第4回日洪シンポジウム
「離散数学とその応用」
２００５年６月３日ー６日
ハンガリ ーブダペシュト

4. japán-magyar konferencia
a véges matematikáról és
alkalmazásairól
2005. június 3-6.
Magyarország, Budapest

Cover design:
Kazuhiko Shiozaki
1999

Preface

The present volume consists of the extended abstracts of the talks presented at the 4th Japanese-Hungarian Symposium on Discrete Mathematics and its Applications (Budapest, June 3-6, 2005).

Based on a long history of cooperation among Japanese and Hungarian scientists in the area of discrete mathematics, the previous symposia in this series took place in Kyoto (March 17-19, 1999), Budapest (April 20-23, 2001) and in Tokyo (January 21-24, 2003).

The 4th Symposium has been jointly organized by the Department of Operations Research, L. Eötvös University of Budapest, by the Department of Computer Science and Information Theory, Budapest University of Technology and Economics, and by the Combinatorics and Discrete Mathematics Research Group of the A. Rényi Institute of Mathematics of the Hungarian Academy of Sciences.

It has been partially supported by the Hungarian National Science Fund (OTKA) and by the Hungarian Academy of Sciences.

The organizers wish to thank all the contributors for submitting papers and all their colleagues, graduate students and sponsors for their assistence and support.

The organizers wish to thank all the contributors for submitting papers and all their colleagues, graduate students and sponsors for their assistence and support.

June 2005

András Frank, Gyula Y. Katona and András Recski
for the Organizing Committee

序言

本書では、第4回日洪シンポジウム「離散数学とその応用」(ブダペスト、2005年6月3日ー6日)の講演予稿がまとめられました。

日本とハンガリーの、離散数学の分野で活動する研究者達の長年に渡る協力により、このシリーズにおける前のシンポジウムは京都(1999年3月17日ー19日)、ブダペスト(2001年4月20日ー23日)、そして東京(2003年1月21日ー24日)で行われました。

第4回のシンポジウムは、ブダペストのL.エトヴェシュ大学（L. Eötvös University of Budapest）のオペレーションズ・リサーチ学科、ブダペスト工科経済大学（Budapest University of Technology and Economics）のコンピュータ科学・情報理論学科、およびハンガリー科学アカデミー（Hungarian Academy of Sciences）のA.レーニ数学研究所（A. Rényi Institute of Mathematics）の組み合わせ論と離散数学の研究グループによって共同で組織されました。

シンポジウムはハンガリー科学アカデミーの科学研究資金（Hungarian National Science Fund 「OTKA」）によって部分的に支援されました。

オーガナイザーは、論文を提出した全ての貢献者の協力、また、全ての同僚、大学院生やスポンサーの援助とサポートを心より感謝します。

2005年6月

組織委員会の
フランク・アンドラーシュ、夏斗南（カトナ）・Y・珠良（ジュラ）、レチキ・アンドラーシュ

Contents

Tighter Bounds on the OBDD size of Integer Multiplication

KAZUYUKI AMANO*
GSIS, Tohoku Univ.
Aoba 6-6-05, Sendai, Miyagi,
980-8579 Japan
ama@ecei.tohoku.ac.jp

AKIRA MARUOKA
GSIS, Tohoku Univ.
Aoba 6-6-05, Sendai, Miyagi,
980-8579 Japan
maruoka@ecei.tohoku.ac.jp

Abstract: We show that the middle bit of the multiplication of two n-bit integers can be computed by an OBDD of size less than $2.8 \cdot 2^{6n/5}$. This improves the previously known upper bound of $(7/3) \cdot 2^{4n/3}$ by Woelfel (STACS, 2001). The experimental results suggest that our exponent of $6n/5$ is (at least very close to) optimal. A general upper bound of $O(2^{3n/2})$ on the OBDD size of each output bit of the multiplication and some conjecture on the structural properties of multipliers inspired by the experimental results are also presented.

Keywords: OBDD, integer multiplication, upper bounds

1 Introduction

Ordered binary decision diagrams (OBDDs), which were first introduced by Bryant [4], are nowadays one of the most well-established computational models for representing and manipulating Boolean functions. OBDDs are widely used in the areas of hardware verification, model checking, and computer aided design (see e.g., [9, 12]).

Definition 1 *Let $X_n = \{x_1, \ldots, x_n\}$ be a set of Boolean variables. A variable ordering π on X_n is a permutation on $\{1, \ldots, n\}$ leading to the ordered list $x_{\pi(1)}, \ldots, x_{\pi(n)}$ of the variables.*

A π-OBDD on X_n is a directed acyclic graph whose sinks are labeled by a constant 0 or 1 and whose inner nodes are labeled by a Boolean variables from X_n. Each inner node has two outgoing edges, one of them labeled by 0, the other by 1. The edges between inner nodes have to respect the variable ordering π, i.e., if an edge leads from an x_i-nodes to an x_j-node, then $\pi^{-1}(i) < \pi^{-1}(j)$. Each node v represents a Boolean function $f_v : \{0,1\}^n \to \{0,1\}$ defined in the following way: An assignment $(a_1, \ldots, a_n) \in \{0,1\}^n$ to variables X_n defines a uniquely determined path from v to one of the sinks. The label of the reached sink gives $f_v(a)$. The size of a π-OBDD is defined as the number of its nodes. The OBDD size of f, denoted by OBDD(f) is the minimum size of all π-OBDDs that compute f. A π-OBDD for some unspecified variable order is simply called OBDD.

For many practically relevant functions, such as symmetric functions, the corresponding OBDD representations are quite small. However, for several important functions, exponential lower bounds on the size of an OBDD representation are known. The most "famous" such function is, probably, (the middle bit of) integer multiplication.

Definition 2 *For each $0 \leq k \leq 2n-1$, let $\mathsf{MUL}_{k,n} : \{0,1\}^{2n} \to \{0,1\}$ denote the Boolean function that outputs z_k of the product $(z_{2n-1} \cdots z_0)$ of two n-bit integers $(x_{n-1} \cdots x_0)$ and $(y_{n-1} \cdots y_0)$, where x_0, y_0 and z_0 are the least significant bits.*

The middle bit of integer multiplication is denoted by $\mathsf{MUL}_{n-1,n}$. Since for any Boolean function on m variables, there exists an OBDD of size $(2+\varepsilon)2^m/m$ [8], the trivial upper bound on OBDD($\mathsf{MUL}_{n-1,n}$) is $O(2^{2n}/n)$. In 1991, Bryant [5] first proved an exponential lower bound of $2^{8/n}$ on OBDD($\mathsf{MUL}_{n-1,n}$). Only recently, Woelfel have succeeded to improve the upper and lower bounds on the size of OBDD for $\mathsf{MUL}_{n-1,n}$ [14]. Precisely, he showed that OBDD($\mathsf{MUL}_{n-1,n}$) is between $2^{n/2}/61$ and $(7/3) \cdot 2^{4n/3}$. His lower bound rules out the possibility of constructing an OBDD for 64-bit multiplication with a reasonable size. Nevertheless, there still exists a considerable gap between the upper and lower bounds.

The main objective of this work is to determine the asymptotic behavior of the size of OBDDs for $\mathsf{MUL}_{n-1,n}$, or more generally, for $\mathsf{MUL}_{k,n}$. Though we started this work by theoretical interest, it may also have an application in the design of multipliers. In addition, we believe that this may help to make progress on the investigations of the complexity of multiplication for more general models of branching programs, which have been extensively studied recently (e.g., [1, 3, 11, 13]).

In the paper, we mainly consider a restricted variant of OBDDs which is called *leveled* OBDDs or *quasi*-OBDDs, denoted by QOBDDs. This is because analyzing the size of QOBDDs is easier than that of OBDDs.

*Corresponding Author. The work was supported in part by Grant-in-Aid for Scientifi c Research on Priority Areas "New Horizons in Computing" from MEXT of Japan.

Definition 3 *A π-QOBDD is a π-OBDD with the additional property that each edge from an $x_{\pi(i)}$-node for $i < n$ reaches an $x_{\pi(i+1)}$-node. In other words, each path in a π-QOBDD examines every variable exactly once in the order of π. Let π-QOBDD(f) denote the minimum size of π-QOBDDs that compute f. The QOBDD size of a Boolean function f, denoted by QOBDD(f) is the minimum size of all π-QOBDDs that compute f, i.e., QOBDD$(f) = \min_\pi \pi$-QOBDD(f). A π-QOBDD for some unspecified variable order is simply called QOBDD.*

Since every π-OBDD can be transformed into a π-QOBDD by inserting dummy nodes on paths from the root to a sink, it is obvious that

$$\mathrm{OBDD}(f) \leq \mathrm{QOBDD}(f) \leq (n+1)\mathrm{OBDD}(f)$$

for every Boolean function f on n variables. Thus, the size of QOBDDs can be considered essentially the same as that of OBDDs, especially for a function having an exponential complexity, such as multiplication. A detailed discussion on the relationship between the OBDD size and the QOBDD size can be found in e.g., [2, 8].

The contributions of the paper are as follows: First, in Section 2, we show that $\mathsf{MUL}_{n-1,n}$ can be computed by a QOBDD of size less than $2.8 \cdot 2^{6n/5}$, which improves the previously known upper bound of $(7/3) \cdot 2^{4n/3}$ [14]. Second, we obtain the optimal QOBDDs for $\mathsf{MUL}_{n-1,n}$ for small values of n by an exhaustive search using a computer, and analyze them. Interestingly, our experimental results suggest that the exponent of $6n/5$ in our upper bound is the true exponent of the QOBDD size of $\mathsf{MUL}_{n-1,n}$. This will be described in Section 3. Next, in Section 4, we give a general upper bound on the QOBDD size of each output bit of integer multiplication. Precisely, we show that $\mathrm{QOBDD}(\mathsf{MUL}_{k,n}) = O(2^c)$ where $c = 6k/5$ for $0 \leq k \leq 5n/4$, $c = 3n/2$ for $5n/4 < k \leq 3n/2$, and $c = 3n-k$ for $3n/2 < k \leq 2n-1$. Finally, in Section 5, we describe some conjecture on the structural properties of multipliers inspired by the results of our experiments, which may lead to a better lower bound on the OBDD size of $\mathsf{MUL}_{n-1,n}$.

2 Upper Bounds for $\mathsf{MUL}_{n-1,n}$

In this section, we show an upper bound of $2.8 \cdot 2^{6n/5}$ on the OBDD size of the middle bit of integer multiplication.

Let $X = (x_{n-1} \cdots x_0)$ be an n-bit binary string. We also use X to denote the integer represented by $x_{n-1} \cdots x_0$, i.e., $X = \sum_{i=0}^{n-1} 2^i x_i$. For $0 \leq i \leq j \leq n-1$, let $[X]_j^i$ be the integer represented by the substring $x_j \cdots x_i$. Formally,

$$[X]_j^i \quad = \quad X \text{ div } 2^i \bmod 2^{j-i+1} = X \bmod 2^{j+1} \text{ div } 2^i.$$

Here we use the operators "mod" that gives the integer reminder of division, and "div" that gives the integer result of division. We abbreviate $[X]_i^i$ by $[X]_i$. For a set S, $|S|$ denotes the cardinality of S.

Theorem 4 *There is a QOBDD for $\mathsf{MUL}_{n-1,n}$ whose size is less than $2.8 \cdot 2^{6n/5}$.*

PROOF: Let $X = (x_{n-1} \cdots x_0)$ and $Y = (y_{n-1} \cdots y_0)$ be the input variables for $\mathsf{MUL}_{n-1,n}$. Let π be the variable ordering $(x_0, y_0, x_1, y_1, \ldots, x_{n-1}, y_{n-1})$. For the sake of simplicity, we suppose that n is a multiple of 5. (Other cases will be discussed later.) Below we show that π-QOBDD$(\mathsf{MUL}_{n-1,n}) \leq 19/7 \cdot 2^{6n/5} < 2.72 \cdot 2^{6n/5}$.

Let $n = 5k$. Let $\mathscr{F}_{i,j}$ denote the set of subfunctions of $\mathsf{MUL}_{n-1,n}$ that obtained by replacing the variables $x_0, \ldots, x_{i-1}$ and $y_0, \ldots, y_{j-1}$ with constants. It is easy to verify that

$$\pi\text{-QOBDD}(\mathsf{MUL}_{n-1,n}) = |\mathscr{F}_{0,0}| + |\mathscr{F}_{1,0}| + |\mathscr{F}_{1,1}| + \cdots + |\mathscr{F}_{n,n}|. \tag{1}$$

This is because that the number of x_i-nodes (y_i-nodes, resp.) in an optimal π-QOBDD for $\mathsf{MUL}_{n-1,n}$ is shown to be $|\mathscr{F}_{i,i}|$ ($|\mathscr{F}_{i+1,i}|$, resp.). Thus, our goal is to bound the number of different subfunctions in $\mathscr{F}_{i,i}$ and in $\mathscr{F}_{i+1,i}$.

We first bound the size of $\mathscr{F}_{i,i}$. Suppose that $n/2 \leq i < n$.

Let $X_L = (x_{i-1} \cdots x_0)$, $Y_L = (y_{i-1} \cdots y_0)$, $X_H = X \backslash X_L = (x_{n-1} \cdots x_i)$ and $Y_H = Y \backslash Y_L = (y_{n-1} \cdots y_i)$, which means that $X = 2^i X_H + X_L$ and $Y = 2^i Y_H + Y_L$. We focus on the "middle part" of $X \cdot Y$, namely, $[X \cdot Y]_{n-1}^i$ of which the most significant bit represents $\mathsf{MUL}_{n-1,n}(X,Y)$. We have

$$\begin{aligned} X \cdot Y \bmod 2^n &= (2^i X_H + X_L) \cdot (2^i Y_H + Y_L) \bmod 2^n \\ &= 2^i (X_H \cdot Y_L + Y_H \cdot X_L) + X_L \cdot Y_L \bmod 2^n \\ &= 2^i (X_H \cdot [Y_L]_{n-i-1}^0 + Y_H \cdot [X_L]_{n-i-1}^0) + [X_L \cdot Y_L]_{n-1}^0 \bmod 2^n. \end{aligned}$$

Further, we have

$$[X \cdot Y]_{n-1}^i = (X \cdot Y \bmod 2^n) \text{ div } 2^i = (X_H \cdot [Y_L]_{n-i-1}^0 + Y_H \cdot [X_L]_{n-i-1}^0) + [X_L \cdot Y_L]_{n-i}^i.$$

This implies that the value of $[X \cdot Y]^i_{n-1}$ and hence $\mathsf{MUL}_{n-1,n}(X,Y)$ is uniquely determined by X_H, Y_H, $[X_L]^0_{n-i-1}$, $[Y_L]^0_{n-i-1}$ and $[X_L \cdot Y_L]^i_{n-1}$. Hence, each subfunction $f_{X_L,Y_L} \in \mathscr{F}_{i,i}$ is uniquely determined by $[X_L]^0_{n-i-1}$, $[Y_L]^0_{n-i-1}$ and $[X_L \cdot Y_L]^i_{n-1}$, each of them has length $n-i$. Therefore, we have $|\mathscr{F}_{i,i}| \le 2^{3(n-i)}$ for $n/2 \le i < n$. Note that this bound is better than the trivial upper bound of $|\mathscr{F}_{i,i}| \le 2^{2i}$ when $i \ge 3n/5$. By an analogous argument to the case of $|\mathscr{F}_{i,i}|$, we can also show that $|\mathscr{F}_{i+1,i}| \le 2^{3(n-i)-1}$ for $n/2 \le i < n$, which is better than the trivial bound of $|\mathscr{F}_{i+1,i}| \le 2^{2i+1}$ for $i \ge 3n/5$.

The upper bound of π-QOBDD($\mathsf{MUL}_{n-1,n}$) is now easily derived by plugging these bounds into Eq. (1). Namely, we have

$$\begin{aligned}
\pi\text{-QOBDD}(\mathsf{MUL}_{n-1,n}) &= \sum_{i=0}^{3k-1}(|\mathscr{F}_{i,i}|+|\mathscr{F}_{i+1,i}|)+\sum_{i=3k}^{5k-1}(|\mathscr{F}_{i,i}|+|\mathscr{F}_{i+1,i}|)+|\mathscr{F}_{n,n}| \\
&\le \sum_{i=0}^{6k-1}2^i+\sum_{i=3k}^{5k-1}(2^{3(5k-i)}+2^{3(5k-i)-1})+2 \\
&= 2^{6k}-1+\left(1+\frac{1}{2}\right)\sum_{i=1}^{2k}2^{3i}+2 \\
&= 2^{6k}+\frac{3}{2}\cdot\frac{8(2^{6k}-1)}{7}+1 < \frac{19}{7}\cdot 2^{6k},
\end{aligned}$$

which completes the proof for the case $n = 5k$.

The other cases, i.e., $n = 5k+\alpha$ for $\alpha \in \{1,2,3,4\}$, can be shown analogously. The upper bounds are $40/7\cdot 2^{6k}$, $96/7\cdot 2^{6k}$, $208/7\cdot 2^{6k}$ and $544/7\cdot 2^{6k}$ for $\alpha = 1,2,3$ and 4, respectively. By a simple case checking, we can see that all these values are bounded by $2.8\cdot 2^{6n/5}$. Note that the largest constant ($= (544/7)/2^{4.8} \sim 2.7897$) is attained when $\alpha = 4$. □

Remark that we use the variable ordering $\pi = (x_0, y_0, \ldots, x_{n-1}, y_{n-1})$ in the proof of Theorem 4, whereas Woelfel [14] used the ordering $\pi = (x_0, \ldots, x_{n-1}, y_0, \ldots, y_{n-1})$ to show the upper bound of $O(2^{4n/3})$. We also remark that Theorem 4 guarantees that the middle bit of a 16-bit multiplication can be computed by a QOBDD with less than 1.5 million nodes. In addition, the proof gives an explicit construction of such a QOBDD. Remarkably, the experimental results suggest that the exponent of $6n/5$ in Theorem 4 is (at least very close to) the true exponent of the QOBDD size of $\mathsf{MUL}_{n-1,n}$, which we will describe in the next section.

3 Empirical Results

In this section, we describe the empirical results supporting the conjecture that the exponent of $6n/5$ in the upper bound in Theorem 4 is optimal.

We did an exhaustive search by using a computer to find the optimal QOBDDs for $\mathsf{MUL}_{n-1,n}$ for small values of n. Note that the best known algorithm for computing an optimal OBDD for a given function has an exponential running time [6, 7]. In addition, we believe that computing an optimal QOBDD is almost as hard as computing an optimal OBDD.

We use a standard dynamic programming approach to compute the size of optimal QOBDDs for $\mathsf{MUL}_{n-1,n}$ which we briefly describe below.

Let f be a Boolean function over the set of variables $X = \{x_1, \ldots, x_n\}$. For $I \subseteq X$, let $\mathrm{sub}(f, I)$ denote the number of subfunctions of f which we obtain by fixing all variables in $X \setminus I$ to constants. It is easy to verify that

$$\mathrm{QOBDD}(f) = \min_{\mathscr{I}=\{I_0,\ldots,I_n\}} \sum_{0\le i\le n} \mathrm{sub}(f, I_i)$$

where the minimum ranges over all sequences of sets $\emptyset = I_0 \subset I_1 \subset \cdots \subset I_n = X$ with $|I_i| = i$. If we define $\mathrm{QOBDD}(f, I)$ for $I \subseteq X$ by the following recursion,

$$\mathrm{QOBDD}(f, I) = \min_{x\in I}\{\mathrm{QOBDD}(f, I\setminus\{x\}) + \mathrm{sub}(f, I)\}, \tag{2}$$

then $\mathrm{QOBDD}(f)$ is given by $\mathrm{QOBDD}(f, X)$. It should be noted that if we replace the term $\mathrm{sub}(f, I)$ in Eq. (2) by $\mathrm{sub}_x(f, I)$, which denotes the number of subfunctions of f obtained by fixing all variables in $X \setminus I$ and that essentially depend on x, then we can obtain $\mathrm{OBDD}(f)$ in a similar fashion [6, 7].

Using the above algorithm, we compute the size of optimal QOBDDs for $\mathsf{MUL}_{n-1,n}$ for $n \le 11$. In addition, we also compute the minimum size of π-QOBDDs for $\mathsf{MUL}_{n-1,n}$ with the variable ordering $\pi = (x_0, y_0, y_1, y_1, \ldots, x_{n-1}, y_{n-1})$, which is used in the proof of Theorem 4.

The results are shown in Table 1. Remarkably, Table 1 shows that the QOBDD size of $\mathsf{MUL}_{n-1,n}$ and also the minimum size of π-QOBDDs for $\mathsf{MUL}_{n-1,n}$ are almost proportional to $2^{6n/5}$. This leads to a conjecture that $\mathrm{QOBDD}(\mathsf{MUL}_{n-1,n}) =$

Table 1: The QOBDD size of $\mathsf{MUL}_{n-1,n}$ is shown in the second column, and the minimum size of π-QOBDDs for $\mathsf{MUL}_{n-1,n}$ with the variable ordering $\pi = (x_0, y_0, \ldots, x_{n-1}, y_{n-1})$ is shown in the fourth column. Note that the computation of $\mathrm{QOBDD}(\mathsf{MUL}_{n-1,n})$ for $n = 12$ has not completed. We have only confirmed that it is not larger than 22,180.

n	QOBDD	QOBDD$/2^{6n/5}$	π-QOBDD	π-QOBDD$/2^{6n/5}$
4	39	1.40	56	2.01
5	72	1.13	109	1.70
6	156	1.06	230	1.56
7	348	1.03	490	1.45
8	797	1.03	1,106	1.43
9	1,808	1.01	2,490	1.40
10	4,106	1.00	5,751	1.40
11	9,796	1.04	13,228	1.41
12	$\leq$22,180	$\leq$1.03	30,862	1.43
13			71,239	1.43
14			166,981	1.46

$\Theta(2^{6n/5})$, which means that the upper bound in Theorem 4 is tight up to a constant factor. Table 1 also shows that the optimal QOBDDs for $\mathsf{MUL}_{n-1,n}$ are almost 30% smaller than the optimal π-QOBDDs.

During the experiments, the optimal variable orderings for $\mathsf{MUL}_{n-1,n}$ are also obtained. For example, one of the optimal variable orderings for $\mathsf{MUL}_{7,8}$ is

$$(x_1, x_2, x_3, x_4, y_3, y_4, y_2, x_5, y_5, y_1, x_6, y_6, x_0, y_7, x_7, y_0),$$

and that for $\mathsf{MUL}_{10,11}$ is

$$(x_3, x_4, x_5, x_6, x_7, y_4, y_5, y_6, y_7, y_3, x_2, y_2, x_1, y_1, x_8, y_8, x_9, x_9, x_0, y_{10}, x_{10}, y_0).$$

We remark that the optimal variable ordering is not unique in general. Generalizing these orders may yield an upper bound with a slightly better constant factor than Theorem 4.

4 General Upper Bounds

In this section, we consider the size of a smallest QOBDD for the k-th bit of integer multiplication for general values of k.

The problem of determining the hardest bit of the multiplication and its complexity is interesting and important since the total complexity of the multiplication may essentially depend on the complexity of the hardest bit. It is well known that the middle bit is the "hardest" bit, in the sense that if it can be computed by OBDDs of size $s(n)$, then any other bit can be computed with size at most $s(2n)$ (e.g., [10]). However, this does not assert that the middle bit is *exactly* the hardest bit. The experimental results suggest that the hardest bit is located higher than the middle. For example, for a 8 bit multiplication, we verified that the 10-th output bit is the hardest for QOBDDs, namely, $\mathrm{QOBDD}(\mathsf{MUL}_{k,8}) = 797, 1623, 1937, 2041, 1755, 1175$ for $k = 7, 8, \ldots, 12$, respectively. Recall that the 0-th bit is the least significant bit.

It is clear that computing $\mathsf{MUL}_{k,n}$ is not harder than computing $\mathsf{MUL}_{k,k+1}$ for every k. This is because if $k < n$, then

$$\mathsf{MUL}_{k,n}(X, Y) \quad = \quad \mathsf{MUL}_{k,k+1}([X]^0_k, [Y]^0_k),$$

and if $k \geq n$, then

$$\mathsf{MUL}_{k,n}(X, Y) \quad = \quad \mathsf{MUL}_{k,k+1}(0^{n-k+1}X, 0^{n-k+1}Y).$$

Hence, the following corollary is a direct consequence of Theorem 4.

Corollary 5 *For every k, there exists a QOBDD for $\mathsf{MUL}_{k,n}$ whose size is $O(2^{6k/5})$.*

Apparently, the upper bound in the above corollary overestimates the actual size of a smallest QOBDD for $\mathsf{MUL}_{k,n}$ if k is close to $2n$. The following theorem asserts that the OBDD size of every single bit of multiplication is bounded by $O(2^{3n/2})$.

Theorem 6 *For every k, there exists a QOBDD for $\mathsf{MUL}_{k,n}$ whose size is $O(2^{3n/2})$.*

For $a \in \{0, \ldots, 2^n - 1\}$, let $\mathsf{MUL}^a_{k,n} : \{0,1\}^n \to \{0,1\}$ be the function that outputs the k-th bit of the product of a with an n-bit number, i.e., $\mathsf{MUL}^a_{k,n}(X) = \mathsf{MUL}_{k,n}(a, X)$. Here the 0-th bit is the least significant bit. To prove the theorem, we use the following lemma, which is a generalization of the results of Woelfel [14]. They showed that $\mathrm{QOBDD}(\mathsf{MUL}^a_{k,n}) = O(2^{n/2})$ for $k = n - 1$.

Lemma 7 *For every k and for every $a \in \{0,\ldots,2^n-1\}$, there exists a QOBDD for $\mathsf{MUL}^a_{k,n}$ whose size is $O(2^{n/2})$.*

PROOF: If $k < n$, then the theorem follows immediately from the result of Woelfel described above. We therefore assume $k \geq n$. Let $Y = (y_{n-1}\cdots y_0)$ be the input variables for $\mathsf{MUL}^a_{k,n}$ and let π be the variable ordering $(y_0,\ldots,y_{n-1})$. Let $\mathscr{F}_i$ be the set of subfunctions of $\mathsf{MUL}^a_{k,n}$ which we obtain by fixing the variables $y_0,\ldots,y_{i-1}$ to constants. We will upper bound the number of subfunctions in $\mathscr{F}_i$. Let $Y_L = (y_{i-1}\cdots y_0)$ and $Y_H = (y_n\cdots y_i)$. Note that

$$\mathsf{MUL}^a_{k,n}(Y) \quad = \quad \mathsf{MUL}^a_{k,n}(Y_H,Y_L) = aY_H2^i + aY_L \bmod 2^{k+1} \operatorname{div} 2^k.$$

For $h \in \{0,\ldots,2^{n-i}-1\}$, let $z_h = ah2^i \bmod 2^k$. Let $\rho : \{0,\ldots,2^{n-i}-1\} \to \{0,\ldots,2^{n-i}-1\}$ be a permutation that satisfies $0 \leq z_{\rho(0)} \leq z_{\rho(1)} \leq \cdots \leq z_{\rho(2^n-1)} < 2^k$. For the sake of simplicity, we denote that $z_{\rho(-1)} = 0$ and $z_{\rho(2^{n-i})} = 2^k$.

We claim that for every two distinct integers $l,l' \in \{0,\ldots,2^i-1\}$ such that $2^k - (al \bmod 2^{k+1})$ and $2^k - (al' \bmod 2^{k+1})$, or $2^{k+1} - (al \bmod 2^{k+1})$ and $2^{k+1} - (al' \bmod 2^{k+1})$ lie in the same interval, precisely,

$$z_{\rho(t-1)} < 2^k - (al \bmod 2^{k+1}) \leq z_{\rho(t)} \text{ and } z_{\rho(t-1)} < 2^k - (al' \bmod 2^{k+1}) \leq z_{\rho(t)}, \tag{3}$$

or

$$z_{\rho(t-1)} < 2^{k+1} - (al \bmod 2^{k+1}) \leq z_{\rho(t)} \text{ and } z_{\rho(t-1)} < 2^{k+1} - (al' \bmod 2^{k+1}) \leq z_{\rho(t)}, \tag{4}$$

for some $0 \leq t \leq 2^{n-i}$, the subfunctions of $\mathsf{MUL}^a_{k,n}$ obtained by fixing Y_L to l and to l' are identical.

The claim is proved as follows. We assume that l and l' satisfy condition (3). (The proof for the case of (4) is analogous to this case.) Since $0 \leq z_{\rho(t-1)} + al \bmod 2^{k+1} < 2^k$ and $0 \leq z_{\rho(t-1)} + al' \bmod 2^{k+1} < 2^k$, we have that $\mathsf{MUL}^a_{k,n}(h,l) = \mathsf{MUL}^a_{k,n}(h,l')$ for every $h = \rho(t')$ with $t' \in \{0,1,\ldots,t-1\}$. Similarly, since $2^k \leq z_{\rho(t)} + al \bmod 2^{k+1} < 2^{k+1}$ and $2^k \leq z_{\rho(t)} + al' \bmod 2^{k+1} < 2^{k+1}$, we have that $\mathsf{MUL}^a_{k,n}(h,l) = \mathsf{MUL}^a_{k,n}(h,l')$ for every $h = \rho(t')$ with $t' \in \{t,\ldots,2^{n-i}-1\}$. Hence, we can conclude that $\mathsf{MUL}^a_{k,n}(h,l) = \mathsf{MUL}^a_{k,n}(h,l')$ for every $h \in \{0,\ldots,2^{n-i}-1\}$, completing the proof of the claim.

The claim implies that the number of subfunctions in $\mathscr{F}_i$ is bounded by the number of such "intervals", i.e., $2(2^{n-i}+1)$, which is smaller than the trivial upper bound of 2^i when $i > (n+1)/2$. Hence, the size of a π-QOBDD is bounded by

$$\pi\text{-QOBDD}(\mathsf{MUL}^a_{k,n}) \quad = \quad \sum_{i=0}^{n} |\mathscr{F}_i| = \sum_{0\leq i\leq \lceil n/2\rceil} 2^i + \sum_{\lceil n/2\rceil < i \leq n} 2(2^{n-i}+1) = O(2^{n/2}).$$

This complete the proof of Lemma 7. □

Theorem 6 follows immediately from Lemma 7.

PROOF:[of Theorem 6] Let $X = (x_{n-1}\cdots x_0)$ and $Y = (y_{n-1}\cdots y_0)$ be the input variables for $\mathsf{MUL}_{k,n}$. We first construct a full binary tree T of depth n which examines all the variables in X. Note that each leaf in T corresponds to an n-bit integer. Then, for each leaf in T that corresponds to an integer a, we connect a QOBDD that computes $\mathsf{MUL}^a_{k,n}$ to the leaf. Lemma 7 guarantees that the resulting QOBDD computes $\mathsf{MUL}_{k,n}$ and whose size is $O(2^n 2^{n/2}) = O(2^{3n/2})$. □

As one might be expected, if the value of k is large enough, then a better upper bound can be obtained.

Theorem 8 *For every $k \geq n$, there exists a QOBDD for $\mathsf{MUL}_{k,n}$ whose size is $O(2^{3n-k})$.*

PROOF: Let $X = (x_{n-1}\cdots x_0)$ and $Y = (y_{n-1}\cdots y_0)$ be the input variables for $\mathsf{MUL}_{k,n}$. Let π be the variable ordering $(x_{n-1},y_{n-1},x_{n-2},y_{n-2},\ldots,x_0,y_0)$. Let $\mathscr{F}_{i,j}$ denote the set of subfunctions of $\mathsf{MUL}_{k,n}$ that obtained by fixing the variables $x_{n-1},\ldots,x_{n-i}$ and $y_{n-1},\ldots,y_{n-j}$ to constants. Note that the suffixes i and j indicate the number of fixed variables in X and Y, respectively.

In the following, we bound the number of subfunctions in $\mathscr{F}_{i,i}$. Let $X_H = (x_{n-1}\cdots x_{n-i})$, $Y_H = (y_{n-1}\cdots y_{n-i})$, $X_L = (x_{n-i-1}\cdots x_0)$ and $Y_L = (y_{n-i-1}\cdots y_0)$. We obtain

$$\begin{aligned} X\cdot Y \bmod 2^{k+1} &= (2^{n-i}X_H + X_L)\cdot(2^{n-i}Y_H + Y_L) \bmod 2^{k+1} \\ &= 2^{2(n-i)}X_H\cdot Y_H + 2^{n-i}(X_H\cdot Y_L + X_L\cdot Y_H) + X_L\cdot Y_L \bmod 2^{k+1}. \end{aligned} \tag{5}$$

Suppose that $k \geq 2n-i+3$. Let $l = k-(2n-i)+1$. For $0 \leq x < 2^n$, let x_H and x_L denote two integers that satisfy $x = 2^{n-i}x_H + x_L$ where $0 \leq x_H < 2^i$ and $0 \leq x_L < 2^{n-i}$. For $0 \leq y < 2^n$, y_H and y_L are defined analogously. For a pair of integers (x_H,y_H), let z_H and z_L be two integers, depending on (x_H,y_H), such that

$$2^{2(n-i)}x_H\cdot y_H \bmod 2^{k+1} = 2^{2n-i}z_H + z_L, \tag{6}$$

where $0 \leq z_H < 2^l$ and $0 \leq z_L < 2^{2n-i}$. From Eqs. (5) and (6), $\mathsf{MUL}_{k,n}(x,y)$ is given by the most significant bit of

$$2^{2n-i}z_H + z_L + 2^{n-i}(x_H \cdot y_L + x_L \cdot y_H) + x_L \cdot y_L \bmod 2^{k+1}.$$

Since

$$z_L + 2^{n-i}(x_H \cdot y_L + x_L \cdot y_H) + x_L \cdot y_L < 3 \cdot 2^{2n-i}, \tag{7}$$

if the j-th bit of the binary representation of z_H is 0 for some $j \geq 2$, then the carry generated by Eq. (7) is not propagated to a higher bit than the j-th bit of z_H. Hence, for every pairs of integers (x_H, y_H) that satisfies $[z_H]_j = 0$ for some $j \geq 2$, the subfunction of $\mathsf{MUL}_{k,n}$ obtained by fixing X_H to x_H and Y_H to y_H is a constant function. Therefore, the number of subfunctions in $\mathscr{F}_{i,i}$ is bounded by 2 plus the number of pairs of integers (x_H, y_H) that satisfy the condition that $[z_H]_j = 1$ for every $2 \leq j \leq l-2$, equivalently, the output bits $i+2$ through $i+l-2$ of the product $x_H \cdot y_H$ are all 1. The number of such pairs is at most

$$\begin{aligned} \sum_{1 \leq y < 2^i} \lceil \frac{2^{i+2}}{y} \rceil \lceil \frac{y2^i}{2^{i+l-1}} \rceil &< \sum_{1 \leq y < 2^i} 2\frac{2^{i+2}}{y} \cdot 2\frac{y2^i}{2^{i+l-1}} \\ &= \sum_{1 \leq y < 2^i} 2^{i-l+5} < 2^{2i-l+5}. \end{aligned}$$

Hence, we have

$$|\mathscr{F}_{i,i}| \leq 2 + 2^{2i-l+5} < 2^{2i-k+(2n-i)+7} = 2^{2n-k+i+7}, \tag{8}$$

which is better than the trivial bound of $|\mathscr{F}_{i,i}| \leq 2^{2i}$ when $i \geq 2n-k+7$. The total size of π-QOBDD for $\mathsf{MUL}_{k,n}$ is given by

$$\pi\text{-QOBDD}(\mathsf{MUL}_{k,n}) = \sum_{0 \leq i < n} (|\mathscr{F}_{i,i}| + |\mathscr{F}_{i+1,i}|) + |\mathscr{F}_{n,n}| \leq 3 \sum_{0 \leq i < n} |\mathscr{F}_{i,i}| + 2. \tag{9}$$

The last inequality follows from the simple fact that $|\mathscr{F}_{i+1,i}| \leq 2|\mathscr{F}_{i,i}|$ for every i. By plugging Eq (8) into Eq. (9), we have

$$\begin{aligned} \pi\text{-QOBDD}(\mathsf{MUL}_{k,n}) &\leq 3\left(\sum_{0 \leq i \leq 2n-k+6} 2^{2i} + \sum_{2n-k+7 \leq i < n} 2^{2n-k+i+7} \right) + 2 \\ &\leq 3(2^{4n-2k+13} + 2^{3n-k+7}) + 2 = O(2^{3n-k}). \end{aligned}$$

The last equality follows from the assumption that $k \geq n$. □

Combining Corollary 5, Theorems 6 and 8, we have the following theorem.

Theorem 9 *The QOBDD size of* $\mathsf{MUL}_{k,n}$ *is* $O(2^c)$ *where*

$$c = \begin{cases} 6k/5, & \text{for } 0 \leq k \leq 5n/4, \\ 3n/2, & \text{for } 5n/4 < k \leq 3n/2, \\ 3n-k, & \text{for } 3n/2 < k \leq 2n-1. \end{cases}$$

The theorem says that every single bit of the multiplication of two n-bit integers can be computed by a QOBDD (and also by an OBDD) of size $O(2^{3n/2})$. Note that the best known lower bound for $\mathsf{MUL}_{k,n}$ is $2^{\lfloor (k+1)/2 \rfloor}/61$ for $k < n$ and $2^{\lfloor (2n-k-1)/2 \rfloor}/61$ for $k \geq n$ by Woelfel [14].

5 Concluding Remarks

Although we have not succeeded to improve the lower bound on the size of OBDDs for $\mathsf{MUL}_{n-1,n}$, the experiments gave us some insight to improvements. Inspired by our experimental results, we make the following conjecture.

Let $s(i)$, $i = 0, 1, \ldots$, be a sequence of integers defined by $s(0) = 1$ and

$$s(i) = \begin{cases} (6s(i-1)-1)/5 & \text{if } i \bmod 4 \text{ is } 0, \\ (5s(i-1)-1)/3 & \text{if } i \bmod 4 \text{ is } 2, \\ 2s(i-1) & \text{if } i \text{ is odd.} \end{cases} \tag{10}$$

The first few numbers of this sequence are: 1,2,3,6,7,14,23,46,55,110,183,366,439,.... Note that if i is a multiple of 4, then $s(i)$ has a simple closed formula, i.e., $s(4j) = (6 \cdot 2^{3j} + 1)/7$. This can be verified by observing that $s(4(j+1)) = 8 \cdot s(4j) - 1$ from Eq. (10).

Conjecture 10 *Let $X = (x_{n-1} \cdots x_0)$ and $Y = (y_{n-1} \cdots y_0)$ be the input variables for $\mathsf{MUL}_{n-1,n}$. For $I \subseteq X \cup Y$, let $\mathscr{F}_I$ be the set of subfunctions of $\mathsf{MUL}_{n-1,n}$ obtained by fixing all variables in I to constants. Define a variable ordering π as*

$$\pi = (x_0, y_{n-1}, x_{n-1}, y_0, x_1, y_{n-2}, x_{n-2}, y_1, \ldots, x_j, y_{n-1-j}, x_{n-1-j}, y_j, \ldots).$$

For every $n \geq 4$, the following are true:

(i) For every $0 \leq k \leq \lfloor 4(n-1)/3 \rfloor$, the k-th level of π-QOBDD with minimum size for $\mathsf{MUL}_{n-1,n}$ has exactly $s(k)$ nodes, i.e.,

$$|\mathscr{F}_{\{\pi(1),\ldots,\pi(k)\}}| = s(k).$$

(ii) For every $0 \leq k \leq \lfloor 4(n-1)/3 \rfloor$, the ordering π is optimal in terms of minimizing the number of nodes at the k-th level of a QOBDD for $\mathsf{MUL}_{n-1,n}$, i.e.,

$$\min_{I:|I|=k} |\mathscr{F}_I| = |\mathscr{F}_{\{\pi(1),\ldots,\pi(k)\}}|.$$

The statement (i) of the conjecture is verified experimentally for every $n \leq 14$, and the statement (ii) is verified for every $n \leq 11$. An affirmative answer to Conjecture 10 implies that the number of nodes at the k-th level in a QOBDD for $\mathsf{MUL}_{n-1,n}$ is at least $s(k)$ for $k \sim 4n/3$. This immediately implies a lower bound of $s(4n/3) = \Omega(2^n)$ on the QOBDD size of $\mathsf{MUL}_{n-1,n}$.

References

[1] F. ABLAYEV AND M. KARPINSKI, A Lower Bound for Integer Multiplication on Randomized Ordered Read-Once Branching Programs, *Inf. Comput.*, **186**, 78–89 (2003)

[2] B. BOLLIG AND I. WEGENER, Asymptotically Optimal Lower Bounds for OBDDs and the Solution of Some Basic OBDD Problems, *J. Comp. Sys. Sci.*, **61**, 558–579 (2000)

[3] B. BOLLIG AND P. WOELFEL, Read-Once Branching Program Lower Bound of $\Omega(2^{n/4})$ for Integer Multiplication Using Universal Hashing, *Proc. 33rd STOC*, 419–424 (2001)

[4] R. E. BRYANT, Graph-Based Algorithms for Boolean Function Manipulation, *IEEE Trans. Comput.*, **35**, 677–691 (1986)

[5] R. E. BRYANT, On the Complexity of VLSI Implementations and Graph Representations of Boolean Functions with Applications to Integer Multiplication, *IEEE Trans. Comput.* **40**, 205–213 (1991)

[6] S.J. FRIEDMAN AND K.J. SUPOWIT, Finding the Optimal Variable Ordering for Binary Decision Diagrams, *IEEE Trans. Comput.* **39**, 710–713 (1990)

[7] N. ISHIURA, H. SAWADA AND S. YAJIMA, Minimization of Binary Decision Diagrams Based on the Exchanges of Variables, *Proc. IEEE International Conf. on Computer-Aided Design*, 472–475 (1991)

[8] H. T. LIAW, C.S. LIN, On the OBDD-Representation of General Boolean Functions, *IEEE Trans. Comput.* **41(6)**, 661–664 (1992)

[9] C. MEINEL AND T. THEOBALD, *Algorithms and Data Structures in VLSI Design – OBDD Foundations and Applications*, Springer-Verlag, Berlin (1998)

[10] S. PONZIO, A Lower Bound for Integer Multiplication with Read-Once Branching Programs, *SIAM J. Comput.* **28(3)**, 798–815 (1998)

[11] M. SAUERHOFF AND P. WOELFEL, Time-Space Tradeoff Lower Bounds for Integer Multiplication and Graphs of Arithmetic Functions, *Proc. 35th STOC*, 186–194 (2003)

[12] I. WEGENER, BDDs – design, analysis, complexity, and applications *Disc. Appl. Math.* **138**, 229–251 (2004)

[13] I. WEGENER AND P. WOELFEL, New Results on the Complexity of the Middle Bit of Multiplication, *ECCC TR04-107* (2004) (also to appear in *Proc. 20th Comput. Complexity* (2005))

[14] P. WOELFEL, New Bounds on the OBDD-Size of Integer Multiplication via Universal Hashing, *Proc. 18th STACS*, LNCS 2010, 563–574 (2001)

Congestion Minimization Confluent Flow Problem: Experimental Evaluation of Algorithms

KOICHI NIHEI

Information and System Engineering Course
Graduate School of Science and Engineering
Chuo University
Bunkyo-ku, Tokyo 112-8551, Japan

TAKAO ASANO[†]

Department of Information Engineering
Chuo University
Bunkyo-ku, Tokyo 112-8551, Japan
asano@ise.chuo-u.ac.jp

Abstract: A flow is called confluent if, at every node, all the flow leaves along a single edge. The congestion minimization confluent flow problem is stated as follows: given a directed graph, sinks and nonnegative demands on nodes, find a minimum congestion confluent flow that routes all the demands to sinks. For this problem, a $(1+\ln k)$-approximation algorithm was proposed by Chen et al. in 2004, where k is the number of sinks. In this paper, we evaluate its experimental performance and show that a greedy post-processing works well.

Keywords: NP-hard, confluent flow, approximation algorithm, congestion

1 Introduction

A flow is called confluent if, at every node, all the flow leaves along a single edge. As Chen et al. [1, 2] described, confluent flows arise in many applications including evacuation problem, CDNs (content delivery networks), and Internet routing. The congestion minimization confluent flow problem is stated as follows: given a directed graph, sinks and nonnegative demands on nodes, find a minimum congestion confluent flow that routes all the demands to sinks. This problem was first posed by Chen, Rajaraman and Sundaram [2]. They presented an $O((\log n)^3)$-approximation algorithm for digraphs with unit demand (i.e. each node has unit demand). Chen, Kleinberg, Lovász, Rajaraman, Sundaram and Vetta [1] proposed a $(1+\ln k)$-approximation algorithm, where k is the number of sinks and each node has arbitrary demand. They also presented a simpler $(1+\log k)$-approximation algorithm and showed that there is no $(\frac{1}{2}\log k)$-approximation algorithm unless $\mathbf{P}=\mathbf{NP}$.

In this paper, we propose a simple greedy post-processing of their algorithms, evaluate experimental performances of algorithms by Chen et al. and algorithms with a greedy post-processing, and show that a greedy post-processing works quite well.

2 Problem Formulation

For a directed graph $G=(V,E)$, let $d: V \to \mathbb{R}_+$ be nonnegative demands of nodes of G and $S=\{s_1,\ldots,s_k\} \subset V$ be a set of k sinks. Throughout this paper we use $n := |V|$, $m := |E|$ and assume every sink is of out-degree 0. Let $\delta_G^+(v)$ ($\delta_G^-(v)$) denote the set of edges in G leaving from (entering into) node v. A function $f: E \to \mathbb{R}_+$ is called a *flow* on G if, for every node $v \in V-S$, the following flow conservation holds:

$$\sum_{e\in\delta_G^+(v)} f(e) - \sum_{e\in\delta_G^-(v)} f(e) = d(v).$$

The *congestion of node v* in flow f is defined by

$$c_f(v) := d(v) + \mathrm{in}_f(v),$$

where $\mathrm{in}_f(v) := \sum_{e\in\delta_G^-(v)} f(e)$. Then the *congestion of flow* f is defined by

$$\max_{v\in V} c_f(v).$$

Flow f is called *confluent* if, at every node $v \in V-S$, at most one edge $e \in \delta_G^+(v)$ has positive $f(e)$. The *support* of a flow f is defined to be the subgraph of G induced by the edges e with $f(e)>0$. From now on, we use the word "*splittable flow*" for "flow" when we need to distinguish "flow" from "confluent flow".

[†]This research is partially supported by the Grant-in-Aid for Scientifi c Research and the 21st Century COE Program of Japan.

The support of a confluent flow f consists of k connected components $\{T_1, \ldots, T_k\}$, where each $T_i = (V(T_i), E(T_i))$ is an anti-arborescence rooted at sink s_i. The maximum congestion of T_i is achieved at s_i and we denote it by $d_f(T_i) := \sum_{v \in V(T_i)} d(v)$. Thus the congestion of f is

$$\max_{i \in \{1,\ldots,k\}} d_f(T_i)$$

and the congestion minimization confluent flow problem is to find a confluent flow f minimizing $\max_{i \in \{1,\ldots,k\}} d_f(T_i)$.

For the congestion minimization confluent flow problem, it is shown by Chen et al. [1] using the reduction from the NP-complete vertex-disjoint path problem that approximation ratio $\frac{1}{2}\log_2 k$ cannot be improved unless $\mathbf{P} = \mathbf{NP}$. Chen, Rajaraman and Sundaram [2] proposed a randomized $O((\log n)^3)$-approximation algorithm for the unit demand case based on randomized rounding. Chen et al. [1] proposed deterministic $(1+\log_2 k)$- and $(1+\ln k)$-approximation algorithms for the general case by using the congestion of a splittable flow as lower bound. They also showed that $H_k = \sum_{i=1}^{k} \frac{1}{i}$ is an integrality gap (i.e., the ratio between the optimal value of a confluent flow and that of a splittable flow) and claimed that $(1+\ln k)$-approximation algorithm is almost tight since $\ln(k+1) < H_k < 1 + \ln k$.

3 Outlines of Chen et al. algorithms [1]

In this section, we present outlines of $(1+\log_2 k)$- and $(1+\ln k)$-approximation algorithms of Chen et al. [1]. The $(1+\ln k)$-approximation algorithm is a modification of the $(1+\log_2 k)$-approximation algorithm.

Both algorithms first find a splittable flow f with minimum congestion (i.e., a splittable flow f minimizing $\max_{s \in S} c_f(s)$). This can be done using a binary search based on a maximum flow algorithm. We assume $\max_{s \in S} c_f(s) = 1$, since all $f(e)$ and $d(v)$ can be divided by $\max_{s \in S} c_f(s)$ when considering an approximation ratio. Let $G(f)$ be the support of f. Then we can assume without loss of generality that $G(f)$ has no directed cycle. For a directed subgraph G' of G with no directed cycle, we call a node *frontier* if it has an edge in G' into a sink.

$(1+\log_2 k)$-**approximation algorithm** [1]

Find a minimum congestion splittable flow f.
Let $c_{\max} := \max_{s \in S} c_f(s)$, $f(e) := f(e)/c_{\max}$ and $d(v) := d(v)/c_{\max}$.
(Note that now $c_{\max} := \max_{s \in S} c_f(s) = 1$.)
For each sink $s_i \in S$, mark it active.

main loop: while $|V(G(f))| > k$ **do** the following:

1. **(node aggregation)**
   if there is a frontier node v such that all edges in $\delta^+_{G(f)}(v)$ go to some sink s_i **then**:
   Mark one of these edges and contract v into s_i (G and $G(f)$ need to be updated).
   Let $d_f(s_i) := d_f(s_i) + d(v)$.
   Go to main loop.

2. **(breaking sawtooth cycle)**
   Let $\hat{G}$ be the graph obtained from $G(f)$ by adding the reverse edge $e_{\text{rev}} = (s, v)$ corresponding to each edge $e = (v, s)$ of $G(f)$ joining a frontier node v to a sink s.
   if $\hat{G}$ has a directed cyle C of length at least 3 **then**:
   Let $f_{\min} := \min_{e \in C \cap E(G(f))} f(e)$.
   Update f as follows: $f(e) := f(e) - f_{\min}$ ($e \in C$) and $f(e) := f(e) + f_{\min}$ ($e_{\text{rev}} \in C$).
   Go to main loop.

3. **(sink deactivation)**
   Let s_j be a sink with exactly one adjacent frontier node v.
   (such a node s_j always exists in this case).
   Let s_ℓ be arbitrary sink adjacent to v.
   if $c(s_j) + f(v, s_\ell) < c(s_\ell) - f(v, s_\ell)$ **then**:
   set $f(v, s_j) := f(v, s_j) + f(v, s_\ell)$ and $f(v, s_\ell) := 0$
   else:
   set $f(v, s_\ell) := f(v, s_\ell) + f(v, s_j)$ and $f(v, s_j) := 0$ and mark s_j inactive.
   Go to main loop.

output: Output the marked edges.

If the conditions in 1 and 2 above are violated, then the subgraph G' of $G(f)$ induced by the sinks and frontier nodes contains no cycles of length more than 1. That is, a simple graph of G' (which is obtained by deleting the self-loops and remaining just one edge of each set of parallel edges) is a forest (if we ignore the direction of edges). Furthermore, it contains more sinks than frontier nodes, and has a sink of degree 1. Thus such a node can be chosen as s_j.

The $(1+\ln k)$-approximation algorithm is obtained just by replacing (sink deactivation) step with the following (parsimonious sink deactivation) step. For completeness, we describe all steps.

$(1+\ln k)$-approximation algorithm [1]

Find a minimum congestion splittable flow f.
Let $c_{\max} := \max_{s\in S} c_f(s)$, $f(e) := f(e)/c_{\max}$ and $d(v) := d(v)/c_{\max}$.
(Now note that $c_{\max} := \max_{s\in S} c_f(s) = 1$.)
For each sink $s_i \in S$, mark it active.

main loop: while $|V(G(f))| > k$ **do** the following:

1. **(node aggregation)**
   if there is a frontier node v such that all edges in $\delta^+_{G(f)}(v)$ go to some sink s_i **then**:
   Mark one of these edges and contract v into s_i (G and $G(f)$ need to be updated).
   Let $d_f(s_i) := d_f(s_i) + d(v)$.
   Go to main loop.

2. **(breaking sawtooth cycle)**
   Let $\hat{G}$ be the graph obtained from $G(f)$ by adding the reverse edge $e_{\text{rev}} = (s,v)$ corresponding to each edge $e = (v,s)$ of $G(f)$ joining a frontier node v to a sink s.
   if $\hat{G}$ has a directed cyle C of length at least 3 **then**:
   Let $f_{\min} := \min_{e\in C\cap E(G(f))} f(e)$.
   Update f as follows: $f(e) := f(e) - f_{\min}$ ($e \in C$) and $f(e) := f(e) + f_{\min}$ ($e_{\text{rev}} \in C$).
   Go to main loop.

3. **(parsimonious sink deactivation)**
   Find a subgraph G_1 of $G(f)$ satisfying the following:

   - all edges of G_1 join frontier nodes and sinks (thus G_1 is a bipartite graph).
   - if a frontier node v is in G_1 then all edges in $\delta^+_{G(f)}(v)$ are contained in G_1 (thus there is no edge of $G(f)$ from v to another frontier node).
   - if a sink s is G_1 then all edges in $\delta^-_{G(f)}(s)$ (except self-loops) are contained in G_1.
   - every frontier node is adjacent to at least two sinks.

   Let S_1 (F_1) be the set of sinks (frontier nodes) in G_1 and do the following:

   i. **(balancing)** Find a splittable flow f_1 in G_1 with $c_f(v) = c_{f_1(v)}$ for each $v \in F_1$ that minimizes $\sum_{s_i\in S_1} e^{c_{f_1}(s_i)}$. Set $f(e) := f_1(e)$ for each edge e in G_1 and delete edges e with $f(e) = 0$ from G_1.
   ii. Find a sink $s \in S_1$ minimizing $\text{in}_f(s)$. For each frontier node v adjacent to s, reroute all the flow $f(v,s)$ to another adjacent sink (flow f needs to be updated) and delete edge (v,s) from G_1. Mark s inactive and set $S_1 := S_1 - \{s\}$.
   iii. **(balancing)** Find a splittable flow f_1 in G_1 with $c_f(v) = c_{f_1(v)}$ for each $v \in F_1$ that minimizes $\sum_{s_i\in S_1} e^{c_{f_1}(s_i)}$. Set $f(e) := f_1(e)$ for each edge e in G_1.

   Go to main loop.

output: Output the marked edges.

If we decompose $\hat{G}$ into the strongly connected components then graph G_1 is a component containing sinks but not containing any edges into other components.

Let x_e and y_s denote $f_1(e)$ and $c_{f_1}(s)$, respectively. Then the minimization problem of $\sum_{s_i\in S_1} e^{c(s_i)}$ can be formulated in

terms of convex programming as follows:

$$\begin{aligned} \min \quad & \sum_{s\in S_1} e^{y_s} && (1)\\ \text{s.t.} \quad & \sum_{e\in\delta^+_{G_1}(v)} x_e \geq c_f(v) && (v\in F_1)\\ & y_s \geq \sum_{e\in\delta^-_{G_1}(s)} x_e + d(s) && (s\in S_1)\\ & x_e \geq 0 && (e\in E(G_1)) \end{aligned}$$

This convex programming problem can be solved in polynomial time. Actually, Chen et al. [1] showed how they solve this problem in polynomial time based on a maximum flow algorithm.

They also presented analysis that the approximation ratios of these algorithms are $1+\log_2 k$ and $1+\ln k$.

4 Greedy Post-Processing

Experimental performance of Chen et al. algorithms may be greatly improved by a greedy post-processing. In the following local search algorithm, for a confluent flow f, we denote by $T_{\max}\subseteq\{T_1,\ldots,T_k\}$ a connected component of the support of f with maximum congestion and by $s_{\max}$ the sink of $T_{\max}$. Let $p(u)$ be the parent of node u in $T_{\max}$.

Greedy Post-Processing

1. For each edge $e=(u,v)\in E(G)$ with $u\in T_{\max}$ and $v\in T_i$ ($T_i\neq T_{\max}$), compute the congestion $r(e)$ of the sink s_i of T_i when we reroute all the flow $f(u,p(u))$ in $T_{\max}$ to v.
2. Find an edge $e_{\min}=(u_{\min},v_{\min})$ minimizing $r(e)$ among all such edges $e=(u,v)$.
3. **If** $c_f(s_{\max})>r(e_{\min})$ **then:** set $f(e_{\min}):=c(u_{\min})$, $f(u_{\min},p(u_{\min})):=0$ and go to 1 **else:** output f.

5 Computational Experiment

In this section, we investigate, through computational experiments, experimental performances of five approximation algorithms as follows:

A. $(1+\log_2 k)$-approximation algorithm [1].

B. $(1+\ln k)$-approximation algorithm [1].

C. Greedy Post-Processing for a solution obtained by Algorithm A.

D. Greedy Post-Processing for a solution obtained by Algorithm B.

E. Greedy Post-Processing for an initial solution obtained by routing each demand to a nearest sink.

We evaluate the experimental performance by considering the ratio of the congestion of a confluent flow obtained by algorithms to the congestion of an optimal splittable flow. We generate one thousand of instances for each triple (n,m,k) and compute their solutions, where n, m, and k are the numbers of nodes, edges, and sinks, respectively. We use the following environments.

Table 1: environment of experiments

CPU	Intel Pentium4 1.7 GHz
memory	1,024 MB RDRAM
OS	Red Hat Linux 7.3
complier	g++ 2.96

5.1 Computational Results

5.1.1 Results for changing the number of sinks

Figure 1 shows the average performance of 1000 instances with $n = 200$, $m = 1000$ and k as parameter. Demands of nodes are randomly generated from a uniform distribution. All the results show that performance is worst when $k = 100 = \frac{n}{2}$. The performance becomes better as $|\frac{k}{n} - \frac{1}{2}|$ becomes large. We have obtained almost the same results for instances with $(n,m,k) = (100,1000,k)$ and instances with $(n,m,k) = (300,1000,k)$. The performance is worst when $k = \frac{n}{2}$.

To investigate this tendency, we try to find optimal confluent flows for instances of small size and compare their performances with those of approximate solutions obtained by Algorithms B and D. Figure 2 shows that the average performance ratios of optimal confluent flows, solutions obtained by algorithms B and D to the congestions of optimal splittable flows. This figure shows that when $k = \frac{n}{2}$ the integrality gap becomes large and the performance ratios of solutions obtained by Algorithms B and D to optimal confluent flows are almost independent of parameter k. This suggests that almost the same results work for instances of large size.

Table 2 shows that, if we fix $k = \frac{n}{2}$, the performance ratios of solutions obtained by Algorithms B and D to optimal splittable flows become worse as n becomes large.

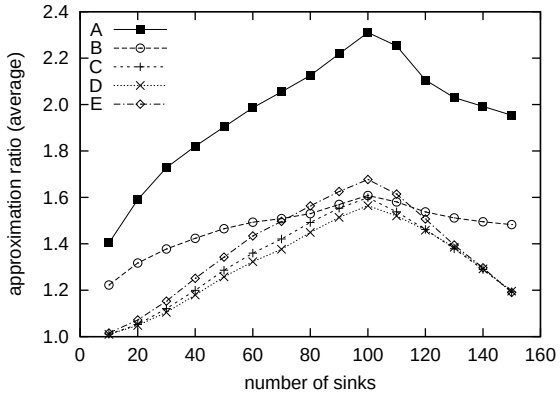


Figure 1: Changing the number of sinks

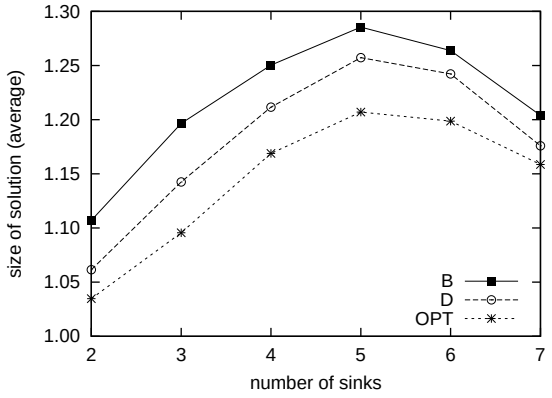


Figure 2: Comparison with optimal splittable flow

5.1.2 Results for changing the number of edges

Figure 3 shows the average performance of 1000 instances with $n = 200$, $k = 20$ and m as parameter. Since $|\frac{k}{n} - \frac{1}{2}| = \frac{2}{5}$ is large, Greedy Post-Processing works quite well. As the number of edges becomes large, the performance becomes slightly better.

Table 2: Changing n with $k = \frac{n}{2}$

instance			performance ratio (average)	
n	m	k	B	D
100	500	50	1.554324	1.527063
200	1,000	100	1.608197	1.564448
300	1,500	150	1.638836	1.588455

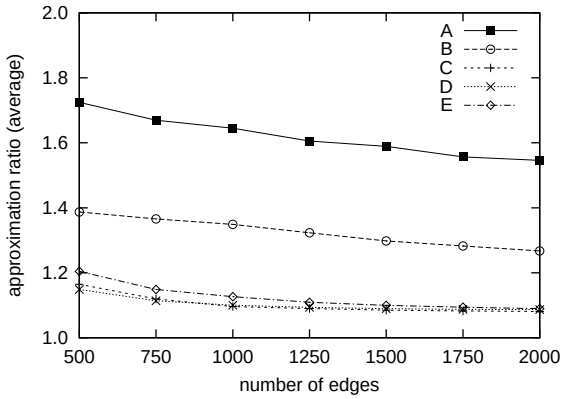


Figure 3: Changing the number of edges

5.1.3 Results for changing demands

In this case, we consider $d(v)$ to be dependant on the distance from v to sinks. Let $h(v)$ be the number of edges of a shortest path from v to a nearest sink. We consider two cases: $d(v) := h(v) + 1$ (increasing demands) and $d(v) := \frac{1}{h(v)+1}$ (decreasing demands). We compare these results with the result for uniformly distributed random $d(v)$. Table 3 shows the average performance of 1000 instances with $n = 200$, $m = 1000$ and $k = 50$.

Algorithms A and B of Chen et al. are almost stable for increasing demands, while they are better for decreasing demands than for uniformly distributed random demands. Since these algorithms fix flows on edges leaving frontier nodes first (i.e., in a kind of nearest-to-sink-first order), they may work well for the decreasing demands.

Algorithm C is worse for increasing demands and decreasing demands than for uniformly distributed random demands. Since Algorithm A is better for decreasing demands than for uniformly distributed random demands, this shows that Greedy Post-Processing does not always work better for better initial solutions.

Algorithm D is better for decreasing demands than for uniformly distributed random demands. For increasing demands, it becomes slightly worse, however, it produces best solutions among all algorithms here for increasing demands.

For uniformly distributed random demands, Algorithm E is better than Algorithms A and B, however, for increasing demands and decreasing demands, it becomes quite worse. The number of iterations in Algorithm E is 57.8 on the average for uniformly distributed random demands while 3.1 on the average for increasing demands. This shows that Algorithm E makes almost no improvement on an initial solution for increasing demands.

5.1.4 Improvement on Initial Solutions by Greedy Post-Processing

Table 4 shows improvements on initial solutions made by Greedy Post-Processing 1000 instances with $n = 200$, $m = 1000$, $k = 50$ and uniformly distributed random demands. Here, initial solutions are produced by Algorithm B. Most solutions obtained by Algorithm D are with performance in $[1.2, 1.3)$. This shows that Greedy Post-Processing works well for all initial solutions produced by Algorithm B. It does not depend much on the goodness of the initial solutions produced by Algorithm B.

Table 3: Changing demands

approx.	uniform random					increasing					decreasing				
ratio	A	B	C	D	E	A	B	C	D	E	A	B	C	D	E
[1.0,1.2)	0	0	96	205	24	0	0	0	5	0	0	22	0	227	0
[1.2,1.4)	0	294	819	754	747	0	370	18	811	0	20	674	388	757	0
[1.4,1.6)	35	597	79	40	207	53	524	392	170	0	355	291	523	16	20
[1.6,1.8)	312	102	6	1	22	257	93	437	13	2	404	12	77	0	261
[1.8,2.0)	379	7	0	0	0	441	13	130	1	26	178	1	12	0	344
[2.0,2.2)	184	0	0	0	0	161	0	19	0	146	38	0	0	0	250
[2.2,2.4)	67	0	0	0	0	70	0	4	0	250	4	0	0	0	102
[2.4,2.6)	21	0	0	0	0	13	0	0	0	213	1	0	0	0	16
[2.6,2.8)	1	0	0	0	0	5	0	0	0	188	0	0	0	0	5
[2.8,3.0)	0	0	0	0	0	0	0	0	0	92	0	0	0	0	2
[3.0,3.2)	1	0	0	0	0	0	0	0	0	42	0	0	0	0	0
[3.2,3.4)	0	0	0	0	0	0	0	0	0	21	0	0	0	0	0
[3.4,3.6)	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0
[3.6,3.8)	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0
[3.8,4.0)	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0
[4.0,4.2)	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

Table 4: Improvement on Initial Solutions by Greedy Post-Processing

Initial Sol-	Improvement by Greedy Post-Processing (Solution of D)							
tion of B	[1.1,1.2)	[1.2,1.3)	[1.3,1.4)	[1.4,1.5)	[1.5,1.6)	[1.6,1.7)	[1.7,1.8)	total
[1.2,1.3)	9	27	0	0	0	0	0	36
[1.3,1.4)	63	151	44	0	0	0	0	258
[1.4,1.5)	77	224	77	18	0	0	0	396
[1.5,1.6)	41	116	35	5	4	0	0	201
[1.6,1.7)	14	42	20	7	1	0	0	84
[1.7,1.8)	1	10	3	3	0	0	1	18
[1.8,1.9)	0	1	2	0	2	0	0	5
[1.9,2.0)	0	1	1	0	0	0	0	2
total	205	572	182	33	7	0	1	1,000

Table 5: Running Time and Number of Iterations

Instance				A	B	C		D		E	
n	m	k	Dmnd.	Time	Time	Time	Iters.	Time	Iters.	Time	Iters.
200	1,000	50	Incr.	0.1017	0.2186	0.1130	1.75	0.2330	2.30	0.0115	3.14
200	1,000	50	Decr.	0.0961	0.2150	0.1068	2.04	0.2260	1.21	0.0114	3.59
200	1,000	50	Rndm.	0.0988	0.2279	0.1572	18.51	0.2492	5.67	0.1346	57.84
200	1,000	100	Rndm.	0.1441	0.4933	0.2087	15.79	0.5006	1.04	0.1718	60.97
200	2,000	50	Rndm.	0.1144	0.2421	0.3289	28.46	0.3355	10.12	0.5061	93.44
500	5,000	50	Rndm.	0.4182	0.5990	2.0120	75.10	1.6599	49.32	3.3198	203.6
1000	10,000	100	Rndm.	1.7189	2.2631	8.0462	138.2	6.3063	84.23	17.469	445.1
5000	50,000	500	Rndm.	78.564	73.333	286.20	574.3	187.95	307.3	646.03	2241

Time (seconds) and Iterations (Iters.) are on the average.

5.1.5 Running Time and Number of Iterations

Finally, we discuss the running time and the number of iterations. Each running time of Algorithms C, D, and E includes the running time of computing an initial solution. Table 5 shows the running time and the number of iterations for various instances.

The running time of Greedy Post-Processing depends on the number of iterations. For increasing demands and decreasing demands, the number of iterations is quite small compared with for uniformly distributed random demands. This may imply that there are a lot of local minimum solutions in an instance for increasing demands and decreasing demands. As the number of edges increases, the number of iterations also increases. As the size of an instance becomes large, the number of iterations in Algorithm D among Algorithms C, D, and E becomes fewest, and its running time becomes best. The actual running time of Algorithm D is almost within three minutes for $n = 5000$ and we think it can be used in practical applications.

5.2 Conclusion of Experiments

Our experiments show that the performances of Chen et al. algorithms are quite better than the theoretically guaranteed performances.

For an instance with $k = \frac{n}{2}$, the performance of all algorithms tested here becomes rather worse. This is partly because the integrality gap becomes worse for instances with $k = \frac{n}{2}$.

For all kinds of instances, Algorithm D works best. For instances with $n = 5000$, $m = 50000$, $k = 500$ and uniformly distributed random demands, the approximation ratio of solution produced by Algorithm B is 1.425, while Algorithm D improves this to 1.065. The time required for Post-Processing is not so large and within practical applications.

The approximation ratio of a solution produced by Algorithm E is quite bad except instances with uniformly distributed random demands. Thus, Greedy Post-Processing does work well for some good initial solutions.

6 Concluding Remarks

We have experimentally tested several algorithms for the congestion minimization confluent flow problem. We have shown that Greedy Post-Processing works quite well for an solution produced by $(1+\ln k)$-approximation algorithm of Chen et al. [1].

We think it is important to make theoretical analysis of the performance and running time of Greedy Post-Processing for a solution produced by $(1+\ln k)$-approximation algorithm of Chen et al.

References

[1] J. Chen, R. D. Kleinberg, L. Lovasz, R. Rajaraman, R. Sundaram and A. Vetta: (Almost) Tight Bounds and Existence Theorems for Confluent Flows. *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, Chicago, Illinois, June 2004, pp. 529–538.

[2] J. Chen, R. Rajaraman and R. Sundaram: Meet and Merge: Approximation Algorithms for Confluent Flows. *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, San Diego, California, June 2003, pp. 373–382.

[3] Y. Dinitz, N. Garg and M. Goemans: On the Single-Source Unsplittable Flow Problem. *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, Palo Alto, California, November 1998, pp. 290–299.

[4] J. Kleinberg: Single-Source Unsplittable Flow. *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, Burlington, Vermont, October 1996, pp. 68–77.

[5] J. Kleinberg, Y. Rabani and E. Tardos: Fairness in Routing and Load Balancing. *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, New York City, New York, October 1999, pp. 568–578.

An Algorithm For Source Location In Directed Graphs

MIHÁLY BÁRÁSZ*

Department of Operations Research
Eötvös University
Pázmány Péter sétány 1/c, Budapest
Hungary, H-1117.
barasz@cs.elte.hu

JOHANNA BECKER*

Department of Operations Research
Eötvös University
Pázmány Péter sétány 1/c, Budapest
Hungary, H-1117.
beckerjc@cs.elte.hu

ANDRÁS FRANK*†

Department of Operations Research
Eötvös University
Pázmány Péter sétány 1/c, Budapest
Hungary, H-1117.
frank@cs.elte.hu

Abstract: Ito, Makino, Arata, Honami, Itatsu, and Fujishige [7] provided a theoretical answer to a source location problem by proving that the minimum cardinality of a subset R of nodes in a directed graph $D = (V,A)$ for which there are k edge-disjoint paths from R to every node $v \in V - R$ and there are l edge-disjoint paths from every node $v \in V - R$ to R is equal to the maximum number of pairwise disjoint deficient sets where a nonempty subset of nodes is deficient if its in-degree is less than k or its out-degree is less than l. They also showed how this theorem gave rise to a polynomial time algorithm to compute the optima in question in case the demands k and l are fixed, and posed as an open problem of developing an algorithm that is polynomial not only in the size of the digraph but in k and l, as well. To describe such an algorithm is the main goal of the present work. The algorithm is strongly polynomial even in the edge-capacitated extension of the problem.

Keywords: source location, edge-connectivity, polynomial algorithm

1 Introduction

Discrete location problems are about finding an optimal placement of some facilities (shops, telecommunication centers, factories) in a network so as to satisfy certain customer demands. Typically it is the distance that matters in defining the constraints and the objective functions. For an annotated bibliography of the topic, see the work of M. Labbe and F.V. Louveoux [8]. Source location is a new type of location problem where the flow-amount or connectivity rather than the distance between facilities and customers is taken into consideration. Source location may serve as a useful optimization framework for designing fault-tolerant telecommunication networks. For example, imagine such a network in which a subset R of nodes is considered a suitable source-set if there are k edge-disjoint paths from R to every node not in R and the objective is to compute a smallest source-set.

There are several versions of source location problems, depending on the type of connectivity used in the constraints. Ito et al. [7] considered and analyzed the source location problem in directed graphs constrained with edge-connectivity or maximum flow-amounts. Their paper is a good overview of other models and results, as well, and it is the starting point of the present work. They proved a min-max theorem for the minimum cardinality of a subset R of nodes of an edge-capacitated digraph $D = (V,A)$ so that, for every node $v \in V - R$, the maximum flow-amount from R to v is at least k and from v to R is at least l. Based on this, they described an algorithm for computing such a minimum set R. The algorithm is polynomial provided that k and l are fixed. (That is, the running time of the algorithm depends polynomially on the size of D but exponentially on k and l.) Throughout we will refer to this problem as the Flow-constrained Directed Source Location (FDSL) problem. To simplify our notation and discussions, we tipically work with the uncapacitated case (when the capacity function is identically one). In this case the maximum flow-amount from s to t is the same as the maximum number of edge-disjoint paths from s to t.

*Research is supported by the Hungarian National Foundation for Scientifi c Research, OTKA T037547 and by European MCRTN Adonet, Contract Grant No. 504438.

†Research is supported by Ericsson Hungary, Laborc u. 1, Budapest, Hungary H-1037.

In the present paper, by developing further the ideas of [7], we describe a strongly polynomial algorithm for solving the FDSL problem. Around the same time as we did, J. van den Heuvel and M. Johnson [6] also developed a polynomial algorithm based on completely different ideas. For a comparision, see Section 5.

1.1 Preliminaries

Let $D = (V,A)$ be a digraph. For elements $s,t \in V$ a subset $X \subset V$ is called a $t\bar{s}$-set if $t \in X \subseteq V - s$. In D the in-degree $\rho(X) = \rho_D(X)$ denotes the number of edges entering X while the out-degree $\delta_D(X) = \delta(X)$ is the number of edges leaving X. If a nonnegative capacity function g is given on the edge set, $\rho_g(X)$ (respectively, $\delta_g(X)$) denotes the sum of capacity values on the edges entering (resp., leaving) X.

Given nonnegative integers k and l, a nonempty proper subset X of nodes is called k-**in-deficient** or simply **in-deficient** if $\rho(X) < k$ (or in the capacitated case $\rho_g(X) < k$) and l-**out-deficient** or simply **out-deficient** if $\delta(X) < l$ (or in the capacitated case $\delta_g(X) < l$). An in- or out-deficient set is called **deficient**. A deficient set Z is called **minimal** if no proper subset of Z is deficient. The hypergraph of minimal deficient sets will throughout be denoted by H_{kl}.

A digraph is called (k,l)-**edge-connected** with respect to a root node r if there are k edge-disjoint directed paths from r to every other node and there are l edge-disjoint directed paths from every node to r. It follows immediately from the directed edge-version of Menger's theorem that there are k edge-disjoint paths from r to *all* other nodes of D (that is, D is $(k,0)$-edge-connected) if and only if the in-degree of all nonempty subsets of $V - r$ is at least k, and an analogous characterization holds for $(0,l)$ edge-connectivity. Therefore (k,l)-edge-connectivity of a digraph is equivalent to requiring that the in-degree and out-degree of all nonempty subsets of $V - r$ is at least k and l, respectively. When $k = l$, this notion is equivalent to the k-edge-connectivity of D, while the case $l = 0$ corresponds to the rooted k-edge-connectivity of D.

Call a subset R of nodes a (k,l)-**source** if the contraction of R into a single node r results in a (k,l)-edge-connected digraph with respect to root node r. Equivalently, there are k edge-disjoint directed paths from R to every node and there are l edge-disjoint directed paths from every node to R. Yet another equivalent formulation is that R covers all deficient sets.

In a hypergraph $H = (V,\mathcal{E})$ a family of pairwise disjoint hyperedges is called a **matching.** The largest cardinality $\nu(H)$ of a matching is the **matching number** of H. A subset Z of nodes intersecting each hyperedge is called a **transversal** of H. The smallest cardinality $\tau(H)$ of a transversal is the **transversal number** of H. A hypergraph H is said to admit the **Helly property** or to be of **Helly-type** if any subset of pairwise intersecting hyperedges has a nonempty intersection. A hypergraph is **laminar** if at least one of the sets $X - Y, Y - X, X \cap Y$ is empty for any two members X, Y. A **subtree hypergraph** (sometimes called an arboreal hypergraph) is one for which there is a tree T on its node set such that each hyperedge induces a subtree of T. The tree is called a **basic** (or representative) **tree** for the hypergraph.

The **line graph** $L(H)$ of a hypergraph H is a graph in which the nodes correspond to the hyperedges, two of them being adjacent if the corresponding hyperedges have a nonempty intersection. It follows from the definitions that $\nu(H)$ is the stability number $\alpha(L(H))$ of $L(H)$ while $\tau(H)$ is at least the clique-covering number of $L(H)$ (which is, by definition, the chromatic number of the complement of $L(H)$) with equality for hypergraphs of Helly type.

An undirected graph is called **chordal** if there is no induced circuit of length at least four, or in other words, every circuit of length at least four admits a chord. Chordal graphs are known to be perfect and F. Gavril [4] constructed an algorithm that computes in a chordal graph a maximum stable set and a minimum clique-covering with the same cardinality.

The following simple theorem was discovered independently by several authors (for references, see [2]).

Theorem 1 *A hypergraph $H = (V,\mathcal{E})$ is a subtree hypergraph if and only if H admits the Helly property and the line graph $L(H)$ of H is chordal.*

It follows that the matchings of a subtree hypergraph H correspond to the stable sets of $L(H)$ and the transversals of H correspond to the clique-coverings of $L(H)$. Therefore $\nu(H) = \tau(H)$ and Gavril's algorithm may be used to compute a maximum matching and a minimum transversal of H. This algorithm is polynomial in $|\mathcal{E}|$.

1.1.1 Flows and cuts

In later sections we need some basic properties of flows and cuts. For two disjoint nonempty sets S and T of V, let $\lambda_g(S,T)$ denote the maximum flow-amount from s to t in the digraph arising from D by contracting S and T into nodes s and t, respectively. By the Max-flow Min-cut (MFMC) theorem, $\lambda_g(S,T)$ is equal to the minimum in-capacity of sets Z with $T \subseteq Z \subseteq V - S$. In the uncapacitated case $\lambda(S,T)$ is the minimum in-degree of sets Z with $T \subseteq Z \subseteq V - S$. By the directed edge-version of Menger's theorem, $\lambda(S,T)$ is equal to the maximum number of edge-disjoint paths from S to T. It is known that the family $\{Z : T \subseteq Z \subseteq V - S,\ \rho(Z) = \lambda(S,T)\}$ of minimizer sets is closed under taking union and intersection. (Indeed, by using the submodularity of ρ_g one has $\lambda_g(S,T) + \lambda_g(S,T) = \rho_g(X) + \rho_g(Y) \geq \rho_g(X \cap Y) + \rho_g(X \cup Y) \geq \lambda_g(S,T) + \lambda_g(S,T)$ from which $\rho_g(X \cap Y) = \lambda_g(S,T) = \rho_g(X \cup Y)$.)

Let Z_{min} and Z_{max} denote the unique minimal and maximal member of this family. With the help of an MFMC computation one can compute not only a maximum flow (or the edge-disjoint paths) from S to T and a minimizer set but the set Z_{min}, as

well. For example, if x is a maximum flow, then Z_{min} is nothing but the set of nodes from which T is reachable in the auxiliary digraph defined by x in the Ford-Fulkerson algorithm. That is, once a maximum flow x is already computed, Z_{min} as well as Z_{max} is computable by a search algorithm in $O(|A|)$ time. Note that a typical MFMC algorithm based on alternating-paths can easily be modified so as to return Z_{max} as a minimizer set. The additional search for Z_{min} or Z_{max} may be needed if another MFMC algorithm is applied. We will use these facts without any further reference.

2 Flow-constrained directed source location (FDSL)

2.1 Known results

As mentioned above, Ito et al. [7] introduced and investigated the FDSL problem. While they pointed out that several closely related problems are NP-complete, they also showed, by proving a fundamental min-max theorem, that FDSL belongs to **NP**∩co-**NP**. We formulate the result for the uncapacitated case.

Theorem 2 (Ito et al. [7]) *The hypergraph H_{kl} of minimal deficient sets form a subtree hypergraph and (hence)*

$$\tau(H_{kl}) = \nu(H_{kl}). \tag{1}$$

The minimum cardinality of a (k,l)-source is equal to the maximum number of pairwise disjoint deficient sets.

What Ito et al. actually proved was that the hypergraph H_{kl} admits the Helly property and its line graph is chordal. This and Theorem 1 implied that H_{kl} is a subtree hypergraph from which (1) followed. The second part follows from the first one by observing that in order to cover all deficient sets, it is sufficient to cover only the minimal ones, that is, the hyperedges of H_{kl}. Note that the hypergraph of *all* deficient sets is not necessarily a subtree hypergraph indicating the value of working with the hypergraph of minimal deficient sets.

The main concern of the FDSL problem is to construct an efficient algorithm to compute a (k,l)-source of smallest cardinality. In this respect, Theorem 2 may be viewed as a stopping rule since it can equivalently be formulated as follows: *A (k,l)-source R is of minimum cardinality if and only if there is a family $\mathscr{M}$ of $|R|$ pairwise disjoint deficient sets.* Such an $\mathscr{M}$ may serve as a certificate for the minimality of a proposed (k,l)-source R.

By applying Gavril's algorithm to the line graph of H_{kl}, Ito et al. [7] obtained an algorithm that computes both a minimum cardinality (k,l)-source and a maximum matching $\mathscr{M}$ of deficient sets. The running time is polynomial in the size of D and of H_{kl}. Unfortunately, the number of minimal deficient sets may be exponential in $|V|$ even if $l = 0$ as shown by the following example of [7]. Define a digraph on n nodes $(n > k)$ with a special node s so that, for every other node v, there is one edge from v to s and there are k parallel edges from s to v. Then the minimal in-deficient sets are precisely those containing s and having $n-k+1$ elements.

Therefore the algorithm for the FDSL problem is polynomial in the size of D but not necessarily in k and l. For fixed k and l, however, the number of minimal deficient sets depends polynomially on the size of D since every deficient set Z is determined by the set of edges entering (or leaving) Z and the number of subsets of edges smaller than $\max\{k,l\}$ is polynomial in $|A|$. That is, the algorithm of [7] outlined above is polynomial for fixed k,l.

2.2 Strategy of a strongly polynomial algorithm

The goal of the present work is to describe an algorithm that is polynomial for the FDSL problem in k and l (and strongly polynomial in the capacitated case). A crucial idea is the introduction of the new concept of solid sets (for a definition, see Section 3). This notion depends only on D and not on k and l and it proves to be a fortunate generalization of that of minimal deficient sets. On one hand, Theorem 2 nicely extends to solid sets (Theorem 5), on the other hand, serious algorithmic difficulties with minimal deficient sets can be overcome by working with solid sets.

Since the line graph of H_{kl} may be exponentially large in the size of D, one must avoid its usage in an algorithm. [3] described an algorithm for computing a minimum transversal and a maximum matching of an arbitrary subtree hypergraph H that works directly on the basic tree T for H, rather than using the line graph of H.

Therefore we must be able to compute a basic tree for H_{kl} as well as to run the algorithm of [3] in a situation where the hypergraph H_{kl} cannot be given explicitly. This difficulty will be overcome by introducing a 'surrogate' subtree hypergraph H'_D that has just a few hyperedges (at most n^2, by a straightforward estimation, and, in fact, at most $2n-2$, by a tricky one [1]). H'_D will have the property that any tree basic for H'_D is automatically basic for H_{kl}. Details will be discussed in Section 3. Once a basic tree for H_{kl} is available, the algorithm of [3] can be mimicked with the help of MFMC computations, see Section 4 for details.

3 Computing a basic tree for H_{kl}

In this section we show how to compute a basic tree for H_{kl} without using an explicit list of the hyperedges of H_{kl}.

3.1 Computing a basic tree for a subtree hypergraph

How can one compute a basic tree for an arbitrary subtree hypergraph $H = (V, \mathscr{E})$? Although the known inductive proof of Theorem 1 may easily be turned into an algorithm that is polynomial in $|\mathscr{E}|$, we outline here another approach, based on the greedy algorithm, for constructing a basic tree. We do so for completeness and because we have not found in the literature this pretty link to the well-known maximum weight spanning tree problem.

Define a weight function $c(uv)$ on the edge-set of the complete graph on V as follows. For every pair $\{u,v\}$ of nodes,

$$\text{let } c(uv) \text{ be the number of hyperedges containing both } u \text{ and } v.$$

Theorem 3 *A hypergraph H admits a basic tree (that is, H is a subtree hypergraph) if and only if a spanning tree of maximum c-weight is a basic tree for H.*

PROOF: To see the non-trivial direction, let Z be a hyperedge and T an arbitrary spanning tree. Then Z induces at most $|Z|-1$ edges of T. Therefore

$$c(T) := \sum_{uv \in E(T)} c(uv) = \sum_{uv \in E(T)} \sum [1 : Z \in \mathscr{E}, \{u,v\} \subseteq Z] \leq \sum_{Z \in \mathscr{E}} (|Z|-1) \tag{2}$$

and equality holds if and only if every hyperedge Z induces precisely $|Z|-1$ edges of T, that is, if T is basic for H. □

It follows that Kruskal's algorithm can be used to compute a basic tree for a subtree hypergraph. Again, this algorithm is polynomial in the number of hyperedges.

3.2 Solid sets and partitions

In order to introduce the small surrogate hypergraph for H_{kl} promised above, first we need to define a hypergraph that is actually larger than H_{kl}. Given a digraph $D = (V,A)$, we call a nonempty subset Z of V **in-solid** (respectively, **out-solid**) if $\rho(X) > \rho(Z)$ (respectively, $\delta(X) > \delta(Z)$) for every nonempty proper subset X of Z. An in- or out-solid set is called **solid**. Singletons are always in- and out-solid, and a minimal k-in-deficient set is in-solid (for any k). A k-in-deficient in-solid set is not necessarily a minimal k-in-deficient set. Let $H_D = (V, \mathscr{E}_D)$ denote the hypergraph of all solid sets. Note that the definition of deficient sets depends on the parameters k and l while that of the solid sets does not. Since an in-solid set Z is a minimal k'-in-deficient set with respect to the parameter $k' := \rho(Z)+1$, the set of in-solid sets is exactly the union of all k-in-deficient sets ($k = 1,2,\dots$). An analogous statement holds for out-solid and solid sets. In other words, H_D may be viewed as the union of hypergraphs H_{kl} for all possible values of k and l.

We remark that an analogous notion for undirected graphs, under the name of extreme sets, was introduced and successfully used to solve the undirected edge-connectivity augmentation problem by Watanabe and Nakamura [10]. But the structure of extreme sets of undirected graphs, as they are laminar, is much simpler than that of the solid sets of digraphs.

As mentioned, [7] proved that minimal deficient sets form a subtree hypergraph. We can use their proof-technique almost word for word to show that the hypergraph H_D of solid sets is also a subtree hypergraph. Let us start with the following useful observation.

Lemma 4 *If X is in-solid and Y is out-solid, then at least one of the subsets $A := X-Y$, $B := Y-X$, $C := X \cap Y$ is empty.*

PROOF: Let $\alpha, \beta, \gamma, \gamma'$ denote, respectively, the number of edges from C to A, from B to C, from $V-(X \cup Y)$ to C, and from C to $V-(X \cup Y)$. If, indirectly, none of A, B, C is empty, then $\rho(A) > \rho(X)$ and $\delta(B) > \delta(Y)$. Therefore $\alpha > \beta + \gamma$ and $\beta > \alpha + \gamma'$ from which the impossible $0 > \gamma + \gamma'$ would follow. □

Theorem 5 *The hypergraph $H_D = (V, \mathscr{E}_D)$ of solid sets is a subtree hypergraph, that is, for every directed graph $D = (V,A)$ there is a spanning tree on the groundset V such that each solid set of D induces a subtree.*

PROOF: We claim that the line graph of H_D is chordal. If, indirectly, it induces a chordless circuit of length at least 4, then there are solid sets $X_1, \dots, X_h$ ($h \geq 4$) so that $X_i \cap X_j \neq \emptyset$ if and only if i and j are consecutive integers where we use the notational convention $X_{h+1} = X_1$. Lemma 4 implies that either all X_i's are in-solid or all X_i's are out-solid. By symmetry, we may assume that the first case occurs. It follows that the h intersections $X_i \cap X_{i+1}$ are pairwise disjoint and hence

$$\sum_{i=1}^{h} \rho(X_i \cap X_{i+1}) \leq \sum_{i=1}^{h} \rho(X_i). \tag{3}$$

Since X_i is in-solid, $\rho(X_i) < \rho(X_i \cap X_{i+1})$ for $i = 1, \dots, h$ and hence $\sum_i \rho(X_i) < \sum_i \rho(X_i \cap X_{i+1})$, contradicting (3).

We claim that H_D admits the Helly property. If it does not, then there is a smallest number $h \geq 3$ along with h solid sets $X_1,\ldots,X_h$ such that any two of these sets intersect each other while the intersection $M = X_1 \cap \cdots \cap X_h$ is empty. Again, by Lemma 4 either the sets $X_1,\ldots,X_h$ are all in-solid or they are all out-solid. By symmetry we may assume that each X_i is in-solid. Let $Y_i = X_1 \cap X_2 \cap \cdots \cap X_{i-1} \cap X_{i+1} \cap \cdots \cap X_h$ ($i = 1,\ldots,h$). By the minimal choice of h, $Y_i \neq \emptyset$, while $M = \emptyset$ implies that $Y_i \cap Y_j = \emptyset$ ($1 \leq i < j \leq h$). If an edge enters one of the sets Y_i, then it enters at least one of the sets X_j. Therefore $\sum_i \rho(Y_i) \leq \sum_i \rho(X_i)$. On the other hand $\rho(Y_i) > \rho(X_{i+1})$ for each i as X_{i+1} is in-solid and $Y_i \subset X_{i+1}$. Hence $\sum_i \rho(Y_i) > \sum_i \rho(X_{i+1}) = \sum_i \rho(X_i)$, a contradiction.

By Theorem 1 H_D is indeed a subtree hypergraph. □

We call a basic tree for H_D a **solid tree** for D. In order to be able to compute a solid tree, we need some further properties of solid sets.

Lemma 6 *If the intersection of two in-solid (out-solid) sets X and Y is nonempty, then $X \cup Y$ is in-solid (out-solid).*

PROOF: If indirectly $X \cup Y$ is not in-solid, then there is a maximal nonempty subset $Z \subset X \cup Y$ with $\rho(Z) \leq \rho(X \cup Y)$.

If Z includes one of X and Y, say X, then $Z \cap Y \subset Y$, $X \cup Y = Z \cup Y$ and hence $\rho(Z \cap Y) > \rho(Y)$, $\rho(Z \cup Y) = \rho(X \cup Y) \geq \rho(Z)$ from which $\rho(Y) + \rho(Z) \geq \rho(Z \cap Y) + \rho(Z \cup Y) > \rho(Y) + \rho(Z)$ would follow. Therefore Z can include neither X nor Y.

If Z is disjoint from X or Y, say from X, that is, $Z \subseteq Y - X$, then $\rho(Z) > \rho(Y)$ which is not possible since $\rho(X) + \rho(Y) \geq \rho(X \cap Y) + \rho(X \cup Y) > \rho(X) + \rho(X \cup Y)$ implies $\rho(Y) > \rho(X \cup Y)$ from which we would have $\rho(Z) > \rho(X \cup Y)$, contradicting the assumption $\rho(Z) \leq \rho(X \cup Y)$. Therefore Z must intersect both X and Y.

It follows that $X \cap Z \neq \emptyset$ and $X \cap Z \subset X$ from which $\rho(X \cap Z) > \rho(X)$ as X is in-solid. Since $Z \subset X \cup Z$, the maximal choice of Z implies $\rho(X \cup Z) \geq \rho(Z)$. Therefore we have $\rho(X) + \rho(Z) \geq \rho(X \cap Z) + \rho(X \cup Z) > \rho(X) + \rho(Z)$, a contradiction. The proof for out-solid sets is analogous. □

By an s-**avoiding in-solid** (**out-solid**) set Z we mean an in-solid (out-solid) subset of $V - s$. The adjective maximal is used if Z is not included in any other s-avoiding in-solid (out-solid) subset of $V - s$. By Lemma 6 the maximal s-avoiding in-solid sets are disjoint. Since each singleton is in-solid, the maximal s-avoiding in-solid sets partition $V - s$. This will be called the **in-solid partition** of $V - s$. The out-solid partition of $V - s$ is defined analogously. It follows from Lemmas 6 and 4 that:

Corollary 7 *The family of maximal s-avoiding solid sets is a partition of $V - s$.*

We call this partition the **solid partition** of $V - s$.

3.3 Computing the solid partition of $V - s$

By Corollary 7 the members of the in-solid partition and the out-solid partition of $V - s$ form a laminar family $\mathscr{L}$. Therefore the solid partition of $V - s$ consists of the maximal members of $\mathscr{L}$. Hence, in order to compute the solid partition of $V - s$, it suffices to compute separately the in-solid and the out-solid partitions of $V - s$. Since the two computations are analogous, we describe only the first one to compute the in-solid partition of $V - s$.

As mentioned in the introduction, the maximum number $\lambda(t) := \lambda(s,t)$ of edge-disjoint paths from s to a node $t \in V - s$ is equal to the minimum in-degree of the t $\overline{s}$-sets, and the minimizer sets are closed under taking union and intersection. Let N_t denote the unique minimal member of this family.

Lemma 8 *If N is a minimal member of the family $\{N_t : t \in V - s\}$, then N is a maximal s-avoiding in-solid set.*

PROOF: We claim that $z \in N_t$ implies $N_z \subseteq N_t$ for any $z,t \in V - s$. Indeed, if we had, indirectly, $N_z - N_t \neq \emptyset$, then $\rho(N_z \cap N_t) > \rho(N_z)$ from which $\rho(N_z) + \rho(N_t) \geq \rho(N_z \cup N_t) + \rho(N_z \cap N_t) > \lambda(t) + \rho(N_z) \geq \rho(N_t) + \rho(N_z)$ would follow.

This and the minimality of N imply that $N = N_t$ for every element $t \in N$ and hence N is in-solid. Furthermore there are $\rho(N)$ edge-disjoint paths from s to t, therefore $\rho(Z) \geq \rho(N)$ whenever $N \subseteq Z \subseteq V - s$, that is, N is maximally in-solid in $V - s$. □

Based on this, the in-solid partition of $V - s$ can be computed as follows. First compute all sets N_t ($t \in V - s$) and choose the smallest of these sets N_t, denoted by N_1. By Lemma 8, N_1 is a maximal s-avoiding in-solid set. Second, contract s and N_1 into a node s_1 and compute in a similar manner a maximal s_1-avoiding in-solid set N_2 in the contracted digraph. Since the maximal s-avoiding in-solid sets in D are disjoint, N_2 is a maximal s-avoiding in-solid set in D. At a general step, contract s and the already computed maximal s-avoiding in-solid sets $N_1,\ldots,N_h$ into a node s_h and compute a maximal s_h-avoiding in-solid set N_{h+1} of the contracted digraph. The algorithm terminates when the union of the current sets $N_1,\ldots,N_h$ is $V - s$.

To describe the algorithm more formally, let $\mathscr{N}$ denote the current family of maximal disjont in-solid subsets of $V - s$. Instead of carrying out the contractions we will maintain a subset S that is the union of the members of $\mathscr{N}$ plus s.

Algorithm for computing the in-solid partition of $V - s$.

INPUT Digraph $D = (V,A)$ and a node $s \in V$.
OUTPUT The in-solid partition $\mathcal{N}$ of $V - s$.

(P1) Set $\mathcal{N} := \emptyset$ and $S := \{s\}$.

(P2) If $V - S$ is empty, output $\mathcal{N}$. STOP. (The algorithm terminates.)

(P3) For each $t \in V - S$, with the help of an MFMC routine, compute $\lambda(S,t)$ and the unique smallest set N_t for which $t \in N_t \subseteq V - S$ and $\rho(N_t) = \lambda(S,t)$. Let N be a smallest member of the family $\{N_t : t \in S - V\}$. Add $\{N\}$ to $\mathcal{N}$. Set $S := S \cup N$. Go to (P2).

3.4 Computing a solid tree for D

Given the solid partition of $V - s$ for every node $s \in V$, let H'_D be the subhypergraph of H_D consisting of those hyperedges which occur in the solid partition of $V - s$ for some $s \in V$. Note that H'_D has at most n^2 hyperedges, that is, H'_D is small even if H_D has exponentially many hyperedges. (In fact, A. Bernáth [1] proved that H'_D has at most $2n-2$ hyperedges.) Therefore one can compute a basic tree T for H'_D as described in Subsection 3.1 and this algorithm is polynomial in the size of D. The nice thing is that T will automatically be a basic tree for H_D and hence for H_{kl}, too.

Theorem 9 *If T is a basic tree for H'_D, then T is basic for the hypergraph H_D of all solid sets (and, in particular, for its subhypergraph H_{kl} of deficient sets).*

PROOF: Suppose indirectly that there is a solid set Z that does not induce a subtree of T. Then there are two elements a, b of Z so that the unique path P in T connecting a and b contains a node s not belonging to Z. That is, Z is an s-avoiding solid set and hence there is a maximal s-avoiding solid set Z' including Z. But T is basic for H'_D and hence the whole P must belong to Z', a contradiction. □

4 Computing a minimum transversal and a maximum matching

Let $H = (V, \mathcal{E})$ be an arbitrary subtree hypergraph and T a basic tree for H. [3] describes an algorithm for computing a minimum transversal R and a maximum matching $\mathcal{M}$ of H that works directly on the basic tree T for H. (Actually, that algorithm settles a weighted case as well but here we need only the unweighted version.) First we exhibit and justify the correctness of the generic form of the algorithm where it does not matter how the input hypergraph is given. A more specific version is then described and shown how it applies to the hypergraph H_{kl} of minimal deficient sets.

We need some notation. Choose an arbitrary node s of T as a root node. Let $\vec{T}$ denote the arborescence arising from T by orienting each edge of T away from s. Define the **height** of a node v to be the distance of v from s in $\vec{T}$. A node v is said to be **above** a node $u \neq v$ if there is a path in $\vec{T}$ from u to v. For a hyperedge Z of H, the **bottom node** $b(Z)$ of Z is the (unique) lowest node of Z. The height of Z is defined to be the height of its bottom node. We say that a hyperedge Z of H is **independent** from a matching $\mathcal{M}$ if Z is disjoint from the members of $\mathcal{M}$, that is, if $\mathcal{M} \cup \{Z\}$ is a matching.

The generic algorithm starts with the empty matching $\mathcal{M}$. In each step, it chooses any of the highest hyperedges that is independent from the current matching $\mathcal{M}$ and adds it to $\mathcal{M}$. The algorithm terminates when no such hyperedge exists anymore. It returns the final $\mathcal{M}$ and the set R of bottom nodes of the members of $\mathcal{M}$. Clearly, $|\mathcal{M}| = |R|$. The correctness of the algorithm as well as a proof of the min-max relation $\nu(H) = \tau(H)$ for subtree hypergraphs follow from the following lemma.

Lemma 10 ([3]) *The set R of bottom nodes output by the algorithm outlined above covers all hyperedges.*

PROOF: Suppose indirectly that there is a hyperedge Y not covered by R. By the termination rule of the algorithm Y must intersect a member of $\mathcal{M}$. Among these members, let Z be the one which was added earliest to $\mathcal{M}$. Then Y is disjoint from each member of $\mathcal{M}$ that has been added to $\mathcal{M}$ prior to Z. Since $b(Z)$ is not in Y but $Z \cap Y \neq \emptyset$, it follows that $b(Y)$ is above $b(Z)$ contradicting the 'choose-the-highest' rule of the algorithm. □

We describe now more specifically how a highest hyperedge independent from $\mathcal{M}$ can be found. Instead of trying to find it directly, the algorithm considers the nodes of H in a decreasing order according to their height, and checks whether or not the current node is the bottom node of a hyperedge Z which is independent from $\mathcal{M}$. If it is, Z is added to $\mathcal{M}$.

Let $A(X)$ denote, for a subset $X \subseteq V$, the set of nodes reachable from X in $\vec{T}$. For a singleton $\{v\}$ we write $A(v)$ and let $B(v) := V - A(v)$. Then $v \in A(v)$ and $V = A(s)$. In addition to the matching $\mathscr{M}$ and the set R of the bottom nodes of the members of $\mathscr{M}$, the algorithm maintains a label assigned to each node. The content of the label is 'marked' or 'unmarked'.

Specific Algorithm for computing a maximum matching and a minimum transversal of a subtree hypergraph H.

INPUT: A subtree hypergraph $H = (V, \mathscr{E})$ along with a basic tree T for H.
OUTPUT: A matching $\mathscr{M}$ and a transversal R of H so that $|\mathscr{M}| = |R|$.

(SA1) Set $\mathscr{M}$ and R to be empty, and set each node unmarked.

(SA2) If there is no unmarked node, output $\mathscr{M}$ and R. STOP. (The algorithm terminates).

(SA3) Choose a highest unmarked node v and mark it. Let $S(v) := B(v) \cup A(R)$.

(SA4) Find a hyperedge Z for which $v \in Z \subseteq V - S(v)$. If no such hyperedge exists, go to Step (SA2).

(SA5) Add Z to $\mathscr{M}$ and add $b(Z)$ to R. Go to Step (SA2).

The correctness of this algorithm follows from that of the generic algorithm since a node v cannot get marked as long as $A(v) - v$ includes a hyperedge disjoint from R. Hence we have:

PROPERTY ($*$) Every hyperedge included in $V - S(v)$ must contain v.

4.1 Realizing Step (SA4) for H_{kl}

Let us return to our initial problem of computing a minimum (k,l)-source set, that is, a minimum transversal of the hypergraph H_{kl} of minimal deficient sets along with a maximum matching. In the preceding section we showed how to compute a basic tree T for H_{kl}. Now we want to apply the algorithm above to H_{kl}, a situation where the list of hyperedges is not explicitly given. The only task is to realize Step (SA4). To this end, let v be the node considered in Steps (SA3) and (SA4).

(SA4.1) Compute $\lambda(S(v),v)$ along with the unique minimal set Z' for which $v \in Z' \subseteq V - S(v)$ and $\rho(Z') = \lambda(S(v),v)$. Compute $\lambda(v,S(v))$ along with the unique minimal set Z'' for which $v \in Z'' \subseteq V - S(v)$ and $\delta(Z'') = \lambda(v,S(v))$.

(SA4.2) If $\lambda(S(v),v) \geq k$ and $\lambda(v,S(v)) \geq l$, then (by Property $(*)$) a hyperedge for (SA4) does not exist. Go to Step (SA2).

(SA4.3) If $\lambda(S(v),v) < k$ and $\lambda(v,S(v)) < l$, then let Z be the smaller of Z' and Z''. If exactly one of $\lambda(S(v),v) < k$ and $\lambda(v,S(v)) < l$ holds, then let Z be, accordingly, Z' or Z''. Turn to Step (SA5) with this Z.

The only property we have to check is that the subset Z constructed this way is a hyperedge of H_{kl}.

Claim 11 *The subset Z is minimal deficient, that is, Z is a hyperedge of H_{kl}.*

PROOF: If $\lambda(S(v),v) < k$ and $\lambda(v,S(v)) < l$, then by Property $(*)$, Z' is minimal in-deficient and Z'' is minimal out-deficient. By Lemma 4 one of them includes the other, hence Z is minimally deficient. If exactly one of $\lambda(S(v),v) < k$ and $\lambda(v,S(v)) < l$ holds, then by Property $(*)$ again, Z is minimal deficient. □

5 Running time and conclusions

In the following estimation of running times we use the notation $n := |V|$, $m := |A|$. The algorithm outlined above for solving the FDSL problem consists of three consecutive phases:

1. Computing the solid partition of $V - s$ for each $s \in V$.

2. Computing a basic tree T for H_D.

3. Computing a minimum (k,l)-source (and a maximum matching) using T.

Note that only the last step depends on k and l, so in order to solve the FDSL problem on the same digraph for several values of k and l, only the third step should be repeated.

Let $F(n,m)$ denote the complexity of an MFMC algorithm on a digraph with n nodes and m edges. As we mentioned in Section 3, one member of the solid partition of $V-s$ may be obtained by running an MFMC algorithm n times. Thus the in-solid partition of $S-v$, and analogously the out-solid partition as well, can be computed in $O(n^2F(n,m))$. Since we need this for all nodes, the total time of Phase 1 is $O(n^3F(n,m))$.

To compute a basic tree for H_D, we first have to determine the weight function c corresponding to H'_D, and find then a maximum weight spanning tree T. So this phase can be bounded by $O(n^3)$.

The third phase of the algorithm applies the MFMC algorithm twice for every node v (to get the maximum flow-amount and the min-cut from v to $S(v)$ and from $S(v)$ to v). Hence this is doable in $O(nF(n,m))$.

We can conclude that the bottleneck of the whole algorithm is Phase 1, therefore we want to improve on this.

5.1 Computing the solid partition via the algorithm of Hao and Orlin

Hao and Orlin [5] invented an $O(nm\log(n^2/m))$ time algorithm to compute the minimum cuts in a digraph between a given node s and all the other nodes $t \in V-s$. With a slight modification of their algorithm (which does not increase its complexity), one can obtain the unique minimal minimizer set N_t. Namely, the Hao-Orlin algorithm maintains a feasible preflow, so when it finds a $t\,\bar{}\,s$-set with $\lambda_g(s,t)$ in-capacity, then one more search algorithm gives rise to N_t. That is, the additional time we need is $O(mn)$ which does not affect the total complexity of the Hao-Orlin algorithm.

Summing up, when the algorithm of Hao and Orlin is used in Phase 1, the total complexity of our algorithm is $O(n^3m\log(n^2/m))$.

Finally, we remark that the algorithm may be applied to the capacitated case without any change. Since the time bound for the Hao-Orlin algorithm concerns capacitated digraphs anyhow, the complexity bound given before remains valid.

J. van den Heuvel and M. Johnson [6] also developed a polynomial algorithm based on completely different ideas. Actually, their algorithm can compute a smallest transversal for *any* subtree hypergraph provided a subroutine is available for deciding whether a subset of nodes is a transversal of H. On the other hand the algorithm does not compute a maximum matching of H_{kl}. The complexity of the algorithm of [6] is $O(n^3S(n))$ where $S(n)$ denotes the complexity of the subroutine, and such a subroutine is indeed available for H_{kl} via a Hao-Orlin computation. Therefore the algorithm of van den Heuvel and Johnson, when specialized to the FDSL problem, is of complexity $O(n^4m\log(n^2/m))$.

5.2 Conclusion

We developed a strongly polynomial time algorithm for the FDSL problem introduced and analyzed in [7]. A useful feature of our approach is that it can be used to solve the following inverse problem: given the digraph D and an integer C, what is the maximum value $k=k(C)$ so that there is a C-element subset of V whose contraction to a node gives rise to a k-edge-connected digraph (or in another version, to a $(k,0)$-edge-connected digraph). In the uncapacitated case this question can be easily answered: simply run the algorithm above for all possible values $1,2,\ldots,M$, where M is the maximum of the in-degrees and the out-degrees of the nodes, and choose the largest k for which the resulting minimum (k,k)-source set has at most C elements. This approach is certainly not strongly polynomial in the capacitated case but an elegant idea of N. Megiddo [9] can be used to show that $k(C)$ can be computed by n applications of our algorithm.

References

[1] A. Bernáth, A note on the directed source location algorithm, Egerváry Research Report, 2004-12.

[2] P. Duchet, Hypergraphs, (Theorem 3.8) in: Handbook of Combinatorics (eds. R. Graham, M. Grötschel, L. Lovász), Elsevier Science B. V. (1995), pp. 381-432.

[3] A. Frank, Some polynomial algorithms for certain graphs and hypergraphs, in: Proceedings of the Fifth British Combinatorial Conference, (1975) Congressus Numerantium XV, Eds. C. Nash-Williams and J. Sheehan, pp. 211-226.

[4] F. Gavril, Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph, SIAM Journal on Computing 1 (1972) 180-187.

[5] J. Hao and J.B. Orlin, A faster algorithm for finding the minimum cut in a graph, J. Algorithms 17 (1994) 424-446.

[6] J. van den Heuvel and M. Johnson, Transversals of subtree hypergraphs and the source location problem in digraphs, CDAM Research Report, LSE-CDAM-2004-10.

[7] Hiro Ito, Kazuhisa Makino, Kouji Arata, Shoji Honami, Yuichiro Itatsu, and Satoru Fujishige, Source location problem with flow requirements in directed networks, Optimization Methods and Software, Vol. 18, No. 4, August 2003, pp. 427-435.

[8] M. Labbe and F.V. Louveaux, Location problems, in: Annotated Bibliographies in Combinatorial Optimization, eds.: M. Dell' Amico, F. Maffioli, and S. Martello, Wiley, 1997, pp. 261-281.

[9] N. Megiddo, Combinatorial optimization with rational objective functions, Mathematics of Operations Research, Vol. 4 (1979) 414-424.

[10] T. Watanabe and A. Nakamura, Edge-connectivity augmentation problems, Computer and System Sciences, **35** No.1, (1987) 96–144.

Stable matching with incremental algorithms The last one gets his best stable partner

PÉTER BIRÓ

Department of Algebra and
Department of Computer Science and Information
Theory
Budapest University of Technology and Economics
H-1521, Budapest, Hungary
pbiro@cs.bme.hu

Abstract: Roth and Vande Vate introduced an algorithm to solve the stable marriage problem by successively adding the vertices to the graph in a random order and successively satisfying blocking edges. Tan and Hsueh solved the more general stable roommates problem with the same basic idea. These algorithms are able to model the mechanisms of a matching market, where players enter and leave the market one after another. Generalizing Cechlárová [2] here we prove that each agent, who makes proposal in the last active phase, gets the best stable partner in both of the above algorithms.

Keywords: stable marriage problem, stable roommates problem, matching mechanism

1 Introduction

Let G be a graph and for every vertex v let $<_v$ be a linear order on the edges incident with v. We say that v prefers edge f to e, in other words f *dominates* e at v if $e <_v f$ holds. A matching M is called *stable* if every edge $e \notin M$ is dominated by some edge $f \in M$. Alternatively, the stable matching can be defined as a matching without *blocking edge*: an edge preferred by both of its vertices to the eventual matching edges.

This problem was introduced and solved for bipartite graphs by Gale and Shapley [3]. In their terminology, the two sets of vertices were that of men and women, and the stable matching is called a *stable marriage*. The solution obtained by the deferred-acceptance algorithm was proven to be optimal for the men if men make proposals. This means that each man gets his *best stable partner*, so none of them can have a better partner in any other stable matching.

The *stable roommates problem*, which is about stable matching in general graphs, is also defined in [3]. It is shown by an example that a stable matching does not always exists. Irving [5] was the first who constructed an algorithm that finds a stable matching if one exists at all. The explanation of the non-existence was given by Tan [12], who described a half-integer solution, the so called *stable partition*. To prove the existence of such a half-matching, an algorithm was given to find one in every instance.

For the bipartite case Knuth [7] posed the question whether it is possible to start from an arbitrary matching and to obtain a stable matching by successively satisfying blocking edges. Roth and Vande Vate [9] introduced a decentralised algorithm to avoid cycling. In this algorithm, pairs or single vertices enter the graph in a random order, and the stability is achieved in $O(m)$ steps in each phase by a natural proposal-rejection process.

Tan and Hsueh [11] constructed an algorithm, that finds a stable half-matching by using a similar incremental method. The difference is that their algorithm works on general graphs and it also handles half-weighted cycles. In the bipartite case, the Tan-Hsueh algorithm is equivalent to the Roth-Vande Vate algorithm. These two algorithms are hereafter abbreviated as *incremental algorithms*. (It is also interesting that the original deferred-acceptance algorithm of Gale and Shapley is a particular incremental algorithm: women enter the game first.)

Ma [6] observed, that there might be stable matchings that the Roth-Vande Vate algorithm never finds if the vertices are added to the graph one after another. Cechlárová [2] found a reason: she proved that one of the vertices must get its best

†Research is partially supported by the Center for Applied Mathematics and Computational Physics, and by the Hungarian National Science Fund (grant OTKA F 037301).

stable partner. Hence, if a stable matching exists where nobody gets his best stable partner, then this stable matching cannot be found. She asked if the last vertex has the above property. In this paper we confirm this for the incremental algorithms.

The interpretation of the results have an economic relevance. The matching markets are well-known applications of the stable matching problem; the detailed description of the two-sided markets can be found in the book of Roth and Sotomayor [8]. The incremental algorithms are able to model the naturally occurring processes when players enter and leave the market one after another and the stability of the market is restored by successively break off and establish new partnerships.

This paper is organized in the following way: In section 2, we describe the Roth-Vande Vate and the Tan-Hsueh algorithm. In section 3, we prove our main results on the properties of a stable half-matching output by an incremental algorithm.

2 Stable matching with incremental algorithms

In a matching market where the eventual matching is stable it is a natural question to ask how will the situation change if a new player enters the game. We suppose that he will make proposals in the order of his preference. If nobody accepts because everybody has a better partner, then the matching remains stable. If somebody accepts it (because he is single or has a worse partner) then they will form a new pair. The previous partner, who is left now, has to continue making proposals. We will show that a stable matching can always be reached in a finite time by this proposal-acceptance process in the two-sided market. In the general case, repetition can occur if a stable matching does not exists. But a stable half-matching can always be found this way.

2.1 The Roth-Vande Vate algorithm

Suppose, that the underlying bipartite graph G is built up step by step in the algorithm by adding vertices to the graph in some order. A *phase* is the period when the stability is restored between the extended set of agents. To describe a phase, let us add a vertex a_0 to G^0, where a stable matching M_0 exists. Our task is to find a stable matching M in $G = G^0 \cup a_0$.

If a_0 is not incident to any blocking edge, then M_0 remains stable for G, too. This trivial case is called the *inactive phase*.

In an *active phase* the new agent a_0 incident to some blocking pair. Let (a_0, b_1) be the best of them according to a_0. If b_1 was uncovered by M_0, then $M_0 \cup (a_0, b_1) = M_1$ is a stable matching for G. In the other case b_1 had a partner a_1 in M_0, whom he leaves after receiving a better proposal. Trivially M_1 is stable in $G^1 = G \setminus a_1$. So we have a similar situation as in the beginning: a_1 enters the market and makes proposals. Continuing the process, an *alternating sequance* is constructed with the following properties:

1. $M_k = M_{k-1} \setminus (a_k, b_k) \cup (a_{k-1}, b_k)$ is a stable matching in $G^k = G \setminus a_k$.
2. a_{k-1} is a better partner for b_k than a_k and
3. b_{k+1} is a worse partner for a_k than b_k.

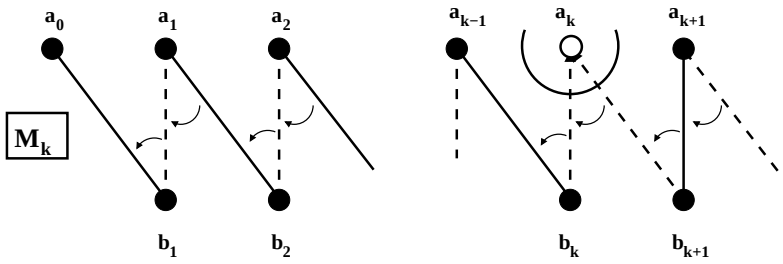


Figure 1: Alternating sequence in the Roth-Vande Vate algorithm

Observe that by this process, each $a_i \in A$ improves his situation and each $b_j \in B$ gets worse off. Consequensly, the same vertices cannot occur as new pairs. So a phase stops in $O(m)$ time and has two possible outcome: either nobody accepts the proposals of some a_i (then the size of M remains the same) or the last b_j was uncovered, hence the size of M increases by one.

Originally, in [9] the agents are allowed to enter the market in pairs if no blocking edge is created. By this method, every stable matching can be obtained in a certain order: when the stable pairs should enter first. But agents tipically enter the market one after another as Ma [6] argued.

2.2 Stable half-matching for the roommates problem

After Aharoni and Fleiner [1] we introduce the definition of stable half-matching. A function x assigning non-negative values to edges of G is called a *fractional matching* if $\sum_{v\in e} x(e) \leq 1$ for every vertex v of G. A fractional matching x is called *stable* if every edge e contains a vertex v such that $\sum_{v\in f, e\leq f} x(f) = 1$. It means that every edge must be *fractionally dominated* at one of its endvertices. The stability can be also defined as the absence of blocking edges, when no pair would like to improve their partnership simultaneously.

The existence of the stable fractional matching is the consequence of the Scarf-lemma [10]. Let us consider the fractional edges $\{e : 0 < x(e) < 1\}$. Obviously, no two of them can be dominated at the same vertex. If a fractional edge e is dominated at v, then another fractional edge f must also contain v. Hence no vertex is incident to exactly one fractional edge. Consequently, fractional edges form vertex disjoint *cycles*. If a cycle is odd, then every weight along the cycle must be $\frac{1}{2}$. If a cycle is even, then the weights can be alternately x and $1-x$, for an arbitrary $x : 0 \leq x \leq 1$. Here, the $x = \{0,1\}$ means that the even cycles can be separated into stable pairs, but if any odd cycle occurs then its agents cause instability in the roommates problem.

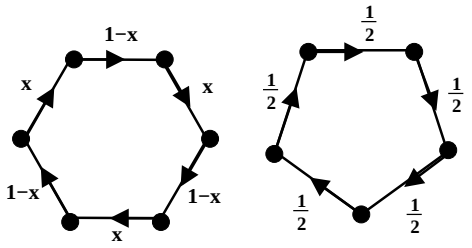


Figure 2: Odd and even cycles in a stable fractional matching

Definition 1 (Tan, 1991) *A stable half-matching is a stable fractional matching, such that $x(e) = \{0, \frac{1}{2}, 1\}$ for all the edges. We can partition the set of vertices:*

a) uncovered vertices

b) cycle-vertices

c) matched vertices

Tan [12] proved the existence of a stable half-matching by a modified version of Irving's algorithm [5]. He also proved, that in any instance the same vertices are uncovered, and the same odd cycles are formed in each stable half-matching. The remaining pair of vertices are each other's *partners*. Later Tan and Hsueh [11] constructed an incremental algorithm to find a stable half-matching.

2.3 The Tan-Hsueh algorithm for the stable roommates problem

In this more general setting we use the terminology of the Roth-Vande Vate algorithm. The only difference is that G is not bipartite, so instead of a matching, we maintain a half-matching M_0 in $G^0 = G \setminus a_0$.

If nobody accepts the new comer's proposal, then the phase is called *inactive* again and the stable half-matching remains the same.

If some b_1 accepts the proposal of a_0 then three cases are possible:

a) If b_1 is uncovered in M_0, then $M = M_0 \cup (a_0, b_1)$ is a stable half-matching in G.

b) If b_1 is a cycle-vertex in M_0, so $b_1 = c_1$ for some cycle $C =< c_1, c_2, \ldots, c_{2k}, c_{2k+1} >$, then $M = M_0 \setminus C \cup (a_0, b_1) \cup (c_2, c_3) \cup, \ldots, \cup (c_{2k}, c_{2k+1})$ is a stable half-matching in G.

c) If b_1 is matched in M_0, then $M_1 = M_0 \setminus (a_1, b_1) \cup (a_0, b_1)$ is a stable half-matching in $G^1 = G \setminus a_1$.

The actual phase end in cases a) and b). The difference between the Roth-Vande Vate and the Tan-Hsueh algorithm is that in the bipartite case repetition cannot occur, because if a man enters then men get worse partners, and women get better ones during the phase. In case of a general graph repetition can in dead happen: an agent, that made a proposal earlier can receive a proposal later, and the alternating sequence might never end. The main result of Tan and Hsueh was that repetition always occurs along an odd cycle.

Theorem 2 (Tan-Hsueh, 1995) *If $a_0, b_1, a_1, \dots$ is an alternating sequence and $a_i = b_k$ is the first return, then this alternating sequence can be extended so it will return to a_k at b_{k+m+1}, and the following properties are true:*
$\{a_k, b_{k+1}, \dots, b_{k+m}, a_{k+m}\}$ are distinct vertices, and in the same order they form an odd cycle C, so $M = M_k \setminus (a_{k+1}, b_{k+1}) \setminus \cdots \setminus (a_{k+m}, b_{k+m}) \cup C$ is a stable half-matching in G.

In the following example illustrates the mechanisms of the Tan-Hsueh algorithm:

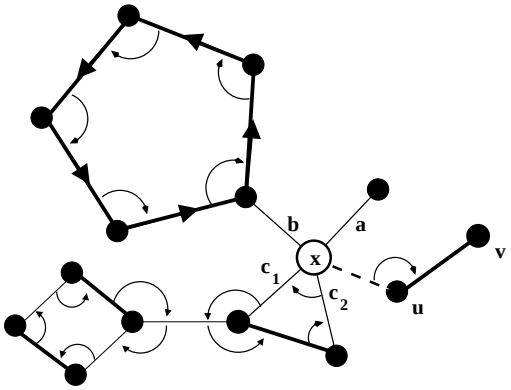


Figure 3: The Tan-Hsueh algorithm in an example

Here, the vertex v enters the graph. The first vertex accepting its proposal is u, and its previous partner x is left alone. In this figure there is a stable half-matching M_x in $G \setminus x$. In the next step x makes proposals. If nobody accepts it x remains uncovered and M_x is stable for G, too. If somebody accepts x's proposal one of the following cases is true:

a) an uncovered vertex accepts it so together with x they form a new pair.

b) a cycle-vertex accepts it so with x they form a new pair, and also the rest of the cycle form stable pairs.

c) a matched vertex accepts x's proposal. The process continues and finally x receives a proposal, so the sequence returns in x. In this case the phase would never end, but by collecting there vertices into an odd cycle, the following stable half-matching is reached:

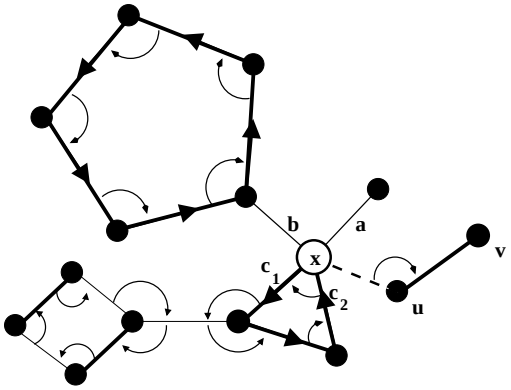


Figure 4: The obtained stable half-matching

3 Properties of the incremental solutions

Ma [6] gave an example showing, that if the agents enter the market successively in the Roth-Vande Vate algorithm not all of the stable matchings can be obtained. Cechlárová [2] extended this result by showing a reason of the non-existence of some stable matchings. She proved the following theorem:

Theorem 3 (Cechlárová, 2002) *If M is a stable matching in a bipartite graph G output by the incremental algorithm, then somebody gets his best stable partner.*

For our extension of Cechlárová's result we observe two lemmas.

Lemma 4 *If a phase is inactive in an incremental algorithm, then the extended graph cannot contain any new stable half-matchings. That is, if M is a stable half-matching in G and a vertex x is not covered by M, then M is a stable half-matching in $G \setminus x$, too.*

PROOF: Since x cannot be covered in a stable half-matching of G then if a matching was blocked in $G \setminus x$, it is blocked in G, too. □

Lemma 5 *For any vertex a_0, if M_0 is a stable half-matching in $G \setminus a_0$, and (a_0, b_1) is not a blocking edge in G for M_0, then a_0 and b_1 cannot be partners in any stable half-matching in G.*

PROOF: Let us suppose that (a_0, b_1) is not a blocking edge in G for M_0 but there is a stable half-matching M in G, where a_0 and b_1 are partners. In other words, b_1 must refuse a_0, because he had a better partner (say a_1) in M_0. So $(a_0, b_1) <_{b_1} (a_1, b_1)$, where $(a_0, b_1) \in M \setminus M_0$. Since (a_1, b_1) cannot be dominated at b_1 in M, it must be dominated at a_1 by an edge $(a_1, b_2) \in M$, which cannot be in M_0, so it must be dominated at b_2 by an edge $(a_2, b_2) \in M_0$, and so on.

If there is no odd cycle in M_0 and M then the alternating sequence $(a_0, b_1, a_1, b_2, \dots)$ has the following property: $(a_{i-1}, b_i) \in M \setminus M_0$ and $(b_i, a_i) \in M_0 \setminus M$, furthermore the domination is also in sequence: $(a_{i-1}, b_i) <_{b_i} (a_i, b_i)$ and $(a_i, b_i) <_{a_i} (a_i, b_{i+1})$ for every i. We call this sequence a *dominated alternating sequence*. Because a_0 is not in the stable half-matching M_0, the sequence cannot return at a_0 and trivially cannot return to any other vertices in the sequence either.

The other case is, when M_0 or M contains odd cycle. The properties of the dominated alternating sequence remain the same, just the edges can be half-edges as well. To avoid the repetition the idea is the following: when an edge $(a_i, b_i) \in M_0$ is dominated at a_i in M by two edges (so a_i is in a cycle in M), then we chose for b_{i+1} that neighbour in the cycle which is less preferred by a_i. The edge (a_i, b_{i+1}) is still not in M_0, so it must be dominated at b_{i+1}. But then the edge(s) that dominate (a_i, b_{i+1}) is (are) better than either of the edges that cover b_{i+1} in M, so they are not in M. This is why every new edge in this sequence will be alternately in $M \setminus M_0$ and $M_0 \setminus M$.

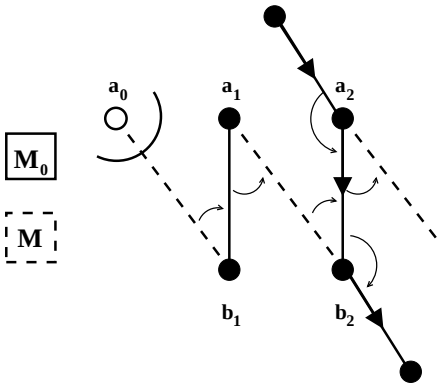


Figure 5: Dominated alternating sequence with half-weighted edges

Repetition cannot happen. Let us suppose that $a_k = a_i$ for some $k \neq i$. This means that (b_k, a_i) and (b_i, a_i) are in the same odd cycle in M_0 but the directions are opposite, because for b_i and also for b_k a_i is the less preferred neighbour in the cycle. In the other case, assume that $a_k = b_i$ for some $k \neq i$. This means that (b_k, b_i) and (b_i, a_i) are in the same odd cycle in M_0 but the directions are opposite again. As a_i is less prefered for b_i in the cycle, and $(b_k, b_i) \in M_0 \setminus M$ it must be dominated at b_i in M, this means $(b_k, b_i) <_{b_i} (a_{i-1}, b_i) <_{b_i} (a_i, b_i)$ a contradiction. □

These two lemmas prove the following theorem. Since a vertex, that receives a partner in the last active phase by a proposal has the same partner at the end of the algorithm. The vertices preferred by him must have refused his proposal so they cannot be his stable partner because of Lemma 5. And this partner remains his best possible, because in an inactive phase a new stable matching cannot be produced.

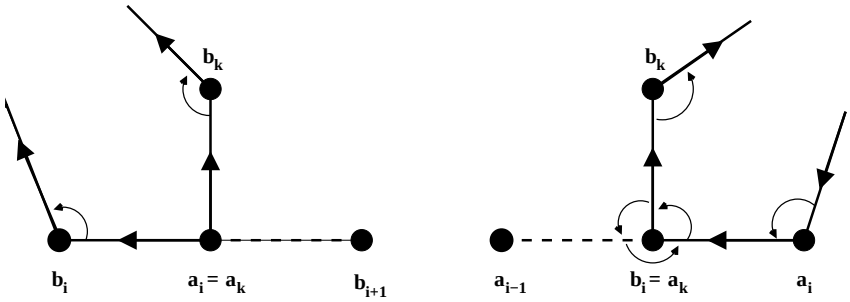


Figure 6: Return is not possible

Theorem 6 *Each matched agent, that gets his partner in the last active phase by a proposal, gets his best stable partner in the stable solution of the incremental algorithm.*

Remark 7 *The vertices that remained uncovered in the last active phase or entered later (in an inactive phase), will still be uncovered at the end of the algorithm, just like they are in every stable matching. The vertices that form an odd cycle in the last active phase will form an odd cycle at the end of the algorithm, just like they do in every stable half-matching. Hence these agents also get best possible partners.*

Remark 8 *In the proofs of the two lemmas we only use that M_0 is a stable half-matching in each phase. So the theorem is also true if we add a new vertex to an instance with an arbitrary stable half-matching.*

Corollary 9 *If an agent is matched in the stable half-matchings and enters the market last, then he gets his best stable partner.*

PROOF: Firstly, he receives a partner by a proposal, so later he cannot accept any proposal because then he would be involved in the cycling. □

Corollary 10 *Everyone who gets a partner by accepting a proposal in the last active phase, gets his worst stable partner in an incremental algorithm.*

PROOF: If v is the best stable partner for u in a stable matching M, then u is the worst stable partner for v. If indirectly, v has a worse partner u' in a stable matching M' than u, then u would also have a worse partner v', because v was his best stable partner, so (u,v) would be a blocking edge for M'. □

Corollary 11 *A stable matching, where no matched agent gets his best stable partner, cannot be obtained by an incremental algorithm.*

Example 12 *Gale and Shapley [3] gave an instance that contain a stable matching, such that nobody gets his best stable partner, but everybody gets the "middle one partner" according his preference. This very stable matching cannot be obtained by the incremental algorithm.*

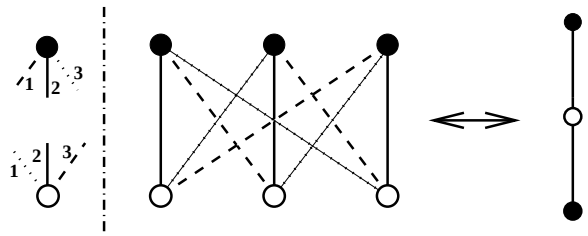


Figure 7: The preferences, the graph and the lattice of the stable matchings

Let us remark that it is still open if for a stable matching where somebody gets his best stable partner, or for a stable half-matching which contains odd cycle whether it can be obtained with an incremental algorithm or not.

Acknowledgement

I would like to thank Katarína Cechlárová who suggested me to consider this problem, Tamás Fleiner, my PhD consultant, for his professional guidance in my research, and Virág Szörényi for helpful comments and corrections.

References

[1] RON AHARONI, TAMÁS FLEINER, On a lemma of Scarf, *J. of Combinatorial Theory B* (2003) **87**

[2] KATARÍNA CECHLÁROVÁ, Randomized matching mechanism revisited. *Preprint* (2002)

[3] D. GALE, L.S. SHAPLEY, Collage admissions and stability of marriage. *Amer. Math. Monthly* (2003) **69**

[4] D. GUSFIELD, R.W. IRVING, The stable marriage problem: structure and algorithms. *The MIT press* (1990)

[5] R.W. IRVING, An efficient algorithm for the "stable roommates" problem. *J. Algorithms* (1985) **6**

[6] JIMPENG MA, On randomised matching mechanism. *Economic Theory* (1996) **8**

[7] DONALD E. KNUTH, Mariages stables. *Les Presses de l'Universite de Montreal* (1976)

[8] A. ROTH, M. SOTOMAYOR, Two-sided matching: A study in game-theoretic modeling and analysis. *Cambridge University Press* (1990)

[9] A. ROTH, J.H. VANDE VATE, Random paths to stability in two-sided matching. *Econometrica* (1990) **58**

[10] H.E. SCARF, The core of an *N* person game. *Econometrica* (1967) **35**

[11] JIMMY J. M. TAN, YUANG-CHEH HSUEH, A generalisation of the stable matching problem, *Applied Mathematics* (1995) **59**

[12] JIMMY J. M. TAN, A necessary and sufficient condition for the existence of a complete stable matching, *J. Algorithms* (1991) **12**

Cryptomorphisms of Closure Systems Axiomatization and Structures

FLORENT DOMENACH

Computer Science Department
Intercollege
46 Makedonitissas Av., 1700 Nicosia, Cyprus
domenach.f@intercollege.ac.cy

Abstract: In this paper, we will focus on some of the many different cryptomorphisms equivalent with closure systems (and so with closure operators), particularly on implications and on overhanging relations (introduced in [3]). Our aim is double. On one hand, to present a characterization of different type of closure systems (distributive, nested, ...) through their associated overhanging relation. On the other hand, we'll introduce the lattice structures of this cryptomorphisms, with some derived structures for particular sets of overhanging relations.

Keywords: Closure systems, Implication, Lattice, Moore family, Overhanging

1 Introduction

Closure systems, together with the many cryptomorphisms associated, occur in quantity of domain: algebra, topologic, logic, relational databases [1], symbolic data analysis [5], mathematics of social sciences [6], etc. In several of these fields, one is interested in closure systems satisfying additional axioms, like topologies or convex geometries. After recalling one particular cryptomorphism introduced in [3], the overhanging relations, we'll present a characterization of different type of closure systems through these relations, and take a brief look at the structure of closure systems.

2 Closure System, Implications and Overhangings

Let's S be a finite set with at least two elements. A *closure system* (or *Moore family*) $\mathscr{F}$ on S is a family of subsets of S satisfying

(CS1) $S \in \mathscr{F}$;

(CS2) for all $A, B \in \mathscr{F}$, $A \cap B \in \mathscr{F}$.

To this set $\mathscr{F}$ we can associate two (between many other) different binary relations $\mathscr{I}$ and $\mathscr{O}$ on the set $\mathscr{P}(\mathscr{S})$ of the subsets of S:

- If a set B is systematicaly associated with a set A in $\mathscr{F}$, i.e. if every set of $\mathscr{F}$ containing A also contain B, then we say that A *imply* B, and denote it by $A \to B$, or $A \mathscr{I} B$. The binary realtion $\mathscr{I}$ on S associated to $\mathscr{F}$ is called an *implicational system*.
- If, for two sets A and B of S, $A \subset B$, there's a set in $\mathscr{F}$ containing A but not B, i.e. the set B is more generalthan a set S relatively to $\mathscr{F}$, then we say that A is *overhanged* in B (or B *overhangs* A). We denote it by $(A,B) \in \mathscr{O}$, or $A \mathscr{O} B$, and $\mathscr{O}$ is the *overhanging relation* on S associated with $\mathscr{F}$.

Implicational systems and overhanging relation can be directly define on S by two different sets of axioms. The binary relation $\mathscr{I}$ satisfies:

(I1) for all $A, B \in S$, $B \subseteq A$ implies $A \mathscr{I} B$;

(I2) for all $A, B, C \in S$, $A \mathscr{I} B$ and $B \mathscr{I} C$ imply $A \mathscr{I} C$;

(I3) for all $A, B, C, D \in S$, $A \mathscr{I} B$ and $C \mathscr{I} D$ imply $(A \cup C) \mathscr{I} (B \cup D)$.

And the overhanging relation $\mathscr{O}$ satisfies:

(O1) for all $A,B \in S$, $A \mathscr{O} B$ implies $A \subset B$;

(O2) for all $A,B,C \in S$, $A \subset B \subset C$ imply $[A \mathscr{O} C \iff A \mathscr{O} B$ or $B \mathscr{O} C]$;

(O3) for all $A,B \in S$, $A \mathscr{O} (A \cup B)$ implies $(A \cap B) \mathscr{O} B$.

The set of all closure systems, closure operators, implicational systems and overhanging relations are in a one-to-one correspondence to each other. For the correspondences between closure systems, closure operators and implications, we will refer to [2]. We give hereunder two correspondences pointed out in [3]: for all $A,B \subseteq S$

$$AOB \iff [A \subset B \text{ and } (A,B) \notin I]$$

$$AOB \iff [A \subset B \text{ and } \varphi(A) \subset \varphi(B)]$$

3 Axiomatization of Particular Types of Closure Systems

Here we will present some particular types (between many other) of closure systems appearing in the litterature, together with their axiomatization in terms of of overhanging relations.

Proposition 1 *Let $\mathscr{F}$ be a closure system on S. $\mathscr{F}$ is* **nested** *if $F,F' \in \mathscr{F}$ imply $F \cap F' \in \{F,F'\}$. It's associated overhanging relation $\mathscr{O}$ satisfies (O1), (O2) and, for all $A,B,C \subseteq S$,*

$$A \mathscr{O} C \text{ and } B \mathscr{O} C \text{ imply } A \cup B \mathscr{O} C$$

Proposition 2 *Let $\mathscr{F}$ be a closure system on S. $\mathscr{F}$ is a* **tree of subsets** *if $F,F' \in \mathscr{F}$ imply $F \cap F' \in \{\emptyset,F,F'\}$. It's associated overhanging relation $\mathscr{O}$ satisfies (O1), (O2) and, for all $A,B,C \subseteq S$,*

$$A \mathscr{O} C \text{ and } B \mathscr{O} C \text{ imply } A \cup B \mathscr{O} C \text{ or } A \cap B = \emptyset$$

Proposition 3 *Let $\mathscr{F}$ be a closure system on S. $\mathscr{F}$ is* **hierarchical** *if it's a tree of subsets and if $\{s\} \in \mathscr{F}$ for any $s \in S$. It's associated overhanging relation $\mathscr{O}$ satisfies (O1), (O2) and, for all $s \in S, A,B,C \subseteq S$,*

$$A \notin \{\emptyset,\{s\}\} \text{ implies} \{s\} \mathscr{O} A \cup \{s\}$$

$$A \mathscr{O} C \text{ and } B \mathscr{O} C \text{ imply } A \cup B \mathscr{O} C \text{ or } A \cap B = \emptyset$$

Proposition 4 *Let $\mathscr{F}$ be a closure system on S. $\mathscr{F}$ is* **distributive** *if $F,F' \in \mathscr{F}$ imply $F \cup F' \in \mathscr{F}$. It's associated overhanging relation $\mathscr{O}$ satisfies (O1), (O2) and, for all $s \in S, A \subseteq S$,*

$$[\{a\} \mathscr{O} \{a,s\} \text{ for any } a \in A] \iff [A \mathscr{O} A \cup \{s\}]$$

Proposition 5 *Let $\mathscr{F}$ be a closure system on S. $\mathscr{F}$ is a* **convex geometry** *if $\emptyset \in \mathscr{F}$ and, for any $F \in \mathscr{F}$, there's a unique minimal (for inclusion) generator of F. It's associated overhanging relation $\mathscr{O}$ satisfies (O1), (O2), (O3) and, for all $s \in S, A,B,C \subseteq S$, $A \subseteq C, B \subseteq C$,*

$$\emptyset \mathscr{O} \{s\}$$

$$A \cap B \mathscr{O} C \text{ implies } A \mathscr{O} C \text{ or } B \mathscr{O} C$$

4 Structures

Let denote by $\mathscr{M}$ the set of all closure systems on S, and $\mathscr{O}$ the set of all overhanging relations on S. From the one-to-one correspondence between closure systems and overhanging relations recalled on Section 2, we can state that (see [4] for a proof):

Proposition 6 *The sets $\mathscr{M}$ and $\mathscr{O}$ are order isomorphics.*

Corollary 7 *The lattice $\mathscr{O}$ of all overhanging realtions is atomistic, lower bounded, lower locally distributive, join-semidistributive, join-pseudocomplemented, lower semimodular and ranked.*

Note that, even if $\mathscr{O}$ is isomorphic to $\mathscr{M}$, $\mathscr{O}$ is not a closure system and, so, not a convex geometry.

References

[1] ARMSTRONG W.W., Dependency structures of data base relationship, *Information Processing* (1974), 580-583.

[2] CASPARD N., MONJARDET B., The lattices of Moore families and closure operators on a finite set: a survey, *Discrete Applied Math.* (2003) **127**, 241-269.

[3] DOMENACH F., LECLERC B., Closure systems, implicational systems, Overhanging relations and the case of Hierarchical Classification, *Mathematical Social Science* (2004) **47 (3)**, 349-366.

[4] DOMENACH F., LECLERC B., The structure of the overhanging relations associated with some types of closure systems, *Discrete Applied Math.* (2005), to appear.

[5] GANTER B., WILLE R., Formal Concept Analysis, *Springer* (1999).

[6] MONJARDET B., The presence of lattice theory in discrete problems of mathematical social sciences, why ?, *Mathematical Social Science* (2003) **46 (2)**, 103-144.

Planar k-sets under insertion

WAEL EL ORAIBY, DOMINIQUE SCHMITT

Laboratoire de Mathématiques,
Informatique et Applications
Faculté des Sciences et Techniques
Université de Haute-Alsace
4, rue des Frères Lumière
68093 Mulhouse Cédex, France
Wael.el-Oraiby@uha.fr
Dominique.Schmitt@uha.fr

Abstract: The k-sets of a set V of points of the plane are the subsets of k points of V that can be separated from the remaining by a straight line. Using the so-called k-set polytope of V, which is a polygon whose vertices are the centroids of the k-sets of V, we characterize the k-sets that disappear as well as those that appear when a new point is added to V. We also use these results to give an incremental algorithm to construct the k-sets of V.

Keywords: linear separabitlity, computational geometry

1 Introduction

Given a finite set V of n points in the Euclidean plane and an integer k ($0 < k < n$), the *k-sets* of V are the subsets of k points of V that can be strictly separated from the remaining by a straight line. k-sets have been used in various areas and under different terminology. For example, they were investigated by Onn and Sturmfels [6] in the field of commutative algebra under the name of corner cuts. Enumeration and construction of k-sets have been studied in combinatorial and computational geometry. Dey [4] has shown that the number of k-sets of n points in the plane is at most $O(nk^{\frac{1}{3}})$ and Toth [10] has constructed sets of points with $n2^{\Omega(\sqrt{\log k})}$ k-sets. The most efficient algorithm to build all the k-sets of a given point set V has been given by Cole, Sharir, and Yap and runs in $O(n\log n + m\log^2 k)$ time, where m is the number of k-sets of V [3].

A classical method in data analysis to measure the distance from a point to a set of k points consists in using the centroid (also called center of gravity) of these k points. Andrzejak and Fukuda [1] used the centroids to reformulate the k-set problem. They showed that finding the k-sets of V comes to find the vertices of the convex hull of the centroids of all the subsets of k elements of V. This convex hull is called the *k-set polytope* of V and has been introduced by Edelsbrunner, Valtr, and Welzl [5].

In this paper we study how the set of k-sets of V is modified when a point v that does not belong to the convex hull of V is added to V. First, we show that the edges of the k-set polytope of V that are no longer edges of the k-set polytope of $V \cup \{v\}$ form an unique polygonal line on the boundary of the k-set polytope of V. In the same way, we characterize the polygonal line of edges that appear when v is added. We give a decomposition of this line in elementary subsets with nice properties. Finally, we show how these properties can be applied to the incremental construction of the k-set polytope of a planar point set. We compare this method to the one given by Cole, Sharir, and Yap [3].

2 k-set polytopes

Let V be a finite set of n points in the Euclidean plane such that $n \geq 2$ and no 3 points of V are colinear. Let k be an integer of $\{1,...,n-1\}$. We denote the k-set polytope of V by $g^k(V)$ and throughout this paper we will consider its boundary to be oriented in counter clockwise direction. Given two points s and t of V, we denote by st the closed oriented segment with endpoints s and t, by (st) the oriented straight line generated by st, and by $(st)^+$ (resp. $(st)^-$) the open half plane on the right (resp. left) of (st). Given any subset ω of the plane, let $conv(\omega)$ be the convex hull of ω, $\overline{\omega}$ be the smallest closed subset of the plane containing ω and $\delta(\omega)$ be the boundary of ω.

Let us now recall two important properties of the vertices and edges of $g^k(V)$ given by Andrzejak and Welzl [2] (see figure 1 for an illustration).

Proposition 1 *T is a k-set of V if and only if its centroid $g(T)$ is a vertex of $g^k(V)$. Moreover, these centroids are distinct points.*

Proposition 2 *T and T' are two k-sets of V such that $g(T)g(T')$ is an oriented edge of $g^k(V)$, if and only if there exist two points s and t of V and a subset P of $k-1$ points of V such that $T = P \cup \{s\}$, $T' = P \cup \{t\}$, and $V \cap (st)^+ = P$.*

Notation 3 *The closed oriented edge $g(P \cup \{s\})g(P \cup \{t\})$ of $g^k(V)$ is denoted by $e_P(s,t)$.*

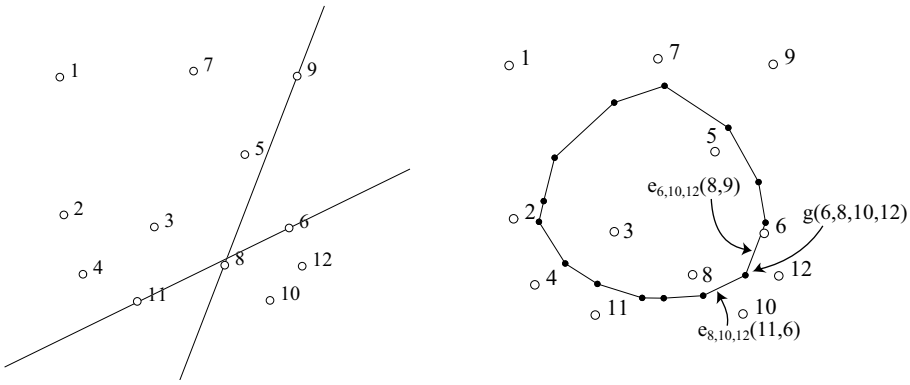


Figure 1: A set of 12 points and its 4-set polytope.

Corollary 4 *If $e_{P_i}(s_i,t_i)$ is an edge of $g^k(V)$ and $e_{P_{i+1}}(s_{i+1},t_{i+1})$ is its successor on $\delta(g^k(V))$, then $P_i \cup \{t_i\} = P_{i+1} \cup \{s_{i+1}\}$, $s_it_i \cap s_{i+1}t_{i+1} \neq \emptyset$, and $s_{i+1} \in \overline{(s_it_i)^+}$.*

PROOF: From proposition 1, since $g(P_i \cup \{t_i\}) = g(P_{i+1} \cup \{s_{i+1}\})$, $P_i \cup \{t_i\} = P_{i+1} \cup \{s_{i+1}\}$ and, from proposition 2, $P_i \cup \{t_i\} \subset \overline{(s_it_i)^+}$. Thus $s_{i+1} \in P_i \cup \{t_i\} \subset \overline{(s_it_i)^+}$. Moreover, since $t_{i+1} \notin P_{i+1} \cup \{s_{i+1}\} = P_i \cup \{t_i\}$, $t_{i+1} \in \overline{(s_it_i)^-}$. Thus $(s_it_i) \cap s_{i+1}t_{i+1} \neq \emptyset$. In the same way $s_it_i \cap (s_{i+1}t_{i+1}) \neq \emptyset$ and thus $s_it_i \cap s_{i+1}t_{i+1} \neq \emptyset$. □

Remark 5 *In the particular case where $k = 1$, $g^k(V)$ is the convex hull of V and its edges are of the form $e_\emptyset(s,t)$. When V is reduced to two points s and t, $g^1(V)$ admits exactly two oriented edges $e_\emptyset(s,t)$ and $e_\emptyset(t,s)$.*

3 k-set polytopes under insertion

Let now S be a subset of V and $v \in V \setminus S$ such that $k+1 \leq |S| \leq n-2$ and v does not belong to $conv(S)$. Let us compare $g^k(S)$ and $g^k(S \cup \{v\})$ (see figure 2).

Lemma 6 *For any straight line Δ that strictly separates v from S and that is not parallel to any straight line passing through any two points of S, there is an unique vertex of $g^k(S)$ closest to (resp. farthest from) Δ and at least one of the edges of $g^k(S)$ incident to this verex is not (resp. is) an edge of $g^k(S \cup \{v\})$.*

PROOF: (i) Let us orientate Δ such that $v \in \Delta^+$. Let Δ_1 be a straight line parallel to Δ, oriented in the same direction as Δ, and such that $|\Delta_1^+ \cap S| = k$. Let $T_{min} = \Delta_1^+ \cap S$ and let Δ_2 be the straight line parallel to Δ_1, passing through $g(T_{min})$, and oriented in the same direction as Δ_1. For every subset $T \neq T_{min}$ of k points of S, at least one point of T belongs to Δ_1^- and $g(T)$ belongs to Δ_2^-. Thus $g(T_{min})$ is the point of $g^k(S)$ closest to Δ and is unique. In the same way, the point $g(T_{max})$ of $g^k(S)$ farthest from Δ is unique.

(ii) Let $e_{P_i}(s_i,t_i)$ and $e_{P_{i+1}}(s_{i+1},t_{i+1})$ be the two edges of $g^k(S)$ with endpoint $g(T_{min})$ such that $P_i \cup \{t_i\} = P_{i+1} \cup \{s_{i+1}\} = T_{min}$. Since $\{s_{i+1},t_i\} \subseteq T_{min} \subset \Delta_1^+$, since $\{s_i,t_{i+1}\} \subseteq S \setminus T_{min} \subset \Delta_1^-$, and since $s_it_i \cap s_{i+1}t_{i+1} \neq \emptyset$ from corollary 4, the straight line Δ_3 parallel to Δ_1, oriented in the same direction as Δ_1, and that passes through $s_it_i \cap s_{i+1}t_{i+1}$, is such that $\overline{\Delta_3^+} \subset \overline{(s_it_i)^+} \cup \overline{(s_{i+1}t_{i+1})^+}$. Since $v \in \Delta^+ \subset \Delta_3^+$, it follows that $v \in (s_it_i)^+$ or $v \in (s_{i+1}t_{i+1})^+$ and thus, from proposition 2, that $e_{P_i}(s_i,t_i)$ or $e_{P_{i+1}}(s_{i+1},t_{i+1})$ is not an edge of $g^k(S \cup \{v\})$.

A symmetric proof shows that at least one of the edges incident to $g(T_{max})$ is also an edge of $g^k(S \cup \{v\})$. □

Lemma 7 *$g^k(S \cup \{v\})$ admits one and only one edge $e_P(s,t)$ such that $s = v$ (resp. $t = v$).*

PROOF: Since $v \notin conv(S)$ and no 3 points of $S \cup \{v\}$ are colinear, there is one and only one point t of S such that $|(vt)^+ \cap S| = k-1$ and thus, from proposition 2, there is one and only one edge $e_P(s,t)$ of $g^k(S \cup \{v\})$ such that $s = v$. In the same way there is one and only one edge of the form $e_P(s,v)$. □

Theorem 8 *(i) $e_P(s,t)$ is an edge of $g^k(S\cup\{v\})$ and not an edge of $g^k(S)$ if and only if $v\in P\cup\{s,t\}$.*

(ii) These edges form an open connected polygonal line whose first (resp. last) edge in counter clockwise direction is the edge $e_P(s,t)$ of $g^k(S\cup\{v\})$ such that $t=v$ (resp. $s=v$).

PROOF: (i) Straightforward from proposition 2.

(ii) From lemma 7, the set $\mathscr{C}$ of edges of $g^k(S\cup\{v\})$ that are not edges of $g^k(S)$ admits at least two edges and, from lemma 6, at least one edge of $g^k(S)$ does not belong to $\mathscr{C}$. Thus, $\mathscr{C}$ admits at least one edge $e_{P_i}(s_i,t_i)$ such that its successor $e_{P_{i+1}}(s_{i+1},t_{i+1})$ on $\delta(g^k(S\cup\{v\}))$ does not belong to $\mathscr{C}$. Hence, from (i), v belongs to $P_i\cup\{s_i,t_i\}$ but does not belong to $P_{i+1}\cup\{s_{i+1}\}=P_i\cup\{t_i\}$. Thus $s_i=v$ and, from lemma 7, $e_{P_i}(s_i,t_i)$ is the only edge of $g^k(S\cup\{v\})$ of the form $e_P(v,t)$. In the same way, there is an unique edge of $\mathscr{C}$ whose predecessor on $\delta(g^k(S\cup\{v\}))$ does not belong to $\mathscr{C}$ and this edge is of the form $e_P(s,v)$. It follows that $\mathscr{C}$ is an open connected polygonal line whose first (resp. last) edge in counter clockwise direction is of the form $e_P(s,v)$ (resp. $e_P(v,t)$). □

Theorem 9 *(i) $e_P(s,t)$ is an edge of $g^k(S)$ and not an edge of $g^k(S\cup\{v\})$ if and only if $v\in(st)^+$.*

(ii) These edges form an open connected and non empty polygonal line.

PROOF: (i) Straightforward from proposition 2.

(ii) From theorem 8, the edges of $g^k(S\cup\{v\})$ that are not edges of $g^k(S)$ form an open polygonal line. Thus, since $g^k(S\cup\{v\})$ and $g^k(S)$ are polygons, the edges of $g^k(S)$ that are not edges of $g^k(S\cup\{v\})$ form also an open connected polygonal line. □

Notation 10 *(i) Let $\mathscr{C}_{S,v}$ (resp. $\mathscr{D}_{S,v}$) denote the oriented polygonal line of edges of $g^k(S\cup\{v\})$ (resp. $g^k(S)$) that are not edges of $g^k(S)$ (resp. $g^k(S\cup\{v\})$).*

(ii) Let $T_1,T_2,...,T_m$ denote the k-sets of S such that $(g(T_1),g(T_2),...,g(T_m))$ is the sequence of vertices of $\mathscr{D}_{S,v}$.

(iii) For every $i\in\{1,...,m\}$, let $e_{P_i}(s_i,t_i)$ denote the oriented edge of $g^k(S)$ whose second endpoint is $g(T_i)$ and let $e_{P_{m+1}}(s_{m+1},t_{m+1})$ denote the oriented edge whose first endpoint is $g(T_m)$.

(iv) For every $i\in\{2,...,m\}$, set $\alpha_i=t_i$ and $\omega_{i-1}=s_i$ and set $\alpha_1=\omega_m=v$.

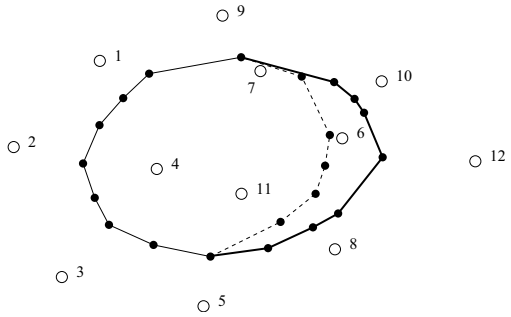


Figure 2: The 4-set polytope of $S=\{1,...,11\}$ and the 4-set polytope of $S\cup\{v\}$, with $v=12$. $\mathscr{C}_{S,v}$ is in bold lines and $\mathscr{D}_{S,v}$ in dashed lines.

Lemma 11 *For every $i\in\{1,...,m\}$, α_i and ω_i are vertices of $conv(T_i\cup\{v\})$.*

PROOF: Since v can be separated from S by a straight line, $\alpha_1=\omega_m=v$ is a vertex of $conv(T_1\cup\{v\})$ and of $conv(T_m\cup\{v\})$. From proposition 2, for every $i\in\{2,...,m\}$, $T_i\setminus t_i=P_i\subset(s_it_i)^+$ and, from theorem 9, $v\in(s_it_i)^+$. Thus, $\alpha_i=t_i$ is a vertex of $conv(T_i\cup\{v\})$, for all $i\in\{2,...,m\}$. In the same way, $\omega_i=s_{i+1}$ is a vertex of $conv(T_i\cup\{v\})$, for all $i\in\{1,...,m-1\}$. □

Notation 12 *(i) For every $i\in\{1,...,m\}$, if $\alpha_i\neq\omega_i$ we denote by $\varphi(T_i)$ the oriented polygonal line that connects α_i to ω_i in counter clockwise direction on $\delta(conv(T_i\cup\{v\}))$ and, if $\alpha_i=\omega_i$, we set $\varphi(T_i)=\alpha_i$.*

(ii) For every $i\in\{1,...,m\}$, let $\mathscr{H}_i$ denote the homothety of center $g(T_i\cup\{v\})$ and ratio $-\frac{1}{k}$.

Lemma 13 *For every $i\in\{1,...,m\}$ such that $\varphi(T_i)$ is not reduced to a single point and for every oriented edge qr of $\varphi(T_i)$, $\mathscr{H}_i(qr)$ is the edge $e_{T_i\cup\{v\}\setminus\{q,r\}}(r,q)$ of $\mathscr{C}_{S,v}$.*

PROOF: Let us give the proof for the case $i \in \{2,...,m-1\}$ (see figure 3). In this case, $\varphi(T_i)$ is the polygonal line that connects t_i and s_{i+1} in counter clockwise direction on $\delta(conv(T_i \cup \{v\}))$ and both $e_{P_i}(s_i,t_i)$ and $e_{P_{i+1}}(s_{i+1},t_{i+1})$ belong to $\mathscr{D}_{S,v}$. Thus $v \in (s_it_i)^+ \cap (s_{i+1}t_{i+1})^+$ and, since $v \notin conv(S)$, $v \in (s_{i+1}t_i)^+$. Since $\varphi(T_i) \subset \overline{\delta(conv(T_i \cup \{v\}))}$, every edge qr of $\varphi(T_i)$ is such that $T_i \cup \{v\} \setminus \{q,r\} \subset (rq)^+$. Moreover, qr is visible from every point of $\overline{(s_it_i)^-} \cap \overline{(s_{i+1}t_{i+1})^-}$ and, from corollary 4, $S \setminus T_i = S \setminus (P_i \cup \{t_i\}) = S \setminus (P_{i+1} \cup \{s_{i+1}\}) \subset \overline{(s_it_i)^-} \cap \overline{(s_{i+1}t_{i+1})^-}$. Thus $S \setminus T_i \subset (rq)^-$. Since $|T_i \cup \{v\} \setminus \{q,r\}| = k-1$, it follows that $e_{T_i \cup \{v\} \setminus \{q,r\}}(r,q)$ is an edge of $g^k(S \cup \{v\})$, from proposition 2, and belongs to $\mathscr{C}_{S,v}$, from theorem 8. Moreover, from barycentric properties, $e_{T_i \cup \{v\} \setminus \{q,r\}}(r,q) = \mathscr{H}_i(qr)$.

The cases $i = 1$ and $i = m$ are treated in the same way. □

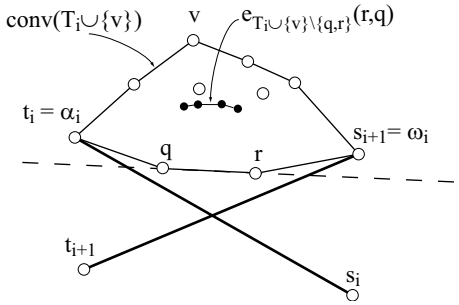


Figure 3: Illustration for the proof of lemma 13.

Theorem 14 *$\mathscr{C}_{S,v}$ is the sequence of polygonal lines $(\mathscr{H}_1(\varphi(T_1)), \mathscr{H}_2(\varphi(T_2)), ..., \mathscr{H}_m(\varphi(T_m)))$.*

PROOF: (i) From lemma 13, for every $i \in \{1,...,m\}$, if $\varphi(T_i)$ admits at least one edge, $\mathscr{H}_i(\varphi(T_i))$ is a connected polygonal line included in $\mathscr{C}_{S,v}$. Moreover, for every $j \in \{1,...,m\}$ such that $j \neq i$, we have $T_i \neq T_j$ and, for every edge q_ir_i of $\varphi(T_i)$ and for every edge q_jr_j of $\varphi(T_j)$, $\mathscr{H}_i(q_ir_i) = e_{T_i \cup \{v\} \setminus \{q_i,r_i\}}(r_i,q_i)$ and $\mathscr{H}_j(q_jr_j) = e_{T_j \cup \{v\} \setminus \{q_j,r_j\}}(r_j,q_j)$ are distinct edges of $\mathscr{C}_{S,v}$. Thus, $\mathscr{H}_i(\varphi(T_i))$ and $\mathscr{H}_j(\varphi(T_j))$ share no edge.

(ii) Since $\alpha_1 = v$ and $\omega_1 = s_2$ are distinct, $\varphi(T_1)$ admits at least one edge: Let vr be its first edge. ¿From lemma 13 and theorem 8, $\mathscr{H}_1(vr) = e_{T_1 \setminus \{r\}}(r,v)$ is then the first edge of $\mathscr{C}_{S,v}$ and $\mathscr{H}_1(\varphi(T_1))$ is an initial subsequence of $\mathscr{C}_{S,v}$. In the same way, $\mathscr{H}_m(\varphi(T_m))$ is a final subsequence of $\mathscr{C}_{S,v}$. Moreover, for all $i \in \{1,...,m-1\}$, $\omega_i = s_{i+1}$ and $\mathscr{H}_i(\omega_i) = g(T_i \cup \{v\} \setminus \{s_{i+1}\})$. Since, from proposition 2, $T_i \setminus \{s_{i+1}\} = T_{i+1} \setminus \{t_{i+1}\}$, it follows that $\mathscr{H}_i(\omega_i) = g(T_{i+1} \setminus \{t_{i+1}\} \cup \{v\}) = \mathscr{H}_{i+1}(\alpha_{i+1})$. Finally, $\mathscr{C}_{S,v} = (\mathscr{H}_1(\varphi(T_1)), ..., \mathscr{H}_m(\varphi(T_m)))$. □

4 An incremental algorithm

The results of the previous section can be used to develop an algorithm that updates the k-set polytope of a set S of points when a new point is added. The boundary of the k-set polytope of S can be stored in a circular list L whose elements represent the edges of $g^k(S)$. To any element e of L that represents an edge $e_P(s,t)$ of $g^k(S)$ are associated the two elements of L that represent the predecessor and the successor of $e_P(s,t)$ on $\delta(g^k(S))$ as well as the two points s and t of S. Let us notice that it suffices to know one k-set T of S and one edge with endpoint $g(T)$ to be able to generate the whole k-sets of S while traversing L.

Let us now recall results given by Overmars [7] and by Overmars and van Leeuwen [8] :

Lemma 15 *There exists a data structure CH of size $O(k)$ to store the convex hull of k points of the plane that allows to:*

- *get the predeccessor and the successor of any edge in constant time,*
- *update the convex hull in $O(\log^2 k)$ time after inserting or deleting a point.*

For any polygonal line $\mathscr{P}$ let $|\mathscr{P}|$ be the number of vertices of $\mathscr{P}$.

Lemma 16 *The elements of L that represent the edges of $\mathscr{D}_{S,v}$ can be reported in $O(|\mathscr{D}_{S,v}|)$ time, provided that one of these edges is known.*

PROOF: From theorem 9, $\mathscr{D}_{S,v}$ is the polygonal line formed by the edges $e_P(s,t)$ of $g^k(S)$ such that v is on the right of the oriented straight line (st). Since testing on which side of (st) v lies simply comes to test the sign of a determinant and since the neighbours of any edge in L can be obtained in constant time, the edges of $\mathscr{D}_{S,v}$ can be reported in $O(|\mathscr{D}_{S,v}|)$ time, provided that one of them is known. □

Lemma 17 *If the convex hull of one k-set T ($T \in \{T_1,...,T_m\}$) is stored in CH and if an edge with endpoint $g(T)$ is given, the edges of $\mathscr{C}_{S,v}$ can be inserted in L in $O(|\mathscr{D}_{S,v}|\log^2 k + |\mathscr{C}_{S,v}|)$ time.*

PROOF: From theorem 14, determining $\mathscr{C}_{S,v}$ comes, for all $i \in \{1,...,m\}$, to determine $\mathscr{H}_i(\varphi(T_i))$, where $\varphi(T_i)$ is a connected subset of $conv(T_i \cup \{v\})$. From lemma 15, $conv(T \cup \{v\})$ can be computed from $conv(T)$ in $O(\log^2 k)$ time. In the same way, for every $i \in \{1,...,m-1\}$ (resp. for every $i \in \{2,...,m\}$), $conv(T_{i+1} \cup \{v\})$ (resp. $conv(T_{i-1} \cup \{v\})$) can be computed from $conv(T_i \cup \{v\})$ in $O(\log^2 k)$ time since, from proposition 2, $T_{i+1} = T_i \setminus \{s_{i+1}\} \cup \{t_{i+1}\}$ (resp. $T_{i-1} = T_i \setminus \{t_i\} \cup \{s_i\}$). From lemma 16, for all $i \in \{1,...,m\}$, the couples $\{s_i,t_i\}$ can be reported in $O(|\mathscr{D}_{S,v}|)$ time and thus, the convex hulls $conv(T_i \cup \{v\})$ can be successively computed in total $O(|\mathscr{D}_{S,v}|\log^2 k)$ time. For every such convex hull, the edges of $\varphi(T_i)$ (i.e. the ones between the vertices α_i and ω_i) can be reported in $O(|\varphi(T_i)|)$ time, from lemma 15. Moreover, from lemma 13, the image by $\mathscr{H}_i$ of every edge of $\varphi(T_i)$ can be inserted in L in constant time per edge and, from theorem 14, can be connected to its neighbours also in constant time. It follows that L can be updated after the insertion of v in total $O(|\mathscr{D}_{S,v}|\log^2 k + \sum_{i=1}^{m} |\varphi(T_i)|)$, that is, $O(|\mathscr{D}_{S,v}|\log^2 k + |\mathscr{C}_{S,v}|)$ time. □

Let now V be a set of n points in the plane, $(v_1,v_2,...,v_n)$ be the lexicographically ordered sequence of these points (i.e. ordered by increasing x and then y coordinate values), and for every $i \in \{1,...,n\}$, let $S_i = \{v_1,...,v_i\}$.

Theorem 18 *The k-set polytope of V can be incrementaly constructed in $O(n\log n + c\log^2 k)$ time, where c is the total number of edges created by the algorithm.*

PROOF: (i) The sequence $(v_1,v_2,...,v_n)$ can be obtained from V in $O(n\log n)$ time. Let us first show that the k-set polytope of $S_{k+1} = \{v_1,...,v_{k+1}\}$ can be computed in $O(k)$ time. Indeed, from proposition 2, st is an edge of $conv(S_{k+1})$ if and only if $e_{S_{k+1}\setminus\{s,t\}}(t,s)$ is an edge of $g^k(S_{k+1})$ and every edge of $g^k(S_{k+1})$ is of this form. Moreover, the convex hull of $O(k)$ lexicographically ordered points can be constructed by an incremental algorithm in $O(k)$ time [9].

(ii) For every $i \in \{k+2,...,n\}$, v_i can be separated from $S_{i-1} = \{v_1,...,v_{i-1}\}$ by a straight line Δ whose slope tends toward $-\infty$ (notice that Δ is not vertical if two points of V have equal x-coordinates). Moreover, the point of $g^k(S_{i-1})$ closest to Δ is the lexicographically maximal vertex of $g^k(S_{i-1})$ that is, the centroid of $\{v_{i-k},...,v_{i-1}\}$. But, from lemma 15, $conv(v_{i-k},...,v_{i-1})$ can be obtained from $conv(v_{i-k-1},...,v_{i-2})$ in $O(\log^2 k)$ time and thus, from lemma 17, $g^k(S_{i-1} \cup \{v_i\})$ can be computed from $g^k(S_{i-1})$ in $O(|\mathscr{D}_{S_{i-1},v_i}|\log^2 k + |\mathscr{C}_{S_{i-1},v_i}|)$ time. Hence, from (i), the incremental algorithm constructs $g^k(V)$ in $O(n\log n + k + \sum_{i=k+2}^{n}(|\mathscr{D}_{S_{i-1},v_i}|\log^2 k + |\mathscr{C}_{S_{i-1},v_i}|))$ time. If c is the total number of edges created by the algorithm, we have $\sum_{i=k+2}^{n} |\mathscr{D}_{S_{i-1},v_i}| \leq c$ and the time complexity of the algorithm is $O(n\log n + c\log^2 k)$. □

Theorem 19 *The total number of edges created by the incremental algorithm to compute the k-set polytope of V is $O(k(n-k))$.*

PROOF: (i) If $k = 1$, the algorithm works as the classical planar convex hull construction algorithm [9]. The two oriented edges of the convex hull of $\{v_1,v_2\}$ are created and then, for every $i \in \{3,...,n\}$, exactly two edges with endpoint v_i are created. Thus, if $k = 1$, the incremental algorithm constructs $2n-2$ edges.

If $k \in \{2,...,n-1\}$, as pointed out in (i) of the proof of theorem 18, the k-set polytope of $\{v_1,...,v_{k+1}\}$ is constructed in the same way as the convex hull of $\{v_1,...,v_{k+1}\}$ and this construction generates $O(k)$ edges.

(ii) Let us now enumerate, for any $k \in \{2,...,n-2\}$ and for all $i \in \{k+2,...,n\}$, the number c_i^k (resp. d_i^k) of edges that are created (resp. deleted) by the incremental construction of the k-set polytope of V when v_i is inserted. From theorem 8, when v_i is inserted, at least two distinct edges are created: The first and the last edge of $\mathscr{C}_{S_{i-1},v_i}$. All other created edges are the edges $e_P(s,t)$ of $g^k(S_{i-1} \cup \{v_i\})$ such that $v_i \in P$. But, from proposition 2, $e_P(s,t)$ is such an edge if and only if $e_{P\setminus\{v_i\}}(s,t)$ is an edge of $g^{k-1}(S_{i-1})$ and is not an edge of $g^{k-1}(S_{i-1} \cup \{v_i\})$. Thus we have $c_i^k = 2 + d_i^{k-1}$. Denoting respectively by $c_V^k = \sum_{i=k+2}^{n} c_i^k$ and by $d_V^k = \sum_{i=k+2}^{n} d_i^k$ the number of edges created and deleted when the points $\{v_{k+2},...,v_n\}$ are successively inserted, we have $c_V^k = \sum_{i=k+2}^{n}(2+d_i^{k-1}) = 2(n-k-1) + \sum_{i=k+2}^{n} d_i^{k-1} \leq 2(n-k-1) + \sum_{i=k+1}^{n} d_i^{k-1} \leq 2(n-k-1) + d_V^{k-1}$. But the number d_V^{k-1} of edges deleted when the points $\{v_{k+1},...,v_n\}$ are inserted while constructing the $(k-1)$-set polytope of V cannot be greater than the number c_V^{k-1} of edges created when these points are inserted increased by the number of edges of the $(k-1)$-set polytope of $\{v_1,...,v_k\}$. Thus, from (i), $d_V^{k-1} \leq c_V^{k-1} + k$. It follows that $c_V^k \leq 2(n-k-1) + c_V^{k-1} + k$ and, after resolution, $c_V^k \leq 2n(k-1) - \frac{(k+6)(k-1)}{2} + c_V^1$. Since, from (i), the number c_V^1 of edges generated by the incremental construction of the convex hull of V is in $O(n)$ and the number of edges created to construct the k-set polytope of $\{v_1,...,v_{k+1}\}$ is in $O(k)$, the whole algorithm constructs $O(k(n-k))$ edges. □

Remark 20 *The time complexity of the incremental algorithm per created edge is $O(\log^2 k)$ and is the same as the one in the algorithm given by Cole, Sharir, and Yap [3]. However, this last algorithm only constructs usefull edges, that is, edges of*

the final k-set polytope of V. This is not the case with our incremental algorithm since it constructs $O(k(n-k))$ edges (and it can be shown that this bound can be achieved) whereas the size of the k-set polytope of n points is known to be $O(nk^{\frac{1}{3}})$ [4]. It follows that an incremental algorithm cannot be optimal as long as k is not considered as a constant. Otherwise the algorithm is optimal, in particular for the case $k = 1$, that is when we are concerned with constructing the convex hull of V.

5 Conclusion

In this paper we have studied, through the notion of *k*-set polytope, how the *k*-sets of a given planar point set *S* are modified when a new point *v* that does not belong to the convex hull of *S* is added to *S*. In particular, we have shown that the edges of the *k*-set polytope that disappear after adding *v* form a connex set. Moreover, we have shown that the polygonal line of the edges that appear can be decomposed into subsequences that are characterizable through the notion of convex hull. Latter, this property allowed us to give an incremental algorithm to generate the *k*-set polytope, or equivalently, the *k*-sets of a planar point set.

In this paper we assumed that no 3 points of $S \cup \{v\}$ are colinear. However, with an appropriate characterization of the edges of the *k*-set polytope, this restriction can be removed and our results as well as our algorithm can be extended. It is the same with the generalisation in higher dimensions. Surprisingly, it is more difficult to deal with the cases where *v* belongs to the convex hull of *S*. Indeed, the edges to be deleted, and hence those to be created, do no more form a connex set, even in the plane. The treatement of this general case would give an incremental algorithm to insert the points in any order.

References

[1] A. Andrzejak and K. Fukuda. Optimization over *k*-set polytopes and efficient *k*-set enumeration. In *Proc. 6th Workshop Algorithms Data Struct.*, volume 1663 of *Lecture Notes Comput. Sci.*, pages 1–12. Springer-Verlag, 1999.

[2] A. Andrzejak and E. Welzl. In between *k*-sets, *j*-facets, and *i*-faces: (i, j)-partitions. *Discrete Comput. Geom.*, 29:105–131, 2003.

[3] R. Cole, Micha Sharir, and C. K. Yap. On *k*-hulls and related problems. *SIAM J. Comput.*, 16:61–77, 1987.

[4] T. K. Dey. Improved bounds on planar *k*-sets and related problems. *Discrete Comput. Geom.*, 19:373–382, 1998.

[5] H. Edelsbrunner, P. Valtr, and Emo Welzl. Cutting dense point sets in half. *Discrete Comput. Geom.*, 17:243–255, 1997.

[6] S. Onn and B. Sturmfels. Cutting corners. *Advances in Applied Mathematics*, 23.

[7] M. H. Overmars. *The Design of Dynamic Data Structures*, volume 156 of *Lecture Notes Comput. Sci.* Springer-Verlag, Heidelberg, West Germany, 1983.

[8] M. H. Overmars and J. van Leeuwen. Maintenance of configurations in the plane. *J. Comput. Syst. Sci.*, 23:166–204, 1981.

[9] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, New York, NY, 1985.

[10] Géza Toth. Point sets with many *k*-sets. *Discrete Comput. Geom.*, 26(2):187–194, 2001.

The distribution of the angles on the plane

ZOLTÁN FÜREDI

Univ. of Illinois, Urbana-Champaign, and Rényi
Institute, Budapest

Abstract:

Conway, Croft and Erdős (1979) investigated the following problem: Let n points in the plane given, no three on a line. They form exactly $N = \frac{1}{2}n(n-1)(n-2)$ angles. Let $0 < \alpha < \pi$ and denote by $f(n,\alpha)$ the greatest integer such that there are at least $f(n,\alpha)$ angles exceeding α for every choice of n such points. They showed that $F(\alpha) = \lim_{n\to\infty} f(n,\alpha)/N$ exists and decreasing. Obviously, $F(\alpha) = 1/3$ for all $0 < \alpha \le \pi/3$, because at least one third of the angles exceeds $\pi/3$. They conjecture, e.g., that the maximum number of acute angled triangles is $(1+o(1))\frac{5}{9}\binom{n}{3}$. In this talk we determine further values of F and consider other related problems. We also point out connections to Turan type extremal hypergraph problems.

Source location with rigidity and tree packing requirements

ZSOLT FEKETE*

Department of Operations Research
Eötvös University
Pázmány Péter sétány 1/C, Budapest, Hungary,
H-1117.
fezso@cs.elte.hu

Abstract: We consider the following two problems. *(i)* Given a graph, add a minimum size clique to the graph such that the resulting graph is rigid in the plane. *(ii)* Given a graph, contract a minimum size vertex-set such that the resulting graph has two edge-disjoint spanning trees. We prove that these problems are polynomially solvable.

Keywords: rigidity of frameworks, pinning down, spanning trees, source location

1 Introduction

We consider undirected graphs which may contain multiple edges but not loops. Let $G=(V,E)$ be a graph, $F\subseteq E$ and $X\subseteq V$. Let $F(X)$ denote the set of edges in F with both end-vertices in X and $i_F(X):=|F(X)|$. $e(X)$ is the number of edges with at least one endpoint in X, that is, $e(X)=|E|-i_E(V-X)$. Let $\mathscr{C}(G)$ denote the set of the components of G. If $(A,B;E)$ is a bipartite graph and $X\subseteq A$, then $\Gamma(X)$ denotes the set of the neighbors of X.

We will define two matroids $\mathscr{M}_{2,2}$ and $\mathscr{M}_{2,3}$, so let $l=2$ or $l=3$ in the rest of the paper. Suppose we are given a graph $G=(V,E)$. We define the following set system:

$$\mathscr{I}_{2,l}:=\{F\subseteq E : i_F(X)\leq 2|X|-l \text{ for every } X\subseteq V, |X|\geq 2\}.$$

The following claim is well-known. We remark that the condition "$i_F(X)\leq k|X|-l$" defines a matroid for many other values of k and l (see the appendix of [14]).

Claim 1 *$\mathscr{I}_{2,l}$ forms the independent sets of a matroid on the underlying set E. In this matroid the rank of an $F\subseteq E$ is the following:*

$$\min_{\mathscr{X}} \sum_{X\in\mathscr{X}} (2|X|-l)+|F-F(\mathscr{X})|$$

Where the minimum is taken over set systems $\mathscr{X}=\{X_1,\dots,X_t\}$ where $X_i\subseteq V$, $|X_i|\geq 2$ and $F(\mathscr{X}):=\cup_{X\in\mathscr{X}}F(X)$.

This matroid will be denoted by $\mathscr{M}_{2,l}$ and let r be its rank function. We remark that if $l=2$, then $\mathscr{X}$ can be chosen to be a sub-partition of V (this is a consequence of matroid union theorem, [13, Theorem 51.1 and Corollary 51.1c]), and if $l=3$, then $\mathscr{X}$ can be chosen such that $F=F(\mathscr{X})$ [9].

If $Z\subseteq V$, then K_Z will denote (the edge-set of) a graph on Z which has $4-l$ parallel edges between every two vertices of Z. We investigate the following problem.

> We are given a graph $G=(V,E)$ and the task is to find a minimum cardinality Z such that $r(E+K_Z)=2|V|-l$.

If $l=2$, then $r(E)=2|V|-2$ if and only if there exist 2 edge-disjoint spanning trees in G. (By Nash-Williams' theorem [11], see also [13, Corollary 51.1c.]) So in this case the problem is the following.

> We are given a graph G. Find a minimum cardinality Z so that $G+K_Z$ has 2 edge-disjoint spanning trees, or equivalently contract a minimum size Z so that G/Z has 2 edge-disjoint spanning trees.

This problem is a variation of the so called source location problem, where G/Z has to satisfy connectivity requirements. (See [1] and [6].)

If $l=3$, then $r(E)=2|V|-3$ if and only if G is generically rigid in the plane (Laman's theorem [7], for basic concepts of rigidity see e.g. [5]). So in this case the problem is the following.

*The author is a member of the Egerv´ary Research Group on Combinatorial Optimization (EGRES). Supported by Communication Networks Laboratory, P´azm´any P´eter s´et´any 1/A, Budapest, Hungary, H-1117. Research is supported by Hungarian National Foundation for Scientifi c Research, OTKA grants T037547, TS049788, by the Egerv´ary Research Group of the Hungarian Academy of Sciences and by European MCRTN Adonet, Contract Grant No. 504438.

Add a minimum size clique to a graph so as to make it generically rigid in the plane.

It can be shown that this problem is equivalent to finding a minimum size vertex set so that pinning down that vertex-set makes the graph generically infinitesimally rigid. The non-generic version of this problem was solved in [8]. (About pinning see also [12, Section 8.2 and 18.2].)

The main result is that these problems are polynomial time solvable (Theorem 3). In fact it is proved that a slight generalization of these problems are solvable, namely we can prescribe a vertex-set T that has to be contained in Z.

2 Main result

The main content of Lemma 2 is that we give an equivalent formulation of our problem in the case when E is independent in $\mathscr{M}_{2,l}$. Using Lemma 2 we are able to reduce our problem to a matching problem.

Lemma 2 *Let E be independent in $\mathscr{M}_{2,l}$ and $|E| < 2|V| - l$, and let $Z \subseteq V$.*

(i) If $|V - Z| \geq 2$, then

$$r(E + K_{V-Z}) = \min_{X \subseteq Z} 2|V - X| - l + e(X)$$

(ii) $r(E + K_{V-Z}) = 2|V| - l$ if and only if $e(X) \geq 2|X|$ holds for every $X \subseteq Z$.

PROOF: *(i)* Let $E' := E + K_{V-Z}$. For a set system $\mathscr{X} = \{X_1, \ldots, X_t\}$, $X_i \subseteq V$, $|X_i| \geq 2$ let $\nu(\mathscr{X}) := \sum_{X \in \mathscr{X}} (2|X| - l) + |E' - E'(\mathscr{X})|$. By Claim 1, $r(E') = \min_{\mathscr{X}} \nu(\mathscr{X})$. It is enough to prove that the minimum is attained on a one element set system $\mathscr{X} = \{X\}$ where $Z - V \subseteq X \subseteq V$ because $\nu(\{X\}) = 2|X| - l + e(V - X)$, thus by complementing X we get *(i)*.

Let the set system $\mathscr{X}$ lexicographically minimize the vector $(\nu(\mathscr{X}), |E' - E'(\mathscr{X})|, |\mathscr{X}|)$. Then the following holds.

$$|X \cap Y| \leq 1 \text{ for } X \neq Y \in \mathscr{X} \tag{1}$$

$$\nexists X, Y, Z \in \mathscr{X} : \ |X \cap Y| = |X \cap Z| = |Y \cap Z| = 1 \text{ and } X \cap Y \cap Z = \emptyset \tag{2}$$

$$\text{for } u \neq v \in V - Z \text{ there exists a unique } X_{uv} \in \mathscr{X} : u, v \in X_{uv} \tag{3}$$

(1) is true because $2|X| - l + 2|Y| - l = 2|X \cup Y| - l + 2|X \cap Y| - l > 2|X \cup Y| - l$ if $|X \cap Y| \geq 2$, so $\mathscr{X} - X - Y + X \cup Y$ would contradict the minimality of $\nu(\mathscr{X})$.

Suppose on the contrary that (2) is not true. Then there exists such a configuration then $2|X| - l + 2|Y| - l + 2|Z| - l = 2(|X \cup Y \cup Z| + 3) - 3l \geq 2|X \cup Y \cup Z| - l$. Thus by replacing X, Y, Z with $X \cup Y \cup Z$ we lexicographically decrease $(\nu(\mathscr{X}), |E' - E'(\mathscr{X})|, |\mathscr{X}|)$, a contradiction. To prove (3) suppose that $u \neq v \in V - Z$ and there does not exist an X containing u, v. Then adding set $\{u, v\}$ lexicographically decrease $(\nu(\mathscr{X}), |E' - E'(\mathscr{X})|, |\mathscr{X}|)$. The unicity of such a set follows by (1).

We prove that if $u, v, w \in V - Z$, then $X_{uv} = X_{vw}$. Suppose not. Then by (1) $X_{uv} \cap X_{vw} = \{v\}$, but then $X_{uv} \cap X_{uw} = \{u\}$ and $X_{vw} \cap X_{uw} = \{w\}$. This contradicts (2).

This implies that $X_{uv} = X_{u'v'}$ for every $u, v, u', v' \in V - Z$, that is, there exists a unique $X \in \mathscr{X}$ containing $V - Z$, and $|X \cap Y| \leq 1$ for every $Y \in \mathscr{X} - X$. It is easy to see that $\nu(\mathscr{X}) \geq \nu(\{X\})$ because $i_{E'}(X') = i_E(X') \leq 2|X'| - l$ for every $X' \in \mathscr{X} - X$. Therefore the one element set system $\{X\}$ satisfies that $r(E') = \nu(\{X\})$.

(ii) If $|V - Z| \geq 2$, then *(i)* implies *(ii)*. If $|V - Z| \leq 1$, then $r(E + K_{V-Z}) = r(E) = |E| < 2|V| - l$ and $e(Z) = |E| < 2|V| - l \leq 2|Z|$. □

We call a set $Z \subseteq V$ a *good subset* if $e(X) \geq 2|X|$ holds for every $X \subseteq Z$. By Lemma 2 (ii) if E is independent in $\mathscr{M}_{2,l}$, then the problem of finding a minimum cardinality subset Y such that $r(E + K_Y) = 2|V| - l$ is equivalent to finding a maximum size good set Z.

Theorem 3 *We are given a graph $G = (V, E)$ and a set $T \subseteq V$. Then there exists an $O\left((|E| + |V|)^{3/2}\right)$ running time algorithm to find a maximum cardinality good set Z not intersecting T, and there exists an $O(|V|^2)$ running time algorithm to find a minimum cardinality set Z containing T so that $r(E + K_Z) = 2|V| - l$.*

PROOF: First we state that in the second problem we can assume that E is independent in $\mathscr{M}_{2,l}$ because if E' is a base of E in matroid $\mathscr{M}_{2,l}$, then $r(E + K_Z) = r(E' + K_Z)$ holds for all $Z \subseteq V$. Finding a base of an edge-set in $\mathscr{M}_{2,l}$ can be done in $O(|V|^2)$ time (if $l = 2$, then see [13, Section 51.5a], if $l = 3$, then see e.g. [3] and [2] for other references).

By Lemma 2 (ii) the two optimization problems are equivalent. We prove that finding a maximum size good set can be reduced to finding a maximum matching. Let us define the following bipartite graph B. Let $\tilde{V}$ denote a set which contains two nodes v', v'' for each $v \in V - T$. For a set $X \subseteq V - T$ let $\tilde{X}$ will denote the set $\{v' : v \in X\} \cup \{v'' : v \in X\}$. The two color classes of B will be $\tilde{V}$ and E. The edge-set is $F := \{(ev') : v \in V - T \text{ is an endpoint of } e\} \cup \{(ev'') : v \in V - T \text{ is an endpoint of } e\}$. By definition it follows that $Z \subseteq V - T$ is good if and only if $|\Gamma(\tilde{X})| \geq |\tilde{X}|$ for each $X \subseteq Z$, and it is easy to see that this is equivalent to the following: $|\Gamma(X)| \geq |X|$ for each $X \subseteq \tilde{Z}$. By Hall's theorem (see e.g. [13, Theorem 16.6]) Z is good if and only if there is a matching $M \subseteq F$ covering $\tilde{Z}$.

We construct graph $G'=(V',E')$ such that $V':=\tilde{V}\cup E$ and $E':=F\cup\{(v'v''):v\in V-T\}$. Let M be a maximum matching in G' and Z a maximum cardinality good set in G. We claim that $|M|=|V-T|+|Z|$. If we have a good set Z' and $M_{Z'}\subseteq F$ is a matching covering Z', then $M_{Z'}\cup\{(v'v''):v\in V-T-Z'\}$ is a matching of cardinality $2|Z'|+|V-T|-|Z'|=|V-T|+|Z'|$. If M' is a maximal matching, then $Z:=\{v\in V-T:v'$ and v'' is covered by $M'\cap F\}$ is good set of cardinality $|M'|-|V-T|$.

We can apply an algorithm finding a maximum matching in G' to find a maximum size good set in G. This matching can be found in $O(\sqrt{|V'|}|E'|)$ time (for literature see [13, Section 24.4a]). By the definition of G': $|V'|\leq 2|V|+|E|$ and $|E'|\leq 4|E|+|V|$, hence $O(\sqrt{|V'|}|E'|)=O\left((|E|+|V|)^{3/2}\right)$. Using the fact that $|E|\leq 2|V|$ if E is independent in $\mathscr{M}_{2,l}$ we get an $O(|V|^{3/2})$ running time algorithm for independent E. So the total running time of the algorithm to find a minimum cardinality Z set containing T so that $r(E+K_Z)=2|X|-l$ is $O(|V|^2)$. □

Without proof we mention that one can use the above reduction and the Berge-Tutte formula (see e.g. [13, Theorem 24.1]) to deduce the following minimax theorem on the maximum size of a good set.

Theorem 4 *If we are given a graph $G=(V,E)$ and a set $T\subseteq V$, then the following equality holds:*

$$\max_{Z \text{ is good}, Z\cap T=\emptyset}|Z|=\min_{Y:T\subseteq Y\subseteq V}\sum_{C\in\mathscr{C}(G-Y)}\left\lfloor\frac{e(C)}{2}\right\rfloor+|Y-T| \tag{4}$$

We present a short proof for the NP-completeness of the source location problem concerning 3 edge-disjoint trees and the 3 dimensional pinning problem. (*(ii)* was already proved in [10].)

Theorem 5 *Let $G=(V,E)$ be a graph.*
(i) The problem of finding a minimum size $Z\subseteq V$ such that G/Z has 3 edge-disjoint spanning trees is NP-complete.
(ii) The problem of finding a minimum size $Z\subseteq V$ such that $G+K_Z$ is rigid in 3 dimensions is NP-complete.

PROOF: *(i)* This problem is clearly in NP. The problem of finding a maximum stable set in a 3-regular graph is NP-complete, see [4]. It is easy to see that if G is 3-regular, then G/Z has 3 edge-disjoint spanning trees if and only if $V-Z$ is stable.

(ii) This problem is in NP because the problem of 3 dimensional rigidity is known to be in NP. It is easy to see that if G is 3-regular, then $G+K_Z$ is rigid in 3 dimensions if and only if $V-Z$ is stable. □

We mention that the problem of finding a minimum cardinality set Z so that $r(E+K_Z)=2|V|-l$ is equivalent to a special matroid parity problem. If $r_0(Z):=r(E+K_{V-Z})-r(K_{V-Z})$ $(Z\subseteq V)$, then r_0 is a 2-polymatroid function on V. It can be checked that if $r(E)<2|V|-l$, then $r_0(V-Z)=2|V-Z|$ holds if and only if $r(E+K_Z)=2|V|-l$. Hence the question of finding a minimum cardinality Z so that $r(E+K_Z)=2|V|-l$ is equivalent to finding a maximum cardinality W so that $r_0(W)=2|W|$.

Acknowledgment

The author wishes to thank Tibor Jordán for useful discussions and for the proof of Theorem 5.

References

[1] M. BÁRÁSZ, J. BECKER, AND A. FRANK, An algorithm for source location in directed graphs, *Oper. Res. Lett.*, 33(3):221–230, 2005.

[2] A. R. BERG AND T. JORDÁN, Algorithms for graph rigidity and scene analysis, In *Algorithms—ESA 2003*, Lecture Notes in Comput. Sci., pages 78–89. Springer, Berlin, 2003.

[3] H. N. GABOW AND H. H. WESTERMANN, Forests, frames, and games: algorithms for matroid sums and applications, *Algorithmica*, 7(5-6):465–497, 1992.

[4] M. R. GAREY AND D. S. JOHNSON, Computers and intractability, W. H. Freeman and Co., San Francisco, Calif., 1979.

[5] J. GRAVER, B. SERVATIUS, AND H. SERVATIUS, Combinatorial rigidity, American Mathematical Society, 1993.

[6] H. ITO, K. MAKINO, K. ARATA, S. HONAMI, Y. ITATSU, AND S. FUJISHIGE, Source location problem with flow requirements in directed networks, *Optim. Methods Softw.*, 18(4):427–435, 2003.

[7] G. LAMAN, On graphs and rigidity of plane skeletal structures, *J. Engrg. Math.*, 4:331–340, 1970.

[8] L. LOVÁSZ, Matroid matching and some applications, *J. Combin. Theory Ser. B*, 28(2):208–236, 1980.

[9] L. LOVÁSZ AND Y. YEMINI, On generic rigidity in the plane, *SIAM J. Algebraic Discrete Methods*, 3(1):91–98, 1982.

[10] A. MANSFIELD, On the computational complexity of a rigidity problem, *IMA J. Appl. Math.*, 27(4):423–429, 1981.

[11] C. S. J. A. NASH-WILLIAMS, Edge-disjoint spanning trees of finite graphs, *J. London Math. Soc.*, 36:445–450, 1961.

[12] A. RECSKI, Matroid theory and its applications in electric network theory and in statics, volume 6 of *Algorithms and Combinatorics*, Springer-Verlag, Berlin, 1989.

[13] A. SCHRIJVER, Combinatorial optimization. Polyhedra and efficiency, volume 24 of *Algorithms and Combinatorics*, Springer-Verlag, Berlin, 2003.

[14] W. WHITELEY, Some matroids from discrete applied geometry, In *Matroid theory (Seattle, WA, 1995)*, volume 197 of *Contemp. Math.*, pages 171–311. Amer. Math. Soc., Providence, RI, 1996.

Linear approximation algorithms and space lower bounds for the SimRank similarity function on massive graphs

DÁNIEL FOGARAS*

Budapest University of
Technology and Economics
Hungary
fd@cs.bme.hu

BALÁZS RÁCZ*

Computer and Automation Research Institute
of the Hungarian Academy of Sciences
Hungary
bracz+s52@math.bme.hu

Abstract: In this paper we give a randomized method for computing an approximation of the SimRank similarity function on vertices of large graphs. Our method uses $2V$ cells of main memory to preprocess the graph in $O(V+E)$ time, for any graph with V vertices and E edges. The resulting $O(V)$ size array can be used to serve similarity queries in constant time. We also give external memory variants of our algorithms. Thus we significantly improve existing methods, which require $O(V^2)$ storage, infeasible for the primary application area of web search. We demonstrate the scalability of our method by evaluating SimRank on the Stanford WebBase graph and show that it supersedes the co-citation similarity function. We justify the approximation approach by proving that any algorithm computing precise SimRank scores must use a database of $\Omega(V^2)$ bits in worst case, and $\Omega(V)$ bits for the approximate problem. The latter lower bound matches our method up to a logarithmic factor.

Keywords: web graph, similarity search, fingerprint, external memory

1 Introduction

The development of graph-based similarity functions for massive graphs is largely motivated by the application of web search: in the "related pages" search method users submit a web page of their interest as the query and as a result they expect a list of similar pages. This functionality has become a basic and often used search method. Implementation may rely on both the text and the hyperlink structure, but as the text of the web is a very heterogeneous, language dependent dataset, using link-based methods should improve search quality just as it improved ranking with the successful PageRank scheme.

Several link-based similarity functions were suggested as graph algorithms over the *web graph*, with vertices correspondig to web pages and arcs to the hyperlinks between pages. These algorithms either come from social network (such as the reference graph of the scientific literature) analysis, and are too simple for the web (they utilize only the 1-2 step neighborhood of the queried pages), or were developed by adapting link-based ranking schemes. The latter have much more potential, but no algorithm was known so far that would allow the computation of these iterative link-based similarity functions over the whole web graph.

In this paper we present a new, robust and scalable method for computing an approximation of an iterative link-based similarity function, SimRank. Our approximation, described in Section 2 uses a Monte Carlo method with simulated random walks. We analyze the convergence of the approximation in Section 3. Furthermore, in Section 4 we prove lower bounds on the storage complexity of SimRank computation: we show that any algorithm calculating exact SimRank scores must use $\Omega(V^2)$ bits of storage for some graphs with V vertices. However, for the approximate problem, both the lower bound and our algorithm's requirement is proportional to V. In Section 5 we experimentally analyze SimRank on the Stanford WebBase web graph to show the scalability of our methods and conclude that SimRank with its recursiveness is significantly better on the web graph than the co-citation similarity measure originating from sociometry.

1.1 SimRank, a recursive notion of link-based similarity

SimRank was introduced by Jeh and Widom [21] to formalize the intuition that "*two pages are similar if they are referenced by similar pages*". The recursive *SimRank iteration* propagates similarity scores with a constant *decay factor* $c \in (0,1)$:

$$\mathrm{sim}_\ell(u,v) = \frac{c}{|I(u)|\,|I(v)|} \sum_{u' \in I(u)} \sum_{v' \in I(v)} \mathrm{sim}_{\ell-1}(u',v'),$$

*Research was supported by grants OTKA T 42481, T 42559, T 42706 and T 44733 of the Hungarian National Science Fund, and from NKFP-2/0017/2002 project Data Riddle.

where u, v denote a pair of vertices with $u \neq v$ and $I(x)$ denotes the set of vertices linked to x. If $I(u)$ or $I(v)$ is empty, the above sum equals to zero by definition. Furthermore $\mathrm{sim}_\ell(u,v) = 1$, if $u = v$ holds. The SimRank iteration starts with $\mathrm{sim}_0(u,v) = 0$ for $u \neq v$ and $\mathrm{sim}_0(u,v) = 1$ otherwise. The *SimRank score* is defined as the limit $\mathrm{sim}(u,v) = \lim_{\ell\to\infty} \mathrm{sim}_\ell(u,v)$; see [21] for the proof of convergence. Throughout this paper we refer to $\mathrm{sim}_\ell(u,v)$ as a SimRank score, and regard ℓ as a parameter of SimRank.

The original SimRank algorithm [21] calculates the scores by iterating over all pairs of web pages, thus each iteration requires $O(V^2)$ time and memory, which is infeasible for massive graphs.

The following alternate formulation of SimRank scores will be applied in our algorithm. Suppose that a random walk starts from each page and takes ℓ uniform steps following the hyperlinks backwards. Let the random variable $X^\ell_{u,v}$ be the number of steps until the walks of u and v first meet, with $X^\ell_{u,v} = \infty$ if the walks never meet. We shall need the following basic result:

Theorem 1 ([21]) *For the SimRank of pages u and v we have* $\mathrm{sim}_\ell(u,v) = \mathbb{E}(c^{X^\ell_{u,v}})$.

1.2 Complexity requirements for computation on massive graphs

In this section we declare the strict computational complexity, memory usage and parallelization requirements for our similarity search algorithms. We assume that the web graph is available from a repository of already downloaded web pages. Similarity search algorithms are divided into the following phases:

Precomputation: preprocess the web graph and compute a *similarity database* to support fast queries later.

Similarity query: calculate the similarity score $\mathrm{sim}(u,v)$ for given web pages u and v.

Top query: for a given query page u enumerate the pages most similar to u, i.e., the pages v with $\mathrm{sim}(u,v)$ score above a given similarity threshold.

The complexity requirements for these phases are listed below.

Computational complexity: the similarity database is computed in $O(V+E)$ time. Furthermore the time complexity of similarity query is constant, and that of top query is proportional to the size of the result list.

Memory usage: In case of the *semi-external memory* model $\Theta(V)$ main memory is available for computation, where V denotes the number of vertices (web pages) [27]. The more restrictive *external memory* model enables to utilize a main memory of constant size.

Accessing the web graph: the edges can only be accessed sequentially as a stream (read from external memory). The process of reading the whole stream of edges will be referred to as an *edge-scan*.

Parallelization: Both precomputation and queries can be implemented to utilize the computing power and storage capacity of tens to thousands of servers interconnected with a fast local network.

Fulfilling these requirements allow the computation on the entire web graph with $4 \cdot 10^9$ vertices and $40 \cdot 10^9$ edges. Analogous requirements appear in [30] for estimating the neighborhood functions of massive graphs, furthermore these requirements describe the computational environment in which Google's PageRank scores are computed [9, 29].

1.3 Related Results

Jeh and Widom [21] introduced SimRank to formalize the recursive reference-based intuition about similarity. They show several other variants within the same framework by different averaging methods in SimRank equations. Independently, Melnik et al. [26], Blondel et al. [7], Heymans and Singh [19] applied the same intuition to define and compute similarities between vertices of graphs G_1 and G_2. These three algorithms were motivated by the applications of matching information models like XML trees, extracting synonyms, and deriving phylogenetic trees from enzyme-enzyme relational graphs, respectively. These algorithms are essentially identical to some special cases of the SimRank framework if $G_1 = G_2$. Furthermore Pivovarov and Trunov [31] apply SimRank-like iterations to obtain a hierarchical clustering of web pages; Nowell and Kleinberg [24] predict the appearance of edges in social networks from SimRank scores.

Besides SimRank, several other link-based algorithms were designed to evaluate node-to-node similarities in networked information spaces. We refer to [24] for an exhaustive list of the available methods. Unfortunately the methods are either very simple, like co-citation, or not scalable to massive graphs. For instance, the HITS-based Companion algorithm [12] and the max-flow/min-cut-based similarities of [25] build the vicinity graph of queried vertices and then perform complex computations in query time; building the vicinity graph requires random access to the web graph. This approach does not seem feasible under the high workload of search engines, though large amount of research was done to achieve random access

with sufficient compression techniques [6, 8]. Another approach would precompute all similarity scores in advance, but this would make running time and space requirement at least quadratic as in the case of SimRank.

As a compromise between random access and external memory algorithms, we provide semi-external memory solutions assuming that $\Theta(V)$ memory is available for graph algorithms. The same assumption was taken by Sibeyn et al. [33] for DFS, Laura et al. [23] for retrieving small bipartite cliques in the web graph, and Abello et al. [2] for finding large cliques. Google's PageRank was also treated as a semi-external memory graph algorithm in [9, 29].

The key idea of our algorithm is Monte Carlo (MC) simulation of random walks, analogously to the text-based MC similarity search algorithms developed by Broder [10] and Heintze [15]. MC methods were successfully applied by Cohen [11] and Palmer et al. [30] to estimate the sizes of transitive closures and neighborhood functions in massive graphs. Simulated random walks also play an important role in a different aspect of web algorithms, when a crawler attempts to download a uniform sample of web pages (Markov Chain Monte Carlo) and compute various statistics [17, 32, 4, 16] or page decay [5]. More theoretical results about approximate counting for combinatorial problems are discussed in Chapter 11 of [28]. A significant difference between our algorithm and those mentioned above is that in our case the random variable behind the Monte Carlo simulation is not of a specific, well-known distribution (like of binomial distribution in [11]). Thus our convergence estimates are weaker, we bound the probability of absolute error. Fortunately, this suffices for our main application area, since web search engines only need to reveal large gaps in similarity scores for top queries.

The lower bounds of Section 4 are based on variants of the techniques of Henzinger et al. [18] proving lower bounds for the space complexities of several graph algorithms with stream access to the edges. We refer to the PhD thesis of Bar-Yossef [3] as a comprehensive survey with several new space complexity lower bounds for massive data set computations obtained by reduction to communication complexity problems.

An earlier stage of our research was reported in [13]. The results are improved by the compact but unbiased representation of Section 2.2, a new proof of the probability of error, lower bounds on the database size and experiments on a significantly larger well-known dataset, the Stanford WebBase graph.

2 Our Algorithm

This section discusses our main result, a novel algorithm for approximating SimRank scores with respect to the strict complexity requirements stated in the introduction. A random walk of length ℓ starting from a vertex v and following the links backwards will be referred to as a *fingerprint* of v. Outlining our approach, we generate and store N independent fingerprints during precomputation. Upon queries meeting times are evaluated and averaged with exponential weights yielding unbiased estimations for SimRank scores by Theorem 1.

Once the fingerprints are precomputed and stored in a similarity database of size $N \cdot V \cdot \ell$, the similarity queries can be evaluated with an external memory algorithm by loading the fingerprints of queried vertices into main memory and scanning the paths for the first vertices in common. Top queries can be evaluated with standard inverted indexing techniques. This also provides an external memory algorithm with constant number of database accesses compared to the graph size. We refer to [13] containing pseudo-code for these algorithms.

Generating random walks (fingerprints) for all vertices of a massive graph is not trivial in the external (semi-external) memory model with stream access to the edges. After summarizing our precomputation algorithm in the next subsection, we will introduce an alternate representation of the fingerprints of total size $N \cdot V$. Such a compact representation is especially advantageous, if SimRank is computed for large values of ℓ. An other advantage of the reduced storage is that with parallelization over N machines in the semi-external memory model queries can be served without disk accesses.

The key idea that allows efficient precomputation and reduced storage is, that we do not need the fingerprint walks to be independent to have an unbiased estimate. We only need them to be pairwise independent until the first meeting time. They can be arbitrarily coupled after that; in our case they will stick together.

Throughout this paper we write V for the number of vertices (web pages), E the number of edges (hyperlinks), N for the number of fingerprints (random walks) and ℓ for the path length (number of SimRank iterations). Typical values in real applications are as follows: $V \approx 10^8 - 10^{10}$, $E \approx 10 \cdot V$, $N \approx 100 - 1000$ and $\ell \approx 5 - 20$. Furthermore, we will slightly abuse the notation of SimRank and use $\mathrm{sim}(\cdot,\cdot)$ instead of $\mathrm{sim}_\ell(\cdot,\cdot)$.

2.1 Precomputation: simultaneously generating random walks

The precomputation phase of our algorithm generates N independent fingerprints for each vertex of the input graph. A naïve algorithm would compute the fingerprints for each vertex one-by-one requiring random access to the edge structure. Instead, we generate one fingerprint for each vertex simultaneously. Suppose that we have already computed partial fingerprints of length k, and the ending vertices of the partial fingerprints are stored in an array of size V. Then by scanning the edges of the web graph, we select for each vertex v an edge from those linking to v at random. If the V randomly generated in-edges are stored in the main memory, we can easily extend each partial fingerprint with ending vertex v by the random in-edge of v.*

*Recall that the random walks of SimRank follow the hyperlinks backwards.

Figure 1: Representing the fingerprints of 1, 2, 3, 4 and 5 with an FPG.

Notice that the outlined procedure guarantees pairwise independence for a pair of fingerprints exactly until they first meet, since both partial fingerprints will be extended by the same random in-edge later.

The above algorithm fulfills the requirements of the semi-external memory model, as it allocates no more than $2V$ memory cells for the ending vertices of walks and the random in-edges. Even an external memory implementation can be achieved by dumping these data to disk, sorting the ending vertices and merging them with the in-edges. The edges of the of the web graph are accessed sequentially $N \cdot \ell$ times. The number of edge-scans can be significantly reduced by generating all the required $N \cdot \ell$ sets of in-edges with a single scan over a sorted stream of edges. We conclude that our complexity requirements are satisfied during precomputation. A more detailed description of the algorithm with pseudo-code is presented in [13].

2.2 Fingerprint graph

The previously outlined algorithm computes a fingerprint for each vertex such that any two fingerprints are coupled to stick together after they first meet. For these walks, we introduce a compact representation of size V, thus the total size of database reduces from $N \cdot V \cdot \ell$ to $N \cdot V$. A further advantage is that in case of the semi-external memory model the representation already fits into main memory. Therefore, we can apply random access to this representation and develop efficient query algorithms.

Given a fingerprint for each web page, the *fingerprint graph, FPG* will be defined over the vertex set $1, 2, \ldots, V$ corresponding to the web pages. Let us denote the fingerprint of vertex u by $\mathrm{FP}(u)$, and recall that $X_{u,v}$ denotes the first meeting time of $\mathrm{FP}(u)$ and $\mathrm{FP}(v)$. There exists an arc (u,v) in FPG, iff $u = \mathrm{argmin}_{u'<v} \left\{ X_{u',v} = \min_{u''<v} X_{u'',v} \right\}$ and $X_{u,v} < \infty$, i.e., $u < v$ and $\mathrm{FP}(v)$ first meets $\mathrm{FP}(u)$ from those corresponding to web pages with $u' < v$. If more than one $\mathrm{FP}(u')$ meet at the same time, then the smallest u' is taken as u. Furthermore, the meeting time $X_{u,v}$ is assigned to the arc (u,v) and this is stored as $\mathrm{dist}(u,v)$. See Fig. 1 for a small sample FPG representation of fingerprints.

Since in an FPG the in-degree of each vertex is at most one and $u < v$ holds for each edge (u,v), the structure of an FPG is very simple:

Fact 2 *Each component of an FPG is a rooted tree, and an FPG has no more than V edges.*

Notice that this bound holds independently of the length ℓ of the paths. Starting from a vertex u, there is a unique path that follows the edges of the FPG backwards. This path visits some vertices with decreasing labels. A vertex w of the path corresponds to one or more steps of $\mathrm{FP}(u)$ and possibly several other fingerprints that sticked together with it, from which $\mathrm{FP}(w)$ is the one with the smallest w. If the unique reverse paths starting from u and v first intersect[†] at w, then the corresponding $\mathrm{FP}(u)$ and $\mathrm{FP}(v)$ also meets on a node of $\mathrm{FP}(w)$, where no walks of smaller index than w reside at that time. This implies that the first meeting time of $\mathrm{FP}(u)$ and $\mathrm{FP}(v)$ can be extracted from the FPG:

Fact 3 *Suppose that the unique reverse paths of FPG starting from u and v first intersect at vertex w through the edges* (w,u') *and* (w,v')*, respectively. Then* $\mathrm{FP}(u)$ *and* $\mathrm{FP}(v)$ *also meet each other and the first meeting time equals to* $\max\{\mathrm{dist}(w,u'), \mathrm{dist}(w,v')\}$*. If the reverse FPG paths intersect at u or v then the fist meeting time of* $\mathrm{FP}(u)$ *and* $\mathrm{FP}(v)$ *equals to* $\mathrm{dist}(u,v')$ *or* $\mathrm{dist}(v,u')$*, respectively.*

Based on the above fact, similarity queries can be evaluated by walking through the unique reverse paths of the queried vertices in the FPG until the first common ancestor is accessed. To evaluate the top query for a given vertex q, we need to traverse the component of q in the FPG. See the pseudo-code of Algorithm 1, which successively accesses the ancestors of q to find vertices with reverse paths that intersect the path of q at the currently visited ancestor.

This algorithm fulfills the stated complexity requirements if the threshold for top query is set to 0, i.e., all pages with positive estimated similarity have to be returned. The required random access to an FPG can be realized in the semi-external memory model, since an FPG has V vertices and less than V edges. However, we can hardly imagine a machine that stores all the N FPGs in main memory, if $V \approx 10^9$. In case of large web-search engines the algorithm can be still applied with *parallel computing* on a cluster of N servers interconnected with some fast local network. Each server stores one FPG in its main memory; computes its SimRank estimation independently, and the result lists are merged by a front-end server. This type of parallelization perfectly suits for the requirements of cluster-computing in terms of *fault tolerance* and *load balancing*; if

[†]Unlike meeting of walks, intersection of two paths is allowed after a different number of steps.

Algorithm 1: Top query (with similarity threshold 0)

Input: q=query page, N=number of fingerprints , c=decay factor, Fingerprint graphs=$FPG_1, \ldots, FPG_N$
Output: $\mathrm{sim}(q,w)$ scores for all vertex w with positive estimated similarity.
FPG_k traversing subroutines: $\mathrm{Child_k}(u)$ returns the set of vertices linked by u, $\mathrm{Parent_k}(u)$ returns the unique parent of u, $\mathrm{Descendant_k}(u)$ returns all the vertices of the subtree with root u. $\mathrm{dist}_k(u,v)$ is the assigned value to arc (u,v) of FPG_k.

for $k := 1,\ldots,N$ **do**
  repeat
    Prev := NULL; Ancestor := q */*Store the previously and currently visited ancestors of q*/*
    for all $v \in \mathrm{Child_k}(\mathrm{Ancestor}) \setminus \{\mathrm{Prev}\}$ **do** */*Siblings of the previously visited ancestor*/*
      for all $w \in \mathrm{Descendant_k}(v)$ **do** */*Vertices first meeting* Ancestor *from the ancestors of q*/*
        $\mathrm{sim}(q,w)$ += $\frac{1}{N} \cdot c^{\max\{\mathrm{dist_k(Ancestor},v),\mathrm{dist_k(Ancestor,Prev)}\}}$ */*Estimate similarity by Fact 3 */*
    Prev := Ancestor; Ancestor := $\mathrm{Parent_k}(\mathrm{Ancestor})$ */*Step to the next ancestor of q*/*
  until Ancestor = NULL */*Terminate at the end of the reverse path of q*/*
return pages sorted by $\mathrm{sim}(q,w)$

one server does not respond to some query, it causes just a small $(N-1)/N$ loss of precision. Furthermore if more than N machines are available for computation, then any N machines can be queried yielding a balanced load on the machines. More details about the advantages of Monte Carlo parallelization are discussed in our previous paper [13].

Notice that a component of an FPG corresponds to a group of paths that finally stick together during precomputation. The above algorithm traverses only one component of the FPG, and all of its vertices appear in the output; thus we can give an external memory implementation with two disk seeks per fingerprint by looking up and loading only the respective component of the FPG into memory.

3 How Many Fingerprints are Needed?

In this section we discuss the convergence of our estimates, and analyze the required amount of fingerprints for proper precision.

It is clear by the law of large numbers that as the number of fingerprints $N \to \infty$, the estimated similarity denoted by $\widehat{\mathrm{sim}}(u,v)$ converges to the actual SimRank score $\mathrm{sim}(u,v)$. The rate of convergence is $\mathcal{O}(\frac{1}{\sqrt{N}})$ in the sense that for each vertices u,v, $\frac{\widehat{\mathrm{sim}}(u,v)-\mathrm{sim}(u,v)}{\sigma\sqrt{N}} \Rightarrow \mathcal{N}(0,1)$ (for some constant σ, depending on u and v but not N), i.e. $\frac{\widehat{\mathrm{sim}}(u,v)-\mathrm{sim}(u,v)}{\sqrt{N}}$ has a limit distribution by the central limit theorem.

Unfortunately, this is not enough for our purposes. To gain proper exact values for $\widehat{\mathrm{sim}}(u,v)$ we would need up to tens of thousands of samples. Notice however, that we do not require the actual values to be very precise, but we need the ordering defined by the approximation match fairly closely the ordering defined by the SimRank values for $\mathrm{top}(u)$ queries. In this sense we have exponential convergence:

Theorem 4 *For any vertices u,v,w assume that* $\mathrm{sim}(u,v) > \mathrm{sim}(u,w)$. *Then the probability of interchanging v and w in the approximated similarity ranking* $\mathrm{top}(u)$ *tends to 0 exponentially in the number of fingerprints used:* $\Pr\widehat{\mathrm{sim}}(u,v) < \widehat{\mathrm{sim}}(u,w) \to 0$ *exponentially with N.*

For theoretical applications it is also important, that this convergence is uniform among those node triplets of the graph, that have a lower bounded similarity difference:

Theorem 5 *For any $\varepsilon,\delta > 0$ there exists an N_0 such that for any $N \geq N_0$ number of fingerprints, for any graph and vertices u,v,w such that* $\mathrm{sim}(u,v) - \mathrm{sim}(u,w) > \delta$, *we have* $\Pr\widehat{\mathrm{sim}}(u,v) < \widehat{\mathrm{sim}}(u,w) < \varepsilon$.

We prove Theorems 4 and 5 together.

PROOF: Consider a fingerprint of u and let Z be the following random variable: $Z = X_{u,v} - X_{u,w}$. ($X_{u,v}$ is the first meeting time of walks starting from u and v.) Then $\mathbb{E}Z = \mathrm{sim}(u,v) - \mathrm{sim}(u,w) > 0$. Estimating the SimRank values from N fingerprints, the event of interchanging v and w in the rankings is equivalent to taking N independent Z_i variables and having $\sum_{i=1}^N Z_i < 0$. This can be upper bounded using Bernstein's inequality and the fact that $Z \in [-c,c]$, thus $\mathrm{Var}(Z) \leq c^2$:

$$\begin{aligned}
\Pr\tfrac{1}{N}\textstyle\sum_{i=1}^N Z_i < 0 &\leq e^{-N\frac{(\mathbb{E}Z)^2}{2\mathrm{Var}(Z)+4c/3\,\mathbb{E}Z}} \\
&\leq e^{-N\frac{(\mathrm{sim}(u,v)-\mathrm{sim}(u,w))^2}{2c^2+4c/3(\mathrm{sim}(u,v)-\mathrm{sim}(u,w))}} \\
&\leq e^{-0.3N\frac{(\mathrm{sim}(u,v)-\mathrm{sim}(u,w))^2}{c^2}} \leq e^{-0.3N\frac{\delta^2}{c^2}}
\end{aligned}$$

From the above inequalities both theorems follow. □

Theorem 4 shows that even a modest amount (e.g. $N = 100$) of fingerprints is enough to make distinction between the high, medium and low ranked pages according to the SimRank scores, and this is exactly what is needed in search engine applications (the related query result list should be sorted using both the global and the similarity ranking).

Theorem 5 has an important theoretical consequence. When we investigate the asymptotic growth of the database size as a function of the graph size, the number of required fingerprints remains constant for fixed ε and δ, thus the database size is linear in the graph size.

4 Lower Bounds for the Similarity Database Size

In this section we will prove several lower bounds on the space complexity of calculating SimRank. In particular, we prove that except for the approximate approach, the required similarity database size is at least $\Omega(V^2)$ bits for some graphs with V vertices. In the approximate problem the lower bound is linear in V, which is matched by our algorithm of Section 2.

More precisely we will consider two-phase algorithms: in the first phase the algorithm has access to the edge set of the graph and has to compute a similarity database, in the second phase the algorithm gets a query, and has to answer using only the similarity database. We will lower bound the similarity database size in this model using reductions to communication complexity problems. We will consider the following types of queries (all statements hold for any fixed $\ell \geq 1$):

(1) Exact: Calculate the SimRank $\mathrm{sim}(u,v)$ of u and v.

(2) Approximate: Estimate $\mathrm{sim}(u,v)$ with a $\widehat{\mathrm{sim}}(u,v)$ such that for fixed $\varepsilon, \delta > 0$

$$\Pr|\widehat{\mathrm{sim}}(u,v) - \mathrm{sim}(u,v)| < \delta \geq 1-\varepsilon$$

(3) Positivity: Decide whether $\mathrm{sim}(u,v)$ is positive with error probability at most ε.

(4) Comparison: Given u and v,w vertices, decide whether v or w is more similar to u according to SimRank, with error probability at most ε.

(5) ε–δ comparison: For fixed $\varepsilon > 0, \delta > 0$ and any triple u,v,w, for which $|\mathrm{sim}(u,v) - \mathrm{sim}(u,w)| > \delta$, decide the comparison problem with error probability at most ε.

Our tool towards the lower bounds will be the asymmetric communication complexity game *bit-vector probing* [18]: there are two players A and B, player A has x, an m-bit vector, B has an index $y \in \{1,2,\ldots,m\}$, and they have to compute the function $f(x,y) = x_y$, i.e., the output is the y^{th} bit of the input vector x. To compute the proper output they have to communicate, and communication is restricted in the direction $A \to B$. The *one-way communication complexity* [22] of this function is the number of bits of transferred in the worst case by the best protocol.

Theorem 6 ([18]) *Any protocol that outputs the correct answer to the bit-vector probing problem with probability at least* $\frac{1+\gamma}{2}$ *must transmit at least* γm *bits.*

Now we are ready to prove our lower bounds.

Theorem 7 *Any algorithm solving the positivity problem (3) with probability at least* $\frac{1+\gamma}{2}$ *must use a database of size* $\Omega(\gamma V^2)$ *bits.*

PROOF: We give a communication protocol for the bit-vector probing problem. Given an input bit-vector x we will create a graph, that "encodes" the bits of this vector. Player A will create a similarity database of this graph, and transmit it to B. Then Player B will use the positivity query algorithm for some vertices (depending on the requested index y) such that the answer to the positivity query will be the y^{th} bit of the input vector x. Thus if the algorithm solves the positivity query with probability $\frac{1+\gamma}{2}$, then this protocol solves the bit-vector probing problem with probability $\frac{1+\gamma}{2}$, so the size of the transferred database is at least γm.

Let x be an input to the bit vector probing problem with length $m = n^2$. We construct a graph on $3n$ vertices $u_1,\ldots,u_n$, $z_1,\ldots,z_n$ and $v_1,v_2,\ldots,v_n$ as follows. For each $1 \leq i \leq n$ and $1 \leq j \leq n$ the edge (z_i,v_j) is in the graph iff bit $(i-1)n+j$ is set in the input vector x.

For any index $1 \leq y \leq m$ let $y = (i-1)n+j$; the SimRank value $\mathrm{sim}(u_i,v_j)$ is positive iff (z_i,v_j) edge was in the graph preprocessed, thus iff bit y was set in the input vector. The theorem follows. □

Corollary 8 *Any algorithm solving the exact SimRank problem (1) must have a similarity database of size* $\Omega(V^2)$ *bits.*

Theorem 9 *Any algorithm solving the approximation problem (2) needs a similarity database of* $\Omega(\frac{1-2\varepsilon}{\delta}V)$ *bits on graphs with* $V = \Omega(\frac{1}{\delta})$ *vertices. For smaller* δ *the similarity database requires* $\Omega((1-2\varepsilon)V^2)$ *bits.*

PROOF: We will modify the construction of Theorem 7 for the approximation problem. We have to achieve that when a bit is set in the input graph, then the queried $\mathrm{sim}(u_i, v_j)$ value should be at least 2δ, so that the approximation will decide the positivity problem, too. If vertex v_i in the input graph of our construction has k edges connected to it, then for each of those z_j, for the respective vertex u_j we have $\mathrm{sim}(u_i, v_j) = \frac{1-c}{k}$. For this to exceed 2δ we may have at most $\frac{1-c}{2\delta}$ possible z_j vertices (the number of v_i vertices is not restricted). With $\frac{1+\gamma}{2} = 1-\varepsilon$ the theorem follows. For small δ the original construction suffices. □

This radical drop in the storage complexity is not surprising, as our approximation algorithm achieves this bound (up to a logarithmic factor): for a fixed ε, δ we can calculate the necessary number of fingerprints N, and then for each vertex in the graph we store exactly N fingerprints, independently of the size of the graph. This is a linear database, though the constant makes it very impractical. In the comparison problems (4) and (5) we have the same results. By Theorems 4 and 5 the constants appear to be modest in the ε–δ comparison problem (5).

Theorem 10 *Any algorithm solving the comparison problem (4) with probability* $\frac{1+\gamma}{2}$ *requires a similarity database of* $\Omega(\gamma V^2)$ *bits.*

PROOF: We will modify the graph of Theorem 7 so that the existence of the specific edge can be queried using the comparison problem. To achieve this we will introduce a fourth set $w_1, \ldots, w_n$ of vertices in the graph construction, one for each $v_1, \ldots, v_n$ vertex such that w_j is the complement of v_j: Player A puts the arc (z_i, w_j) in the graph iff (z_i, v_j) is not an arc, which means bit $(i-1)n+j$ was not set in the input vector.

Then upon quering bit $y = (i-1)n+j$, exactly one of $\mathrm{sim}(u_i, v_j), \mathrm{sim}(u_i, w_j)$ will be positive (depending on the input bit x_y), thus the comparison query $\mathrm{sim}(u_i, v_j) > \mathrm{sim}(u_i, w_j)$ will yield the required output for the bit-vector probing problem. □

Corollary 11 *Any algorithm solving the* ε–δ *comparison problem (5) needs a similarity database of* $\Omega(\frac{1-2\varepsilon}{\delta}V)$ *bits on graphs with* $V = \Omega(\frac{4}{\delta})$ *vertices. For smaller* δ *the similarity database needs* $\Omega((1-2\varepsilon)V^2)$ *bits.*

PROOF: Modifying the proof of Theorem 10 along the lines of the proof of Theorem 9 yields the necessary results. □

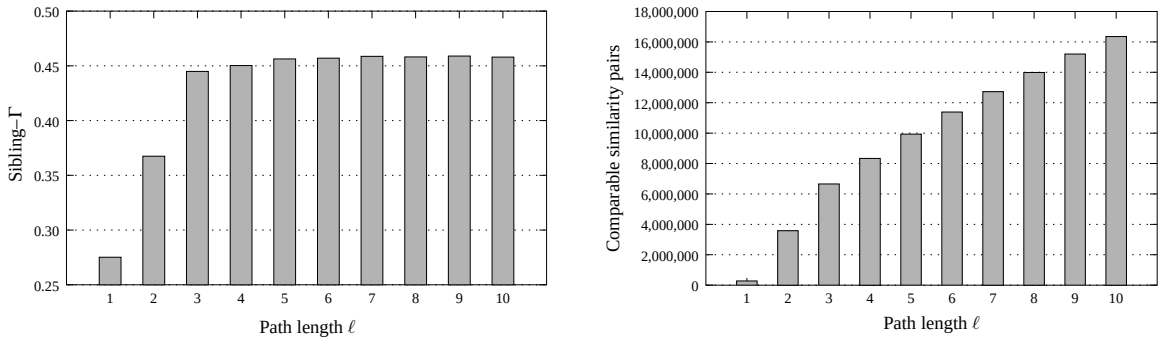


Figure 2: Measuring the quality of SimRank as a function of path length ℓ.

5 Experiments

The fundamental question of hyperlink-based similarity search is addressed by our experiments: *do the ℓ-neighborhoods*[‡] *of vertices hold similarity information relevant to human users?* The question is translated into measuring the quality of SimRank top lists as a function of path length ℓ. This will also answer the question: do iterative similarity functions have advantages over simple ones (like co-citation, almost equivalent to the $\ell = 1$ case)? The effects of further parameters such as the number N of fingerprints and decay factor c are reported by experiments of [13] on a significantly smaller data set.

Similarity top lists were computed on the web graph created by the Stanford WebBase project [20] in 2001 containing 80M vertices and 800M edges after removing dangling links. The quality of SimRank scores is expressed by sibling Γ measure [14]. Due to space constraints, we will not define sibling Γ formally; it compares the similarity scores to ground truth similarities obtained from the Open Directory Project (ODP, [1]), a hierarchically categorized directory of web pages. The category tree of ODP provides ground truth *similarity pairs* by claiming that $\mathrm{sim}(u,v) < \mathrm{sim}(u,w)$ should hold for ODP pages u, v and w, if v is a closer "relative" of u in the hierarchy than w (familial distance). A similarity pair will be referred to as *comparable* by an algorithm if both $\mathrm{sim}(u,v)$ and $\mathrm{sim}(u,w)$ are computed to be larger than a minimal similarity threshold, which was set to zero in our experiments. To evaluate sibling Γ for SimRank scores we check if the above inequality holds for the comparable similarity pairs. The resulting $\Gamma \in (-1,1)$ value is 1, if all the above inequalities hold for the computed SimRank scores; while $\Gamma = -1$, if all pairs are ordered reversely. For our algorithms $\Gamma \approx 0.5$, implying that a comparable similarity pair chosen at random from a SimRank top list is ordered correctly by our algorithm with probability over 0.75. Notice that high Γ quality can be achieved even with very few comparable pairs, which may result in extremely short top lists. Therefore we decided to measure the number of comparable similarity pairs in addition to sibling Γ.

From a web graph of 80M vertices and 800M edges a similarity database was computed for $N = 100$ fingerprints and path length $\ell = 10$. The database was then truncated to the ODP pages resulting in a size of 1.8Gbytes. The precomputation took four hours on a machine with 2.8GHz Intel Pentium 4 processor, 2Gbyte main memory and Linux OS. Sibling Γ was evaluated for different values of ℓ. The results are summarized in Fig. 2. The Γ values suggest that the quality of SimRank significantly increases in the range $\ell = 1 \ldots 5$, and Γ changes very little afterwards. The right side of the figure shows that the larger the ℓ is the more similarity pairs become comparable, i.e., the longer the results lists for top queries get.

Our experiments strongly support the affirmative answer to the main question of this section. However, to determine an optimal value of ℓ would require measurements with a cost function of the computation overhead for larger values. In addition, non-ODP pages should also be taken into account as they form the majority of the web.

6 Conclusion

SimRank is a similarity function over the nodes of a graph, with primary application area being the graph of web pages and hyperlinks. Previous algorithms have memory requirement and running time quadratic in the number of graph nodes. In this paper we propose a scalable algorithm using a linear database and constant query time for computing approximate values of SimRank, based on Monte Carlo simulation of random walks over the graph. The estimate is unbiased and can be either served from disk or on the main application platform (a cluster of PC category machines) from main memory using a suitable representation of the database.

Furthermore, we have justified the relaxation of the problem to the approximate version by proving that any algorithm calculating exact similarity scores requires a similarity database with size quadratic in the number of vertices, while the approximate solution requires only a linear database.

Acknowledgement

We wish to thank András Benczúr, Katalin Friedl, Lajos Rónyai, Tamás Sarlós for valuable discussions and comments, and Glen Jeh, Taher Haveliwala and the Stanford WebBase project for providing the graph for our experiments.

References

[1] Open Directory Project (ODP). `http://www.dmoz.org`.

[2] J. Abello, P. Pardalos, and M. G. C. Resende. On maximum clique problems in very large graphs. In *External memory algorithms*, pages 119–130. American Mathematical Society, 1999.

[3] Z. Bar-Yossef. The complexity of massive data set computations, 2002. PhD Thesis, UC Berkeley.

[4] Z. Bar-Yossef, A. Berg, S. Chien, J. Fakcharoenphol, and D. Weitz. Approximating aggregate queries about web pages via random walks. In *Proceedings of the 26th International Conference on Very Large Data Bases*, pages 535–544. Morgan Kaufmann Publishers Inc., 2000.

[‡]The *ℓ-neighborhood* of a vertex refers to the set of vertices, where from the given vertex can be reached within ℓ steps.

[5] Z. Bar-Yossef, A. Z. Broder, R. Kumar, and A. Tomkins. Sic transit gloria telae: towards an understanding of the web's decay. In *Proceedings of the 13th World Wide Web Conference (WWW)*, pages 328–337. ACM Press, 2004.

[6] K. Bharat, A. Broder, M. Henzinger, P. Kumar, and S. Venkatasubramanian. The connectivity server: Fast access to linkage information on the web. In *Proceedings of the 7th World Wide Web Conference, Brisbane, Australia*, 1998.

[7] V. D. Blondel, A. Gajardo, M. Heymans, P. Senellart, and P. V. Dooren. A measure of similarity between graph vertices. with applications to synonym extraction and web searching, 2004. To appear in: SIAM Review.

[8] P. Boldi and S. Vigna. The webgraph framework I: compression techniques. In *Proceedings of the 13th World Wide Web Conference (WWW)*, pages 595–602. ACM Press, 2004.

[9] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998.

[10] A. Z. Broder. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences (SEQUENCES'97)*, pages 21–29. IEEE Computer Society, 1997.

[11] E. Cohen. Size-estimation framework with applications to transitive closure and reachability. *J. Comput. Syst. Sci.*, 55(3):441–453, 1997.

[12] J. Dean and M. R. Henzinger. Finding related pages in the World Wide Web. *Computer Networks (Amsterdam, Netherlands: 1999)*, 31(11–16):1467–1479, 1999.

[13] D. Fogaras and B. R´acz. A scalable randomized method to compute link-based similarity rank on the web graph. In *Proceedings of the Clustering Information over the Web workshop, Conference on Extending Database Technology*, 2004.

[14] T. H. Haveliwala, A. Gionis, D. Klein, and P. Indyk. Evaluating strategies for similarity search on the web. In *Proceedings of the 11th World Wide Web Conference (WWW)*, pages 432–442. ACM Press, 2002.

[15] N. Heintze. Scalable document fi ngerprinting. In *1996 USENIX Workshop on Electronic Commerce*, November 1996.

[16] M. R. Henzinger, A. Heydon, M. Mitzenmacher, and M. Najork. Measuring index quality using random walks on the Web. In *Proceedings of the 8th World Wide Web Conference, Toronto, Canada*, pages 213–225, 1999.

[17] M. R. Henzinger, A. Heydon, M. Mitzenmacher, and M. Najork. On near-uniform url sampling. In *Proceedings of the 9th international World Wide Web conference on Computer networks*, pages 295–308, 2000.

[18] M. R. Henzinger, P. Raghavan, and S. Rajagopalan. Computing on data streams. *External memory algorithms*, pages 107–118, 1999.

[19] M. Heymans and A. K. Singh. Deriving phylogenetic trees from the similarity analysis of metabolic pathways. *Bioinformatics*, 19(90001):138i–146, 2003.

[20] J. Hirai, S. Raghavan, H. Garcia-Molina, and A. Paepcke. Webbase: a repository of web pages. In *Proceedings of the 9th World Wide Web Conference (WWW)*, pages 277–293. North-Holland Publishing Co., 2000.

[21] G. Jeh and J. Widom. SimRank: A measure of structural-context similarity. In *Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2002.

[22] E. Kushilevitz and N. Nisan. *Communication complexity*. Cambridge University Press, 1997.

[23] L. Laura, S. Leonardi, S. Millozzi, U. Meyer, and J. F. Sibeyn. Algorithms and experiments for the webgraph. In *Proceedings of the 11th Annual European Symposium on Algorithms (ESA03)*, 2003.

[24] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *Proceedings of the twelfth international conference on Information and knowledge management*, pages 556–559. ACM Press, 2003.

[25] W. Lu, J. Janssen, E. Milios, and N. Japkowicz. Node similarity in networked information spaces. In *Proceedings of the 2001 conference of the Centre for Advanced Studies on Collaborative research*, page 11. IBM Press, 2001.

[26] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity fl ooding: A versatile graph matching algorithm and its application to schema matching. In *Proceedings of the 18th International Conference on Data Engineering (ICDE.02)*, 2002.

[27] U. Meyer, P. Sanders, and J. Sibeyn. *Algorithms for Memory Hierarchies, Advanced Lectures*. LNCS, Springer, 2003.

[28] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[29] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[30] C. R. Palmer, P. B. Gibbons, and C. Faloutsos. ANF: a fast and scalable tool for data mining in massive graphs. In *Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 81–90. ACM Press, 2002.

[31] G. Pivovarov and S. T. Seus. Eqrank: a self-consistent equivalence relation on graph vertexes. *SIGKDD Explor. Newsl.*, 5(2):185–190, 2003.

[32] P. Rusmevichientong, D. M. Pennock, S. Lawrence, and C. L. Giles. Methods for sampling pages uniformly from the world wide web. In *AAAI Fall Symposium on Using Uncertainty Within Computation*, pages 121–128, 2001.

[33] J. F. Sibeyn, J. Abello, and U. Meyer. Heuristics for semi-external depth fi rst search on directed graphs. In *Proceedings of the fourteenth annual ACM symposium on Parallel algorithms and architectures*, pages 282–292. ACM Press, 2002.

Submodularity and Polyhedra

SATORU FUJISHIGE

Research Institute for Mathematical Sciences
Kyoto University
Kyoto 606-8502
Japan
fujishig@kurims.kyoto-u.ac.jp

Abstract:

We briefly give a survey on recent developments in theory and algorithms for submodular functions and their associated polyhedra.

The aim of the present talk is to introduce some fundamental, polyhedral and algorithmic (open) problems related to, or motivated by, submodularity.

Keywords: submodular function, base polyhedron, algorithms, discrete convexity

Degree Conditions and Disjoint Cycles in Graphs

SHINYA FUJITA

Department of Mathematics
Keio University
Yokohama 223-8522., Japan
shinyaa@comb.math.keio.ac.jp

Abstract: Recent results on degree conditions and the existence of vertex-disjoint cycles in graphs will be reviewed. Moreover, we give an outline of the proof of a theorem concerning a sufficient condition for the existence of a specified number of vertex-disjoint cycles and isolated vertices covering all vertices in a graph.

Keywords: vertex-disjoint cycles, degree conditions, partition of a graph

1 Introduction

In this paper, we consider only finite, simple, undirected graphs with no loops and no multiple edges. For a graph G, we denote by $V(G)$ and $E(G)$ the vertex set and the edge set of G, respectively. For a vertex x of a graph G, the neighborhood of x in G is denoted by $N_G(x)$, and we let $d_G(x) := |N_G(x)|$. For a graph G, let $\alpha(G)$ be the independence number of G. Let $\sigma_k(G) :=\min\{\sum_{x\in S} d_G(x)|\ S$ is an independent set of G with $|S| = k\}$; if $\alpha(G) < k$, let $\sigma_k(G) := \infty$. For an integer $n \geq 1$, we let K_n denote the complete graph of order n. In this paper, "disjoint" means "vertex-disjoint".

So far, we have some known results concerning the existence of disjoint cycles in graphs. The following result which was independently obtained by Enomoto[5] and Wang[11] is a basic result in this thesis.

Theorem 1 ([5], [11]) *Let k be an integer with $k \geq 2$. Let G be a graph of order at least $3k$ with $\sigma_2(G) \geq 4k-1$. Then G contains k disjoint cycles.*

Recently, the author, Matsumura, Tsugaki and Yamashita[8] obtained the following result which is the σ_3-version of Theorem 1.

Theorem 2 ([8]) *Let k be an integer with $k \geq 2$. Let G be a graph of order at least $3k+2$ with $\sigma_3(G) \geq 6k-2$. Then G contains k disjoint cycles.*

Since now we have above results whose conclusions are "G contains k disjoint cycles", it would be an interesting thesis for us to consider the problem concerning covering vertices as many as possible by k disjoint cycles as a next step. As for this topic, Egawa, Hagita, Kawarabayashi and Wang[3] obtained the following result:

Theorem 3 ([3]) *Let k,d,n be integers with $k \geq 3, d \geq 4k-1$ and $n \geq 3k$. Let G be a graph of order n, and suppose that $\sigma_2(G) \geq d$. Then G contains k disjoint cycles covering at least $\min\{d,n\}$ vertices of G.*

In fact, Theorem 3 holds for $k = 2$ as well. Recently, Egawa, the author, Kawarabayashi and Wang[2] proved it.

Theorem 4 ([2]) *Let d,n be integers with $d \geq 7$ and $n \geq 6$. Let G be a graph of order n, and suppose that $\sigma_2(G) \geq d$. Then G contains two disjoint cycles covering at least $\min\{d,n\}$ vertices of G.*

Proofs of Theorems 3 and 4 are essentially different. The following theorem is a result concerning a sufficient condition for the existence of a specified number of disjoint cycles covering all vertices. This was given by S.Brandt, G.Chen, R.Faudree, R.J.Gould and L.Lesniak in [1].

Theorem 5 ([1]) *Let k,n be integers with $n \geq 4k-1$. Let G be a graph of order n, and suppose that $\sigma_2(G) \geq n$. Then G contains k disjoint cycles H_i, $1 \leq i \leq k$, such that $V(H_1) \cup \ldots \cup V(H_k) = V(G)$.*

The next result which was obtained by Enomoto and Li[6] asserts that if we regard K_1 and K_2 as degenerated cycles, the assumption of Theorem 5 can be weakened without an exceptional graph.

Theorem 6 ([6]) *Let k,n be positive integers with $n \geq k$. Let G be a graph of order n, and suppose that $\sigma_2(G) \geq n-k+1$. Then unless $k = 2$ and G is a cycle of length 5, G contains k disjoint subgraphs H_i, $1 \leq i \leq k$, such that $V(H_1) \cup \ldots \cup V(H_k) = V(G)$ and such that for each $1 \leq i \leq k$, H_i is either a cycle or isomorphic to K_1 or K_2.*

Enomoto[4] conjectured that G can be partitioned into cycles and isolated vertices if n is sufficiently large compared with k. Actually there are several results which justify this conjecture.

Theorem 7 (Kawarabayashi [10]) *Let k,n be integers with $k \geq 2$ and $n \geq 4k$. Let G be a graph of order n, and suppose that $\sigma_2(G) \geq n-1$. Then one of the following holds:*

(i) *G contains k disjoint cycles H_i, $1 \leq i \leq k$, such that $V(H_1) \cup \ldots \cup V(H_k) = V(G)$;*

(ii) *G has a vertex set $S \subset V(G)$ with $|V(S)| = \frac{n-1}{2}$ such that $G-S$ is independent; or*

(iii) *G is isomorphic to the graph obtained from K_{n-1} by adding a vertex and join it to precisely one vertex of K_{n-1} (i.e., G is isomorphic to $(K_{n-2} \cup K_1)+K_1$).*

Theorem 8 (Hu & Li [9]) *Let k,n be positive integers with $n \geq 10k+3$. Let G be a graph of order n, and suppose that $\sigma_2(G) \geq n-k+1$. Then G contains k disjoint subgraphs H_i, $1 \leq i \leq k$, such that $V(H_1) \cup \ldots \cup V(H_k) = V(G)$ and such that for each $1 \leq i \leq k$, H_i is either a cycle or isomorphic to K_1.*

Theorem 9 *Let k,r,n be integers with $2 \leq r \leq k-2$ and $n \geq 7k$. Let G be a graph of order n, and suppose that $\sigma_2(G) \geq n-r$. Then G contains k disjoint subgraphs H_i, $1 \leq i \leq k$, such that $V(H_1) \cup \ldots \cup V(H_k) = V(G)$ and such that H_i is a cycle or isomorphic to K_1 for each i with $1 \leq i \leq r$, and H_i is a cycle for each i with $r+1 \leq i \leq k$.*

Combining Theorems 5, 7, 8 and 9, we obtain the following corollary:

Corollary *Let k,r,n be integers with $k \geq 2, 0 \leq r \leq k-1$ and $n \geq 10k+3$. Let G be a graph of order n, and suppose that $\sigma_2(G) \geq n-r$. Then G contains k disjoint subgraphs H_i, $1 \leq i \leq k$, such that $V(H_1) \cup \ldots \cup V(H_k) = V(G)$ and such that H_i is a cycle or isomorphic to K_1 for each i with $1 \leq i \leq r$, and H_i is a cycle for each i with $r+1 \leq i \leq k$.*

Theorem 9 is the main result in this paper. We give the outline of the proof of Theorem 9 in the rest of this paper.

2 Preparation of the proof of Theorem 9

Our notation is standard except possibly for the following. Let G be a graph. For a subset L of $V(G)$, the subgraph induced by L is denoted by $\langle L \rangle$. For a subset M of $V(G)$, we let $G-M = \langle V(G)-M \rangle$ and, for a subgraph H of G, we let $G-H = \langle V(G)-V(H) \rangle$. For subsets L and M of $V(G)$, we let $E(L,M)$ denote the set of edges of G joining a vertex in L and a vertex in M. A vertex x is often identified with the set $\{x\}$. Thus if $x \in V(G)$, then $\langle x \rangle$ means $\langle \{x\} \rangle$, $G-x$ means $G-\{x\}$, and $E(x,M)$ means $E(\{x\},M)$ for $M \subset V(G)$. We say that G is pancyclic if $|V(G)| \geq 3$ and G contains a cycle of length l for each l with $3 \leq l \leq |V(G)|$. For a cycle $C = x_1x_2\ldots x_{|V(C)|}x_1$ and for a vertex $x = x_i \in V(C)$, we define $x^{+j} = x_{i+j}$ and $x^{-j} = x_{i-j}$ (indices are to be read modulo $|V(C)|$). Also, we let $x^+ = x^{+1}, x^- = x^{-1}$.

Throughout the rest of this paper, let n,k,r be as in Theorem 9, and let G be a counterexample to Theorem 9. Let $L = \{v \in V(G) |\ d_G(v) < \frac{n-r}{2}\}$. Note that $xy \in E(G)$ for any $x,y \in L$ by the assumption that $\sigma_2(G) \geq n-r$.

Here, we list basic lemmas without proofs.

Lemma 10 *Let r,α be integers with $\alpha \geq r+2 \geq 4$. Let F be a graph of order α, and suppose that $\max\{d_F(x), d_F(y)\} > \lfloor \frac{\alpha}{2} \rfloor$ for any $x,y \in V(F)$ with $x \neq y$ and $xy \notin E(F)$. In the case where $r=2$, suppose further that $|V(F)| \leq 6$. Then F contains r disjoint subgraphs $A_1,\ldots,A_r$ such that $V(A_1) \cup \ldots \cup V(A_r) = V(F)$ and such that for each $1 \leq j \leq r$, A_j is either a cycle or isomorphic to K_1.*

Lemma 11 *In G, there exist $k-r$ disjoint cycles $H_1,\ldots,H_{k-r}$ such that $n-3r \leq |\cup_{i=1}^{k-r} V(H_i)| \leq n-r$.*

Let $H_1,\ldots,H_{k-r}$ be as in Lemma 11. We choose $H_1,\ldots,H_{k-r}$ so that

(a) $|\cup_{i=1}^{k-r} V(H_i)|$ is maximum (subject to the condition that $|\cup_{i=1}^{k-r} V(H_i)| \leq n-r$) and,

subject to condition (a),

(b) $|(\cup_{i=1}^{k-r} V(H_i)) \cap L|$ is maximum

(we make use of (b) only in the proof of Lemma 17).

Let $H = \langle \cup_{i=1}^{k-r} V(H_i) \rangle$ and let $\alpha = |V(G-H)|$. If $\alpha = r$, then $\{H_1,\ldots,H_{k-r}\} \cup \{\langle v \rangle |\ v \in V(G-H)\}$ forms a collection of subgraphs having the properties required in Theorem 9. Thus we may assume $\alpha \geq r+1$.

We use following lemmas in estimating the degree of various vertices.

Lemma 12 *Let $v \in V(G-H)$. Then $|E(v,V(H))| \leq (n-\alpha)/2$.*

Lemma 13 *Let $vv' \in E(G-H)$. Then the following hold.*

(i) $|E(\{v,v'\},V(H))| \leq (2(n-\alpha)+4(k-r))/3$.

(ii) *If $N_{G-H}(v) \cap N_{G-H}(v') \neq \emptyset$, then $|E(\{v,v'\},V(H))| \leq ((n-\alpha)+(k-r))/2$.*

Lemma 14 *Let $v \in V(G-H)$, and let $1 \leq i \leq k-r$. Let $x \in V(H_i)$, and suppose that $N_G(v) \supseteq \{x,x^{+2}\}$. Then $d_H(x^+) \leq (n-\alpha)/2$.*

The following lemma is used when we choose an appropriate vertex in H where degree is to be estimated.

Lemma 15 *Let $v \in V(G-H)-L$ and $v' \in N_{G-H}(v)$, and suppose that either $d_{G-H}(v) \leq \frac{\alpha}{2}$ or $\alpha \leq r+2$. Then for some i with $1 \leq i \leq k-r$, there exists $x \in V(H_i)$ such that $x,x^{+2} \in N_G(v)$, $v,v' \notin N_G(x^+)$ and $|E(x^+,V(G-H))| \leq \frac{\alpha-2}{2}$.*

Besides, we need the following two lemmas in considering the case where $V(G-H) \subseteq L$.

Lemma 16 *Suppose that $\alpha = r+1$ and there exists a triangle T in $G-H$. Let $1 \leq i \leq k-r$ with $|V(H_i)| \geq 4$, and let $x \in V(H_i)$.Then $d_H(x)+d_H(x^+) \leq n-\alpha$.*

Lemma 17 *Suppose that $V(G-H) \subseteq L$, and let $1 \leq i \leq k-r$.*

(i) *If $z \in V(H_i)$ and $E(z,V(G-H)) \neq \emptyset$, then $E(z^{+2},V(G-H)) = \emptyset$.*

(ii) *There exists $x \in V(H_i)$ such that $E(x,V(G-H)) = \emptyset$ and $E(x^+,V(G-H)) = \emptyset$.*

3 Proof of Theorem 9

We continue with the notation of the preceding section, and complete the proof of Theorem 9. We divide the proof into two cases.

Case 1: $V(G-H) \nsubseteq L$

Subcase 1.1. $r+3 \leq \alpha \leq 3r$.

If $d_{G-H}(z) > \alpha/2$ for all $z \in V(G-H)-L$, then by Lemma 10, $G-H$ contains r disjoint subgraphs $A_1,\ldots,A_r$ such that $V(A_1) \cup \ldots \cup V(A_r) = V(G-H)$ and A_j is either a cycle or isomorphic to K_1 for each $1 \leq j \leq r$ (note that we have $|V(G-H)| \leq 3r = 6$ in the case where $r = 2$), and they together with $H_1,\ldots,H_{k-r}$ yield subgraphs with the desired properties. Thus we may assume there exists $v \in V(G-H)-L$ such that $d_{G-H}(v) \leq \alpha/2$. We first consider the case where there exists $v' \in N_{G-H}(v)$ such that $N_{G-H}(v) \cap N_{G-H}(v') \neq \emptyset$. By Lemma 15, there exists a cycle H_i and there exists $x \in V(H_i)$ such that $x,x^{+2} \in N_G(v)$ and $v,v' \notin N_G(x^+)$. Since $\alpha \geq r+3$, we see from the maximality of $|\sum_{j=1}^{k-r} V(H_j)|$ that $N_G(x^+) \cap N_G(v) \cap V(G-H) = \emptyset$ and $N_G(x^+) \cap N_G(v') \cap V(G-H) = \emptyset$, and hence $|N_G(x^+) \cap V(G-H)| + |N_G(v) \cap V(G-H)| \leq \alpha$ and $|N_G(x^+) \cap V(G-H)| + |N_G(v') \cap V(G-H)| \leq \alpha$. Since $|N_G(x^+) \cap V(H)| \leq (n-\alpha)/2$ by Lemma 14 and $|N_G(v) \cap V(H)| + |N_G(v') \cap V(H)| \leq ((n-\alpha)+(k-r))/2$ by Lemma 13(ii), this implies $2d_G(x^+)+d_G(v)+d_G(v') \leq 2\alpha+(n-\alpha)+((n-\alpha)+(k-r))/2 = 3n/2+k/2-r/2+\alpha/2$. On the other hand, since $v,v' \notin N_G(x^+)$, $2d_G(x^+)+d_G(v)+d_G(v') \geq 2n-2r$ by the assumption that $\sigma_2(G) \geq n-r$. Consequently $2n-2r \leq 3n/2+k/2-r/2+\alpha/2$, which implies $n \leq k+3r+\alpha \leq k+6r < 7k$, a contradiction. We now consider the case where $N_{G-H}(v) \cap N_{G-H}(z) = \emptyset$ for every $z \in N_{G-H}(v)$. In this case, we have $|N_G(v) \cap (L-V(H))| \leq 1$ by the fact that $\langle L-V(H) \rangle$ is a complete graph. Since $d_{G-H}(v) = d_G(v) - |N_G(v) \cap V(H)| \geq (n-r)/2-(n-\alpha)/2 > 1$ by Lemma 12 and the assumption of Subcase 1.1, this implies $N_{G-H-L}(v) \neq \emptyset$. Take $v' \in N_{G-H-L}(v)$. Since $N_{G-H}(v) \cap N_{G-H}(v') = \emptyset$, $|N_G(v) \cap V(G-H)| + |N_G(v') \cap V(G-H)| \leq \alpha$. Since $|N_G(v) \cap V(H)| + |N_G(v') \cap V(H)| \leq (2(n-\alpha)+4(k-r))/3$ by Lemma 13(i), this implies $d_G(v)+d_G(v') \leq \alpha+(2(n-\alpha)+4(k-r))/3$. On the other hand, we get $d_G(v)+d_G(v') \geq n-r$ from $v,v' \notin L$. Consequently $n-r \leq 2n/3+4k/3-4r/3+\alpha/3$, which implies $n \leq 4k-r+\alpha \leq 4k+2r < 6k$, a contradiction.

Subcase 1.2. $r+1 \leq \alpha \leq r+2$.

Let $v \in V(G-H)-L$. By Lemma 12, $d_{G-H}(v) = d_G(v) - |N_G(v) \cap V(H)| \geq \frac{n-r}{2} - \frac{n-\alpha}{2} > 0$. Take $v' \in N_{G-H}(v)$. By Lemma 15, we can find a cycle H_i for which there exists $x \in V(H_i)$ such that $x,x^{+2} \in N_G(v)$, $v,v' \notin N_G(x^+)$, and $|N_G(x^+) \cap V(G-H)| \leq \frac{\alpha-2}{2}$. If $N_{G-H}(v) \cap N_{G-H}(v') \neq \emptyset$, then by Lemma 13(ii) and Lemma 14, $2n-2r \leq 2d_G(x^+)+d_G(v)+d_G(v') \leq$

$2(\frac{n-\alpha}{2}+\frac{\alpha-2}{2})+\frac{(n-\alpha)+(k-r)}{2}+2(\alpha-1)$, which implies $n \le k+3r+3\alpha-8 \le k+6r-2 < 7k$, a contradiction. Thus we may assume $N_{G-H}(v) \cap N_{G-H}(v') = \emptyset$. Then $|N_G(v) \cap V(G-H)| + |N_G(v') \cap V(G-H)| \le \alpha$. Hence by Lemma 13(i) and Lemma 14, $2n-2r \le 2d_G(x^+)+d_G(v)+d_G(v') \le 2(\frac{n-\alpha}{2}+\frac{\alpha-2}{2})+\frac{2(n-\alpha)+4(k-r)}{3}+\alpha$, which implies $n \le 4k+2r+\alpha-6 \le 4k+3r-4 < 7k$. This is a contradiction, which completes the discussion for Case 1.

Case 2: $V(G-H) \subseteq L$

In this case, $G-H$ is a complete graph by the definition of L. If $\alpha \ge r+2$, then $G-H$ contains a cycle C of length $\alpha-(r-1) \ge 3$, and hence $\{H_1,\dots,H_{k-r},C\} \cup \{\langle v \rangle |\ v \in V(G-H-C)\}$ forms a collection of desired subgraphs of G. Thus we may assume $\alpha = r+1$. Since $|V(H)| = n-(r+1) > 3k$, there exists a H_i with $|V(H_i)| \ge 4$. By Lemma 17(ii), there exists $x \in V(H_i)$ such that $N_G(x) \subseteq V(H)$ and $N_G(x^+) \subseteq V(H)$. Take $v,v' \in V(G-H)$. Note that $\{v,v'\}$ is contained in a triangle of $G-H$ because $|V(G-H)| = r+1 \ge 3$. Hence by Lemma 16, $d_G(x)+d_G(x^+) = d_H(x)+d_H(x^+) \le n-r-1$. By Lemma 13(ii), we also have $|N_G(v) \cap V(H)| + |N_G(v') \cap V(H)| \le \frac{(n-r-1)+(k-r)}{2}$. Since we clearly have $|N_G(v) \cap V(G-H)| + |N_G(v') \cap V(G-H)| \le 2(|V(G-H)|-1) = 2r$, this implies $d_G(v)+d_G(v') \le \frac{n+k+2r-1}{2}$. Consequently $2n-2r \le d_G(x)+d_G(x^+)+d_G(v)+d_G(v') \le \frac{3n+k-3}{2}$, and we therefore obtain $n \le k+4r-3 < 5k$, which is a contradiction.

This completes the proof of Theorem 9.

References

[1] S.BRANDT, G.CHEN, R.FAUDREE, R.J.GOULD AND L.LESNIAK, Degree conditions for 2-factors, *J.Graph Theory* (1997) **24**

[2] Y.EGAWA, S.FUJITA, K.KAWARABAYASHI AND H.WANG, Existence of two disjoint long cycles in graphs, submitted.

[3] Y.EGAWA, M.HAGITA, K.KAWARABAYASHI AND H.WANG, Covering vertices of a graph by k disjoint cycles, to appear in *Discrete Math.*

[4] H.ENOMOTO, Private communication.

[5] H.ENOMOTO, On the existence of disjoint cycles in a graph, *Combinatorica* (1998) **18**

[6] H.ENOMOTO AND H.LI, Partition of a graph into cycles and degenerated cycles, *Discrete Math.* (2004) **276**

[7] S.FUJITA, Partition of a graph into cycles and isolated vertices, to appear in *Australas. J. Combinatorics*

[8] S.FUJITA, H.MATSUMURA, M. TSUGAKI AND T.YAMASHITA, Degree sum condisions and vertex-disjoint cycles in a graph, submitted.

[9] Z.HU AND H.LI, Weak cycle partition involving degree sum conditions, preprint.

[10] K.KAWARABAYASHI, Degree sum conditions and graphs which are not covered by k cycles, to appear in *Discrete Math.*

[11] H.WANG, On the maximum number of independent cycles in a graph, *Discrete Math.* (1999) **205**

Edge packing problem with edge capacity constraints

TAKURO FUKUNAGA

Department of
Applied Mathematics and Physics,
Graduate School of Informatics,
Kyoto University, Japan
e-mail takuro@amp.i.kyoto-u.ac.jp

HIROSHI NAGAMOCHI*

Department of
Applied Mathematics and Physics,
Graduate School of Informatics,
Kyoto University, Japan
e-mail nag@amp.i.kyoto-u.ac.jp

Abstract: The edge dominating set problem is one of the fundamental covering problems in the field of combinatorial optimization. In this paper, we consider a packing version of this problem. Given a graph with an edge cost, the problem asks to find a maximum cost edge set such that each edge in the graph is adjacent to at most one edge in the set. We show inapproximability of the problem by a reduction from the independent set problem. We also consider a capacitated case of this problem and propose an approximation algorithm based on a relation between two polytopes for an LP relaxation of the problem and the capacitated b-matching problem.

Keywords: approximation algorithm, edge dominating set, packing, polytopes

1 Introduction

Let $G = (V,E)$ be a simple undirected graph. We say that an edge $e = (u,v)$ *dominates* an edge incident to u or v, and define an *edge dominating set* to be a set F of edges such that each edge in E is dominated by at least one edge in F. Given a cost vector $w \in \mathbb{Q}_+^E$ together with G, the *edge dominating set problem* asks to find an edge dominating set with minimum cost. The problem with a cost vector w with $w(e) = 1$, $e \in E$ is called the uniform-cost case, or the cardinality case; otherwise the problem is called the cost case. The edge dominating set problem is one of the fundamental covering problems in the field of combinatorial optimization and has some useful applications [14].

It is known that the cardinality case of the edge dominating set problem is NP-complete even for some restricted classes of graphs such as planar or bipartite graphs of maximum degree 3 [14]. In addition, it is proven that the cardinality case of this problem is hard to approximate within a factor of $\frac{7}{6} - \delta$ unless P=NP [3]. Moreover, an arbitrary algorithm that outputs a maximal matching is a 2-approximation algorithm for the cardinality case of the edge dominating set problem [2][10].

We also know that the cost case of the edge dominating set is approximable within factor of $2r$ if there is an r-approximation algorithm for the minimum cost vertex cover problem [2], where currently $r \leq 2$ is known. Carr et al. [2] presented a 2.1-approximation algorithm for the minimum cost edge dominating set problem. This algorithm constructs an instance of the minimum cost *edge cover problem* from the original instance and finds an optimal edge cover for the resulting instance. A key property for this method is that an edge cover in the resulting instance is also an edge dominating set for the original instance and that its cost is at most 2.1 times of the minimum cost of an edge dominating set in the original instance. The property is proved based on a relation between the fractional edge dominating set polyhedron and the edge cover polyhedron. The former is a polyhedron containing all incidence vectors of edge dominating sets, which may not be the convex hull of these vectors. In contrast the edge cover polyhedron is the convex hull of all incidence vectors of edge covers, which is shown to be an integer polyhedron [12]. Afterward by using a refined edge dominating set polyhedron, Fujito and Nagamochi [6] gave a 2-approximation algorithm to the edge dominating set problem. Moreover, the capacitated version of the problem is considered in [7], where a $\frac{8}{3}$-approximation algorithm is proposed.

As we have described above, the edge dominating set problem is well studied in the relation to the edge cover problem. However, a packing version of the edge dominating set problem has not been studied from the algorithmic point of view so far, especially in the relation to the matching problem, which is considered as a packing version of the edge cover problem. In this paper, we introduce a packing version of the edge dominating set problem, and investigate its approximability. For an edge e, let $\delta(e)$ denote an edge set $\{e' \in E \mid e \cap e' \neq \phi\}$, i.e., a set of edges adjacent to e (including e itself). An *edge packing* is defined as an edge set such that at most one edge is allowed to be chosen from $\delta(e)$ for each edge $e \in E$. The objective of the *edge packing problem* is to find a maximum cost edge packing in a given graph. We show that the edge packing problem is NP-complete and is hard to approximate within a constant factor even for its cardinality case by a reduction from the independent set problem. We also discuss a capacitated case of the edge packing problem, which we call the (b,c)-*edge packing problem* for two given capacity functions $b,c \in \mathbb{Z}_+^E$, where $\mathbb{Z}_+$ denotes the set of nonnegative integers. In this problem, we are allowed

*This research was partially supported by the Scientifi c Grant-in-Aid from Ministry of Education, Science, Sports and Culture of Japan.

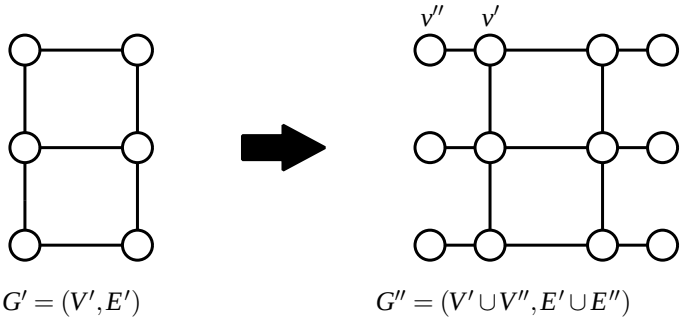


Figure 1: Reduction from the independent set problem to the edge packing problem

to choose multiple copies of edges. A component of b corresponding to edge e denotes an upper bound on the number of chosen edges in $\delta(e)$ and a component of c corresponding to e denotes an upper bound on the the allowed number of multiple edges between two end vertices of e. If $b(e) = 1$ for all $e \in E$, then the (b,c)-edge packing problem is equivalent to the edge packing problem. Given an instance of the edge packing problem, our approximation algorithm constructs an instance of the capacitated b-matching problem and finds its optimal solution as an approximation solution to the original instance. We analyze the approximation guarantee of our algorithm under the assumption that $\beta = \min\{b(e) | \ e \in E\}$ is at least 2 because of its approximation hardness. When $\beta = 2$, we prove that the approximation factor of our algorithm is $\frac{2}{9}$ based on a relation of two polytopes for an LP relaxation of the problem and the capacitated b-matching problem.

The paper is organized as follows. Section 2 introduces notations used in this paper. Section 3 describes a formulation of the edge packing problem and derives a reduction from the independent set problem to the edge packing problem. Section 4 proposes an approximation algorithm to the (b,c)-edge packing problem and analyzes its approximation factor.

2 Preliminaries

Let $\mathbb{Z}$, $\mathbb{Q}$ and $\mathbb{R}$ denote the sets of integers, rational numbers and real numbers, respectively, and $\mathbb{Z}_+$, $\mathbb{Q}_+$ and $\mathbb{R}_+$ denote the sets of nonnegative numbers in $\mathbb{Z}$, $\mathbb{Q}$ and $\mathbb{R}$, respectively.

Let $G = (V,E)$ denote a simple undirected graph with a vertex set V and an edge set E. An edge $e = (u,v) \in E$ in G is defined as a pair of distinct vertices u and v. For a vertex $v \in V$, $\delta(v)$ denotes the set of edges incident to v. For an edge $e = (u,v) \in E$, $\delta(e)$ denotes the set of edges incident to u or v, i.e., $\delta(e)\{e' \in E \mid e \cap e' \neq \phi\}$. For a subset $S \subseteq V$, $\delta(S)$ denotes the set of edges $e = (u,v)$ with $u \in S$ and $v \in V - S$, and $E[S]$ denotes the set of edges contained by S, i.e., $E[S] = \{e \in E \mid e \subseteq S\}$. Let x be an $|E|$-dimensional real vector, i.e., $x \in \mathbb{R}^E$. Then we indicate the entry in x corresponding to an edge e by $x(e)$. For a subset F of E, we denote $x(F) = \sum_{e \in F} x(e)$.

3 Problems

This section introduces the edge packing problem and its capacitated version.

3.1 The edge packing problem

Let us suppose that an undirected graph $G = (V,E)$ and a cost vector $w \in \mathbb{Q}_+^E$ are given. An *edge packing* is defined as an edge set such that at most one edge is allowed to be chosen from $\delta(e)$ for each edge $e \in E$. The *edge packing problem* asks to find a maximum cost edge packing in G, which is formulated as the following linear integer programming problem.

$$\begin{array}{lll} \text{maximize} & w^T x & \\ \text{subject to} & x(\delta(e)) \leq 1 & \text{for each } e \in E, \\ & x \in \{0,1\}^E. & \end{array} \tag{1}$$

We first show that there is a polynomial reduction from the independent set problem [10] to the edge packing problem (see Figure 1). An *independent set* is a set of vertices which contains no pairs of adjacent vertices. The *independent set problem* asks to find a maximum cost independent set for a given graph with costs on vertices. Let (G', w') be an instance of the independent set problem which consists of a graph $G' = (V', E')$ and a cost vector $w' \in \mathbb{Q}_+^{V'}$. We construct an instance (G'', w'') of the edge packing problem from (G', w') as follows. Let V'' be a copy of V', where $v'' \in V''$ denotes the copy of

$v' \in V'$, and let $E'' = \{(v', v'') \mid v' \in V', v'' \in V''\}$, defining a cost vector $w'' \in \mathbb{Q}_+^{E' \cup E''}$ by

$$w''(e) = \begin{cases} 0 & \text{if } e \in E', \\ w'(v') & \text{if } e = (v', v'') \in E''. \end{cases}$$

In the resulting instance (G'', w''), we can assume without loss of generality that a maximum edge packing contains no edges in E' because discarding any edges in E' from an edge packing does not decrease the cost of the edge packing. Then the set of vertices v' such that (v', v'') is contained in such an edge packing is an independent set for the given instance (G', w'). Conversely, given an independent set in (G', w'), we can construct an edge packing for (G'', w'') with the same cost, by selecting each edge $(v', v'') \in E''$ such that $v' \in V'$ belongs to the independent set. Therefore this transformation reduces the independent set problem to the edge packing problem. Moreover this reduction preserves the approximation factor.

In addition, the way of constructing G'' also gives a reduction from the cardinality case of the independent set problem to the cardinality case of the edge packing problem. Suppose that an edge packing for G'' contains an edge e in E'. After replacing such an edge e with an edge in E'' adjacent to e, the resulting edge set remains to be an edge packing with the same number of edges. Then the resulting edge set gives an independent set for G' in the same way as the above and vice versa.

The independent set problem is known to be NP-complete [8] and admits no approximation algorithm whose approximation factor is better than $|V|^{\frac{1}{2}-\varepsilon}$ for any $\varepsilon > 0$ if P $\neq$ NP [11]. Furthermore, it is not approximable within $|V|^{1-\varepsilon}$ for any $\varepsilon > 0$ if P $\neq$ NP and NP $\neq$ ZPP [11], where ZPP denotes a class of languages for which a membership computation by a probabilistic Turing machine halts in polynomial time with no false acceptances or rejections, but randomly halts with some "I don't know" answers. As the above reductions preserve the approximation guarantee, these hardness can be carried out to the edge packing problem.

Theorem 1 *If P $\neq$ NP, the edge packing problem is not approximable within $|V|^{\frac{1}{2}-\varepsilon}$ for any $\varepsilon > 0$. In addition, if* NP $\neq$ ZPP, *then it is not approximable within $|V|^{1-\varepsilon}$.*

3.2 The (b, c)-edge packing problem

We now generalize the edge packing problem so that multiple edges between two vertices are allowed to be chosen, introducing two capacity functions $b, c \in \mathbb{Z}_+^E$ for a graph $G = (V, E)$. Let $b(e)$ be an upper bound on the number of edges dominating e and $c(e)$ be an upper bound on the number of multiple edges between the two end vertices of e. The objective is to maximize the sum of costs of chosen edges. Formally the problem is described as follows.

$$\begin{array}{lll} \text{maximize} & w^T x & \\ \text{subject to} & x(\delta(e)) \leq b(e) & \text{for each } e \in E, \\ & x(e) \leq c(e) & \text{for each } e \in E, \\ & x \in \mathbb{Z}_+^E. & \end{array} \tag{2}$$

We call a feasible solution of this problem by a (b, c)-edge packing and this problem by the (b, c)-*edge packing problem*.

Let EP(G, b, c) be the set of vectors $x \in \mathbb{R}_+^E$ such that

$$\begin{array}{lll} \text{(a)} & 0 \leq x(e) \leq c(e) & \text{for each } e \in E, \\ \text{(b)} & x(\delta(e)) \leq b(e) & \text{for each } e \in E. \end{array}$$

Observe that EP(G, b, c) represents the feasible region of the linear programming problem obtained from problem (2) by relaxing its integer constraints. Although EP(G, b, c) contains all feasible solutions of (2), the set of all optimal solutions over the region may not include any integer solutions.

4 The approximation algorithm

To approximate the (b, c)-edge packing problem, we consider the packing version of the edge cover. The matching problem in a graph is one of the well-studied problems in the combinatorial optimization. This problem is generalized into the following capacitated $\tilde{b}$-matching problem.

$$\begin{array}{lll} \text{maximize} & w^T x & \\ \text{subject to} & x(\delta(v)) \leq \tilde{b}(v) & \text{for each } v \in E, \\ & x(e) \leq c(e) & \text{for each } e \in E, \\ & x \in \mathbb{Z}_+^E, & \end{array} \tag{3}$$

where $\tilde{b} \in \mathbb{Z}_+^V$ and $c \in \mathbb{Z}_+^E$ are given capacities. We call the capacitated $\tilde{b}$-matching problem with capacities $\tilde{b}$ and c the $(\tilde{b}, c)$-*matching problem* and its feasible solution $(\tilde{b}, c)$-*matching*.

It is known that the (b,c)-matching problem can be solved in strongly polynomial time [1][12]. Let $\mathrm{MA}(G,\tilde{b},c)$ be the set of vectors $x \in \mathbb{R}^E_+$ such that

$$\begin{array}{lll} \text{(c)} & 0 \le x(e) \le c(e) & \text{for each } e \in E, \\ \text{(d)} & 0 \le x(\delta(v)) \le \tilde{b}(v) & \text{for each } v \in V, \\ \text{(e)} & x(E[U]) + x(F) \le \left\lfloor \frac{\tilde{b}(U)+c(F)}{2} \right\rfloor & \text{for each } U \subseteq V,\ F \subseteq \delta(U) \\ & & \text{with } \tilde{b}(U)+c(F) \text{ odd.} \end{array}$$

$\mathrm{MA}(G,\tilde{b},c)$ is an integer polytope, whose all extreme points are represented by integer vectors [13]. Since every integer vector satisfying conditions (c) and (d) is a $(\tilde{b},c)$-matching in G, maximizing $w^T x$ over the polytope $\mathrm{MA}(G,\tilde{b},c)$ is essentially equivalent to solving problem (3). Note that, in the (b,c)-edge packing problem, b is an $|E|$-dimensional vector and $b(e)$ is a constraint on $x(\delta(e))$, while $\tilde{b}$ is defined as a $|V|$-dimensional vector and $\tilde{b}(v)$ is a constraint on $x(\delta(v))$ in the $(\tilde{b},c)$-matching problem. To construct an approximate solution to a given instance (G,b,c) of the (b,c)-edge packing problem, we solve an instance $(G,\tilde{b},c)$ of the $(\tilde{b},c)$-matching problem. The capacity vector $\tilde{b}$ will be defined so that a $(\tilde{b},c)$-matching is also a (b,c)-edge packing in G. The algorithm is described as follows.

Algorithm PACK

Input: An undirected graph $G=(V,E)$, capacity functions $b,c \in \mathbb{Z}^E_+$, and a cost vector $w \in \mathbb{Q}^E_+$.

Output: A (b,c)-edge packing.

Step 1: For each $e=(u,v) \in E$, $b'(e,u) := \left\lfloor \frac{b(e)}{2} \right\rfloor$ and $b'(e,v) := \left\lceil \frac{b(e)}{2} \right\rceil$.

Step 2: For each $v \in V$, $\tilde{b}(v) := \min_{e \in \delta(v)} b'(e,v)$.

Step 3: Compute a maximum cost $(\tilde{b},c)$-matching $\bar{x} \in \mathbb{Z}^E_+$ for the graph G and the cost vector w, and output $\bar{x}$ as a (b,c)-edge packing.

Integer vectors $x \in \mathbb{Z}^E$ satisfying (c) and (d) of $\mathrm{MA}(G,\tilde{b},c)$ are (b,c)-edge packings because $x(\delta(e)) \le x(\delta(u)) + x(\delta(v)) \le \tilde{b}(u) + \tilde{b}(v) \le b(e)$, In the following, we analyze the approximation factor of algorithm PACK.

Lemma 2 *Let $x \in \mathrm{EP}(G,b,c)$, $\beta = \min_{e \in E} b(e)$, and $\tilde{b} \in \mathbb{Z}^V_+$ be a vector obtained in Step 2 of algorithm* PACK. *Then vector*

$$\frac{1}{2}\left(1 - \frac{1}{2\lfloor \beta/2 \rfloor + 1}\right) x$$

satisfies conditions (c) *and* (d) *for* $\mathrm{MA}(G,\tilde{b},c)$.

PROOF: Let $x' = \frac{1}{2}\left(1 - \frac{1}{2\lfloor \beta/2 \rfloor + 1}\right) x$. Since $x \in \mathrm{EP}(G,b,c)$ satisfies $0 \le x(e) \le c(e)$ for each $e \in E$, it is immediate to see that x' satisfies (c) for $\mathrm{MA}(G,\tilde{b},c)$. Then, we show that $x'(\delta(v)) \le \tilde{b}(v)$ holds for each $v \in V$.

Let $v \in V$. There is an edge $e=(u,v) \in E$ such that $\tilde{b}(v) = b'(e,v)$. Note that $x(\delta(v)) \le x(\delta(e)) \le b(e)$ hold by (b) for $\mathrm{EDS}(G,b,c)$. If $b'(e,v) = \left\lceil \frac{b(e)}{2} \right\rceil$, then it holds

$$x'(\delta(v)) \le \frac{1}{2} x(\delta(v)) \le \frac{x(\delta(e))}{2} \le \frac{b(e)}{2} \le \left\lceil \frac{b(e)}{2} \right\rceil = \tilde{b}(v),$$

This implies that $x'(\delta(v))$ satisfies (d) in $\mathrm{MA}(G,\tilde{b},c)$,

Consider the other case, $b'(e,v) < \left\lceil \frac{b(e)}{2} \right\rceil$, i.e., $b(e)$ is odd and $b'(e,v) = \left\lfloor \frac{b(e)}{2} \right\rfloor$. Since $x(\delta(v)) \le b(e)$ and $\tilde{b}(v) = b'(e,v) = \frac{b(e)-1}{2}$, we have

$$\frac{\tilde{b}(v)}{x(\delta(v))} \ge \frac{b(e)-1}{2b(uv)} = \frac{1}{2} - \frac{1}{2b(e)}.$$

From the assumption, $b(e) \ge \beta$. In addition, we have $b(e) \ge \beta + 1$ if β is even because $b(e)$ is assumed to be odd. These two inequality can be represented by $b(e) \ge 2\lfloor \beta/2 \rfloor + 1$. Then

$$\frac{1}{2} - \frac{1}{2b(e)} \ge \frac{1}{2}\left(1 - \frac{1}{2\lfloor \beta/2 \rfloor + 1}\right).$$

Therefore $x'(\delta(v))$ satisfies (d) for $\mathrm{MA}(G,\tilde{b},c)$. □

Lemma 3 *Let $x \in \mathbb{R}_+^E$ satisfy* (c) *and* (d) *for* MA$(G, \tilde{b}, c)$ *and* $\beta = \min_{e\in E} b(e)$. *Then vector*

$$\left(1 - \frac{1}{2\lfloor 3\beta'/2 \rfloor + 1}\right) x$$

satisfies (e) *for* MA$(G,\tilde{b},c)$, *where* $\beta' = \left\lfloor \frac{\beta}{2} \right\rfloor$.

PROOF: Let U be a nonempty subset of V, and F be a subset of $\delta(U)$ which can be empty. It suffices to show that

$$x(E[U]) + x(F) \leq \left\lfloor \frac{\tilde{b}(U) + c(F)}{2} \right\rfloor.$$

Since x satisfies (d) for MA$(G, \tilde{b}, c)$, it holds

$$2x(E[U]) + x(\delta(U)) = \sum_{v\in U} x(\delta(v)) \leq \sum_{v\in U} \tilde{b}(v) = \tilde{b}(U),$$

from which we have

$$x(E[U]) \leq \frac{\tilde{b}(U) - x(\delta(U))}{2} \tag{4}$$

From (c), $x(F) = \sum_{e\in F} x(e) \leq \sum_{e\in F} c(e) = c(F)$ holds. From this inequality and (4), we have

$$x(E[U]) + x(F) \leq \frac{\tilde{b}(U) + c(F) - (x(\delta(U)) - x(F))}{2}.$$

Since $x(\delta(U)) - x(F) \geq 0$ holds by $F \subseteq \delta(U)$, it holds

$$x(E[U]) + x(F) \leq \frac{\tilde{b}(U) + c(F)}{2}. \tag{5}$$

The gap between $\frac{\tilde{b}(U)+c(F)}{2}$ and $\left\lfloor \frac{\tilde{b}(U)+c(F)}{2} \right\rfloor$ depends on the parity of $\tilde{b}(U) + c(F)$. Therefore we only have to consider the case where $\tilde{b}(U) + c(F)$ takes a minimum odd value. We consider the following three subcases.

Case 1: $|U| = 1$. Let $U = \{v\}$. Then $x(E[U]) = 0$. Therefore the left hand side of (e) equals to $x(F)$. Since $\tilde{b}(U) + c(F) = \tilde{b}(v) + c(F)$ is assumed to be odd, it holds $\tilde{b}(v) \neq c(F)$, which implies $\tilde{b}(u) + c(F) \geq 2\min\{\tilde{b}(u), c(F)\} + 1$. From (c), $x(F) \leq c(F)$ holds. Moreover, $x(F) \leq x(\delta(v)) \leq \tilde{b}(v)$ holds since $F \subseteq \delta(v)$. Therefore we have

$$x(F) \leq \min\{c(F), \tilde{b}(v)\} \leq \frac{\tilde{b}(U) + c(F) - 1}{2} = \left\lfloor \frac{\tilde{b}(U) + c(F)}{2} \right\rfloor.$$

Case 2: $|U| = 2$. Let $U = \{v_1, v_2\}$, $F_1 = \delta(v_1) \cap F$, and $F_2 = \delta(v_2) \cap F$. Then $\tilde{b}(U) + c(F) = \tilde{b}(v_1) + \tilde{b}(v_2) + c(F_1) + c(F_2)$. From the facts that $\delta(v_1) \cup F_2 \supseteq E[U] \cup F$ and that $\delta(v_2) \cup F_1 \supseteq E[U] \cup F$, we have

$$x(E[U]) + x(F) \leq \min\{x(\delta(v_1)) + x(F_2), x(\delta(v_2)) + x(F_1)\}. \tag{6}$$

It holds that $x(\delta(v_1)) \leq \tilde{b}(v_1)$ and $x(\delta(v_2)) \leq \tilde{b}(v_2)$ from (d). Moreover, we have $x(F_1) \leq c(F_1)$ and $x(F_2) \leq c(F_2)$ from (c). These relations and inequality (6) lead to

$$x(E[U]) + x(F) \leq \min\{\tilde{b}(\delta(v_1)) + c(F_2), \tilde{b}(\delta(v_2)) + c(F_1)\}. \tag{7}$$

On the other hand, since $\tilde{b}(U) + c(F)$ is assumed to be odd, it holds $\tilde{b}(\delta(v_1)) + c(F_2) \neq \tilde{b}(\delta(v_2)) + c(F_1)$, which implies that

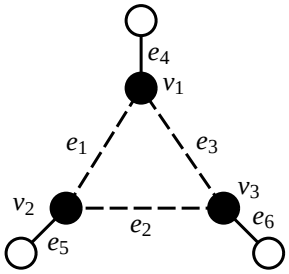
$$\min\{\tilde{b}(\delta(v_1)) + c(F_2), \tilde{b}(\delta(v_2)) + c(F_1)\} \leq \left\lfloor \frac{\tilde{b}(U) + c(F)}{2} \right\rfloor. \tag{8}$$

From (7) and (8), we have (e) for MA$(G, \tilde{b}, c)$.

Case 3: $|U| \geq 3$. Since $b(e) \geq \beta$ for all $e \in E$, it holds $\tilde{b}(v) \geq \left\lfloor \frac{\beta}{2} \right\rfloor$ for all $v \in V$. Hence $\tilde{b}(U) \geq 3\left\lfloor \frac{\beta}{2} \right\rfloor$. Considering that $\tilde{b}(U) + c(F)$ is odd, we have

$$\tilde{b}(U) + c(F) \geq 2\left\lfloor \frac{3\beta'}{2} \right\rfloor + 1,$$

where $\beta' = \left\lfloor \frac{\beta}{2} \right\rfloor$. From (5) and the above inequality,



$$b(e_1) = b(e_2) = b(e_3) = \infty$$
$$b(e_4) = b(e_5) = b(e_6) = 3$$
$$\tilde{b}(v_1) = \tilde{b}(v_2) = \tilde{b}(v_3) = 1$$
$$x^*(e_1) = x^*(e_2) = x^*(e_3) = \tfrac{3}{2}$$

Figure 2: An tight example for the analysis in Corollary 6

$$\frac{\left\lfloor \left(\tilde{b}(U) + c(F)\right)/2 \right\rfloor}{x(E[U]) + x(F)} \geq 1 - \frac{1}{\tilde{b}(U) + c(F)} \geq 1 - \frac{1}{2\lfloor 3\beta'/2 \rfloor + 1}.$$

This completes the proof of the lemma. □

Theorem 4 *Let $\beta = \min_{e\in E} b(e)$ and $\tilde{b}$ be a vector constructed in Step 2 of algorithm* PACK. *Then $MA(G,\tilde{b},c)$ is a polyhedron whose maximum cost extreme points are $f(\beta)$-approximate solutions of the (b,c)-edge packing problem for a graph G, where*

$$f(\beta) = \frac{1}{2}\left(1 - \frac{1}{2\beta'+1}\right) \cdot \left(1 - \frac{1}{2\lfloor 3\beta'/2 \rfloor + 1}\right)$$

and $\beta' = \left\lfloor \frac{\beta}{2} \right\rfloor$.

PROOF: We have already observed that an integer vector $x \in \mathbb{Z}_+^E$ in $\mathrm{MA}(G,\tilde{b},c)$ is a (b,c)-edge packing. Since $\mathrm{MA}(G,\tilde{b},c)$ is an integer polyhedron, every extreme point is a (b,c)-edge packing.

Denote by OPT the maximum cost of a solution to problem (2) with a graph $G = (V,E)$, a cost vector w, and capacity functions b and c. In what follows, we show that $f(\beta)\mathrm{OPT} \leq w^T \bar{x}$ holds, where $\bar{x}$ is a maximum cost extreme point of $\mathrm{MA}(G,\tilde{b},c)$. Let $x^* \in \mathrm{EP}(G,b,c)$ be a vector of the maximum cost $w^T x^*$ for the cost $w \in \mathbb{Q}_+^E$. Because $\mathrm{EP}(G,b,c)$ contains an optimal solution to problem (2), it holds

$$\mathrm{OPT} \leq w^T x^*.$$

By Lemmas 2 and 3, we can see that vector $f(\beta)x^*$ belongs to $\mathrm{MA}(G,\tilde{b},c)$. By the maximality of $w^T \bar{x}$ over $\mathrm{MA}(G\tilde{b},c)$, it holds

$$f(\beta) w^T x^* \leq w^T \bar{x}.$$

From the above two inequalities, we have

$$f(\beta)\mathrm{OPT} \leq w^T \bar{x},$$

as required. □

The above theorem is equivalent to saying that the approximation factor of algorithm PACK is $f(\beta)$ because algorithm PACK outputs a maximum cost vector over the polyhedron $\mathrm{MA}(G,\tilde{b},c)$.

Corollary 5 *Let $\beta = \min_{e\in E} b(e)$. Then the approximation factor of algorithm* PACK *is $f(\beta)$.*

Note that $f(\beta) = 0$ for $\beta \leq 1$. It would be appropriate to assume that $\beta \geq 2$ because of the approximation hardness stated in Section 3. In particular, for $\beta = 2$, $f(\beta) = \frac{2}{9}$ holds.

Corollary 6 *If $b(e) \geq 2$ for all $e \in E$, then algorithm* PACK *achieves an approximation factor of $\frac{2}{9}$.*

Figure 2 shows a tight example for the above analysis in the case of $\beta = 2$. The example consists of a graph $G = (V,E)$, a cost vector w, and capacity functions $b, c \in \mathbb{Z}_+^E$. The vertex set V consists of six vertices. Three vertices in V, called v_1, v_2, v_3 which are represented by black circles and the other vertices are represented by white circles. The edge set E consists of six edges e_1, e_2, e_3 (represented by dotted lines) and e_4, e_5, e_6 (represented by solid lines). Capacity $b(e_i)$ is set to be

$$b(e_i) = \begin{cases} \infty & \text{for } i = 1,2,3, \\ 3 & \text{for } i = 4,5,6. \end{cases}$$

Let $c(e_i) = \infty$ for $i = 1, \ldots, 6$. Let x^* be a vector maximizing $w^T x^*$ over EP(G, b, c). If the value of $w(e_1) = w(e_2) = w(e_3)$ is enough large, then

$$x^*(e_i) = \begin{cases} \frac{3}{2} & \text{for } i = 1, 2, 3, \\ 0 & \text{for } i = 4, 5, 6. \end{cases}$$

Algorithm PACK computes $\tilde{b}(v_i) = 1$ for $i = 1, 2, 3$ and $\tilde{b}(v_i) = 0$ for $i = 4, 5, 6$ after Step 2. For the resulting instance $(G, \tilde{b}, c)$, we need to multiply x^* by $\frac{2}{9}$ in order to satisfy (e) of MA$(G, \tilde{b}, c)$ for $U = \{v_1, v_2, v_3\}$ and $F = \phi$.

5 Conclusion

We have shown that the edge packing problem is NP-hard and that for any $\varepsilon > 0$, it is inapproximable within $|V|^{\frac{1}{2}-\varepsilon}$ and $|V|^{1-\varepsilon}$ unless P = NP and P = NP = ZPP, respectively, by using a reduction from the independent set problem.

We have also considered the (b, c)-edge packing problem and proposed an approximation algorithm. The algorithm computes a capacity function $\tilde{b} \in \mathbb{Z}_+^V$ from $b \in \mathbb{Z}_+^E$ and outputs a maximum cost $(\tilde{b}, c)$-matching. Based on a relation of two polytopes for the (b, c)-edge packing and the $(\tilde{b}, c)$-matching problems, we have analyzed the approximation factor of the algorithm in term of $\beta = \min\{b(e) \mid e \in E\}$. We have shown that the approximation factor is $\frac{2}{9}$ when $\beta = 2$.

A remaining problem is whether there exists an algorithm with a better approximation guarantee. The algorithm proposed in this paper constructs $\tilde{b}$ without using any information on optimal solutions x that maximize $w^T x$ over the polytope EP(G, b, c), which might be helpful to improve the current approximation factor. We are also interested in whether there is an approximation algorithm whose factor becomes better than $\frac{1}{2}$ when β is sufficiently large. The approximation factor we have obtained in Theorem 4 is a monotonically increasing function on β. But it does not become larger than $\frac{1}{2}$ however large β gets.

References

[1] R. P. ANSTEE, A polynomial algorithm for b-matchings: an alternative approach, *Information Processing Letters* (1987) **24**, pp. 153–157.

[2] R. CARR, T. FUJITO, G. KONJEVOD, AND O. PAREKH, A $2\frac{1}{10}$-approximation algorithm for a generalization of the weighted edge-dominating set problem, *Proceedings of the Eighth ESA* (2000), pp. 132–142.

[3] M. CHLEBÍK AND J. CHLEBÍKOVÁ, pproximation hardness of minimum edge dominating set and minimum maximal matching, *Proceedings of the 14th Annual International Symposium on Algorithms and Computation (ISAAC2003)* (2003), pp. 415–424.

[4] I. DIRNUR AND S. SAFRA, The importance of being biased, *Proceedings of the 34th annual ACM symposium on Theory of computing* (2002), pp. 33–42.

[5] J. EDMONDS, Maximum matching and a polyhedron with 0,1-vertices, *Journal of research of the National Bureau of Standards B* (1965) **69**, pp. 125–130.

[6] T. FUJITO AND H. NAGAMOCHI, A 2-approximation algorithm for the minimum weight edge dominating set problem, *Discrete Applied Mathematics* (2002) **118**, pp. 199–207.

[7] T. FUKUNAGA AND H. NAGAMOCHI, Approximation algorithms for the b-edge dominating set problem and its related problems, to appear.

[8] M. R. GAREY AND D. S. JOHNSON, Computers and intractability; a guide to the theory of NP-completeness, W. H. Freeman & Co. (1979).

[9] J. HORTON AND K. KILAKOS, Minimum edge dominating sets, *SIAM Journal on Discrete Mathematics* (1993) **6**, pp. 375–387.

[10] F. HARARY, *Graph Theory*, Addison-Wesley (1969).

[11] J. HÅSTAD, Clique is hard to approximate within $n^{1-\varepsilon}$, *Acta Mathematica* (1999) **182**, pp. 105–142.

[12] A. SCHRIJVER, Combinatorial optimization: polyhedra and efficiency, volume A, paths, flows, matchings, chapter 1-38. Springer, 2003.

[13] A. SRINIVASAN, K. MADHUKAR, P. NAGAVAMSI, C. PANDU RANGAN, AND M-S. CHANG, Edge domination on bipartite permutation graphs and contriangulated graphs, *Information Processing Letters* (1995) **56**, pp. 165–171.

[14] M. YANNAKAKIS AND F. GAVRIL, Edge dominating sets in graphs, *SIAM Journal on Applied Mathematics* (1980) **38**, pp. 364–372.

Hajós Calculus on Planar Graphs

YOICHI HANATANI, TAKASHI HORIYAMA, KAZUO IWAMA

Graduate School of Informatics,
Kyoto University,
Kyoto 606-8501, Japan.
{hanatani,horiyama,iwama}@kuis.kyoto-u.ac.jp

Abstract: The Hajós calculus is a nondeterministic procedure which generates the class of non-3-colorable graphs [3]. If all non-3-colorable graphs can be constructed in polynomial steps by the calculus, NP = co-NP holds. Up to date, however, it remains open whether there exists a family of graphs that can be generated in polynomial steps. To attack this problem, we propose two graph calculi $\mathscr{PHC}$ and $\mathscr{PHC}^*$ that generate non-3-colorable planar graphs, where intermediate graphs in the calculi are also restricted to be planar. Then we prove that $\mathscr{PHC}$ and $\mathscr{PHC}^*$ are *sound* and *complete*. We also show that $\mathscr{PHC}^*$ can polynomially simulate $\mathscr{PHC}$.

Keywords: Hajós calculus, Planar graph, Coloring, Proof systems

1 Introduction

Graph k-coloring problem is the problem to decide whether we can assign one of k colors to each vertex so that adjacent pairs of vertices are assigned different colors [15]. This problem is one of the most fundamental NP-complete problems [5, 9]. Even when $k \geq 3$, it is NP-complete. When $k \leq 2$, we can solve the problem in polynomial time. If graphs are restricted to be planar, it is believed for a long time that every graph is 4-colorable [10]. Appel and Haken finally proved the Four-Color Theorem, i.e., every planar graph is 4-colorable [7, 8, 12, 19]. Therefore, when $k \geq 4$, we can decide whether given planar graph is k-colorable in polynomial time. When $k = 3$, the problem is still NP-complete.

In order to characterize k-colorable graphs, many approaches have been attempted. The most typical one is Hadwiger's conjecture to relate the non-k-colorability and the $(k+1)$-cliques [1]. Let k be the fewest number of colors necessary to color vertices in a given graph. Then, we can obtain a k-clique by contracting adjacent vertices. This conjecture is true for $k \leq 5$ [1, 2, 4].

Another approach is the Hajós calculus. The calculus is a nondeterministic procedure that constructs all non-k-colorable graphs from a $(k+1)$-clique [3]. A graph calculus is a collection of initial graphs, together with a finite set of rules which allows us to derive new graphs. A construction of a graph G is a sequence of graphs $(G_1, G_2, \ldots, G_\ell)$ such that the sequence ends with G (i.e., $G_\ell = G$) and every graph in the sequence is one of the initial graphs, or follows from its previous graphs by applying one of the rules.

The complexity of the Hajós calculus was first studied by Mansfield and Welsh [11]: If all non-3-colorable graphs have polynomial size Hajós constructions then, NP = co-NP holds, thus there may exist graphs that cannot be constructed in polynomial steps. A construction of a graph in the Hajós calculus gives the proof of the non-k-colorability of the graph.

Our Contribution: Our motivation is to give intermediate subsystems that are more powerful than bounded-depth Frege system and yet we can prove super-polynomial lower bounds. For this purpose, we consider the calculus on planar graphs, more precisely, the calculus that generates the class of non-3-colorable planar graphs, where intermediate graphs in the calculus are also restricted to be planar. Although the Hajós calculus can generate all non-3-colorable planar graphs, intermediate graphs are not guaranteed to be planar. When restricting the intermediate graphs to be planar, by adding only one new rule, we can obtain a sound and complete calculus $\mathscr{PHC}$. By modifying the second rule (edge elimination rule) in the Hajós calculus, we can obtain another sound and complete calculus $\mathscr{PHC}^*$. We compare the powers of the two calculi.

Previous work: It is known that the Hajós calculus is polynomially bounded if and only if Extended Frege proof systems are polynomially bounded [16]. This result links an open problem in graph theory to an important open problem in the complexity of propositional proof systems: Is there a strong system to produce a short proof of any tautology? As formalized by Cook and Reckhow [6], there exists a propositional proof system giving rise to short (polynomial-size) proofs of all tautologies if and only if NP equals co-NP. Since Extended Frege system is powerful enough that obtaining super-polynomial lower bounds is beyond our current technique, research interests shift into subsystems of the calculus. For example, Aijtai and others showed exponential lower bounds for bounded-depth Frege proofs [13, 14, 18], which lead exponential lower bounds on the subsystems of the Hajós calculus [16, 17].

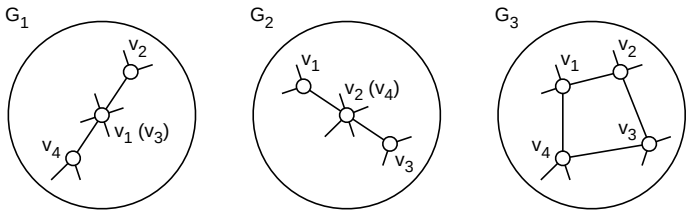


Figure 1: G_1, G_2 and G_3 of Rule 4 in $\mathscr{PHC}$

2 Hajós Calculus

We describe the Hajós calculus for $k = 3$. The set of initial graphs in Hajós calculus contains all graphs isomorphic to complete graph K_4. There are three rules for generating new graphs:

1. **Vertex/edge introduction rule**: Add (any number of) vertices and edges.
2. **Join rule**: Let G_1 and G_2 be disjoint graphs, a_1 and b_1 adjacent vertices in G_1, and a_2 and b_2 adjacent vertices in G_2. Constrct a graph G_3 from $G_1 \cup G_2$ as follows. First, remove edges (a_1, b_1) and (a_2, b_2); then add an edge (b_1, b_2); lastly, contract vertices a_1 and a_2 into a single vertex, named a_1.
3. **Contraction rule**: Contract two nonadjacent vertices into a single vertex, and remove any resulting duplicated edges.

Vertex/edge introduction rule implies that if a subgraph of G has a construction, G also has a construction. Rules 1 and 2 increase vertices and/or edges, but Rule 3 reduces vertices and edges, thus the construction may not be polynomially bounded.

We consider a minor revision of the Hajós calculus, $\mathscr{HC}$. The system $\mathscr{HC}$ has the same set of initial graphs, as well as Rules 1 and 3 of the Hajós calculus, but now Rule 2 of the Hajós calculus is replaced by the following rule:

2. **Edge elimination rule**: Let G_1 and G_2 be two graphs with common vertex set $\{v_1, \ldots, v_n\}$ which are identical except that G_1 contains edges (v_1, v_2) and (v_2, v_3) and not (v_1, v_3), whereas G_2 contains edges (v_1, v_2) and (v_1, v_3) and not (v_2, v_3). Then from G_1 and G_2, we can construct a graph G_3 that is identical to G_1 but does not contain (v_2, v_3).

To associate two calculi, Hajós calculus and $\mathscr{HC}$, we define a binary relation: Let $\mathscr{C}$ and $\mathscr{C}'$ be two graph calculus systems, then $\mathscr{C}$ *p-simulates* $\mathscr{C}'$ if there is a polynomial-time computable function f so that for all graphs G, if s is a graph construction of G in $\mathscr{C}'$, then $f(s)$ is a graph construction of G in $\mathscr{C}$. $\mathscr{C}$ and $\mathscr{C}'$ are *p-equivalent* if $\mathscr{C}$ p-simulates $\mathscr{C}'$ and $\mathscr{C}'$ p-simulates $\mathscr{C}$.

Fact 1 *$\mathscr{HC}$ is p-equivalent to the Hajós calculus.*

3 Planar Calculus $\mathscr{PHC}$

First, we propose planar calculus $\mathscr{PHC}$. The set of initial graphs in $\mathscr{PHC}$ contains all graphs isomorphic to K_4. There are four rules, where Rules 1 to 3 are same as the system $\mathscr{HC}$, but edge addition and vertex contraction are restricted so that the resulting graphs are planar. Rule 4 is as follows:

4. **Quadrilateral rule**: Let G_3 be a graph with vertex set $\{v_1, \ldots, v_n\}$ that contains a face v_1, v_2, v_3, v_4. Let G_1 be a graph obtained by contracting vertices v_1 and v_3 of G_3. Let G_2 be a graph obtained by contracting vertices v_2 and v_4 of G_3. Then from G_1 and G_2, we can construct the graph G_3 (See Figure 1).

Rule 4 is important when given graph consists of only triangle and quadrilateral faces.

For example, we show that the graph G_3 of Figure 2 has a construction in $\mathscr{PHC}$. Let G_1 and G_2 be the graphs shown in Figure 2. G_1 contains K_4 as a subgraph induced by $\{v_1, v_2, v_3, v_5\}$. G_2 also contains K_4 as a subgraph induced by $\{v_1, v_3, v_4, v_5\}$. Therefore G_1 and G_2 can be constructed in $\mathscr{PHC}$. G_1 can be constructed from G_1 and G_2 by Rule 4, since v_1, v_2, v_7, v_6 is a quadrilateral face and G_1 is identical to G_3 with v_2 and v_6 contracted and G_2 is identical to G_3 with v_1 and v_7 contracted.

Since G_3 is edge minimal with respect to the 3-colorability, G_3 cannot be constructed directly by Rule 1. Each face of G_3 is triangle or quadrilateral, thus there is not a triplet of vertices v, v', v'' of satisfying the condition of Rule 2. This means that G_3 cannot be constructed directly by Rule 2. Contraction rule cannot break the structure of non-3-colorablility. Therefore, probably G_3 is an example of graphs that essentially need Rule 4 in $\mathscr{PHC}$.

In the rest of this section, we prove the soundness and the completeness of $\mathscr{PHC}$.

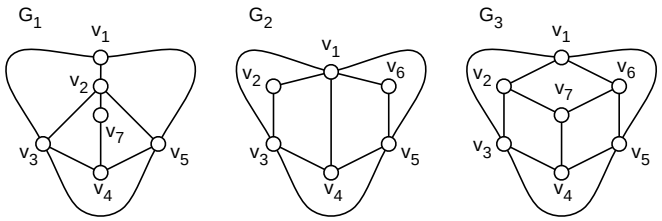


Figure 2: Example of the system $\mathscr{PHC}$

Theorem 2 *$\mathscr{PHC}$ is sound.*

PROOF: We only need to show that Rule 4 is sound since other rules also appear in $\mathscr{HC}$ and are shown to be sound [3]. Assume that there exists a 3-colorable graph G_3 generated by Rule 4. Then, its face v_1, v_2, v_3, v_4 has a coloring satisfying one of the following cases:

Case 1: $\text{color}(v_1) = \text{color}(v_3)$.

Case 2: $\text{color}(v_2) = \text{color}(v_4)$.

Note that, if neither of the cases are satisfied, we have $\text{color}(v_1) \neq \text{color}(v_3)$ and $\text{color}(v_2) \neq \text{color}(v_4)$. In this case, we need more than four colors to the face, which contradicts the 3-colorability of G_3. Cases 1 (Case 2, respectively) implies that G_1 (G_2, respectively) is 3-colorable. Therefore, only non-3-clorable graphs are generated. □

Theorem 3 *$\mathscr{PHC}$ is complete.*

PROOF: We prove this theorem by induction on the size n of the graph. In case $n < 4$, all graphs are colorable by at most 3 colors. In case $n = 4$, K_4 is the initial graph of $\mathscr{PHC}$. Other graphs of size 4 are all 3-colorable, so we do not care them.

Assume that all non-3-colorable graphs of size $n-1$ can be constructed in $\mathscr{PHC}$. We assume that there exists a nonempty set $\mathscr{G}$ of non-3-colorable graphs of size n that cannot be constructed in $\mathscr{PHC}$. Then we lead a contradiction that edge maximal graph $G \in \mathscr{G}$ can be constructed. By considering the size of the faces in G, we have the following three cases.

Case 1: All faces are triangle. According to Theorem 5 (we prove this theorem later), G can be constructed in $\mathscr{PHC}$.

Case 2: There is a face f of size $k \geq 5$. Let $v_1, v_2, v_3, v_4, \ldots, v_k$ be the vertices of face f. $G' = G + (v_1, v_3)$ and $G'' = G + (v_1, v_4)$ can be constructed, since G is a edge maximal graph in $\mathscr{G}$. Therefore we can construct G from G' and G'' applying by Rule 2.

Case 3: G is composed of triangle or quadrilateral faces. Let $f = v_1, v_2, v_3, v_4$ be a quadrilateral face of G. Let G' be a graph obtained by contracting vertices v_1 and v_3 of G. Let G'' be a graph obtained by contracting vertices v_2 and v_4 of G. G' and G'' can be constructed in $\mathscr{PHC}$ because of the assumption. Therefore we can construct G from G' and G'' applying the Rule 4.

In any case, G ($\in \mathscr{G}$) can be constructed in $\mathscr{PHC}$, which contradict to the definition of $\mathscr{G}$. Thus, any non-3-colorable graph can be constructed in $\mathscr{PHC}$. □

Now, we prove that any triangulate non-3-colorable planar graph can be constructed in polynomial number of steps. First we prove the following lemma, which construct an essential component of triangulate planar graphs.

Lemma 4 *Let $G_n = (V, E)$ be a graph of $3n+1$ vertices, where $n \geq 1$,*

$$V = \{a_0\} \cup \{a_i,\ b_i,\ c_i \mid i \in \{1, \ldots, n\}\},$$

$$E = \{(a_0, a_n)\} \cup \{(a_{i-1}, b_i), (a_{i-1}, c_i), (a_i, b_i), (a_i, c_i), (b_i, c_i) \mid i \in \{1, \ldots, n\}\}$$

then, G has a linear size construction in $\mathscr{PHC}$.

PROOF: We prove this lemma by induction on n. In case $n = 1$, the lemma obviously holds since G_1 is isomorphic to an initial graph K_4.

We prove that G_n can be constructed by the assumption that G_{n-1} can be constructed. $G' = G_n + (a_0, a_{n-1})$ can be constructed in $\mathscr{PHC}$ since G_{n-1} is subgraph of G'. $G'' = G_n + (a_{n-1}, a_n)$ can be constructed in $\mathscr{PHC}$ since subgraph of G'' induced by $\{a_{n-1}, a_n, b_n, c_n\}$ is isomorphic to K_4. Therefore we can construct G_n by applying Rule 2 to G' and G''. Since we apply Rule 1 twice and Rule 2 once at each induction step, the whole construction is linearly bounded. □

Theorem 5 *Triangulate non-3-colorable planar graphs have a polynomial size construction in $\mathscr{PHC}$.*

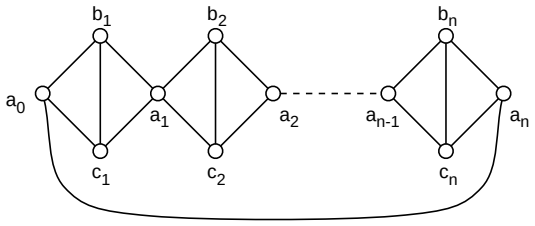


Figure 3: G_n

PROOF: Our goal is to find a structure G_n of Lemma 4 as a subgraph of a given graph G. We try to assign colors to vertices of G. Initially, we choose a triangle face v_1, v_2, v_3 and assign different color to each vertex. color$(v_1) = R$, color$(v_3) = G$ and color$(v_3) = B$. We introduce three trees T_R, T_G, T_B. The root node of each tree is one of the vertices v_1, v_2, v_3. The face that its vertices are already assigned a color is called a colored face. Next, we repeat the following procedure until all vertices are assigned a color or adjacent vertices are assigned the same color. Choose a non-colored triangle face f' adjacent to colored face f. We need not to think about the case that non-colored faces exist but are not adjacent to colored face because the given graph is connected and trianglate. Let v be a vertex that belongs to f and does not belong to f'. Let v' be a vertex that belongs to f' and does not belong to f. Vertices v and v' are uniquely determined. Then we assign $c =$ color(v) to v' and add the vertex v' to the tree T_c as a child node of v. This replication stops before all vertices assigned a color because G is non-3-colorable. When the repetition stops, we find adjacent vertices v' and v'' on G that are assigned the same color c. The tree T_c includes v' and v'' so that there is a path p from v' to v'' in T_c. An Edge $(v_i, v_j) \in T_c$ corresponds to a subgraph of G as the Figure 4. Let G' be a subgraph of G that corresponds to path p (G' of Figure 5 is an example) that corresponds to path p (dotted line of the figure). Let G'' be a graph $G_{|p|}$ of Lemma 4. G'' can be constructed because of Lemma 4. G' can be constructed from G'' with some vertices contracted. G can be constructed from G' by Rule 1. Therefore G has a construction in $\mathscr{PHC}$. □

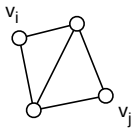


Figure 4: Relation between v_i and v_j in G

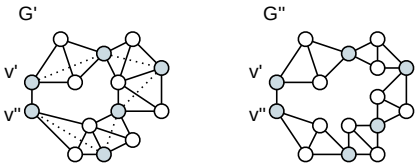


Figure 5: Subgraph of G and its structure

4 Planar Calculus $\mathscr{PHC}^*$

In this section, we propose another planar calculus $\mathscr{PHC}^*$. The set of initial graphs in $\mathscr{PHC}^*$ contains all graphs isomorphic to K_4. There are three rules for generating new graphs. Rule 1 and Rule 3 are same as the system $\mathscr{PHC}$. Our new Rule 2 is as follows:

2. **Vertex division/edge elimination rule**: Let G_1 be a graph with n vertices $\{v_1, \ldots, v_n\}$ that contains an edge (v_1, v_2), and G_2 be the graph obtained by contracting v_1 and v_2 of G_1.Then from G_1 and G_2, we can construct a graph G_3 graph that are identical to G_1 but does not contain (v_1, v_2).

Rule 2 is simple but powerful to generate non-3-colorable graphs. This rule means that none adjacent vertices v_1 and v_2 can be assigned the same color or different colors.

In the rest of this section, we prove the soundness and the completeness of $\mathscr{PHC}^*$.

Theorem 6 *$\mathscr{PHC}^*$ is sound.*

PROOF: We only need to show the soundness of Rule 2 since other rules also appear in $\mathscr{HC}$ and are shown to be sound [3]. Assume that there exists a 3-colorable graph G_3 generated by Rule 2. Then, its vertices v_1 and v_2 has a coloring satisfying one of the following two cases:

Case 1: $\mathrm{color}(v_1) \neq \mathrm{color}(v_2)$.

Case 2: $\mathrm{color}(v_1) = \mathrm{color}(v_2)$.

In Case 1, the coloring is also valid for G_1, i.e., G_1 is 3-colorable. In Case 2, we can contract vertices v_1 and v_2 in G_3, i.e., G_2 is also 3-colorable. Therfore, in $\mathscr{PHC}^*$, all graphs generated by Rule 2 are non-3-colorable. □

Theorem 7 *$\mathscr{PHC}^*$ is complete.*

PROOF: We prove this theorem by induction on the size n of the graph. In case $n < 4$, all graphs are colorable by at most 3 colors. In case $n = 4$, K_4 is the initial graph of $\mathscr{PHC}$. Other graphs of size 4 are all 3-colorable, so we do not care them.

Assume that all non-3-colorable graphs of size $n-1$ can be constructed in $\mathscr{PHC}^*$. We assume that there exists a nonempty set $\mathscr{G}$ of non-3-colorable graphs of size n that cannot be constructed in $\mathscr{PHC}$. Let G be an edge maximal graph in $\mathscr{G}$, i.e., by adding any edge to G, we can construct the graph in $\mathscr{PHC}^*$. Then we lead a contradiction that we can construct graph G in $\mathscr{PHC}^*$. By considering the size of the faces, we have the following two cases:

Case 1: All faces of G are triangle. According to Theorem 9 (we prove this theorem later), G can be constructed in $\mathscr{PHC}^*$.

Case 2: There is a face f of size $k \geq 4$. Let $v_1, v_2, v_3, \ldots, v_k$ be the vertices of face f. G' and G'' are graphs identical to G except that G' has an edge (v_1, v_3) and that G'' is contracted the vertices v_1 and v_3 of G. G' can be constructed since G is an edge maximal graph in $\mathscr{G}$. G'' can be constructed because of the assumption. In any case, we have a construction G from G' and G'' applying Rule 2.

In any case, G ($\in \mathscr{G}$) can be constructed in $\mathscr{PHC}^*$, which contradict to the definition of $\mathscr{G}$. Thus, any non-3-colorable graph can be constructed in $\mathscr{PHC}^*$. □

Now, we prove that any triangulate non-3-colorable planar graph can be constructed in polynomial number of steps. Similary to the proof of Theorem 5, we first prove the following lemma, which construct an essential component of triangulate planar graphs.

Lemma 8 *Let $G_n = (V, E)$ be a graph of $3n+1$ vertices, where $n \geq 1$,*

$$V = \{a_0\} \cup \{a_i,\ b_i,\ c_i \mid i \in \{1, \ldots, n\}\},$$

$$E = \{(a_0, a_n)\} \cup \{(a_{i-1}, b_i), (a_{i-1}, c_i), (a_i, b_i), (a_i, c_i), (b_i, c_i) \mid i \in \{1, \ldots, n\}\}$$

then, G has a linear size construction in $\mathscr{PHC}^*$.

PROOF: We prove this lemma by induction on n. In case $n = 1$, the lemma obviously holds since G_1 is isomorphic to an initial graph K_4.

We prove that G_n can be constructed in $\mathscr{PHC}^*$ by the assumption that G_{n-1} can be constructed. $G' = (V', E')$ is defined as follows:

$$V' = V_{n-1} \cup \{b_n, c_n\},$$

$$E' = E_{n-1} \cup \{(a_{n-1}, b_n), (a_{n-1}, c_n), (b_n, c_n).\}$$

G' can be constructed in $\mathscr{PHC}^*$ since G_{n-1} is a subgraph of G' induced by $\{a_i, b_i, c_i \mid i \in \{1, \ldots, n-1\}\}$. Let G'' be a graph that is identical to G_n with an edge (a_{n-1}, a_n). G'' can be constructed in $\mathscr{PHC}^*$ since it contains K_4 as a subgraph induced by $\{a_{n-1}, a_n, b_n, c_n\}$. Therefore we can construct G_n by applying Rule 2 to G' and G''. Since we apply Rule 1 twice and Rule 2 once at each induction step, the whole construction is linearly bounded. □

Theorem 9 *Triangulate non-3-colorable planar graphs have a polynomial size construction in $\mathscr{PHC}^*$.*

PROOF: Similary to the argument in the proof of Theorem 5, we can find a structure G_n of Lemma 8 as a subgraph of a given graph G. By contracting same vertices and adding some vertices and edges, G can be constructed in the system $\mathscr{PHC}^*$ □

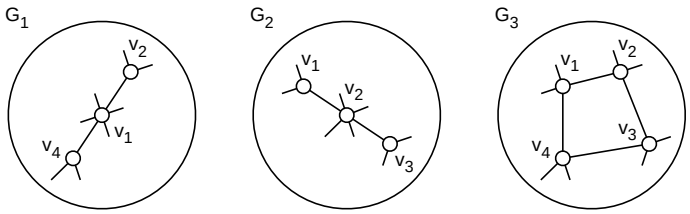


Figure 6: G_1, G_2 and G_3 of Rule 4 in $\mathscr{PHC}$

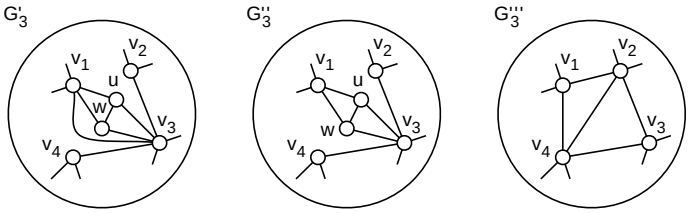


Figure 7: Intermediate graphs of simulation in $\mathscr{PHC}^*$

5 Polynomial-Time Simulation

We show the relationship between $\mathscr{PHC}$ and $\mathscr{PHC}^*$. First direction is that we simulate $\mathscr{PHC}$ by $\mathscr{PHC}^*$.

Theorem 10 *$\mathscr{PHC}^*$ p-simulates $\mathscr{PHC}$.*

PROOF: Rules 1 and 3 are common in $\mathscr{PHC}^*$ and $\mathscr{PHC}$. We only need to simulate Rule 2 and Rule 4 in $\mathscr{PHC}$ by $\mathscr{PHC}^*$. According to Lemma 11, Rule 2 can be simulated. According to Lemma 12, Rule 4 can be simulated. In each case the series of simulating steps can be constructed in polynomial time. Therefore, $\mathscr{PHC}^*$ p-simulates $\mathscr{PHC}$. □

Lemma 11 *$\mathscr{PHC}^*$ p-simulates Rule 2 of $\mathscr{PHC}$.*

PROOF: We prove that a graph G_3 can be constructed from G_1 and G_2 in $\mathscr{PHC}^*$. Let G_1 and G_2 be two graphs with common vertex set $\{v_1,\ldots,v_n\}$ which are identical except that G_1 contains edges (v_1,v_2) and (v_2,v_3) and not (v_1,v_3), whereas G_2 contains edges (v_1,v_2) and (v_1,v_3) and not (v_2,v_3). G_3 is identical to G_1 but does not contain (v_2,v_3). Let G'_1 be a graph identical to G_1 with vertices v_1 and v_3 are contracted. (G_1,G'_1,G_2,G_3) is a subsequence of a construction in $\mathscr{PHC}^*$ since G'_1 can be constructed from G_1 by Rule 3 and G_3 can be constructed from G'_1 and G_2 by Rule 2 with paticular vertices v_1 and v_3. □

Lemma 12 *$\mathscr{PHC}^*$ p-simulates Rule 4 of $\mathscr{PHC}$.*

PROOF: Let G_1, G_2 and G_3 be graphs as Figure 6. G_3 is a graph with vertex set $\{v_1,\ldots,v_n\}$ that contains a face v_1,v_2,v_3,v_4. G_1 is a graph obtained by contracting vertices v_1 and v_3 of G_3. G_2 is a graph obtained by contracting vertices v_2 and v_4 of G_3. We prove that a graph G_3 can be constructed from G_1 and G_2 in $\mathscr{PHC}^*$. Let G'_1 be the graph as Figure 7. G'_1 is identical to G_1, but has two additional vertices u and w and three additional edges $(v_1,u),(v_1,w),(u,w)$. G'_1 can be constructed from G_1 by Rule 1, since G'_1 is a subgraph of G'_3. Let G'_3, G''_3 and G'''_3 be graphs as Figure 7 that are identical to G_3 but some vertices and edges in the figure is different from G_3. G'_3 can be constructed from K_4 by Rule 1, since a subgraph induced by $\{v_1,v_3,u,w\}$ is isomorphic to K_4. G''_3 can be constructed from G'_1 and G'_3 by Rule 2 in $\mathscr{PHC}^*$ with paticular vertices v_1 and v_3 (G'_3 has an edge (v_1,v_3) and G'_1 is identical to G''_3 with v_1 and v_3 contracted). G'''_3 can be constructed from G''_3 by contracting two pairs of vertices (v_2, u) and (v_4, w). This construction needs twice of applying Rule 3 in $\mathscr{PHC}^*$. Then G_3 can be constructed from G'''_3 and G_2 by Rule 2 in $\mathscr{PHC}^*$ with paticular vertices v_2 and v_4. Thus Rule 4 in $\mathscr{PHC}$ can be p-simulated by $\mathscr{PHC}^*$. □

Theorem 10 implies that the modified rule (Rule 2) is at least as powerful as the original one in $\mathscr{HC}$.

Corollary 13 *$\mathscr{HC}$ can be p-simulated by $\mathscr{PHC}^*$ without planarity ristiction on the intermediate graphs.*

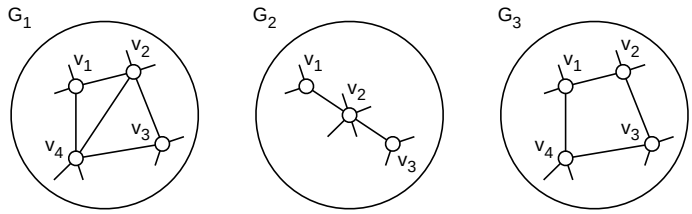


Figure 8: Rule 2 in $\mathscr{PHC}^*$

It is difficult to show that $\mathscr{PHC}$ p-simulates $\mathscr{PHC}^*$. For example, as shown in Figure 8, Rule 2 in $\mathscr{PHC}^*$ can generate a new quadrilateral of G_3 from G_1 and G_2. To simulate this construction, we must remove an edge (v_2, v_4) of G_1 by rules in $\mathscr{PHC}$, but edge elimination rule cannot be applied since (v_2, v_4) are sandwiched between triangle faces and the other rules cannot eliminate edges.

6 Concluding Remarks

We show that there exist a system of generating non-3-colorable planar graphs, where intermediate graphs in the system are restricted to be planar. Two calculi $\mathscr{PHC}$ and $\mathscr{PHC}^*$ are sound and complete graph construction system that generates the class of non-3-colorable planar graphs. $\mathscr{PHC}^*$ is simple but powerful calculus, since $\mathscr{PHC}^*$ p-simulates $\mathscr{PHC}$.

Relationship between construction in planar graph calculus and general graph calculus may be interesting. There is a structure that can replace crossing edges keeping the colorability condition, so that non-planar graphs can be mapped to planar graphs. Thus a class of graphs that have super-polynomial lower bound in $\mathscr{HC}$ may be associated with a class of graphs in a planar calculus. For future discussion, we would like to consider polynomial-time simulation of $\mathscr{PHC}^*$ by $\mathscr{PHC}$. Also lower bound of planar graph calculus is an interesting work.

References

[1] H. HADWIGER, "Über eine Klassifikation der Streckenkomplexe," Vierteljahrsschr Naturforsch, Ges. Zurich, 88, pp.133–142 (1943)

[2] G. A. DIRAC, "A property of 4-chromatic graphs and some remarks on critical graphs," J. London Math. Soc., 27, pp.85–92, (1952)

[3] G. HAJÓS, "Üver eine Konstruktion nicht n-färbbarer Graphen," Wiss. Zeitschr Martin Luther Univ. Halle-Wittenberg, A 10 (1961)

[4] K. WAGNER, "Beweis einer Abschwächung der Hadwiger Vermutung," Math. Ann., 153, pp.139–141 (1964)

[5] R. M. KARP, "Reducibility among Combinatorial Problems," Proc. Symp. Complexity of Computer Computations, ED. R. E. MILLER, J. W. THATCHER, pp.85–103 (1972)

[6] S. A. COOK AND R. A. RECKHOW, "Time bounded random access machines," Journal of Computer and System Sciences, 7 (4), pp.354–375 (1973)

[7] K. APPEL AND W. HAKEN, "Every planar map is four colorable," Part I: Discharging, Illinois J. Math., 21, pp.429–490 (1977)

[8] K. APPEL, W. HAKEN, AND J. KOCH, "Every planar map is four colorable," Part II: Reducibility, Illinois J. Math., 21, pp.491–567 (1977)

[9] M. R. GAREY AND D. S. JOHNSON, "Computers and Intractability: A guide to the theory of NP-Completeness," W. H. FREEMAN, (1979)

[10] F. GUTHRIE, "Note on the colouring of maps," *Proc. R. Soc. Edinb.*, 10, pp.727–728 (1980)

[11] A. J. MANSFIELD, D. J. A. WELSH, "Some coloring problems and their complexity," Annals of Discrete Mathematics, 13 (1982)

[12] K. APPEL AND W. HAKEN, "Every Planar Map is Four Colorable, Providence," Rhode Island, AMS, (1989)

[13] T. PITASSI, P. W. BEAME, AND R. IMPAGLIAZZO, "Exponential lower bounds for the pigeonhole principle," Computational Complexity, 3 (2), pp.97–140 (1993)

[14] M. AJTAI, "The complexity of the pigeonhole principle," Combinatorica, 14 (4), pp.417–433 (1994)

[15] T. R. JENSEN AND B. TOFT, "Graph Coloring Problems", Wiley (1995)

[16] T. PITTASI AND A. URQUHART, “The complexity of the Hajos Calculus,” SIAM Journal on Discrete Mathematics, Vol. 8 (1995)

[17] K. IWAMA AND T. PITASSIRT, “ Exponential lower bounds for the tree-like Hajos Calculus” Information Processing Letters, Vol. 54, pp. 289-294 (1995)

[18] J. KRAJÍČEK, P. PUDLÁK, AND A. WOODS, “Exponential lower bounds to the size of bounded depth Frege proofs of the pigeonhole principle,” Random Structures and Algorithms, 7 (1) (1995)

[19] N. ROBERTSON, D. SANDERS, P. D. SEYMOUR, AND R. THOMAS, “The four-colour theorem,” J. Combin. Theory, B 70, pp.2–44 (1997)

Online Allocation with Risk Information

SHIGEAKI HARADA

Graduate School of Information Sciences
Tohoku University
Sendai 980-8579, Japan
sig@maruoka.ecei.tohoku.ac.jp

EIJI TAKIMOTO*

Graduate School of Information Sciences
Tohoku University
Sendai 980-8579, Japan
t2@maruoka.ecei.tohoku.ac.jp

AKIRA MARUOKA

Graduate School of Information Sciences
Tohoku University
Sendai 980-8579, Japan
maruoka@maruoka.ecei.tohoku.ac.jp

Abstract: We consider the problem of dynamically apportioning resources among a set of options in a worst-case online framework. The model we investigate is a generalization of the well studied online learning model. In particular, we allow the learner to see as additional information how high the risk of each option is. This assumption is natural in many applications like horse-race betting, where gamblers know odds for all options before placing bets. We apply the Aggregating Algorithm to this problem and give a tight performance bound. The bound we give intuitively implies that the algorithm performs better when faced with options of various risks than when faced with options of the same risk.

Keywords: online learning, resource allocation, Hedge algorithm, aggregating algorithm

1 Introduction

Consider the following scenario for horse-race betting. A learner is given access to N expert gamblers who are highly successful in horse-race betting. In every race, the experts give advice on how to bet money and the learner somehow combines their advice and decides his own way to apportion the wager. His goal is to allocate each race's wager in such a way that his total winnings for the season will be reasonably close to what he would have won had he bet everything with the luckiest of the experts.

Such online allocation problems are usually formalized as a repeated game between the learner and the environment as described below. At each time step $t = 1, 2, \ldots T$, the learner (an algorithm A) chooses a probability distribution $\boldsymbol{v}_t = (v_{1,t}, \ldots, v_{N,t})$ over the set $\{1, \ldots, N\}$ of experts. After the choice is made, the environment determines experts' losses $\boldsymbol{l}_t = (l_{1,t}, \ldots, l_{N,t})$, and the learner A suffers loss $\boldsymbol{v}_t \cdot \boldsymbol{l}_t = \sum_{i=1}^{N} v_{i,t} l_{i,t}$. This loss can be interpreted as the loss if the learner apportions the wager among the experts according to the distribution $\boldsymbol{v}_t$ and let the experts place bets on behalf of the learner. Let $L_{A,T} = \sum_{t=1}^{T} \boldsymbol{v}_t \cdot \boldsymbol{l}_t$ denote the total loss of the learner A and $L_{i,T} = \sum_{t=1}^{T} l_{i,t}$ denote the total loss of expert i. The goal of the learner is to minimize the following *regret*

$$L_{A,T} - \min_{i \in \{1,\ldots,N\}} L_{i,T} = \sum_{t=1}^{T} \boldsymbol{v}_t \cdot \boldsymbol{l}_t - \min_{i \in \{1,\ldots,N\}} \sum_{t=1}^{T} l_{i,t},$$

which is the total loss of the learner relative to the total loss suffered by the best expert.

This problem has been extensively studied since Hannan [4] and Blackwell [1]. Freund and Schapire [3] propose a strategy for the learner called the Hedge Algorithm that assigns $v_{i,t}$ proportional to $\beta^{\sum_{s=1}^{t-1} l_{i,s}}$, where $\beta \in (0,1)$ is the *learning rate*. They show that if the losses $l_{i,t}$ are all bounded between 0 and 1, then

$$L_{\mathrm{Hedge},T} \leq \frac{\ln(1/\beta)}{1-\beta} \left(\min_i L_{i,T} + \frac{\ln N}{\ln(1/\beta)} \right). \tag{1}$$

*Research is supported by Grant-in-Aid for Scientifi c Research on Priority Area "New Horizons in Computing" from the Japanese Ministry of Education, Culture, Sports, Science and Technology.

Cesa-Bianchi and Lugosi [2] give the following regret bound for the Hedge Algorithm

$$\frac{\ln N}{\ln(1/\beta)} + \frac{T\ln(1/\beta)}{8} = \sqrt{(T/2)\ln N} \tag{2}$$

with an appropriate choice of β. (For this choice, the horizon T of the game needs to be known in advance). Kalai and Vempala [6] propose a family of algorithms based on Hannan's "Follow the Perturbed Leader" approach that have many applications. The regret for this problem is shown to be of the same form as (2) with an additional small constant factor [5]. Vovk [8] applies his Aggregating Algorithm (AA, for short) to this problem and gives a better bound

$$L_{\mathrm{AA},T} \le c(\beta)\left(\min_i L_{i,T} + \frac{\ln N}{\ln(1/\beta)}\right), \tag{3}$$

where

$$c(\beta) = \frac{\ln(1/\beta)}{N\ln\frac{N}{N+\beta-1}} < \frac{\ln(1/\beta)}{1-\beta}.$$

Note that in the model discussed above, the learner is only allowed to interact with experts who actually make decisions against the environment, whereas the interactions between experts and the environment are hidden and just summarized as the loss vectors $\boldsymbol{l}_t$. Moreover, we assume that the losses $l_{i,t}$ are uniformly bounded to $[0,1]$.

In this paper, we generalize the online allocation model in two ways. Firstly, we allow the learner to see experts' advice as well as their losses and to make its own decisions. Secondly, we do not require experts' losses to be uniformly bounded but allow the learner to see upper and lower bounds on $l_{i,t}$ before making its decisions. This assumption is natural in many applications like horse-race betting, where gamblers know *odds* for all options before placing bets. Clearly, the information of odds determines the maximum return for each option.

To be more specific, let us consider the case where K options are to be bet in every time step, and the environment determines losses for the options by which the losses for the experts are defined. At time step t, the following happens.

1. Each expert i suggests a K-dimensional probability vector $\boldsymbol{x}_{i,t}$ that indicates how to apportion the wager among options.
2. The learner observes the *risk information* $a_{j,t}$ and $b_{j,t}$ that are upper and lower bounds on the loss for all options j.
3. The learner chooses a probability vector $\boldsymbol{p}_t$ of K dimension.
4. The losses $\boldsymbol{y}_t = (y_{1,t},\ldots,y_{K,t})$ for the options are revealed, where $y_{j,t} \in [a_{j,t}, b_{j,t}]$.
5. The learner suffers loss $\boldsymbol{p}_t \cdot \boldsymbol{y}_t$ and each expert i suffers loss $l_{i,t} = \boldsymbol{x}_{i,t} \cdot \boldsymbol{y}_t$.

Note that if the learner's decision $\boldsymbol{p}_t$ is given by $\boldsymbol{p}_t = \sum_{i=1}^N v_{i,t}\boldsymbol{x}_{i,t}$ for a distribution $\boldsymbol{v}_t$ over the experts, then we have $\boldsymbol{p}_t \cdot \boldsymbol{y}_t = \boldsymbol{v}_t \cdot \boldsymbol{l}_t$, which coincides with the learner's loss in the previous model. Therefore, if we assume the uniform risk (i.e., $a_{j,t} = 0$ and $b_{j,t} = 1$), then all the previous results remain to hold in the new model, regardless of the number K of options.

Vovk [8] also considers this model with the uniform risk and shows that the performance of the Aggregating Algorithm is again given by (3) but now the leading factor is

$$c(\beta) = \frac{\ln(1/\beta)}{K\ln\frac{K}{K+\beta-1}}, \tag{4}$$

which is monotonically increasing in K. So, if $K < N$, this gives a tighter bound. Unfortunately, however, there is a technical problem to apply the Aggregating Algorithm to the general case where the risk is not fixed.

In this paper, we assume a fixed but non-uniform risk, namely, $a_{j,t} = 0$ and $b_{j,t} = b_j$, where $\boldsymbol{b} = (b_1,\ldots,b_K)$ is a fixed *risk vector* that is time invariant. Under this assumption, we analyze the performance of the Aggregating Algorithm and give a leading factor $c(\beta)$ in the loss bound in terms of the risk vector $\boldsymbol{b}$. We then show some interesting behaviors of $c(\beta)$. In particular, it turns out that our $c(\beta)$ is smaller than (4) even if some of b_j is larger than 1.

2 Aggregating Algorithm

The Aggregating Algorithm is a very general strategy that works for various games. In this section, we describe the algorithm with its performance bound in a generic form. Vovk shows that under some mild assumptions, the bound cannot be essentially improved and thus the Aggregating Algorithm is optimal [8].

First we describe a game that involves the learner, N experts, and the environment. A game is specified by a triple (Γ,Ω,λ), where Γ is a fixed prediction space, Ω is a fixed outcome space, and $\lambda : \Omega\times\Gamma \to [0,\infty]$ is a fixed loss function. At each trial $t = 1,2,\ldots$, the following happens.

1. Each expert i makes a prediction $\gamma_{i,t} \in \Gamma$.
2. The learner, who allowed to see all $\gamma_{i,t}$, makes his own prediction $\gamma_t \in \Gamma$.
3. The environment chooses some outcome $\omega_t \in \Omega$.
4. Each expert i suffers loss $\lambda(\omega_t, \gamma_{i,t})$ and the learner suffers loss $\lambda(\omega_t, \gamma_t)$.

Next we give the assumptions about the game that makes the Aggregating Algorithm well-defined and perform optimally.

Assumption 1 *We assume that the game $(\Gamma, \Omega, \lambda)$ satisfies the following conditions.*

- *Γ is a compact topological space.*
- *For each ω, the function $\gamma \mapsto \lambda(\omega, \gamma)$ is continuous.*
- *There exists γ such that, for all ω, $\lambda(\omega, \gamma) < \infty$.*
- *There exists no γ such that, for all ω, $\lambda(\omega, \gamma) = 0$.*

It is known that lots of games considered in the literature satisfies the assumptions.

We define a *simple probability distribution* in Γ to be a function Q that assigns to each element γ of its finite domain $\mathrm{dom}\, Q \subseteq \Gamma$ a positive weight $Q(\gamma)$ so that $\sum_\gamma Q(\gamma) = 1$ (γ ranging over $\mathrm{dom}\, Q$). Let $\beta \in (0,1)$. A *pseudoprediction* (with respect to Q) is a function from Ω to the set of real numbers given by

$$g^Q(\omega) = \log_\beta \left(\sum_{\gamma \in \mathrm{dom}\, Q} Q(\gamma) \beta^{\lambda(\omega,\gamma)} \right).$$

We will omit the superscript Q when it is clear from context. Let

$$c(\beta) = \sup_Q \inf_{\gamma \in \Gamma} \sup_{\omega \in \Omega} \frac{\lambda(\omega, \gamma)}{g^Q(\omega)}. \tag{5}$$

Lemma 2 ([8]) *Under the assumptions given in Assumption 1, there exists a function Σ_β called a* substitution function *that maps a pseudoprediction to a prediction in Γ such that for any simple distribution Q and any $\omega \in \Omega$,*

$$\lambda(\omega, \gamma) \le c(\beta) g^Q(\omega),$$

where $\gamma = \Sigma_\beta(g^Q)$. Moreover, the following minimax prediction is a substitution function:

$$\Sigma_\beta(g^Q) = \arg \inf_{\gamma \in \Gamma} \sup_{\omega \in \Omega} \frac{\lambda(\omega, \gamma)}{g^Q(\omega)}. \tag{6}$$

Now we show how the Aggregating Algorithm behaves. It maintains a weight $v_{i,t}$ for each expert i so that $\boldsymbol{v}_t = (v_{1,t}, \ldots, v_{N,t})$ is a probability vector. When the experts make predictions $\gamma_{i,t}$, we consider $\boldsymbol{v}_t$ as a simple distribution Q_t such that $Q_t(\gamma_{i,t}) = v_{i,t}$. Then, the Aggregating Algorithm predicts with $\gamma_t = \Sigma_\beta(g^{Q_t})$ with some substitution function Σ_β. When an outcome ω_t is given, the weights are updated according to $v_{i,t+1} = v_{i,t} \beta^{\lambda(\omega_t, \gamma_{i,t})} / Z$, where Z is for normalization. We give a pseudocode in Figure 1.

The loss bound of the Aggregating Algorithm is represented by $c(\beta)$.

Theorem 3 ([8]) *Assume the assumptions given in Assumption 1. Then, for any horizon T of the game and for any outcome sequence $\omega_1, \ldots, \omega_T$,*

$$L_{AA(\beta),T} \le c(\beta) \left(\min_i L_{i,T} + \frac{\ln N}{\ln(1/\beta)} \right).$$

Moreover, for any pair (c, a) with $c < c(\beta)$ and $a < c(\beta)/\ln(1/\beta)$, no algorithm A achieves $L_{A,T} \le c \min_i L_{i,T} + a \ln N$.

Algorithm AA(β)
begin
$\boldsymbol{v}_1 = (1/N, \ldots, 1/N)$;
for $t = 1$ **to** T **do begin**
receive experts' predictions $(\gamma_{1,t}, \ldots, \gamma_{N,t})$;
let $g_t : \omega \mapsto \log_\beta \sum_{i=1}^{N} v_{i,t} \beta^{\lambda(\omega, \gamma_{i,t})}$;
output $\gamma_t = \Sigma_\beta(g_t)$;
observe an outcome ω_t and suffer loss $\lambda(\omega_t, \gamma_t)$;
for $i = 1$ **to** N **do**
$v_{i,t+1} = \dfrac{v_{i,t} \beta^{\lambda(\omega_t, \gamma_{i,t})}}{\sum_{j=1}^{N} v_{j,t} \beta^{\lambda(\omega_t, \gamma_{j,t})}}$;
end
end

Figure 1: Aggregating Algorithm

3 The Aggregating Algorithm for Our Game

It is easy to see that the following game $(\Gamma, \Omega, \lambda)$ corresponds to our online allocation problem: The prediction space Γ is the K-dimensional probability simplex, the outcome space is $\Omega = [0,1]^K$, and the loss function is given by

$$\lambda(\boldsymbol{\omega}, \boldsymbol{p}) = \sum_{j=1}^{K} b_j \omega_j p_j,$$

where $\boldsymbol{b} = (b_1, \ldots, b_K)$ is a fixed risk vector. Note that the losses for options at trial t are given by $y_{j,t} = b_j \omega_{j,t}$ for some $\boldsymbol{\omega}_t \in \Omega$ so that $y_{j,t} \in [0, b_j]$ and $\lambda(\boldsymbol{\omega}_t, \boldsymbol{p}_t) = \boldsymbol{p}_t \cdot \boldsymbol{y}_t$.

We can show that the assumptions in Assumption 1 are satisfied for this game. So the Aggregating Algorithm and the loss bound given in Theorem 3 apply. Vovk analyzes only the case where $\boldsymbol{b} = 1^K = (1, \ldots, 1)$ and show that $c(\beta)$ is given by (4). In what follows, we assume, without loss of generality, that $b_1 \leq \cdots \leq b_K$.

First we claim that it suffices to consider (5) for Q concentrated on the extreme points $\boldsymbol{e}_j$ ($j = 1, \ldots, K$) of the simplex Γ, where the m-th component of $\boldsymbol{e}_j$ is 1 if $m = j$ and 0 otherwise.

Lemma 4 *Let Q be any simple distribution and the weighted average with respect Q be denoted $\hat{\boldsymbol{p}}$. That is,*

$$\hat{\boldsymbol{p}} = (\hat{p}_1, \ldots, \hat{p}_K) = \sum_{\boldsymbol{x} \in \mathrm{dom}\, Q} Q(\boldsymbol{x}) \boldsymbol{x}.$$

Let Q' be a simple distribution that assigns to each $\boldsymbol{e}_j$ the weight $Q'(\boldsymbol{e}_j) = \hat{p}_j$. Then, for any $\boldsymbol{\omega} \in \Omega$,

$$g^Q(\boldsymbol{\omega}) \geq g^{Q'}(\boldsymbol{\omega}).$$

PROOF: Note that each $\boldsymbol{x} = (x_1, \ldots, x_K) \in \mathrm{dom}\, Q$ is a probability vector. The convexity of the function $\zeta \mapsto \beta^\zeta$ implies

$$\begin{aligned} \sum_{\boldsymbol{x}} Q(\boldsymbol{x}) \beta^{\lambda(\boldsymbol{\omega}, \boldsymbol{x})} &= \sum_{\boldsymbol{x}} Q(\boldsymbol{x}) \beta^{\sum_j b_j \omega_j x_j} \leq \sum_{\boldsymbol{x}} Q(\boldsymbol{x}) \sum_j x_j \beta^{b_j \omega_j} \\ &= \sum_j \sum_{\boldsymbol{x}} x_j Q(\boldsymbol{x}) \beta^{\lambda(\boldsymbol{\omega}, \boldsymbol{e}_j)} = \sum_j \hat{p}_j \beta^{\lambda(\boldsymbol{\omega}, \boldsymbol{e}_i)} = \sum_{\boldsymbol{x}} Q'(\boldsymbol{x}) \beta^{\lambda(\boldsymbol{\omega}, \boldsymbol{x})}, \end{aligned}$$

which completes the lemma. □

Now we give a prediction $\boldsymbol{p}_t$ of the learner of a closed form. Let g_t be the pseudoprediction defined at trial t in the Aggregating Algorithm. That is,

$$g_t(\boldsymbol{\omega}) = \log_\beta \sum_{i=1}^{N} v_{i,t} \beta^{\lambda(\boldsymbol{\omega}, \boldsymbol{x}_{i,t})}$$

where $\boldsymbol{x}_{i,t}$ is the suggestion from expert i. By Lemma 4, we have another pseudoprediction

$$g'_t(\boldsymbol{\omega}) = \log_\beta \sum_{j=1}^{K} \hat{p}_{j,t} \beta^{\lambda(\boldsymbol{\omega}, \boldsymbol{e}_j)}$$

such that $g_t(\boldsymbol{\omega}) \geq g'_t(\boldsymbol{\omega})$, where $\hat{\boldsymbol{p}}_t = (\hat{p}_{1,t}, \ldots, \hat{p}_{K,t}) = \sum_{i=1}^N v_{i,t}\boldsymbol{x}_{i,t}$. So, we compute $\boldsymbol{p}_t = \arg\inf_{\boldsymbol{p}} \sup_{\boldsymbol{\omega}} \lambda(\boldsymbol{\omega}, \boldsymbol{p})/g'_t(\boldsymbol{\omega})$ since Lemma 2 implies $\lambda(\boldsymbol{\omega}, \boldsymbol{p}_t) \leq c(\beta) g'_t(\boldsymbol{\omega}) \leq c(\beta) g_t(\boldsymbol{\omega})$ as desired.

Theorem 5 *The vector $\boldsymbol{p}_t$ given by*

$$p_{j,t} = \frac{\frac{1}{b_j}\log_\beta\left(1-(1-\beta^{b_j})\hat{p}_{j,t}\right)}{\sum_{j=1}^K \frac{1}{b_j}\log_\beta\left(1-(1-\beta^{b_j})\hat{p}_{j,t}\right)} \tag{7}$$

attains the infimum of $\arg\inf_{\boldsymbol{p}} \sup_{\boldsymbol{\omega}} \lambda(\boldsymbol{\omega}, \boldsymbol{p})/g'_t(\boldsymbol{\omega})$.

PROOF: Since $g'_t(\boldsymbol{\omega})$ is concave, it suffices to consider only for $\boldsymbol{\omega}$ that are extreme points of the cube $[0,1]^K$. Using the notation of $I = \{j \mid \omega_j = 1, 1 \leq j \leq K\}$, we write

$$g'_t(\boldsymbol{\omega}) = \log_\beta \sum_{j=1}^K \hat{p}_{j,t}\beta^{b_j\omega_j} = \log_\beta\left(\sum_{j\in I}\beta^{b_j}\hat{p}_{j,t} + \sum_{j\notin I}\hat{p}_{j,t}\right) = \log_\beta\left(1 - \sum_{j\in I}(1-\beta^{b_j})\hat{p}_{j,t}\right).$$

So,

$$\inf_{\boldsymbol{p}}\sup_{\boldsymbol{\omega}} \frac{\lambda(\boldsymbol{\omega}, \boldsymbol{p})}{g'_t(\boldsymbol{\omega})} = \inf_{\boldsymbol{p}}\max_{I} \frac{\sum_{j\in I} b_j p_j}{\log_\beta\left(1-\sum_{j\in I}(1-\beta^{b_j})\hat{p}_{j,t}\right)}. \tag{8}$$

Using the inequality $\sum_j \log_\beta(1-a_j) = \log_\beta \prod_j (1-a_j) \leq \log_\beta(1-\sum_j a_j)$ for $a_j > 0$, we have

$$\frac{\sum_{j\in I} b_j p_j}{\log_\beta\left(1-\sum_{j\in I}(1-\beta^{b_j})\hat{p}_{j,t}\right)} \leq \frac{\sum_{j\in I} b_j p_j}{\sum_{j\in I}\log_\beta\left(1-(1-\beta^{b_j})\hat{p}_{j,t}\right)} \leq \max_{j\in I}\frac{b_j p_j}{\log_\beta\left(1-(1-\beta^{b_j})\hat{p}_{j,t}\right)}.$$

Hence

$$\inf_{\boldsymbol{p}}\sup_{\boldsymbol{\omega}} \frac{\lambda(\boldsymbol{\omega}, \boldsymbol{p})}{g'_t(\boldsymbol{\omega})} = \inf_{\boldsymbol{p}}\max_{j\in\{1,\ldots,K\}} \frac{p_j}{\frac{1}{b_j}\log_\beta\left(1-(1-\beta^{b_j})\hat{p}_{j,t}\right)}. \tag{9}$$

Note that for any $\boldsymbol{p} \in \Gamma$, it must hold that

$$\max_{j\in\{1,\ldots,N\}} \frac{p_j}{\frac{1}{b_j}\log_\beta\left(1-(1-\beta^{b_j})\hat{p}_{j,t}\right)} \geq \frac{1}{\sum_{j=1}^K \frac{1}{b_j}\log_\beta\left(1-(1-\beta^{b_j})\hat{p}_{j,t}\right)}$$

because otherwise we would have some $\boldsymbol{p} \in \Gamma$ such that

$$p_j < \frac{\frac{1}{b_j}\log_\beta\left(1-(1-\beta^{b_j})\hat{p}_{j,t}\right)}{\sum_{j=1}^K \frac{1}{b_j}\log_\beta\left(1-(1-\beta^{b_j})\hat{p}_{j,t}\right)}$$

for all $1 \leq j \leq K$, which implies $\sum_{j=1}^N p_j < 1$, a contradiction. Therefore, our choice of

$$p_{j,t} = \frac{\frac{1}{b_j}\log_\beta\left(1-(1-\beta^{b_j})\hat{p}_{j,t}\right)}{\sum_{j=1}^K \frac{1}{b_j}\log_\beta\left(1-(1-\beta^{b_j})\hat{p}_{j,t}\right)}$$

attains the infimum. □

Next we estimate the value of $c(\beta)$. Since $c(\beta)$ depends on the risk vector $\boldsymbol{b}$ as well, we write it as $c(\beta, \boldsymbol{b})$ to explicitly specify $\boldsymbol{b}$. By virtue of Lemma 4 we can only consider a distribution Q on the extreme points. Let $\boldsymbol{q} = (q_1, \ldots, q_K)$ be the probability vector induced by Q, i.e., $q_j = Q(\boldsymbol{e}_j)$. From the proof above, it follows that

$$c(\beta, \boldsymbol{b}) = \sup_Q \inf_{\boldsymbol{p}}\sup_{\boldsymbol{\omega}} \frac{\lambda(\boldsymbol{\omega}, \boldsymbol{p})}{g^Q(\boldsymbol{\omega})} = \sup_{\boldsymbol{q}} \frac{1}{\sum_{j=1}^K \frac{1}{b_j}\log_\beta\left(1-(1-\beta^{b_j})q_j\right)}. \tag{10}$$

In other words, we need to solve the following optimization problem:

$$\text{minimize } -\sum_{j=1}^K \frac{1}{b_j}\ln\left(1-(1-\beta^{b_j})q_j\right)$$

subject to $\sum_j q_j = 1$ and $q_j \geq 0$ for all j. Since the objective function and the domain are both convex, it is well known that $\boldsymbol{q}$ is an optimal point if and only if the following Kursh-Kuhn-Tucker (KKT) conditions are satisfied:

$$\begin{aligned}
\frac{1-\beta^{b_j}}{b_j\left(1-(1-\beta^{b_j})q_j\right)} - s_j + t &= 0, \quad (j=1,\ldots,K) && (11)\\
\sum_{j=1}^K q_j - 1 &= 0, && (12)\\
s_j \cdot (-q_j) &= 0, \quad (j=1,\ldots,K) && (13)\\
-q_j &\leq 0, \quad (j=1,\ldots,K) && (14)\\
s_j &\geq 0, \quad (j=1,\ldots,K) && (15)
\end{aligned}$$

for some t and s_j for $j=1,\ldots,K$. Note that the first condition (11) is derived from $\nabla_{\boldsymbol{q}} L(\boldsymbol{q},t,\boldsymbol{s}) = 0$ where L is the Lagrangian function.

Solving $\boldsymbol{q}$ that satisfies these conditions, we get $c(\beta,\boldsymbol{b})$ which is given in the next theorem.

Theorem 6 *Assume that $b_1 \leq \cdots \leq b_K$. Then,*

$$c(\beta,\boldsymbol{b}) = \frac{\ln(1/\beta)}{\sum_{j=z}^K \frac{1}{b_j}\ln\frac{b_j}{1-\beta^{b_j}} + \left(\sum_{j=z}^K \frac{1}{b_j}\right)\ln\frac{\sum_{k=j}^K (1/b_j)}{\sum_{j=z}^K \frac{1}{1-\beta^{b_j}}-1}} \tag{16}$$

where

$$z = \arg\max_{k\in\{1,\ldots,K\}} \frac{\sum_{j=k}^K \frac{1}{1-\beta^{b_j}} - 1}{\sum_{j=k}^K \frac{1}{b_j}}. \tag{17}$$

We call z satisfying (17) the threshold index.

PROOF: Let $\boldsymbol{q}$ be a solution that satisfies the KKT conditions. By conditions (13) and (14), if $q_j > 0$ then $s_j = 0$. So, conditions (11) and (15) imply that

$$q_j > 0 \quad \Rightarrow \quad q_j = \frac{1}{t(1-\beta^{b_j})}\left(t + \frac{1-\beta^{b_j}}{b_j}\right), \tag{18}$$

$$q_j = 0 \quad \Rightarrow \quad s_j = t + \frac{1-\beta^{b_j}}{b_j} \geq 0. \tag{19}$$

Plugging (18) into condition (12), we get

$$t = -\frac{\sum_{j:q_j>0}\frac{1}{b_j}}{\sum_{j:q_j>0}\frac{1}{1-\beta^{b_i}} - 1}, \tag{20}$$

which is negative. So, we have that $q_j > 0 \Leftrightarrow (1-\beta^{b_j})/b_j < -t$. Since the function $(1-\beta^{\zeta})/\zeta$ is monotonically decreasing and we assume $b_1 \leq \cdots \leq b_K$, it follows that there exists $z \in \{1,\ldots,K\}$ such that $q_j = 0$ for all $j < z$ and $q_j > 0$ for all $j \geq z$. Equivalently, z must satisfy

$$\frac{1-\beta^{b_z}}{b_z} < -t \leq \frac{1-\beta^{b_{z-1}}}{b_{z-1}},$$

where

$$t = -\frac{\sum_{j=z}^K \frac{1}{b_j}}{\sum_{j=z}^K \frac{1}{1-\beta^{b_j}} - 1}. \tag{21}$$

It is straightforward to confirm that

$$z = \arg\max_k \frac{\sum_{j=k}^K \frac{1}{1-\beta^{b_j}} - 1}{\sum_{j=k}^K \frac{1}{b_j}}$$

actually satisfies the above inequalities. Now all the KKT conditions are satisfied. Plugging (18) with (21) into (10) we get the theorem. □

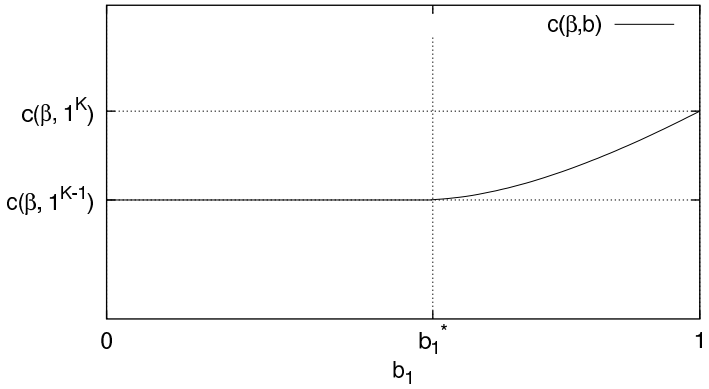


Figure 2: The curve of $c(\beta, \boldsymbol{b})$ with $b_2 = \cdots = b_K = 1$. The critical value b_1^* is given by the solution of $\frac{b_1^*}{1-\beta^{b_1^*}} = \frac{1}{1-\beta} - \frac{1}{K-1}$. Here we set $N = 5$ and $\beta = 0.8$.

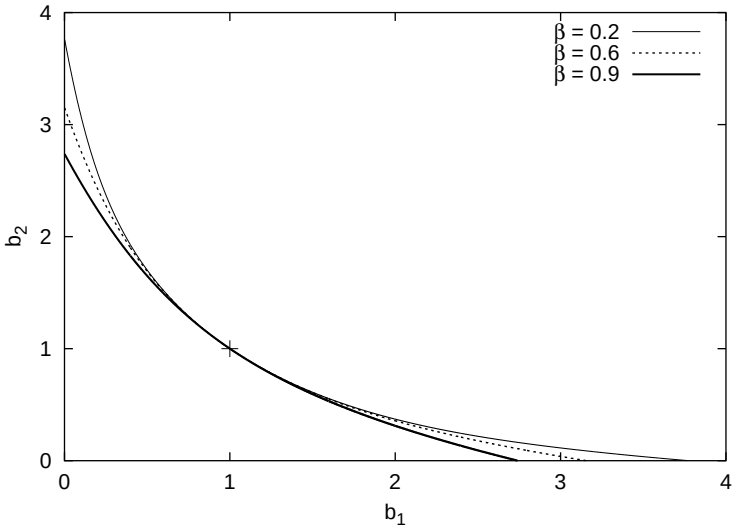


Figure 3: Curves of $\boldsymbol{b} = (b_1, b_2)$ that satisfies $c(\beta, \boldsymbol{b}) = c(\beta, 1^2)$ for various β.

4 Behaviors of $c(\beta, \boldsymbol{b})$

Here we observe some interesting properties of $c(\beta, \boldsymbol{b})$. Firstly, $c(\beta, \boldsymbol{b})$ with the uniform risk vector $\boldsymbol{b} = 1^K$ coincides with the classical value (4). (In this case, the threshold index is $z = 1$.) Secondly, it is easy to verify that $c(\beta, \boldsymbol{b})$ is monotonically increasing with respect to all b_j. So for any $\boldsymbol{b} \leq 1^K$, $c(\beta, \boldsymbol{b}) \leq c(\beta, 1^K)$.

Let us look at this more closely. Let $b_2, \ldots, b_K$ be fixed to 1 and let b_1 leave as a variable. Curiously, if $b_1 \leq b_1^*$ for some $b_1^* > 0$, then the threshold index becomes $z = 2$ and hence $c(\beta, \boldsymbol{b}) = c(\beta, 1^{K-1})$. This implies that when a risk of one option is too small, then the Aggregating Algorithm performs as well as the game where the number of options is $K - 1$. In Figure 2 we show the curve of $c(\beta, \boldsymbol{b})$ as a function of b_1.

Next we investigate a condition that makes $c(\beta, \boldsymbol{b}) \leq c(\beta, 1^K)$. As stated above, $\boldsymbol{b} \leq 1^K$ is a trivial condition. Interestingly, if some of risks b_j are much larger than 1, $c(\beta, \boldsymbol{b})$ can be smaller than $c(\beta, 1^K)$. Figure 3 describes the curves of $\boldsymbol{b} = (b_1, b_2)$ such that $c(\beta, \boldsymbol{b}) = c(\beta, 1^K)$ with $K = 2$ for various values of β. If a point $\boldsymbol{b}$ lays in the lower left part of the curve for β, then $c(\beta, \boldsymbol{b}) < c(\beta, 1^K)$. Intuitively, the curves show that the algorithm performs better when faced with options of various risks than when faced with options of the same risk.

5 Concluding Remarks

We generalize the online allocation model so that the learner is allowed to see the risk information about the options. We apply the Aggregating Algorithm and give a tight loss bound. Unfortunately, we have a result only for a restricted case where

the risks are all lower bounded by zero and time invariant. Yet, the model has a nice application, e.g., the online shortest path problem [7]: When given a network, the learner tries to choose a routing path in every trial in an attempt to minimize the total expected time delay. In this case, each path corresponds to an option and the number of nodes in the path can be regarded as its risk. (The more nodes it has, the more time is expected to take for sending a packet through that path).

To treat a more general case, we perhaps need to extend the Aggregating Algorithm since it does not seem to satisfy the assumptions we need as shown in Section 2.

References

[1] D. BLACKWELL, An analog of the minimax theorem for vector payoffs, *Pacific Journal of Mathematics* (1956) **6(1):1–8**

[2] N. CESA-BIANCHI AND G. LUGOSI, On prediction of individual sequences, *Annals of Statistics* (1999) **27:1865–1895**

[3] YOAV FREUND AND ROBERT E. SCHAPIRE, A decision-theoretic generalization of on-line learning and an application to boosting, *Journal of Computer and System Sciences* (1997) **55(1):119–139**

[4] J. HANNAN, Approximation to Bayes risk in repeated play, *Contributions to the Theory of Games* (1957) **3:97—139**

[5] M. HUTTER AND J. POLAND, Prediction with Expert Advice by Following the Perturbed Leader for General Weights, *LNAI (Proc. 15th ALT)* (2004) **3244:279–293**

[6] A. KALAI AND S. VEMPALA, Efficient algorithms for online decision problems, *LNAI (Proc. 16th COLT)* (2003) **2777:26–40**

[7] E. TAKIMOTO AND M. WARMUTH, Path Kernels and Multiplicative Updates, *Journal of Machine Learning Research* (2003) **4:773–818**

[8] VLADIMIR VOVK, A Game of Prediction with Expert Advice, *Journal of Computer and System Sciences* (1998) **56:153–173**

Compactness of Classifiers by Iterative Compositions of Features

KAZUYA HARAGUCHI

Department of Applied Mathematics and Physics,
Graduate School of Informatics, Kyoto University,
Japan
kazuyah@amp.i.kyoto-u.ac.jp

HIROSHI NAGAMOCHI

Department of Applied Mathematics and Physics,
Graduate School of Informatics, Kyoto University,
Japan
nag@amp.i.kyoto-u.ac.jp

TOSHIHIDE IBARAKI

Department of Informatics, School of Science and Technology,
Kwansei Gakuin University, Japan
ibaraki@ksc.kwansei.ac.jp

Abstract: Classification is one of the most important issues in machine learning. In the n-dimensional binary space $\{0,1\}^n$, we are given a set of examples $x \in \{0,1\}^n$, where each example x is labeled as $y(x)$ by a Boolean function $y : \{0,1\}^n \mapsto \{0,1\}$, called an oracle, and the classification problem asks to find a classifier c, a Boolean function $c : \{0,1\}^n \mapsto \{0,1\}$ that is (approximately) identical to y. We aim at representing y as a compact concept in the spirit of Occam's Razor. As a tool to describe classifiers, we assume a representation model R, on which classifiers can be implemented and the complexity of a representation is defined as its length of description. From our assumption on oracles, we wish to construct a classifier c with small complexity. In this paper, we discuss two representation models, iteratively composed features and decision trees, and prove that, for any classifier c, the former has a representation which is at least as compact as that by the latter.

Keywords: decision trees, iterative composition of features, learning algorithms, machine learning, representation complexity

1 Introduction

Learning Scheme There have been many attempts for understanding mechanism of human learning [1, 9] and for constructing learning models that can provide a high performance machine learning systems [5, 6, 7, 8]. To discuss this problem, we usually assume that we get limited information on a hidden structure called an *oracle*, and we wish to construct an approximately equivalent structure of the oracle. The process of capturing the characteristic of an oracle is called *learning*. Usually the exact form of an oracle is not presented to us. We only assume that an oracle is a compact concept in the spirit of Occam's Razor [2].

A mathematical framework for the above process is given as follows by using Boolean functions. An oracle $y : \{0,1\}^n \mapsto \{0,1\}$ is defined to be a Boolean function in the n-dimensional binary space. y is a hidden Boolean function (i.e., its exact form is not presented to us), but for some vectors $x \in \{0,1\}^n$, the output values $y(x)$ are available. We call such a vector x an *example*. Let X denote the set of examples, called *training set*, and X^1 (resp., X^0) denote the set of examples $x \in X$ such that $y(x) = 1$ (resp., 0); i.e., $X^1 = \{x \in X \mid y(x) = 1\}$ and $X^0 = \{x \in X \mid y(x) = 0\}$. The problem of finding a Boolean function $c = y$ (or $c \simeq y$) from training set X is referred to as *classification*.

We define a *classifier* c as a Boolean function $c : \{0,1\}^n \mapsto \{0,1\}$. The error rate of c for training set X is defined to be

$$e(c) = \frac{1}{|X|}|\{x \in X \mid c(x) \neq y(x)\}|.$$

For a given $\varepsilon \geq 0$, a classifier c with $e(c) \leq \varepsilon$ is called an *ε-classifier*. In particular, we say that a classifier c with $e(c) = 0$ is *consistent* with y under training set X.

A classifier $c : \{0,1\}^n \mapsto \{0,1\}$ with a small error rate seems a good approximation to an oracle y. However, it is rather easy just to choose a consistent classifier c unless its choice is restricted (for example, a consistent c is obtained as a DNF consisting of $|X^1|$ terms each of which corresponds to example $x \in X^1$). In this paper, we assume the following.

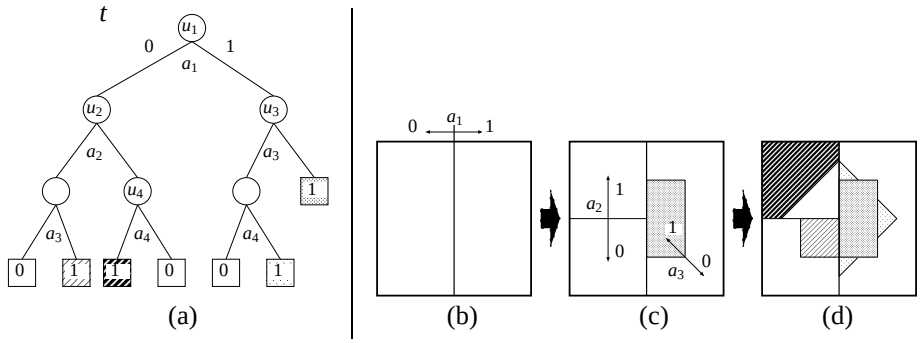


Figure 1: (a) A decision tree $t = \bar{a}_1\bar{a}_2a_3 \vee \bar{a}_1a_2\bar{a}_4 \vee a_1\bar{a}_3a_4 \vee a_1a_3$; (b)-(d) Construction of t

Assumption 1 *An oracle y is a Boolean function that admits a compact form in some representation framework using terminologies of natural languages and/or mathematical expressions.*

We consider that a classifier c is a good approximation of oracle y if the representation complexity needed for c is not much higher than that for y; we say that c "overfits" training set X otherwise. In this sense, we consider that a learning is *successful* if the resulting classifier has not only a small error rate but also has a compact representation.

Machine Learning System Based on the above learning scheme, we now construct a machine learning system. There are two requirements for this purpose. First, we need the way of representing Boolean functions; we have to specify a *representation model R*, which consists of the set of representations of Boolean functions (e.g., graphs, hyperplanes) that can be implemented on computers and the *classifying algorithm* by which an input vector $x \in \{0,1\}^n$ is classified as either 0 or 1 based on some selected representation of R. The other requirement is a *construction algorithm* C_R that constructs one representation of R as our classifier, based on the information of training set X. A *learning model* (R, C_R) is a pair of a representation model R and a construction algorithm C_R. We now see that there are two tasks in order to construct a machine learning system.

1. Selection of a representation model R for classifiers.
2. Design of a construction algorithm C_R for constructing a classifier representation in the framework of R.

This paper focuses on the first task. For this task, there have been proposed a number of representation models such as decision trees T [6], iteratively composed features F [4], and neural networks [7], and so on. Among the above models, F is a new representation model which together with a construction algorithm C_F was recently proposed by Haraguchi et al. [4]. They also conducted numerical experiments on their new learning model (F, C_F) and C4.5 [6], a learning model (T, C_T) based on decision trees, and reported that the former is competitive with the latter in terms of error rates. However, from theoretical or experimental point of view, there has not been the comparison of compactness of classifier representations by these two representation models, T and F. In this paper, we introduce a theoretical framework for defining the complexity of a representation of a Boolean function and its "compactness". Then, in this compactness, for given training set X and oracle y, we prove that the representation model F has a compact ε-classifier whenever T has such one, and also prove that the converse does not hold.

2 Representation Models

This section reviews two representation models, T and F. For a representation r of a classifier, let $\gamma(r)$ denote its complexity. Let $A = \{a_1, a_2, \ldots, a_n\}$ denote the set of n attributes, where $a_j(x) = x_j$, $j = 1, 2, \ldots, n$, for a vector $x \in \{0,1\}^n$.

2.1 Decision Trees

Among many representation models proposed so far, T [3] is known to offer representations whose construction is intuitively easy to understand.

Representation Model A decision tree $t = ((V_t, E_t), \ell)$, a representation of T, consists of a rooted binary tree (V_t, E_t) and a label $\ell : V_t \mapsto A \cup \{0,1\}$ where V_t and E_t denote the set of nodes and edges, respectively. We denote by V_t^{inner} and V_t^{leaf} $(= V \backslash V_t^{inner})$ the sets of inner nodes and leaves in V_t, respectively. We denote the left (resp., right) child of an inner

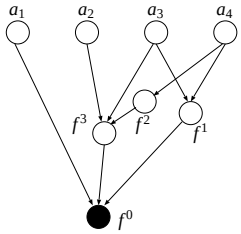


Figure 2: An iteratively composed feature $f = f^0$

node $u \in V_t^{inner}$ by u_{left} (resp., u_{right}). Each inner node $u \in V_t^{inner}$ is associated with an attribute $\ell(u) = a_j \in A$, and each leaf $w \in V_t^{leaf}$ is labeled as $\ell(w) \in \{0,1\}$ (see Fig. 1(a)). The depth of a node u in a rooted tree is defined as the number of edges in the path between u and the root of the tree. The height $h(t)$ of t is defined as the length of the longest path from the root to a leaf. Let V_t^h denote the set of nodes with depth $h \leq h(t)$, and let $V_t^{inner,h}$ (resp., $V_t^{leaf,h}$) denote the set of the inner nodes (resp., leaves) with depth $h \leq h(t)$.

Classifying Algorithm With a decision tree $t = ((V_t, E_t), \ell)$, an input vector $x \in \{0,1\}^n$ is classified as $t(x) \in \{0,1\}$ as follows. We let x visit the tree (V_t, E_t) from the root to a leaf $w \in V_t^{leaf}$, by which $t(x)$ is set to be $\ell(w)$. At each inner node $u \in V_t^{inner}$ which x has just visited, we test whether the attribute $a_j = \ell(u)$ of x is 0 or 1; if $a_j(x) = 1$ (resp., 0), then let x move to the right child u_{right} (resp., left child u_{left}) of the current node u. By repeating this until x reaches a leaf $w \in V_t^{leaf}$, we set $t(x) := \ell(w)$. In this paper, we assume that each attribute $a_j \in A$ appears at most once in the path from the root to any leaf, since all binary vectors $x \in \{0,1\}^n$ encountering the second test of the same attribute in a path exactly move to the same child; the unvisited child and its descendants are redundant. It follows that $h(t) \leq n$ holds for any decision tree t.

Complexity of a Representation We see that a decision tree $t = ((V_t, E_t), \ell)$ can be implemented in $O(|V_t| + |E_t|)$ space. Then we define the complexity $\gamma(t)$ of a decision tree t by

$$\gamma(t) = |V_t| + |E_t|.$$

Construction Algorithm A typical algorithm for constructing a decision tree t from given training set first prepares a single node as a rooted tree, and repeats adding a pair of new leaves to a leaf of the current tree, which is called a *branching*. Note that the root corresponds to the entire binary space $\{0,1\}^n$. Branching at the root u divides the space into two subspaces $\{x \in \{0,1\}^n \mid a_j(x) = 0\}$ and $\{x \in \{0,1\}^n \mid a_j(x) = 1\}$ for the attribute $a_j = \ell(u)$, where the resulting subspaces respectively correspond to new children u_{left} and u_{right} of u. Similarly, branching at a leaf u in the current tree divides its subspace into two subspaces according to the value of associated attribute $a_j = \ell(u)$. Finally, the resulting subspace corresponding to each leaf w is labeled as either 0 or 1. This process is illustrated in Fig. 1(b)-(d), where those subspaces labeled as 1 are depicted by shaded areas. There have been proposed many learning models (T, C_T) of decision trees such as C4.5 [6], SPRINT [8], and so on [5], where different methods of choosing the nodes to be branched or branching attributes are used among these models.

Before closing this subsection, we introduce some terminology for the next section. Let X^v denote the set of examples $x \in X$ that visit a node $v \in V_t$, and let $X^{v,1} = \{x \in X^v \mid y(x) = 1\}$ and $X^{v,0} = \{x \in X^v \mid y(x) = 0\}$. Consider two examples $x, x' \in X^u$ for an inner node u. We say that x and x' are *separated* at u if x and x' visit different children of u. Clearly, if a decision tree represents a consistent classifier, any two examples $x^1 \in X^1$ and $x^0 \in X^0$ reach different leaves, and thus are separated at some inner node.

2.2 Iteratively Composed Features

Representation Model Let $S = \{z_1, z_2, \ldots, z_k\}$ denote a set of k binary variables, where each z_j, $j = 1, 2, \ldots, k$, is either an attribute in A or what we call a *feature*, which is also a binary variable. We write the set of all vectors whose components are the values of variables in S by $\{0,1\}^S$ (for example, if $S = \{z_1, z_2\}$, then $\{0,1\}^S = \{(0,0),(0,1),(1,0),(1,1)\}$). A feature f_S, composed of a subset S of original attributes A and the features already available, is a Boolean function of variables in S; i.e., $f_S : \{0,1\}^S \mapsto \{0,1\}$. Note that this f_S can also be used as a variable of other features.

An iteratively composed feature f, a representation of F, consists of acyclic digraph $f = (V_f = A \cup \mathscr{F}, E_f)$, where V_f and E_f denote the set of nodes and arcs respectively. Each node of V_f corresponds to either an attribute $a_j \in A$ or to a feature $f_S \in \mathscr{F}$, where $\mathscr{F}$ denotes a feature f and the set of features used in the composition of f. In the digraph of f, there are n sources, each of which is an attribute $a_j \in A$, and there is one sink, which is f. The input variables of a feature $f_S \in V_f$ is

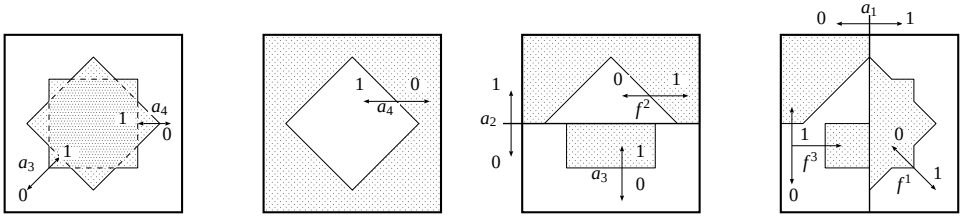


Figure 3: Construction of f^1, f^2, f^3 and f^0 in Fig. 2

given by incoming arcs in E_f. Figure 2 illustrates a representation of a feature $f = f^0$; e.g., feature f^1 is composed of a_3 and a_4 (i.e., $f^1 = f_{\{a_3,a_4\}}$), and f^1 is used for the construction of $f^0 = f_{\{a_1,f^3,f^1\}}$.

Classifying Algorithm For an input vector $x \in \{0,1\}^n$, x is classified as $f(x) \in \{0,1\}$ as follows: For all nodes of V_f in the graph $f = (V_f, E_f)$, we determine their values as follows. For each source of f, that is an attribute a_j $(j = 1,2,\ldots,n)$, its value is set to be $a_j(x) \in \{0,1\}$. After determining the values of all sources, we repeat choosing a node $f_S \in \mathscr{F}$ such that the values of all its fan-ins (i.e., the set of nodes that have arcs entering f_S) have already been determined, and determine the value of f_S according to the values of its fan-ins. The value of the sink f gives $f(x)$ for the input vector x.

Complexity of a Representation A feature f_S can be implemented by storing $2^{|S|}$ values $f_S(z)$ for all vectors $z \in \{0,1\}^S$. Then the complexity $\gamma(f)$ of a feature f is given by

$$\gamma(f) = |V_f| + |E_f| + \sum_{f_S \in V_f} 2^{|S|}.$$

Construction Algorithm Construction of a feature f_S in the n-dimensional binary space $\{0,1\}^n$ is to divide the space into $2^{|S|}$ subspaces and to assign either 0 or 1 to each subspace. This is illustrated in Figure 3. In the construction of $f^1 = f_{\{a_3,a_4\}}$, for example, the binary space $\{0,1\}^n$ is divided into four subspaces according to the values of a_3 and a_4, and $f_S(z)$ in this case is defined to be $f_{\{a_3,a_4\}} = a_3 \vee a_4$, as shown in the figure.

3 Compactness of Representations

This section compares compactness of two representation models, F and T. Let $\mathscr{U} = \{c \mid c : \{0,1\}^n \mapsto \{0,1\}\}$ denote the set of all n-dimensional Boolean functions (hence $|\mathscr{U}| = 2^{2^n}$), and let c denote a Boolean function in $\mathscr{U}$. $R(c)$ denotes the set of all representations for c in a representation model R (possibly $R(c) = \emptyset$). Let us call $\gamma_R^*(c)$ *compactness* of c by R, defined as the smallest complexity of a representation for c by R, i.e.,

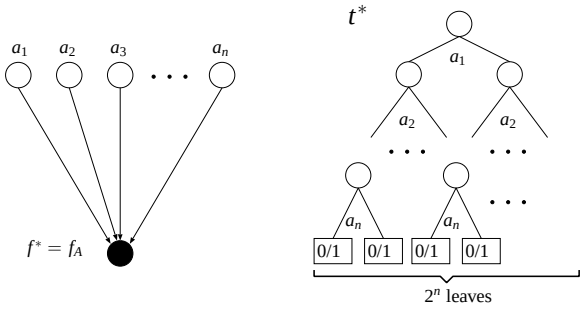
$$\gamma_R^*(c) = \min\{\gamma(r) \mid r \in R(c)\}.$$

First, we show that F and T have the ability to represent any Boolean function in $\mathscr{U}$. Let $\mathscr{U}_F$ (resp., $\mathscr{U}_T$) denote the set of Boolean functions in $\mathscr{U}$ for which F (resp., T) has some representations (i.e., $\mathscr{U}_F = \{c \in \mathscr{U} \mid F(c) \neq \emptyset\}$, $\mathscr{U}_T = \{c \in \mathscr{U} \mid T(c) \neq \emptyset\}$).

Theorem 2 *It holds* $\mathscr{U}_F = \mathscr{U}_T = \mathscr{U}$.

PROOF: It is sufficient to show that there exists a representation $f^* \in F(c)$ and $t^* \in T(c)$ for any Boolean function $c \in \mathscr{U}$. Let $f^* = f_A$ be a feature for the set of n attributes A, where, for each $x \in \{0,1\}^n$, the value of $f_A(x)$ is set to be $c(x)$ (see Fig. 4(a)). Then we see that c is represented by this $f^* = f_A$. Now let t^* be a decision tree with $h(t^*) = n$, where all the inner nodes $u \in V_{t^*}^{inner,h}$, $h = 0,1,\ldots,n-1$, are associated with the $(h+1)$-st attribute (i.e., $\ell(u) = a_{h+1}$). See Fig. 4(b). G_{t^*} has 2^n leaves and each leaf w corresponds to a binary vector $x \in \{0,1\}^n$. For each leaf $w \in V_{t^*}^{leaf}$, label $\ell(w)$ is set to be $c(x)$ for the corresponding vector x. Then we see that c is represented by this t^*. □

Note that $\gamma(f^*) = \Omega(2^n)$ and $\gamma(t^*) = \Omega(2^n)$ for the representations f^* and t^* in the proof of this theorem. From our assumption on the oracle y, f^* and t^* may not be the classifier representations of our interest; we are interested in more compact representations. The following theorem states that, for any Boolean function $c \in \mathscr{U}$, we can construct a feature $f \in F(c)$ from any decision tree $t \in T(c)$ such that its complexity $\gamma(f)$ is bounded by $O(\gamma(t))$.



(a) (b)

Figure 4: Representations of (a) $f^* = f_A$ and (b) t^*

Theorem 3 *For any Boolean function $c \in \mathscr{U}$ and any decision tree $t \in T(c)$, there exists a feature $f \in F(c)$ such that $\gamma(f) = O(\gamma(t))$.*

PROOF: Let t^v denote the subtree of $t = ((V_t, E_t), \ell)$ which has node $v \in V_t$ as its root node, and let c^v denote a Boolean function represented by t^v (i.e., $t^v \in T(c^v)$). For any node $v \in V_t$, we show that there exists a feature $f^v \in F(c^v)$ such that $\gamma(f^v) \leq d\gamma(t^v)$ for some constant $d > 0$.

We prove this by an induction on the height h of node v. Assume $h = h(t)$. Then, v is a leaf in t, and subtree t^v consists of one node v, which represents a constant function c^v that outputs $\ell(v)$ for any input vector $x \in \{0,1\}^n$. We choose a feature $f^v \in F(c^v)$ that has one fan-in (e.g., $f^v = f_{\{a_j\}}, \forall a_j \in A$) and that outputs $\ell(v)$ for any input $\{0,1\}$. Then, $\gamma(f^v) \leq d \leq d\gamma(t^v)$ for some constant $d > 0$.

Assume that the above statement holds for $h = h(t), h(t)-1, \ldots, h'$, and let $v \in V_t^{h'-1}$. If v is a leaf, we obtain $f^v \in F(c^v)$ such that $\gamma(f^v) \leq d\gamma(t^v)$ in the same way as above. Otherwise (i.e., if v is an inner node), subtree t^v represents a Boolean function $c^v = \bar{a}_j c^{v_{left}} \vee a_j c^{v_{right}}$, where $a_j = \ell(v)$, $a_j \in A$. Then, we choose a feature $f^v = f_{\{a_j, f^{v_{left}}, f^{v_{right}}\}} = \bar{a}_j f^{v_{left}} \vee a_j f^{v_{right}}$. Since $f^{v_{left}} \in F(c^{v_{left}})$ and $f^{v_{right}} \in F(c^{v_{right}})$ hold from the assumption, $f^v \in F(c^v)$ holds. Therefore, we have $\gamma(f^v) = \gamma(f^{v_{left}}) + \gamma(f^{v_{right}}) + 1 + 2 + 2^1 \leq d\gamma(t^{v_{left}}) + d\gamma(t^{v_{right}}) \leq d\gamma(t^v)$. □

Note that the feature $f = f^0 \in F(c)$ in Fig. 2 is constructed from the decision tree $t \in T(c)$ in Fig. 1, where $c = \bar{a}_1 \bar{a}_2 a_3 \vee \bar{a}_1 a_2 \bar{a}_4 \vee a_1 \bar{a}_3 a_4 \vee a_1 a_3$; e.g., for four nodes u_1 to u_4 in Fig. 1, $f^{u_1} = f^0$, $f^{u_2} = f^3$, $f^{u_3} = f^1$, and $f^{u_4} = f^2$ in Fig. 2, respectively. (Features for constant functions are omitted in Fig. 2.)

Corollary 4 *Given a training set X, for any ε-classifier c for some $\varepsilon \in [0,1]$ and any decision tree $t \in T(c)$, there exist an ε-classifier c' and a feature $f \in F(c')$ such that $\gamma(f) = O(\gamma(t))$.*

In what follows, we consider the converse of the above relation; is there any compact representation of T for such a classifier c that admits a compact representation of F?

Theorem 5 *Given training set $X = \{0,1\}^n$, there exists an oracle $y \in \mathscr{U}$ such that $\gamma_F^*(c) = O(n)$ and $\gamma_T^*(c) = \Omega(2^n)$ hold for any classifier $c \in \mathscr{U}$ with $e(c) = 0$.*

PROOF: Let oracle y be a parity function such that

$$y(x) = \begin{cases} 1 & \text{if } \sum_{j=1}^n x_j \text{ is odd,} \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Let $c \in \mathscr{U}$ denote a classifier such that $e(c) = 0$. First, we show that there exists a feature $f \in F(c)$ whose complexity $\gamma(f)$ is bounded by $O(n)$. For each attribute $a_j \in A$, $j = 1, 2, \ldots, n-1$, we construct feature f^j as follows:

$$f^j = \begin{cases} f_{\{a_j, a_{j+1}\}} = \bar{a}_j a_{j+1} \vee a_j \bar{a}_{j+1} & \text{if } j = 1, \\ f_{\{f^{j-1}, a_{j+1}\}} = \bar{f}^{j-1} a_{j+1} \vee f^{j-1} \bar{a}_{j+1} & \text{otherwise.} \end{cases} \tag{2}$$

Clearly, $f^j(x) = 1$ if and only if $\sum_{j'=1}^{j+1} x_{j'}$ is odd; it follows that $f^{n-1} \in F(c)$. The complexity $\gamma(f)$ of $f = f^{n-1}$ is bounded as follows: As for the graph $f = (V_f, E_f)$, $|V_f|$, the number of attributes and features in the composition of f, is $n + (n-1) = 2n-1$, the number of fan-ins of each inner node $u \in V_f^{inner}$ is 2, and thus $|E_f| = 2(n-1)$. Since $\gamma(f) = O(n)$, $\gamma_F^*(c) = \min\{\gamma(f) \mid f \in F(c)\} = O(n)$.

Next, we show that complexity of any decision tree $t \in T(c)$ for classifier c, consistent with y under $X = \{0,1\}^n$, requires $\Omega(2^n)$. For this, we show the following two lemmas.

Lemma 6 *For any $A' \subset A$ and any $x^1 \in X^1$, there exists an example $x^0 \in X^0$ such that $x^1|_{A'} = x^0|_{A'}$ (i.e., $a_j(x^1) = a_j(x^0)$ holds for any $a_j \in A'$).*

PROOF: It is sufficient to show that such an x^0 exists for any $A' = A\backslash\{a_j\}$, $j = 1,2,\ldots,n$, and any $x^1 \in X^1$. Since we have all the n-dimensional vectors as $X = (X^1, X^0)$, we find exactly one example $x^0 \in X^0$ satisfying $x^1|_{A'} = x^0|_{A'}$, by taking $x^0 = (x^1_1, x^1_2, \ldots, x^1_{j-1}, \overline{x^1_j}, x^1_{j+1}, \ldots, x^1_n)$. □

Lemma 7 *For any $A' = A\backslash\{a_j\}$, $j = 1,2,\ldots,n$, and any $x^1 \in X^1$, there exists no $x \in X^1$ $(x^1 \neq x)$ such that $x^1|_{A'} = x|_{A'}$ holds.*

PROOF: Assume that there exist two examples $x^1, x \in X^1$ such that $x^1|_{A'} = x|_{A'}$ for some $A' = A\backslash\{a_j\}$. Since $x^1 \neq x$, $a_j(x^1) \neq a_j(x)$ holds; it follows that either $x^1 \in X^0$ or $x \in X^0$ holds as y is a parity function, which contradicts the assumption. □

From Lemma 6, for each $x^1 \in X^1$, there exists an example $x^0 \in X^0$ which visits the same nodes as x^1 visits, in the depth of less than $n-1$. Then, since decision tree t is consistent, x^1 and x^0 are separated at an inner node $u \in V_t^{inner,n-1}$ in the depth of $n-1$. (Note that $h(t) \leq n$.) From Lemma 7, there is no $x \in X^1$ $(x^1 \neq x)$ that visits u, and thus there is no example $x' \in X^0$ that visits u. Since there are $|X^1| = 2^n/2 = 2^{n-1}$ examples for x^1, $|V_t^{inner,n-1}| = 2^{n-1}$ holds. Then the number of leaves $|V_t^{leaf}|$ equals $2 \cdot 2^{n-1} = 2^n$. Hence, $\gamma(t) = \Omega(2^n)$. □

Theorem 8 *Given a training set $X \subseteq \{0,1\}^n$ of 2^k examples (i.e., $|X| = 2^k$), for any integer $k \in [1,n]$, there exists an oracle $y \in \mathscr{U}$ such that $\gamma^*_F(c) = O(k)$ and $\gamma^*_T(c) = \Omega(2^k)$ hold for any classifier $c \in \mathscr{U}$ with $e(c) = 0$.*

PROOF: We associate each k-dimensional vector z in $\{0,1\}^k$ with a n-dimensional vector x^z whose the first k attribute values $a_1(x), a_2(x), \ldots, a_k(x)$ are the same as in z and the values of for the rest of $n-k$ attributes are set by duplicating these k values, i.e., $a_{sk+j}(x) = a_j(x)$, $j = 1,2,\ldots,k$, and $sk+j \leq n$.

Let oracle y be the parity function of the first k attributes $a_1, a_2, \ldots, a_k$, as given in (1). Clearly, we can construct a feature $f \in F(c)$ from the first k attributes such that $e(c) = 0$ and $\gamma(f) = O(k)$ as in the proof for Theorem 5.

On the other hand, we see that there is no decision tree $t \in T(c)$ such that $e(c) = 0$ and $\gamma(t) = o(2^k)$, since otherwise we would have obtained a decision tree t with $\gamma(t) = o(2^n)$ in Theorem 5. Thus $\gamma^*_T(c) = \Omega(2^k)$. □

Theorem 9 *Given a training set $X = \{0,1\}^n$, for every $\varepsilon \in [0, 1/2)$, there exists an oracle $y \in \mathscr{U}$ such that $\gamma^*_F(c) = O(n)$ and $\gamma^*_T(c) = \Omega((1-2\varepsilon)2^n)$ hold for any classifier $c \in \mathscr{U}$ with $e(c) \leq \varepsilon$.*

PROOF: Let oracle y be the parity function as given in (1). From the proof for Theorem 5, there is a feature $f \in F(c)$ such that $e(c) = 0 \leq \varepsilon$ and $\gamma(f) = O(n)$, and thus $\gamma^*_F(c) = O(n)$.

Now we consider a decision tree $t = ((V_t, E_t), \ell) \in T(c)$. It suffices to show that t contains at least $|X|(1-2\varepsilon) = (1-2\varepsilon)2^n$ leaves, from which $\gamma(t) = \Omega((1-2\varepsilon)2^n)$follows. By definition of decision trees, the error rate of c is given by

$$\begin{aligned} e(c) &= \frac{1}{|X|}\Big(\sum_{w \in V_t^{leaf}:\, \ell(w)=1} |X^{w,0}| + \sum_{w \in V_t^{leaf}:\, \ell(w)=0} |X^{w,1}|\Big) \\ &\geq \frac{1}{|X|}\Big(\sum_{w \in V_t^{leaf} \backslash V_t^{leaf,n}:\, \ell(w)=1} |X^{w,0}| + \sum_{w \in V_t^{leaf} \backslash V_t^{leaf,n}:\, \ell(w)=0} |X^{w,1}|\Big). \end{aligned} \tag{3}$$

Claim 10 *For each $w \in V_t^{leaf} \backslash V_t^{leaf,n}$, it holds $|X^{w,0}| = |X^{w,1}|$.*

PROOF: Let $w \in V_t^{leaf,h}$ be a leaf whose depth is $h(< n)$, and A^w denote the set of all attributes used in the path between w and the root. Suppose that we repeat branching at w and its resulting children recursively, by the attributes $A' = A\backslash A^w$ until each of the resulting leaves has depth n. the parent u of each of these leaves has depth $n-1$, we see that $|X^{u,1}| = |X^{u,0}| = 1$ holds from the proof of Theorem 5. This means that $|X^{w,1}| = |X^{w,0}|$ holds. □

By this claim, we see that (3) can be written as follows.

$$e(c) \geq \frac{1}{|X|} \sum_{w \in V_t^{leaf} \backslash V_t^{leaf,n}} \frac{|X^w|}{2} = \frac{1}{2|X|}(|X| - |V_t^{leaf,n}|) = \frac{1}{2^{n+1}}(2^n - |V_t^{leaf,n}|).$$

Since c satisfies $e(c) \leq \varepsilon$, we have $\varepsilon \geq \frac{1}{2^{n+1}}(2^n - |V_t^{leaf,n}|)$. Hence $|V_t^{leaf,n}| \geq 2^n - \varepsilon 2^{n+1}$ holds, as required. □

Theorem 11 *Given a training set $X \subseteq \{0,1\}^n$ of 2^k examples (i.e., $|X| = 2^k$), for any integer $k \in [1,n]$ and every $\varepsilon \in [0,1/2)$, there exists an oracle $y \in \mathscr{U}$ such that $\gamma_F^*(c) = O(k)$ and $\gamma_T^*(c) = \Omega((1-2\varepsilon)2^k)$ hold for any classifier $c \in \mathscr{U}$ with $e(c) \leq \varepsilon$.*

PROOF: Let X be the set of examples given in Theorem 8, and oracle y be the parity function of the first k attributes $a_1, a_2, \ldots, a_k$ as given in (1). Then, we can clearly construct a feature $f \in F(c)$ from the first k attributes such that $e(c) = 0 \leq \varepsilon$ and $\gamma(f) = O(k)$ as in the proof for Theorem 5.

Since $a_j(x) = a_{k+j}(x) = a_{2k+j}(x) = \ldots$ $(j = 1,2,\ldots,k)$ for all examples $x \in X$, any decision tree can be constructed by using the first k attributes. Then, it is sufficient to consider constructing an ε-classifier from training set $X' = \{0,1\}^k$ of 2^k examples with the first k attributes of A. From Theorem 9, the required complexity is $\Omega((1-2\varepsilon)2^k)$. □

4 Conclusion

In this paper, we defined the compactness of a representation model, and showed that iteratively composed features F is not inferior to decision trees T in terms of this compactness. However, this does not immediately imply that the representation model F always provides a learning model that produces a compact representation; there remains an important task to discover a construction algorithm that actually finds a compact representation by F for a given set of examples. The current algorithm employed in the learning model (F, C_F) [4] constructs a representation by repeatedly attaching new features without trying to reduce the complexity of representations being constructed. It is our future work to design a learning model by F by taking the complexity of representations into account, and to conduct a computational experiment on the resulting error rate by the new model.

References

[1] D. ANGLUIN, Queries and Concept Learning, *Machine Learning* (1988) **2**, pp. 319-342.

[2] A. BLUMER, A. ENRENFEUCHT, D. HAUSSLER, M. K. WARMUTH, Occam's Razor, *Information Processing Letters* (1987) **24**, pp. 377-380.

[3] L. BREIMAN, J. H. FREIDMAN, R. A. OLSHEN, C. J. STONE, Classification and Regression Trees, Wadsworth International Group (1984).

[4] K. HARAGUCHI, T. IBARAKI, E. BOROS, Classifiers Based on Iterative Compositions of Features, *Proceedings of the 1st International Conference on Knowledge Engineering and Decision Support* (2004), pp. 143-150.

[5] M. GAROFALAKIS, D. HYUN, R. RASTOGI, K. SHIM, Building Decision Trees with Constraints, *Data Mining and Knowledge Discovery* (2003) **7**, pp. 187-214.

[6] J. R. QUINLAN, C4.5: Programs for Machine Learning, Morgan Kaufmann (1993).

[7] B. D. RIPLEY, Pattern Recognition And Neural Networks, Cambridge University Press (1996).

[8] J. C. SHAFER, R. AGRAWAL, M. MEHTA, SPRINT: A Scalable Parallel Classifier for Data Mining, *Proceedings of the 22nd International Conference on Very Large Data Bases* (1996), pp. 544-555.

[9] L. G. VALIANT, A Theory of the Learnable, *Communications of the ACM* (1984) **27**, pp. 1134-1142.

Greedy Fans: A geometric approach to dual greedy algorithms

HIROSHI HIRAI

Research Institute for Mathematical Sciences
Kyoto University
Kyoto, 606-8502, Japan
hirai@kurims.kyoto-u.ac.jp

Abstract: The purpose of this paper is to understand greedily solvable linear programs in a geometric way. Such linear programs have recently been considered by Faigle and Kern, and Krüger for antichains of posets, and by Frank for a class of lattice polyhedra, and by Kashiwabara and Okamoto for extreme points of abstract convex geometries. Our guiding principle is that solving linear programs is equivalent to finding a normal cone of a polyhedron which contains a given vector. Motivated by this observation, we introduce and investigate a class of simplicial fans, called *greedy fans*, whose membership problem can be greedily solvable.

Keywords: dual greedy algorithm, regular triangulation, submodularity

1 Introduction

In this paper, given a finite set V, a nonempty family $\mathscr{A} \subseteq 2^V$ and a function $f : \mathscr{A} \to \mathbf{R}$, we consider the following dual pair of linear programs

$$\begin{array}{l|l} \text{[P]} & \text{[D]} \\ \max. \quad \sum_{e\in V} w(e)x(e) & \min. \quad \sum_{A\in\mathscr{A}} \lambda(A) f(A) \\ \text{s.t.} \quad \sum_{e\in A} x(e) \le f(A)\ (A \in \mathscr{A}), & \text{s.t.} \quad \sum_{A\in\mathscr{A}: e\in A} \lambda(A) = w(e)\ (e \in V), \\ \quad x \in \mathbf{R}^V, & \quad \lambda(A) \ge 0\ (A \in \mathscr{A}). \end{array} \tag{1}$$

Since many combinatorial optimization problems can reduced to this form, it is important to characterize efficiently solvable classes of this type of LPs. A polymatroid [2] is such an example. Namely, linear programs over polymatroids are greedily solvable.

Since a recent work by Faigle and Kern [3] about submodular programs on rooted forests, several researchers [4], [10], [1], [9], [6] have investigated greedily solvable linear programs, where so-called *dual greedy algorithms* construct a dual optimal solution in a greedy way. In particular, Frank [4] considered dual greedy algorithms for a class of lattice polyhedra. Krüger [10] considered for antichains of posets (see also [3]), and Kashiwabara and Okamoto [9] considered for extreme points of abstract convex geometries (see also [6]).

The purpose of this paper is to understand these greedily solvable linear programs in a geometric way. Our guiding principle is the following fact.

> Solving this dual linear program [D] is equivalent to finding a normal cone of the primal feasible polyhedron which contains a given cost vector w.

That is a membership problem of the normal fan of a polyhedron. Motivated by this observation, we introduce a special class of simplicial fans whose membership problem can be greedily solved. We call such a simplicial fan a *greedy fan*. With the aid of recent developments of regular triangulations [7], [12], we provide a unified framework for these dual greedy algorithms.

This paper is organized as follows. In Section 2, we introduce the concept of greedy fans. We show that every greedy fan can be represented by certain multiple-choice function. We define *submodular functions* for a greedy fan and derive their defining inequalities (*submodularity inequalities*). We show that the set of all submodular functions for a greedy fan has an interior point if and only if the greedy fan is *regular*, i.e., there exists some polyhedron whose normal fan coincides with the greedy fan. In Subsection 2.3, we investigate a special class of greedy fans called *acyclic greedy fans*, which can be represented by some posets. We show that every acyclic greedy fan is *regular*. Analogously to the *secondary fan* [7], the set of all acyclic greedy fans can be naturally regarded as a certain kind of polyhedral fan. We call this polyhedral fan a *secondary greedy fan*. Furthermore, we show that if this secondary greedy fan is regular, it coincides with the normal fan of the base polyhedron associated with some (ordinary) submodular function.

2 Greedy Fans

In this section, motivated by dual greedy algorithms, we introduce the concept of a greedy fan. We need some basic notation. Let V be a (nonempty) finite set. $\mathbf{R}$ and $\mathbf{R}_+$ denote the set of real numbers and nonnegative real numbers, respectively. For a function $f : \mathscr{A} \to \mathbf{R}$ on a set $\mathscr{A}$, the *nonzero support* $\operatorname{supp} f$ is defined by $\{A \in \mathscr{A} \mid f(A) \neq 0\}$. For a subset $A \subseteq V$, the characteristic vector $\chi_A \in \mathbf{R}^V$ is defined as

$$\chi_A(e) = \begin{cases} 1 & \text{if } e \in A, \\ 0 & \text{otherwise.} \end{cases}$$

We need to recall the basic definitions about polyhedral subdivisions (see [17] for details). A set of polyhedral cones Δ is said to be a *polyhedral fan* if every face of any $P \in \Delta$ is in Δ, and the intersection of any two member $P, Q \in \Delta$ is the common face of P and Q. We denote by $|\Delta|$ the union of all member of Δ. Δ is also called a *polyhedral cone subdivision* of $|\Delta|$. If every member of Δ is a simplicial cone, we call Δ a *simplicial fan* or a *simplicial cone subdivision*.

2.1 Greedy Fans, Dual Greedy Algorithms, and Submodular Functions

We consider a simplicial cone subdivisions Δ of $\mathbf{R}_+^V$ with the following additional property, where we call a 1-dimensional cone a *vertex*.

Each vertex of Δ can be expressed by $\mathbf{R}_+\chi_A$ for some nonempty set $A \subseteq V$.

Let $\mathscr{A} = \mathscr{A}_\Delta \subseteq 2^V$ be a nonempty family defined as

$$\mathscr{A} = \{A \subseteq V \mid \mathbf{R}_+\chi_A \text{ is a vertex of } \Delta\}.$$

In particular, Δ can be regarded as an *abstract simplicial complex* on the vertex set $\mathscr{A}$ which is denoted by $\hat{\Delta}$. We shall often identify Δ with $\hat{\Delta}$. For a nonempty $X \subseteq V$, we define a restriction Δ^X to be $\{\mathscr{C} \in \Delta \mid \mathscr{C} \subseteq \mathbf{R}_+^X\}$. Note that Δ^X is a simplicial cone subdivision of $\mathbf{R}_+^X$.

We define *greedy fans* recursively. If $\#V = 1$, trivial simplicial cone subdivision of $\mathbf{R}_+^V$ is defined to be greedy. Now we suppose that we have already defined the set of all greedy fans of $\mathbf{R}_+^U$ with $\#U < \#V$. A simplicial cone subdivision Δ of $\mathbf{R}_+^V$ is said to be *greedy* if there exists a nonempty $A \subseteq V$ such that

(G1) every maximal cone of Δ contains $\mathbf{R}_+\chi_A$ as a vertex and

(G2) for any $e \in A$, a restriction $\Delta^{V\setminus\{e\}}$ is a greedy fan of $\mathbf{R}_+^{V\setminus\{e\}}$.

We call a vertex satisfying (G1) and (G2) of a greedy fan Δ a *center vertex*. First, we see that every restriction of a greedy fan is a greedy.

Lemma 1 *For a greedy fan Δ of $\mathbf{R}_+^V$ and a nonempty $X \subseteq V$, a restriction Δ^X is also a greedy fan.*

From this lemma, we can define a function $\Phi_\Delta : 2^V \to 2^{\mathscr{A}}$ as

$$\Phi_\Delta(X) = \{A \subseteq V \mid A \text{ is a center vertex of } \Delta^X\} \quad (X \subseteq V,\ X \neq \emptyset) \tag{2}$$

and $\Phi_\Delta(\emptyset) = \emptyset$ for convenience.

Next we consider the membership problem of finding a member of Δ which contains a given vector $w \in \mathbf{R}_+^V$. For this, we try to find a conical expression of w of a member of Δ. Consider the following procedure, where $\Phi = \Phi_\Delta$.

Procedure: `Dual_Greedy`

Input: A vector $w \in \mathbf{R}_+^V$.

Output: $\lambda \in \mathbf{R}_+^{\mathscr{A}}$ with $w = \sum_{A\in\mathscr{A}} \lambda(A)\chi_A$.

Initialization: $w' \leftarrow w$, $X \leftarrow V$, $\lambda(A) \leftarrow 0$ $(\forall A \in \mathscr{A})$.

step1: If $X = \emptyset$, then stop.

step2: Pick arbitrary $A^* \in \Phi(X)$ and $e^* \in \operatorname{Argmin}\{w'(e) \mid e \in A^*\}$.

step3: Put $\lambda(X) \leftarrow w'(e^*)$ and $w' \leftarrow w' - w'(e^*)\chi_{A^*}$.

step4: Put $X \leftarrow X \setminus \{e^*\}$ and go to **step1**.

Variants of this procedure have been considered by several authors [3], [4], [10], [1], [9], [6] to obtain an optimum of linear program [D]. This formulation using $\Phi : 2^V \to 2^{\mathscr{A}}$ is essentially due to Fujishige [6]. In fact, this procedure solves the membership problem of Δ as follows.

Proposition 2 *For a greedy fan Δ with vertex set $\mathscr{A} \subseteq 2^V \setminus \{\emptyset\}$ and a nonnegative vector $w \in \mathbf{R}^V_+$, let $\lambda : \mathscr{A} \to \mathbf{R}$ be an output of* `Dual_ Greedy` *for input vector w. Then we have*

$$\mathrm{cone}\{\chi_A \mid A \in \mathrm{supp}\,\lambda\} \in \Delta.$$

Next we try to construct a polyhedron whose normal fan coincides with a given greedy fan Δ. For a function $f : \mathscr{A} \to \mathbf{R}$, we define a polyhedron $P(f)$ as

$$P(f) = \{x \in \mathbf{R}^V \mid \sum_{e \in A} x(e) \leq f(A)\ (A \in \mathscr{A})\} \tag{3}$$

Note that $P(f)$ is the feasible region of [P] in 1. A greedy fan Δ with vertex set $\mathscr{A} \subseteq 2^V$ is *regular* if there exists a function $f : \mathscr{A} \to \mathbf{R}$ such that Δ coincides with the normal fan of polyhedron $P(f)$. Let $(\hat{\Delta})^* \subseteq 2^{\mathscr{A}}$ be a family defined as

$$(\hat{\Delta})^* = \{\mathscr{F} \subseteq \mathscr{A} \mid \mathscr{F} \notin \hat{\Delta},\ \forall \mathscr{F}' \subset \mathscr{F},\ \mathscr{F}' \in \hat{\Delta}\}. \tag{4}$$

For $\mathscr{F} \in (\hat{\Delta})^*$, let $\lambda^{\mathscr{F}} : \mathscr{A} \to \mathbf{Z}$ be defined as the solution of `Dual_Greedy` for input vector $w = \sum_{A \in \mathscr{F}} \chi_A$. We define *submodularity inequalities* for Δ as

$$\sum_{A \in \mathscr{F}} f(A) \geq \sum_{A \in \mathscr{A}} \lambda^{\mathscr{F}}(A) f(A) \quad (\mathscr{F} \in (\hat{\Delta})^*) \tag{5}$$

A function $f : \mathscr{A} \to \mathbf{R}$ is said to be *submodular* if it satisfies submodularity inequalities. Let *submodular cone* $\mathscr{S}_\Delta \subseteq \mathbf{R}^{\mathscr{A}}$ be defined as the set of all submodular functions on $\mathscr{A}$. We denote $\mathrm{int}\,\mathscr{S}_\Delta$ by the set of all interior point of $\mathscr{S}_\Delta$.

Theorem 3 *Suppose the submodular cone $\mathscr{S}_\Delta$ has interior points. Consider linear program* [D] *for a function $f : \mathscr{A} \to \mathbf{R}$.* `Dual_Greedy` *produces an optimal dual solution of* [D] *for any nonnegative cost vector if and only if f is submodular for Δ. In particular, $f \in \mathrm{int}\,\mathscr{S}_\Delta$ if and only if Δ coincides with the normal fan of $P(f)$.*

PROOF: Only-if-part of the first statement follows from the definition of the submodularity inequalities (5). We show if-part. We can take $g \in \mathrm{int}\,\mathscr{S}_\Delta$. Consider linear program [D] with a submodular function f and a cost vector $w \in \mathbf{R}^V_+$. Since both [P] and [D] are feasible, [D] has an optimal solution. We take an optimal solution λ^* of [D] which minimizes the value $\sum_{A \in \mathscr{A}} \lambda^*(A) g(A)$. We claim $\mathrm{supp}\,\lambda^* \in \hat{\Delta}$. If so, λ^* must be the output of `Dual_Greedy`. Suppose that $\mathrm{supp}\,\lambda^* \notin \hat{\Delta}$. Then there exists $\mathscr{F} \in (\hat{\Delta})^*$ such that $\mathscr{F} \subseteq \mathrm{supp}\,\lambda^*$. Let $\tilde{\lambda}$ be defined as

$$\tilde{\lambda}(A) = \begin{cases} \lambda^*(A) - \mu & \text{if } A \in \mathscr{F}, \\ \lambda^*(A) + \mu\lambda^{\mathscr{F}}(A) & \text{if } A \in \mathrm{supp}\,\lambda^{\mathscr{F}}, \\ \lambda^*(A) & \text{otherwise}, \end{cases} \qquad (A \in \mathscr{A}), \tag{6}$$

where $\mu = \min\{\lambda^*(A) \mid A \in \mathscr{F}\} > 0$. From $\sum_{A \in \mathscr{F}} \chi_A = \sum_{A \in \mathscr{A}} \lambda^{\mathscr{F}}(A)\chi_A$, We see that $\tilde{\lambda}$ is also feasible to [D]. Furthermore, by submodularity of f, the objective value of [D] for $\tilde{\lambda}$ is given by

$$\sum_{A \in \mathscr{A}} \tilde{\lambda}(A) f(A) = \sum_{A \in \mathscr{A}} \lambda^*(A) f(A) - \mu \left(\sum_{A \in \mathscr{F}} f(A) - \sum_{A \in \mathscr{A}} \lambda^{\mathscr{F}}(A) f(A) \right) \leq \sum_{A \in \mathscr{A}} \lambda^*(A) f(A).$$

Hence $\tilde{\lambda}$ is also optimal to [D]. Similarly, we have

$$\sum_{A \in \mathscr{A}} \tilde{\lambda}(A) g(A) = \sum_{A \in \mathscr{A}} \lambda^*(A) g(A) - \mu \left(\sum_{A \in \mathscr{F}} g(A) - \sum_{A \in \mathscr{A}} \lambda^{\mathscr{F}}(A) g(A) \right) < \sum_{A \in \mathscr{A}} \lambda^*(A) g(A).$$

This contradicts the definition of λ^*. The above argument implies that $f \in \mathrm{int}\,\mathscr{S}_\Delta$ if and only if for any cost vector $w \in \mathbf{R}^V_+$ the optimal solution of [D] is unique. From this, we obtain the latter statement of this theorem. □

Corollary 4 *A greedy fan is regular if and only if its submodular cone has interior points.*

2.2 Greedy Multiple-Choice Functions

In this subsection, we discuss properties of the map Φ_Δ associated with greedy fan Δ and try to define greedy fans by means of a certain map $\Phi : 2^V \to 2^{2^V}$. Our purpose is to derive the conditions of Φ which determine a greedy fan. First, we see that Φ_Δ has the following properties.

Proposition 5 *For a greedy fan Δ with vertex set $\mathscr{A}$, a map $\Phi = \Phi_\Delta : 2^V \to 2^{\mathscr{A}}$ has the following properties.*

(C1) *for a nonempty $X \subseteq V$, $\Phi(X)$ is nonempty and any $A \in \Phi(X)$ is nonempty.*

(C2) *for $X \subseteq V$ and $A \in \Phi(X)$, we have $A \subseteq X$.*

(M1) *for $X, Y \subseteq V$ and $A \in \Phi(X)$, if $A \subseteq Y \subseteq X$, then we have $A \in \Phi(Y)$.*

(M2) *for $X \subseteq V$ and $A, B \in \Phi(X)$, we have $A = B$ or $A \cap B = \emptyset$.*

If a function $\Phi : 2^V \to 2^{2^V}$ satisfies (C1) and (C2), we call $\Phi : 2^V \to 2^{2^V}$ a *multiple-choice function*. Given a multiple-choice function $\Phi : 2^V \to 2^{2^V}$, we can apply `Dual_Greedy` for any nonnegative input vector. However, the output λ depends on the choices of A^* and e^* in **step 1**. Next we discuss the uniqueness of outputs of `Dual_Greedy` for general multiple-choice functions. For a multiple-choice function Φ and an input vector $y \in \mathbf{R}^V_+$, a sequence $\{(A_i, e_i)\}_{i=1}^n \subseteq 2^V \times V$ is said to be *feasible* if $A_i = A^*$ and $e_i = e^*$ can be chosen by the **step 1** of the ith iteration step for input vector y. We see the following.

Lemma 6 *If a sequence $\{(A_i, e_i)\}_{i=1}^n$ is feasible to some input vector, then it is also feasible to any vector in* $\mathrm{cone}\{\chi_{A_i}\}_{i=1}^n$.

Let Δ_Φ be a set of simplicial cones defined as

$$\Delta_\Phi = \{\mathrm{cone}\{\chi_A\}_{A \in \mathrm{supp}\,\lambda} \mid \lambda \text{ is an output for some } w \in \mathbf{R}^V_+\}. \tag{7}$$

Lemma 6 above implies that any face of any member of Δ_Φ is also contained by Δ_Φ. Hence, if the output λ is uniquely determined for any $w \in \mathbf{R}^V_+$, Δ_Φ forms a simplicial cone subdivision of $\mathbf{R}^V_+$. In particular, this fan is greedy, and its center vertices are determined by the multiple-choice function Φ. In fact, the conditions (M1) and (M2) are sufficient for this uniqueness as follows, where for a function $\Phi : 2^V \to 2^{2^V}$, we define the *image* $\mathrm{Im}\,\Phi \subseteq 2^V$ as

$$\mathrm{Im}\,\Phi = \{A \subseteq V \mid \exists \text{ nonempty } X \subseteq V,\ A \in \Phi(X)\}. \tag{8}$$

Theorem 7 *If a multiple-choice function $\Phi : 2^V \to 2^{2^V}$ satisfies the conditions* (M1) *and* (M2)*, the solution of* `Dual_Greedy` *is determined independently of the choices A^* and e^* in* **step 1** *for any nonnegative input vector. Furthermore, the vertex set of Δ_Φ is given by* $\mathrm{Im}\,\Phi$.

PROOF: We use induction on $\#\mathrm{supp}\,w$ for a nonnegative input vector w. In the case of $\#\mathrm{supp}\,w \leq 1$, the statement clearly holds. Consider a nonnegative input vector w with $\#\mathrm{supp}\,w > 1$. Let $\{A_i, e_i\}_{i=1}^n$ and $\{B_i, d_i\}_{i=1}^n$ be two feasible sequences for w. Let λ and μ be outputs of w using feasible sequences $\{A_i, e_i\}_{i=1}^n$ and $\{B_i, d_i\}_{i=1}^n$, respectively. We define X_i and Y_i as

$$X_1 = Y_1 = V, \quad X_{i+1} = X_i \setminus \{e_i\},\ Y_{i+1} = Y_i \setminus \{d_i\} \tag{9}$$

for $i = 1, \ldots, n$. We show $\lambda = \mu$. Let i be the smallest number satisfying $\lambda(A_i) > 0$. Similarly, let j be the smallest number satisfying $\mu(B_j) > 0$. Then we have $A_i \subseteq \mathrm{supp}\,w \subseteq X_i$ and $B_j \subseteq \mathrm{supp}\,w \subseteq Y_j$. By (M1), we have $A_i, B_j \in \Phi(\mathrm{supp}\,w)$. Hence $A_i = B_j$ or $A_i \cap B_j = \emptyset$ follows from (M2). If $A_i = B_j$, then we have $\lambda(A_i) = \mu(B_j)$. If $A_i \cap B_j = \emptyset$, then we have $A_i \subseteq \mathrm{supp}\,w \setminus \{d_j\} \subseteq X_i$ and $B_{j+1} \subseteq \mathrm{supp}\,w \setminus \{d_j\} \subseteq Y_{j+1}$. Similarly we have $A_i = B_{j+1}$ or $A_i \cap B_{j+1} = \emptyset$. Repeating this process, there exists some number k such that $A_i = B_{j+k}$ or $A_i = \mathrm{supp}\,w \setminus \{d_j, d_{j+1}, \ldots d_{j+k-1}\}$. In the latter case, we have $B_{j+k} \in \Phi(A_i)$. Hence $B_{j+k} = A_i$ holds. Since $A_i \cap B_{j+1} = A_i \cap B_{j+2} = \cdots = A_i \cap B_{j+k-1} = \emptyset$, we have $\lambda(A_i) = \mu(B_{j+k})$. We define a modified input vector $w' = w - \lambda(A_i)\chi_{A_i} = w - \mu(B_{j+k})\chi_{B_{j+k}}$. Then we have $\#\mathrm{supp}\,w' < \#\mathrm{supp}\,w$. Two sequences $\{(A_i, e_i)\}_{i=1}^n$ and $\{(B_i, d_i)\}_{i=1}^n$ are also feasible to the modified input vector w' by Lemma 6. Let λ' and μ' be two outputs of `Dual_Greedy` for input vector w' using feasible sequences $\{(A_i, e_i)\}_{i=1}^n$ and $\{(B_i, d_i)\}_{i=1}^n$, respectively. Then we have

$$\lambda'(A) = \begin{cases} 0 & \text{if } A = A_i = B_{j+k} \\ \lambda(A) & \text{otherwise} \end{cases}, \ \mu'(A) = \begin{cases} 0 & \text{if } A = A_i = B_{j+k} \\ \mu(A) & \text{otherwise} \end{cases} \qquad (A \subseteq V).$$

By induction, $\lambda' = \mu'$ holds. This implies $\lambda = \mu$.

Finally, we show that the set of vertices of Δ_Φ coincides with $\mathrm{Im}\,\Phi$. It suffices to show that for any $A \in \mathrm{Im}\,\Phi$, the output $\lambda \in \mathbf{R}^{2^V}$ of `Dual_Greedy` for input vector χ_A is given as $\chi_{\{A\}} \in \mathbf{R}^{2^V}$. Let $\{(A_i, e_i)\}_{i=1}^n$ be a feasible sequence to χ_A and λ be the resulting output. Let j be the smallest number having $\lambda(A_i)$. Similarly to above discussions, we have $A_i \subseteq \mathrm{supp}\,\chi_A = A \subseteq X_i$, where X_i is defined by (9). Hence we have $A_i \in \Phi(A)$. This implies $A_i = A$ and $\lambda = \chi_{\{A\}}$. □

A multiple-choice function is said to be *greedy* if it satisfies (M1) and (M2). Hence, we obtain greedy fans by greedy multiple-choice functions. The proof of Theorem 7 gives the following criterion when two greedy multiple-choice functions produce same greedy fan.

Proposition 8 *For two greedy multiple-choice functions $\Phi_1, \Phi_2 : 2^V \to 2^{2^V}$, the following conditions are equivalent.*

(1) $\Delta_{\Phi_1} = \Delta_{\Phi_2}$.

(2) *for any $X \subseteq V$, $A \in \Phi_1(X)$, and $B \in \Phi_2(X)$, we have $A = B$ or $A \cap B = \emptyset$.*

2.3 Acyclic Greedy Fans

Here, we introduce a certain class of greedy fans which can be represented by posets. This approach is motivated by dual greedy systems by Frank [4]. Throughout this subsection, $\mathscr{A}$ denotes a subset of $2^V \setminus \{\emptyset\}$. A pair of $A, B \subseteq V$ is said to be *intersecting* if it satisfies $A \cap B \neq \emptyset$, $A \not\subseteq B$, and $B \not\subseteq A$.

Let $\mathscr{P} = (\mathscr{A}, \leq)$ be a poset on $\mathscr{A}$. We define a function $\Phi_{\mathscr{P}} : 2^V \to 2^{\mathscr{A}}$ associated with $\mathscr{P}$ as

$$\Phi_{\mathscr{P}}(X) = \{A \in \mathscr{A} \mid A \subseteq X,\ \forall B \in \mathscr{A}, B > A \Rightarrow B \not\subseteq X\} \quad (X \subseteq V), \tag{10}$$

that is, $\Phi_{\mathscr{P}}(X)$ is a set of maximal members of $\mathscr{A}$ contained in X. Such a function $\Phi_{\mathscr{P}}$ was used by Frank [4] in his dual greedy algorithm.

A poset $\mathscr{P} = (\mathscr{A}, \leq)$ is said to be *greedy* if $\Phi_{\mathscr{P}}$ is a greedy multiple-choice function and satisfies $\mathrm{Im}\,\Phi_{\mathscr{P}} = \mathscr{A}$. Hence, from a greedy poset $\mathscr{P}$, we obtain a greedy fan, which is denoted by $\Delta_{\mathscr{P}}$. A greedy fan Δ is said to be *acyclic* if there exists a greedy poset $\mathscr{P}$ such that $\Delta = \Delta_{\mathscr{P}}$. A greedy poset $(\mathscr{A}, \leq)$ is a refinement of poset $(\mathscr{A}, \subseteq)$ as follows.

Lemma 9 *Let $\mathscr{P} = (\mathscr{A}, \leq)$ be a greedy poset. For any $A, B \in \mathscr{A}$, if $A \subseteq B$, then $A \leq B$.*

PROOF: $\mathrm{Im}\,\Phi_{\mathscr{P}} = \mathscr{A}$ and (M1) for $\Phi_{\mathscr{P}}$ implies $B \in \Phi_{\mathscr{P}}(B)$. Suppose $A \not\leq B$. Then A and B are noncomparable. Then there exists $C \in \mathscr{A}$ such that $C \neq B$ and $C \in \Phi_{\mathscr{P}}(B)$. Hence we have $C \cap B \neq \emptyset$. This is a contradiction to (M2). □

For a poset $\mathscr{P} = (\mathscr{A}, \leq)$, a *linear extension* $(\mathscr{A}, \leq^*)$ of $\mathscr{P}$ is a totally ordered set which satisfies

$$A \leq B \Rightarrow A \leq^* B \quad (A, B \in \mathscr{A}).$$

Then we observe the following.

Lemma 10 *If $\mathscr{A}$ contains every singleton. then any linear extension of $(\mathscr{A}, \subseteq)$ is greedy.*

Furthermore, from the condition of Proposition 8, we have the following.

Proposition 11 *Let $\mathscr{P}$ is a greedy poset. Then, $\Delta_{\mathscr{P}} = \Delta_{\mathscr{P}^*}$ holds for any linear extension $\mathscr{P}^*$ of $\mathscr{P}$.*

Theorem 12 *Acyclic greedy fans are regular.*

PROOF: By Proposition 11, it suffices to show the present theorem when the greedy poset $(\mathscr{A}, \leq)$ is a totally ordered set. Suppose that $\mathscr{A} = \{A_1, A_2, \ldots, A_m\}$ is ordered by

$$A_1 > A_2 > \cdots > A_m.$$

Consider the submodularity inequalities (5) for $\Delta_{(\mathscr{A}, \leq)}$. Then, for any $\mathscr{F} \in (\hat{\Delta}_{(\mathscr{A}, \leq)})^*$, there exists $A^* \in \mathrm{supp}\,\lambda^{\mathscr{F}}$ such that $A^* > A$ holds for any $A \in \mathscr{F}$. Hence we define a function $f : \mathscr{A} \to \mathbf{R}$ as

$$f(A_i) = -\varepsilon^i \quad (A_i \in \mathscr{A} = \{A_1, A_2, \ldots, A_m\}),$$

where $\varepsilon \in \mathbf{R}_+$ is a sufficiently small positive real. Then f satisfies strict submodularity inequalities. This implies that submodular cone $\mathscr{S}_{\Delta_{(\mathscr{A}, \leq^*)}}$ has interior points. By Corollary 4, $\Delta_{(\mathscr{A}, \leq)}$ is regular. □

For two posets $\mathscr{P}_1 = (\mathscr{A}, \leq_1)$ and $\mathscr{P}_2 = (\mathscr{A}, \leq_2)$ with a common ground set $\mathscr{A}$, we define *meet* $\mathscr{P}_1 \wedge \mathscr{P}_2 = (\mathscr{A}, \leq_{1\wedge 2})$ as

$$A \leq_{1\wedge 2} B \overset{\text{def.}}{\Longleftrightarrow} A \leq_1 B \text{ and } A \leq_2 B \quad (A, B \in \mathscr{A}). \tag{11}$$

The next theorem shows that if two greedy posets define the same greedy fan, then their meet also defines the same one.

Theorem 13 *Let $\mathscr{P}_1 = (\mathscr{A}, \leq_1)$ and $\mathscr{P}_2 = (\mathscr{A}, \leq_2)$ be greedy posets on $\mathscr{A}$. If $\Delta_{\mathscr{P}_1} = \Delta_{\mathscr{P}_2}$, $\mathscr{P}_1 \wedge \mathscr{P}_2$ is also greedy and satisfies $\Delta_{\mathscr{P}_1 \wedge \mathscr{P}_2} = \Delta_{\mathscr{P}_1} = \Delta_{\mathscr{P}_2}$.*

PROOF: From the definitions of $\Phi_{\mathscr{P}}$ and the meet $\mathscr{P}_1 \wedge \mathscr{P}_2$, we see

$$\Phi_{\mathscr{P}_1 \wedge \mathscr{P}_2}(X) \subseteq \Phi_{\mathscr{P}_1}(X) \cup \Phi_{\mathscr{P}_2}(X) \quad (X \subseteq V).$$

Hence, it suffices to show that $\Phi_{\mathscr{P}_1 \wedge \mathscr{P}_2}$ satisfies (M2). Suppose that there exist $A, B \in \Phi_{\mathscr{P}_1 \wedge \mathscr{P}_2}(X)$ such that $A \cap B$ is nonempty. Then we have $A, B \in \Phi_{\mathscr{P}_1 \wedge \mathscr{P}_2}(A \cup B)$ by (M2). Then A or B is not contained by $\Phi_{\mathscr{P}_1}(A \cup B) \cup \Phi_{\mathscr{P}_2}(A \cup B)$. We assume $A \notin \Phi_{\mathscr{P}_1}(A \cup B) \cup \Phi_{\mathscr{P}_2}(A \cup B)$. Then there exists $C \in \Phi_{\mathscr{P}_1}(A \cup B)$ such that $C \geq_1 A$ holds. We claim that A and C are disjoint. Suppose that A and C intersect. Then we have $C \in \Phi_{\mathscr{P}_1}(A \cup C)$. We show $A \notin \Phi_{\mathscr{P}_2}(A \cup C)$ and $C \not\geq_2 A$.

If $A \in \Phi_{\mathscr{P}_2}(A \cup C)$, then $A, C \in \Phi_{\mathscr{P}_2}(A \cup C)$ must be disjoint. This contradicts the assumption. If $C \geq_2 A$, then we have $C \geq_{1\wedge 2} A$. This contradicts $A \in \Phi_{\mathscr{P}_1 \wedge \mathscr{P}_2}(A \cup B)$. Hence there exists $D \in \Phi_{\mathscr{P}_2}(A \cup C)$ such that $D \geq_2 A$ and $D \neq C$. Then $D \cap C = \emptyset$ and hence $D \subset A$ (strict inclusion). This implies $D <_2 A$, which contradicts $D \in \Phi_{\mathscr{P}_2}(A \cup C)$. Hence we have $A \cap C = \emptyset$ and $C \subset B$ (strict inclusion). By Lemma 9, we have $C <_1 B$. This contradicts $C \in \Phi_{\mathscr{P}_1}(A \cup B)$. □

For a poset $\mathscr{P}$, we denote by $\mathscr{L}_{\mathscr{P}}$ the set of all linear extensions of $\mathscr{P}$. The above theorem implies that for any family $\mathscr{A} \subseteq 2^V \setminus \{\emptyset\}$ which contains every singleton, there uniquely exists a set of greedy posets $\mathscr{H}(\mathscr{A})$ on the ground set $\mathscr{A}$ such that

(H1) the family of linear extensions $\{\mathscr{L}_{\mathscr{P}} \mid \mathscr{P} \in \mathscr{H}(\mathscr{A})\}$ forms a partition of $\mathscr{L}_{(\mathscr{A},\subseteq)}$.

(H2) for two linear extensions $\mathscr{P}_1^*, \mathscr{P}_2^* \in \mathscr{L}_{(\mathscr{A},\subseteq)}$, we have

$$\Delta_{\mathscr{P}_1^*} = \Delta_{\mathscr{P}_2^*} \Leftrightarrow \exists \mathscr{P} \in \mathscr{H}(\mathscr{A}),\ \mathscr{P}_1^*, \mathscr{P}_2^* \in \mathscr{L}_{\mathscr{P}} \tag{12}$$

A set of posets on a common ground set which satisfies the condition (H1) above is called *holometry*, which was introduced by Tomizawa [14], [15], [16] in 1983 as a combinatorial abstraction of normal fans of base polyhedra. For poset $(\mathscr{A}, \leq)$ we define an *order cone* $\mathscr{C}_{(\mathscr{A},\leq)} \subseteq \mathbf{R}^{\mathscr{A}}$ as

$$\mathscr{C}_{(\mathscr{A},\leq)} = \{x \in \mathbf{R}^{\mathscr{A}} \mid x(A) \leq x(B) \quad (A, B \in \mathscr{P},\ A \leq B)\}. \tag{13}$$

The set of polyhedral cones consisting of order cones $\{\mathscr{C}_{\mathscr{P}} \mid \mathscr{P} \in \mathscr{H}(\mathscr{A})\}$ and their faces is denoted by $\Sigma^g(\mathscr{A})$. In fact, $\Sigma^g(\mathscr{A})$ forms a polyhedral fan as follows, where detailed proof will be given in [8].

Theorem 14 *For a family $\mathscr{A}$ which contains every singleton, $\Sigma^g(\mathscr{A})$ forms a polyhedral cone subdivision of order cone $\mathscr{C}_{(\mathscr{A},\subseteq)}$.*

A holometry is called a *hypergeometry* if its associated set of order cones given above forms a polyhedral cone subdivision [14], [15], [16]. The above theorem states that $\mathscr{H}(\mathscr{A})$ is a hypergeometry. Analogously to the *secondary fan* [7], [12], we call this polyhedral fan $\Sigma^g(\mathscr{A})$ the *secondary greedy fan* of $\mathscr{A}$. So it is natural to ask whether there exists some polyhedron $P \subseteq \mathbf{R}^{\mathscr{A}}$ whose normal fan coincides with $\Sigma^g(\mathscr{A})$. If such a polyhedron P exists, each edge vector is parallel to $\chi_{\{A\}} - \chi_{\{B\}}$ for some $A, B \in \mathscr{A}$. A well-known characterization of base polyhedra by edge directions [13] implies that P is a base polyhedron associated with some (ordinary) submodular function defined on the set of lower ideals of the poset $(\mathscr{A}, \subseteq)$ (see [5]).

Problem 15 *Does there exist a base polyhedron whose normal fan coincides with a secondary greedy fan ?*

2.4 Greedy Fans by Set Systems

Here, we discuss the case when $\mathscr{A}$ is a greedy poset ordered by inclusion $(\subseteq)$. In this case, the associated holometry $\mathscr{H}(\mathscr{A})$ is a singleton, i.e., $\mathscr{H}(\mathscr{A}) = \{(\mathscr{A}, \subseteq)\}$. We observe the following.

Proposition 16 *$(\mathscr{A}, \subseteq)$ is greedy if and only if it satisfies*

(S0) *for any $e \in V$, we have $\{e\} \in \mathscr{A}$.*

(S1) *for any intersecting pair $A, B \in \mathscr{A}$, we have $A \cup B \in \mathscr{A}$.*

The condition (S1) implies that $\Phi_{(\mathscr{A},\subseteq)}(X)$ forms the unique maximal partition of X, where "unique maximal" means that any partition $\Pi \subseteq 2^{\mathscr{A}}$ of X is a refinement of $\Phi_{(\mathscr{A},\subseteq)}(X)$, that is, for any $C \in \Pi$ there exists $C' \in \Phi_{(\mathscr{A},\subseteq)}(X)$ such that $C \subseteq C'$. The submodularity inequalities for a greedy fan $\Delta_{(\mathscr{A},\subseteq)}$ are explicitly given as follows.

Theorem 17 *Let $(\mathscr{A}, \subseteq)$ be a greedy poset ordered by inclusion. The submodularity inequalities for $\Delta_{(\mathscr{A},\subseteq)}$ are given by*

$$f(A) + f(B) \geq f(A \cup B) + \sum_{C \in \Phi_{(\mathscr{A},\subseteq)}(A \cap B)} f(C) \quad (A, B \in \mathscr{A} : \text{intersecting}), \tag{14}$$

$$\sum_{A \in \mathscr{F}} f(A) \geq f\Big(\bigcup_{A \in \mathscr{F}} A\Big)$$
$$(\mathscr{F} \subseteq 2^{\mathscr{A}} : \text{pairwise disjoint with } \textstyle\bigcup_{A \in \mathscr{F}} A \in \mathscr{A} \text{ and } \forall \mathscr{F}' \subset \mathscr{F} \Rightarrow \bigcup_{A \in \mathscr{F}'} A \notin \mathscr{A}\ ). \tag{15}$$

In the case of $\mathscr{A} = 2^V \setminus \{\emptyset\}$, equations (14) and (15) coincide with ordinary submodularity inequalities.

Concluding Remarks

We can show a stronger statement than Theorem 12 that acyclic greedy fans are reverse-lexicographic triangulation. In addition, dual greedy systems by Kashiwabara and Okamoto [9] and by Frank [4] can be derived as examples of acyclic greedy fans. Detailed discussions will be given in the forthcoming full version of this paper [8].

Acknowledgement

The author thanks Satoru Fujishige for helpful comments and suggesting Tomizawa's works [14], [15], [16].

References

[1] K. ANDO, K-submodular functions and convexity of their Lovász extension *Discrete Applied Mathematics* (2002) **122** 1–12.

[2] J. EDMONDS, Submodular functions, matroids, and certain polyhedra, in: R. Guy, H. Hanani, N. Sauer, and J. Schönheim, eds., *Combinatorial Structures and Their Applications*, Gordon and Breach, New York, 1970, 69–87.

[3] U. FAIGLE AND W. KERN, Submodular linear programs on forests, *Mathematical Programming* (1996) **72** 195–206.

[4] A. FRANK, Increasing the rooted-connectivity of a digraph by one, *Mathematical Programming* (1999) **84** 565–576.

[5] S. FUJISHIGE, *Submodular Functions and Optimizations*, North-Holland, Amsterdam, 1991.

[6] S. FUJISHIGE, Dual greedy polyhedra, choice functions, and abstract convex geometries, *Discrete Optimization* (2004) **1** 41–49.

[7] I. M. GEL'FAND, M. M. KAPRANOV AND A. V. ZELEVINSKY, *Discriminants, Resultants, and Multidimensional Determinants*, Birkhäuser, Boston, MA, 1994.

[8] H. HIRAI, Greedy fans, in preparation.

[9] K. KASHIWABARA AND Y. OKAMOTO, A greedy algorithm for convex geometries, *Discrete Applied Mathematics* (2003) **131** 449–465.

[10] U. KRÜGER, Structural aspects of ordered polymatroids, *Discrete Applied Mathematics* (2000) **99** 125–148.

[11] M. QUEYRANNE, F. SPIEKSMA AND F. TERDELLA, A general class of greedily solvable linear programs, *Mathematics of Operations Research* (1998) **23** 892–908.

[12] B. STURMFELS, *Gröbner Bases and Convex Polytopes*, American Mathematical Society, Providence, RI, 1996.

[13] N. TOMIZAWA, Theory of hyperspaces (XVI) — On the structure of hedrons (in Japanese), Papers of the Technical Group on Circuit and System Theory, Institute of Electronics and Communication Engineer of Japan, CAS82–174, 1983.

[14] N. TOMIZAWA, Theory of hyperspaces (XIX) — An Introduction to a New Geometry (in Japanese), Papers of the Technical Group on Circuit and System Theory, Institute of Electronics and Communication Engineer of Japan, CAS83–76, 1983.

[15] N. TOMIZAWA, Theory of hyperspaces (XXI) — On the Holometry (in Japanese), Papers of the Technical Group on Circuit and System Theory, Institute of Electronics and Communication Engineer of Japan, CAS83–142, 1983.

[16] N. TOMIZAWA, Theory of hyperspaces (XXII) — Hypergeometry (in Japanese), Papers of the Technical Group on Circuit and System Theory, Institute of Electronics and Communication Engineer of Japan, CAS83–160, 1983.

[17] G. M. ZIEGLER, *Lectures on Polytopes*, Springer-Verlag, New York, 1995.

Road to “Problem Solving Engines”

TOSHIHIDE IBARAKI

Department of Informatics Kwansei Gakuin
University Sanda, Japan 669-1337
ibaraki@ksc.kwansei.ac.jp

Abstract:

We describe our attempts to build problem solving engines that cover a large portion of combinatorial optimization problems encountered in real world applications. For this, we select a list of standard problems, and develop their solvers which are based on local search and metaheuristics. As standard problems, we have chosen so far CSP (constraint satisfaction problem), RCPSP (recource constrained project scheduling problem), GAP (generalized assignment problem), VRP (vehicle routing problem), SCP (set covering problem), MAX-SAT (maximum satisfiability problem), 2PP (2-dimensional packing problem) and others. In this talk, we outline definitions of some of these problems, algorithmic contents of engines, and some computational results, putting emphasis on VRP, 2PP and RCPSP.

Bisecting a Four-Connected Graph with Three Resource Sets

TOSHIMASA ISHII

Department of Information and Computer Sciences
Toyohashi University of Technology
Aichi 441-8580, Japan
ishii@ics.tut.ac.jp

KENGO IWATA

Department of Information and Computer Sciences
Toyohashi University of Technology
Aichi 441-8580, Japan
iwata@algo.ics.tut.ac.jp

HIROSHI NAGAMOCHI

Department of Applied Mathematics and Physics
Graduate School of Informatics
Kyoto University
Kyoto 606-8501, Japan
nag@amp.i.kyoto-u.ac.jp

Abstract: Let $G = (V,E)$ be an undirected graph with a node set V and an arc set E. G has k pairwise disjoint subsets $T_1, T_2, \ldots, T_k$ of nodes, called resource sets, where $|T_i|$ is even for each i. The partition problem with k resource sets asks to find a partition V_1 and V_2 of the node set V such that the graphs induced by V_1 and V_2 are both connected and $|V_1 \cap T_i| = |V_2 \cap T_i| = |T_i|/2$ holds for each $i = 1,2,\ldots,k$. It is known that the problem of testing whether such a bisection exists is NP-hard even in the case of $k = 1$, and that in the case of $k = 1,2$, a bisection in a $(k+1)$-connected graph can be found in polynomial time. In this paper, we show that in the case of $k = 3$, if G is 4-connected and has K_4 as its subgraph, then a bisection can be found in $O(|V|^3 \log |V|)$ time, while we also show that there is a 4-connected graph which has no bisection.

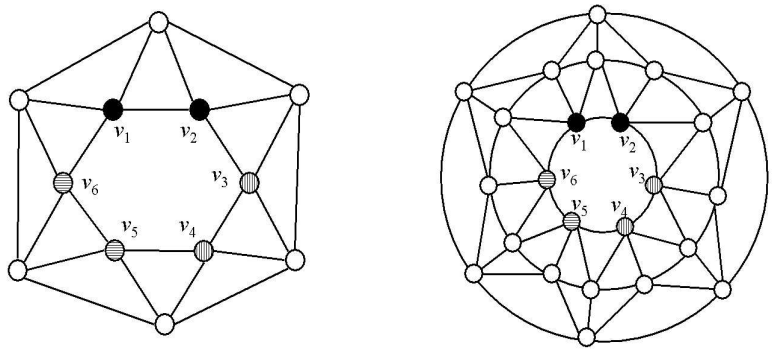
Keywords: 4-connected graph, bisection, ham-sandwich cut, graph embedding

1 Introduction

In this paper, we consider the following graph partition problems: given an undirected graph $G = (V,E)$ with a set V of nodes a set E of arcs, and k pairwise disjoint sets $T_1, T_2, \ldots, T_k$ of nodes, called *resource sets*, where each $|T_i|$ is even, find a partition V_1 and V_2 of V such that the graphs induced by V_1 and V_2 are both connected and $|V_1 \cap T_i| = |V_2 \cap T_i| = |T_i|/2$ holds for each $i = 1,2,\ldots,k$. This problem is called *the bisection problems with k resource sets*, and such a bisection is called *k-bisection* (*with respect to* $T_1, \ldots, T_k$). This problem has applications in the fair-division type problems. For general graphs, the problem was shown to be NP-hard even if $k = 1$ holds, since it is NP-hard to test whether a 1-bisection exists or not [3, 4]. On the other hand, when $k = 1,2$, it is known that such a k-bisection in a $(k+1)$-connected graph exists and it can be found in linear time for $k = 1$ by Suzuki et al. [10] and by Wada and Kawaguchi [11], and in $O(|V|^2 \log |V|)$ time for $k = 2$ by Nagamochi et al. [9]. For a general $k \geq 3$, to our knowledge, any nontrivial sufficient condition for which a k-bisection exists is not known, while Nagamochi et al. [9] conjectured that every $(k+1)$-connected graph has a k-bisection.

On the other hand, as shown in Figure 1, there exist 4-connected graphs which have no 3-bisection. This indicates a negative answer to the conjecture for $k = 3$ given by Nagamochi et al. Moreover, the graph in Figure 1(b) is also 5-connected, and even 5-connected graphs may have no 3-bisection (this also indicates a negative answer to the above conjecture for $k = 4$). Instead, in this paper, we give a sufficient condition for which a 3-bisection exists; we prove that if G is 4-connected and has a complete graph K_4 of four nodes as its subgraph, then a 3-bisection exists. We also show that it can be found in $O(|V|^3 \log |V|)$ time.

A key technique of the proof, which is an extension of the method by Nagamochi et al. [9], is a reduction of the problem to a geometrical problem. We first prove that every 4-connected graph containing a complete graph K^* of four nodes as its subgraph can be embedded in the 3-dimensional space $\Re^3$, in such a way that the following (i)(ii) hold: (i) the convex hull of its nodes is a trigonal pyramid corresponding to the K^*, (ii) every node not in K^* is in the convex hull of its neighbors (precise definition is given in Section 2.2). This will guarantee that, for any given plane H in $\Re^3$, each of the two subgraphs of G separated by H remains connected. Given such an embedding in $\Re^3$, we apply the so-called ham-sandwich cut algorithm, which is well known in computational geometry, to find a plane H^* that bisects T_1, T_2, and T_3 simultaneously. Consequently,



(a) (b)

Figure 1: Illustration of instances of 4-connected graphs which have no 3-bisection, where $T_1 = \{v_1, v_2\}$, $T_2 = \{v_3, v_4\}$, and $T_3 = \{v_5, v_6\}$ in both (a) and (b). Note that the graph (b) is also 5-connected.

the two subgraphs by the plane H^* indicates a 3-bisection. We give an algorithm for finding such a plane H^* in $O(|V|^3 \log |V|)$ time.

2 Preliminaries

Let $G = (V, E)$ stand for an undirected simple graph with a set V of *nodes* and a set E of *arcs*, where we denote $|V|$ by n and $|E|$ by m. A singleton set $\{x\}$ may be simply written as x, and "$\subset$" implies proper inclusion while "$\subseteq$" means "$\subset$" or "$=$". For a subgraph G' of G, the sets of nodes and arcs in G' are denoted by $V(G')$ and $E(G')$, respectively. For a set X of nodes in G, a node $v \in V - X$ is called a *neighbor* of X if it is adjacent to some node in X, and the set of all neighbors of X is denoted by $N_G(X)$.

For an arc $e = (u, v)$, we denote by G/e the graph obtained from G by contracting u and v into a single node (deleting any resulted self-loop), and by $G - e$ the graph obtained from G by removing e. We also say that G/e is obtained from G by contracting the arc e. A graph G is *k-connected* if and only if $|V| \geq k+1$ and the graph $G - X$ obtained from G by removing any set X of $(k-1)$ nodes remains connected.

The main result of this paper is described as follows.

Theorem 1 *Let $G = (V, E)$ be a 4-connected graph which contains a complete graph with four nodes as its subgraph. Let T_1, T_2, T_3 be pairwise disjoint subsets of V such that $|T_i|$ is even for $i = 1, 2, 3$. Then G has a 3-bisection with respect to T_1, T_2, and T_3, and it can be found in $O(n^3 \log n)$ time.* □

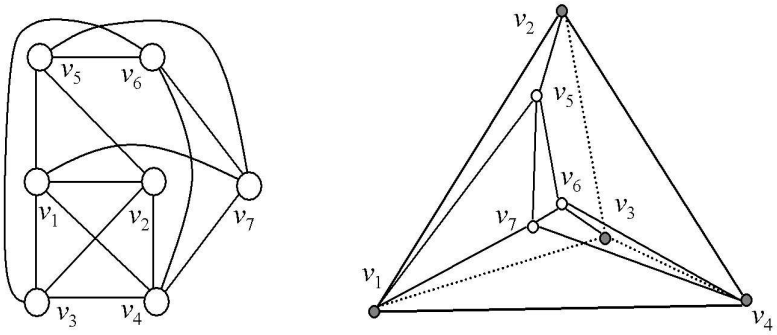
In the sequel, we give a constructive proof of this theorem by reducing the problem to a geometrical problem as mentioned in Section 1. For this, we give some geometric notations in the next two subsections.

2.1 Convex hull and ham-sandwich cut

Consider the d-dimensional space $\Re^d$. For a non-zero $a \in \Re^d$ and a real $b \in \Re^1$, $H(a, b) = \{x \in \Re^d \mid \langle a \cdot x \rangle = b\}$ is called a *hyperplane*, where $\langle a \cdot x \rangle$ denotes the inner product of $a, x \in \Re^d$. Moreover, $H^+(a, b) = \{x \in \Re^d \mid \langle a \cdot x \rangle \geq b\}$ (resp., $H^-(a, b) = \{x \in \Re^d \mid \langle a \cdot x \rangle \leq b\}$) is called a *positive closed half space* (resp., *negative closed half space*) with respect to $H = H(a, b)$.

For a set $P = \{x_1, \ldots, x_k\}$ of points in $\Re^d$, a point $x' = \alpha_1 x_1 + \cdots + \alpha_k$ with $\sum_{i=1,\ldots,k} \alpha_i = 1$ and $\alpha_i \geq 0$, $i = 1, \ldots, k$ is called a *convex combination of P*, and the set of all convex combinations of P is denoted by *conv*(P). If $P = \{x_1, x_2\}$, then *conv*(P) is called a *segment* (connecting x_1 and x_2), denoted by $[x_1, x_2]$. A subset $S \subseteq \Re^d$ is called a *convex set* if $[x, x'] \subseteq S$ for any two points $x, x' \in S$. For a convex set $S \subseteq \Re^d$, a point $x \in S$ is called a *vertex* if there is no pair of points $x', x'' \in S - x$ such that $x \in [x', x'']$. For two vertices $x_1, x_2 \in S$, the segment $[x_1, x_2]$ is called an *edge* of S if $\alpha x' + (1 - \alpha) x'' = x \in [x_1, x_2]$ for some $0 \leq \alpha \leq 1$ implies $x', x'' \in [x_1, x_2]$. The intersection S of a finite number of closed half spaces is called a *convex polyhedron*, and is called a *convex polytope* if S is non-empty and bounded.

Given a convex polytope S in $\Re^d$, the *vertex-edge graph* $G_S = (V_S, E_S)$ is defined to be an undirected graph with node set V_S corresponding to the vertices of S and arc set E_S corresponding to those pairs of vertices x, x' for which $[x, x']$ is an edge of S. For a convex polyhedron S, a hyperplane $H(a, b)$ is called a *supporting hyperplane* of S if $H(a, b) \cap S \neq \emptyset$ and either $S \subseteq H^+(a, b)$ or $S \subseteq H^-(a, b)$. We say that a point $p \in S$ is *strictly inside S* if there is no supporting hyperplane of



(a) (b)

Figure 2: Illustration of an instance of an SC-embedding; (b) shows an SC-embedding of the graph in (a) with boundary $(\{v_1,v_2,v_3,v_4\},\bigcup_{1\le i,j\le 4}(v_i,v_j))$ into $\Re^3$.

S containing p. If S has a point strictly inside S in $\Re^d$, S is called *full-dimensional* in $\Re^d$. The set of points strictly inside $conv(P)$ is denoted by $int(conv(P))$.

Let $P_1,\ldots,P_d$ be d sets of points in $\Re^d$. We say that a hyperplane $H=H(a,b)$ in $\Re^d$ *bisects* P_i if $|P_i\cap H^+(a,b)|\ge\lceil|P_i|/2\rceil$ and $|P_i\cap H^-(a,b)|\ge\lceil|P_i|/2\rceil$ hold. Thus if $|P_i|$ is odd, then any bisector H of P_i contains at least one point of P_i. If H bisects P_i for each $i=1\ldots,d$, then H is called a *ham-sandwich cut* with respect to the sets $P_1,\ldots,P_d$. The following results are well-known.

Theorem 2 [5] *Given d sets $P_1,\ldots,P_d$ of points in the d-dimensional space $\Re^d$, there exists a hyperplane which is a ham-sandwich cut with respect to the sets $P_1,\ldots,P_d$.* □

In [2], Chi-Yuan et al. showed that a ham-sandwich cut with respect to given sets $P_1,P_2,\ldots,P_d$ of points in $\Re^d$ with $\sum_{i=1}^d|P_i|=p$ can be found in $O(p^{3/2})$ time for $d=3$, $O(p^{8/3})$ time for $d=4$, and $O(p^{d-1-a(d)})$ time with certain small constant $a(d)>0$ for $d\ge 5$.

2.2 Convex embedding of a graph

In this section, we introduce a strictly convex embedding of a graph in $\Re^d$, which was first defined by Nagamochi et al. [9].

Given a graph $G=(V,E)$, an *embedding* of G in $\Re^d$ is an mapping $f:V\to\Re^d$, where each node v is represented by a point $f(v)\in\Re^d$, and each arc $e=(u,v)$ by a segment $[f(u),f(v)]$, which may be written by $f(e)$. For two arcs $e,e'\in E$, segments $f(e)$ and $f(e')$ may cross each other. For a set $\{v_1,\ldots,v_p\}=Y\subseteq V$ of nodes, we denote by $f(Y)$ the set $\{f(v_1),\ldots,f(v_p)\}$ of points, and we denote $conv(f(Y))$ by $conv_f(Y)$.

A strictly convex embedding of a graph is defined as follows (see Figure 2).

Definition 3 [9] *Let $G=(V,E)$ be a graph without isolated nodes and let $G'=(V',E')$ be a subgraph of G. A* strictly convex embedding (*or* SC-embedding, *for short*) *of G with* boundary *G' is an embedding f of G into $\Re^d$ in such a way that*

(i) *the vertex-edge graph of the full-dimensional convex polytope $conv_f(V')$ is isomorphic to G'* (*such that f itself defines an isomorphism*),

(ii) *$f(v)\in int(conv_f(N_G(v)))$ holds for all nodes $v\in V-V'$,*

(iii) *the points of $\{f(v)\mid v\in V\}$ are in general position.* □

From this definition, we can see that the vertices of $conv_f(V)$ are precisely the points in the boundary $f(V')$.

The following lemma implies that given an SC-embedding of $G=(V,E)$ into $\Re^d$, each two sets of nodes obtained by bisecting $f(V)$ with an arbitrary hyperplane in $\Re^d$ induce connected graphs.

Lemma 4 [9, Lemma 4.2] *Let $G=(V,E)$ be a graph without isolated nodes and let f be an SC-embedding of G into $\Re^d$. Let $f(V_1)\subseteq H^+(a,b)$ and $f(V)\cap(H^+(a,b)-H(a,b))\subseteq f(V_1)$ hold for some hyperplane $H=H(a,b)$ and for some $\emptyset\ne V_1\subseteq V$. Then V_1 induces a connected graph.* □

By Theorem 2 and this lemma, if there is an SC-embedding of a given graph $G=(V,E)$ into $\Re^d$, then by bisecting the embedded graph with a hyperplane which is a ham-sandwich cut with respect to $T_1,\ldots,T_d$, we can obtain a d-bisection. Based on this observation, for proving Theorem 1, we show in the next section that if G is 4-connected and contains a complete graph with four nodes as its subgraph, then there is an SC-embedding of G into $\Re^3$.

3 SC-embedding of a graph into $\Re^3$

In this section, given a 4-connected graph G which contains a complete graph with four nodes, denoted by K, we propose an algorithm, named EMBED3, for finding an SC-embedding of G with boundary K into $\Re^3$ in $O(n^3 \log n)$ time. Figure 2 shows an instance of such an SC-embedding of a 4-connected graph into $\Re^3$.

The algorithm EMBED3, which is an extension of the algorithm in $\Re^2$ given in [9], consists of two steps. First, we contract arcs in $E - E(K)$, one by one, while preserving the 4-connectivity until a complete graph G^* with 5 nodes containing K is obtained. Then we can easily obtain an SC-embedding f of G^* with boundary K into $\Re^3$; we find an embedding f' of $V(K)$ by putting them in general position (which shapes a trigonal pyramid), and we embed the node v with $\{v\} = V(G^*) - V(K)$ in $int(conv_{f'}(V(K)))$. Next, by tracing the process of the contraction reversely and embedding the contracted arcs into $\Re^3$, we convert the embedding f into the one for the original graph. The outline of algorithm EMBED3 is described as follows.

Algorithm EMBED3
Input: A 4-connected graph $G = (V,E)$ which has a complete graph K with 4 nodes.
Output: An SC-embedding of G with boundary K into $\Re^3$.
Step 1: While $|V(G)| \geq 6$ holds, execute the following procedure.

Find an arc $e \in E(G) - E(K)$ such that G/e remains 4-connected, and contract the arc e.

Let $G := G/e$.

/** The current graph G obtained by Step 1 is a complete graph with 5 nodes containing K. **/
Step 2: Embed G into $\Re^3$ so that its embedding is an SC-embedding f with boundary K. Next, by tracing the process of the contraction in Step 1 reversely and embedding the contracted arcs into $\Re^3$, one by one, we convert the embedding f into the one for the original graph. □

In the subsequent sections, we prove the correctness of algorithm EMBED3 by describing the details for each step, and also analyze the time complexity of each step.

3.1 Correctness of Step 1

In this section, we give a proof of the following theorem for the correctness of Step 1.

Theorem 5 *Let $G = (V,E)$ be a 4-connected graph which has a complete graph K with 4 nodes. Then there exists an arc $e \in E - E(K)$ such that G/e is 4-connected.* □

Before proving this, we introduce the following preparatory theorem about the contraction of arcs in 4-connected graphs.

Definition 6 *A graph G is called* uncontractible k-connected *if G is k-connected and G/e is not k-connected for any arc $e \in E(G)$.* □

Theorem 7 [7] *A graph G is uncontractible 4-connected if and only if G satisfies the following properties*:
(i) *G is 4-connected,*
(ii) *the degree of each node in $V(G)$ is exactly 4, and*
(iii) *for each arc $(u,v) \in E(G)$, there exists a node $w \in V(G) - \{u,v\}$ with $\{(u,w),(v,w)\} \subseteq E(G)$.* □

PROOF OF THEOREM 5: Let $V_1 = V - V(K)$. We construct the new graph H from $G = (V,E)$, defined as follows. $V(H) = V_1 \cup V(K) \cup V_2$, where V_2 is a copy of V_1. An arc (u_1,u_2) belongs to $E(H)$ if and only if (a) $(u_1,u_2) \in E$, (b) for $i = 1,2$, $u_i \in V_2$ and u_i is the copy of $v_i \in V_1$ with $(v_1,v_2) \in E$, or (c) $u_1 \in V(K)$, $u_2 \in V_2$, and u_2 is the copy of $v_2 \in V_1$ with $(u_1,v_2) \in E$. Note that H is also 4-connected. We first claim that in H, the degree of a node $v \in V(K)$ is at least 5. In G, from the 4-connectivity, it follows that $|N_G(v)| \geq 4$, from which and $|V(K) - v| = 3$, we can observe that v is adjacent to some node $u \in V_1$. In H, v is adjacent also to the copy $u' \in V_2$ of u. From this and $V(K) - v \subseteq N_G(v)$, it follows that $|N_H(v)| \geq 5$.

From Theorem 7, H has an arc $e \in E(H)$ such that H/e is 4-connected. Now, since $H - V(K)$ is not connected, the contraction of any arc in $E(K)$ violates the 4-connectivity of H. It follows that $e \notin E(K)$. Without loss of generality, let $e \in E - E(K)$. We claim that G/e remains 4-connected, proving the theorem.

Assume by contradiction that G/e is not 4-connected. Then there is a node set $X \subset V$ with $|X| \leq 3$ such that $G - X$ is not connected. Let C_i, $i = 1,\ldots,t$ (≥ 2) denote the component of $G - X$. We claim that there is a C_ℓ with $V(C_\ell) \subseteq V_1$. This follows since if there are two distinct components C_i, C_j with $V(K) \cap V(C_i) \neq \emptyset \neq V(K) \cap V(C_j)$, then two nodes $u \in V(K) \cap V(C_i)$ and $v \in V(K) \cap V(C_j)$ satisfy $(u,v) \notin E$, which contradicts that K is a complete graph. Now also in H/e, the set X satisfies $|X| \leq 3$ and the component C_ℓ satisfies $N_{H/e}(V(C_\ell)) \subseteq X$. From this and $V(H) - V(C_\ell) - X \neq \emptyset$, it follows that $(H/e) - X$ is not connected, contradicting the 4-connectivity of H/e. □

Finally, we show that Step 1 can be implemented to run in $O(n^3\alpha(n,n))$ time. First, we compute a sparse spanning 4-connected subgraph G' of G with $V(G)=V(G')$ and $O(n)$ arcs. Such a sparse spanning subgraph exists and it can be computed in linear time [8]. In the subsequent arguments about the time complexity of algorithm, let us assume that $|E|=O(n)$.

Now it was shown in [6] that it can be checked in $O(n\alpha(n,n))$ time whether G is 4-connected or not, where α denotes the inverse of the Ackermann's function. From $|E|=O(n)$, we can find a contractible edge in $E(G)-E(K)$ in $O(n^2\alpha(n,n))$ time. The number of the contraction is $O(n)$, and it follows that Step 1 can be implemented to run in $O(n^3\alpha(n,n))$ time.

3.2 Correctness of Step 2

In this section, for a graph $G=(V,E)$ and a subgraph G_1 of G, we consider a situation where a graph G/e obtained from G by contracting some arc $e=(u_1,u_2)$ with $\{u_1,u_2\}-V(G_1)\neq\emptyset$ has an SC-embedding f' of G/e with boundary G_1 into $\Re^d$. For proving the correctness of Step 2, we will show by the following Lemma 8 that if $|N_G(u_i)|\geq d+1$ holds for $i=1,2$, then we can find an SC-embedding of G with boundary G_1 into $\Re^d$. Since Step 1 in algorithm EMBED3 contracts arcs while preserving the 4-connectivity, it follows that the degree of every node is always at least 4 in the current graph. Also note that any arc in boundary K is not contracted through the algorithm. Hence, we can observe that the following Lemma 8 proves the correctness of Step 2.

Lemma 8 *Let $G=(V,E)$ be a graph without isolated nodes and let f' be an SC-embedding of G/e with boundary G_1 into $\Re^d$ for an arc $e=(u_1,u_2)$ with $\{u_1,u_2\}-V(G_1)\neq\emptyset$. Assume that for each node u_i, $i=1,2$, $|N_G(u_i)|\geq d+1$ holds if $u_i\in V-V(G_1)$. Then there is an SC-embedding of G with boundary G_1 into $\Re^d$.* □

Before proving Lemma 8, we give some notations and one preparatory lemma for an embedding of a new point into $\Re^d$. For a convex polyhedron S in $\Re^d$, a supporting hyperplane H of S is called a *facet* of S if the dimension of $H\cap S$ is $d-1$. It is well-known that every full-dimensional convex polyhedron can be uniquely represented by all of its facets.

Definition 9 *For a full-dimensional convex polyhedron S in $\Re^d$, let x be a vertex of S. Let $\mathscr{H}_x$ denote the family of all facets $H(a,b)$ of S containing the point x such that $S\subseteq H^+(a,b)$. Define the following polyhedron*:

$$D(x,S)=\bigcap_{H(a,b)\in\mathscr{H}_x}(H^-(a,b)-H(a,b)).$$ □

It is not hard to see the following property.

Lemma 10 *Let P be a set of points in $\Re^d$ such that $conv(P)$ is full-dimensional, and let x be a vertex of $conv(P)$. Then for a point $y\in\Re^d$, $x\in int(conv(P\cup\{y\})$ if and only if $y\in D(x,conv(P))$.*

PROOF: Assume that $y\notin D(x,conv(P))$. Then from the definition of $D(x,conv(P))$, there exists a facet $H(a,b)$ of $conv(P)$ such that $P\cup\{y\}\subseteq H^+(a,b)$. This indicates that $x\notin int(conv(P\cup\{y\})$.

Assume by contradiction that $y\in D(x,conv(P))$ and there is a facet $H(a,b)$ of $conv(P\cup\{y\})$ containing x. If $H(a,b)$ contains y, then it follows from the definition of $D(x,conv(P))$ that $(H^+(a,b)-H(a,b))\cap conv(P)\neq\emptyset\neq(H^-(a,b)-H(a,b))\cap conv(P)$, a contradiction. Hence, $H(a,b)$ does not contain y, and without loss of generality $conv(P\cup\{y\})\subseteq H^+(a,b)$ holds. It follows that $H(a,b)$ is also a facet of $conv(P)$, which contradicts that $y\in H^-(a,b)-H(a,b)$ holds (from the definition of $D(x,conv(P))$). □

PROOF OF LEMMA 8: Let $u^*\in V(G/e)$ denote the node obtained by contracting u_1 and u_2 in G. Without loss of generality, assume $u_2\in V-V(G_1)$ (this is possible from the assumption $\{u_1,u_2\}-V(G_1)\neq\emptyset$). Hence $|N_G(u_2)|\geq d+1$ holds. We give a constructive proof of the lemma; we show a way of finding an SC-embedding f of G with boundary G_1 into $\Re^d$. Let $f(v):=f'(v)$ for each node $v\in V(G/e)-\{u^*\}=V(G)-\{u_1,u_2\}$ and $f(u_1):=f'(u^*)$. Note that G_1 also plays the role as G' in Definition 3 (i), and that every node $v\in V(G)-(N_G(u_2)\cup\{u_1,u_2\})$ satisfies $v\in int(conv_f(N_G(v)))$. In the sequel, we consider the position of the node u_2 to be embedded into $\Re^3$, taking the convexity for each node in $\{u_2\}\cup N_G(u_2)$ into account (note that $u_1\in N_G(u_2)$ holds).

First, observe that u_2 needs to be embedded in $int(conv_f(N_G(u_2)))$ for the convexity for u_2 (note that the position of each node $v\in V(G)-\{u_2\}$ has been fixed, so $int(conv_f(N_G(u_2)))$ is well-defined). Since $|N_G(u_2)|\geq d+1$ holds and the points of $\{f(v)\mid v\in N_G(u_2)\}$ are in general position, it follows that $int(conv_f(N_G(u_2)))\neq\emptyset$.

Now, for each node $v\in N_G(u_2)$, we define the subspace D_v in $\Re^d$ such that $D_v=\Re^d$ if $v\in int(conv_f(N_G(v)-\{u_2\}))$ or $v\in V(G_1)$ hold, and $D_v=D(v,conv_f(N_G(v)\cup\{v\}-\{u_2\}))$ if v is a vertex of $conv_f(N_G(v)\cup\{v\}-\{u_2\})$ and $v\in V(G)-V(G_1)$. Note that in the case of $D_v=D(v,conv_f(N_G(v)\cup\{v\}-\{u_2\}))$, $conv_f(N_G(v)\cup\{v\}-\{u_2\})$ is full-dimensional in $\Re^d$. If $v=u_1$ holds, then $u_1\notin V(G_1)$ indicates $|N_G(u_1)|\geq d+1$ and hence we have $|N_G(u_1)\cup\{u_1\}-\{u_2\}|\geq d+1$. If $v\neq u_1$ holds, then $v\in int(conv_{f'}(N_{G/e}(v)))$ indicates that $|N_{G/e}(v)|\geq d+1$, $|N_G(v)-\{u_2\}|\geq|N_{G/e}(v)-\{u^*\}|\geq d$, and $|N_G(v)\cup\{v\}-\{u_2\}|\geq d+1$ hold (note that the points of $\{f'(w)\mid w\in V(G/e)\}=\{f(w)\mid w\in V(G)-\{u_2\}\}$ are in general position). Let $D^*=\bigcap_{v\in N_G(u_2)}D_v$. Lemma 10 implies that we need to embed the node u_2 in D^* for the convexity for each node $v\in N_G(u_2)$.

Consequently, for proving the lemma, it suffices to show that $D^* \cap int(conv_f(N_G(u_2))) \neq \emptyset$ holds; we can embed u_2 in $D^* \cap int(conv_f(N_G(u_2)))$ (while satisfying that the points of $\{f(v) \mid v \in V(G)\}$ are in general position). There are the following two possible cases: (I) $f(u_1) \in int(conv_f(N_G(u_1) - \{u_2\}))$ or $u_1 \in V(G_1)$, (II) otherwise.

(I) In this case, we have $D_{u_1} = \Re^d$; we do not have to consider the convexity for u_1. Since each node $v \in N_G(u_2) - \{u_1\}$ satisfies $v \in N_{G/e}(u^*)$ and $f(v) = f'(v) \in int(conv_{f'}(N_{G/e}(v)))$, it follows that D^* contains the point $f'(u^*) = f(u_1)$. This implies that $D^* \neq \emptyset$. Moreover, since D^* is an open set, we can observe that D^* contains points sufficiently close to $f(u_1)$. This and $u_1 \in N_G(u_2)$ indicate that $D^* \cap int(conv_f(N_G(u_2))) \neq \emptyset$.

(II) In this case, $f(u_1)$ is a vertex of $conv_f(N_G(u_1) \cup \{u_1\} - \{u_2\})$ and $u_1 \in V(G) - V(G_1)$ holds, and hence $D_{u_1} = D(u_1, conv_f(N_G(u_1) \cup \{u_1\} - \{u_2\}))$ holds. Let $D' = \bigcap_{v \in N_G(u_2) - \{u_1\}} D_v$ (note that $D^* = D' \cap D_{u_1}$). Similarly to the arguments in (I), we can observe that D' contains the point $f(u_1)$ and points sufficiently close to $f(u_1)$. From $u_2 \in N_G(u_1)$ and the definition of D_{u_1}, we can observe that if $D_{u_1} \cap int(conv_f(N_G(u_2))) \neq \emptyset$, then some points sufficiently close to $f(u_1)$ included in D' are also contained in $D_{u_1} \cap int(conv_f(N_G(u_2)))$. Based on this observation, for proving $D^* \cap int(conv_f(N_G(u_2))) \neq \emptyset$, it suffices to show that $D_{u_1} \cap int(conv_f(N_G(u_2))) \neq \emptyset$.

Assume by contradiction that $D_{u_1} \cap int(conv_f(N_G(u_2))) = \emptyset$. From the definition of $D_{u_1} = D(u_1, conv_f(N_G(u_1) \cup \{u_1\} - \{u_2\}))$, it follows that there exists a supporting hyperplane $H(a,b)$ of $conv_f(N_G(u_1) \cup \{u_1\} - \{u_2\})$ containing the point $f(u_1)$ such that without loss of generality, $conv_f(N_G(u_1) \cup \{u_1\} - \{u_2\}) \cup conv_f(N_G(u_2)) \subseteq H^+(a,b)$ holds. This and $N_{G/e}(u^*) = (N_G(u_1) - \{u_2\}) \cup (N_G(u_2) - \{u_1\})$ indicate that $H(a,b)$ is a supporting hyperplane of $conv_{f'}(N_{G/e}(u^*) \cup \{u^*\})$ containing $f'(u^*)$ $(= f(u_1))$ in $\Re^d$. It follows that $f'(u^*) \in int(conv_{f'}(N_{G/e}(u^*)))$ cannot hold and it violates the statement (ii) in Definition 3 about f', which contradicts that f' is an SC-embedding of G/e (note that $u^* \notin V(G_1)$ from $\{u_1, u_2\} \subseteq V(G) - V(G_1)$). □

Finally, in the case of $d = 3$, we show that Step 2 can be implemented to run in $O(n^3 \log n)$ time. Since the number of the contraction in Step 1 is $O(n)$, it suffices to show that given an SC-embedding f' of G/e with boundary G_1 into $\Re^3$ for an arc (u_1, u_2) with $\{u_1, u_2\} - V(G_1) \neq \emptyset$, we can find an SC-embedding f of G with boundary G_1 in $O(n^2 \log n)$ time. According to the proof of Lemma 8, we can observe that the time complexity of this procedure depends on that of choosing a location for u_2.

First we need to compute $conv_f(N_G(u_2))$ and D_v for each node $v \in N_G(u_2)$. It is known in [5] that for a set P of points in $\Re^3$, $conv(P)$ can be computed in $O(n \log n)$ time. Moreover, from Definition 9, we can observe that the time complexity of computing $D(v, conv(P))$ depends on that of computing $conv(P)$. Hence it follows that $conv_f(N_G(u_2))$ and D_v for each node $v \in N_G(u_2)$ can be computed in $O(n^2 \log n)$ time. Now since the number of facets representing $conv(P)$ is $O(n)$, the number of hyperplanes representing $D^* \cap conv_f(N_G(u_2))$ is $O(n^2)$. This implies that $D^* \cap conv_f(N_G(u_2))$ can be computed in $O(n^2 \log n)$ time. Moreover, we can show that we can choose $f(u_2) \in D^* \cap conv_f(N_G(u_2))$ in $O(n^2 \log n)$ time so that all points in $f(V(G))$ are in general position. We omit the details.

4 Algorithm for bisecting resource sets

Summarizing the arguments given so far, we give an algorithm, named BISECT3 for finding a 3-bisection as follows.

Algorithm BISECT3
Input: A 4-connected graph $G = (V, E)$ which has a complete graph K with 4 nodes, and three pairwise disjoint node sets T_1, T_2, and T_3 where each $|T_i|$ is even.
Output: A 3-bisection of G with respect to T_1, T_2, and T_3.
Step 1: By executing EMBED3, find an SC-embedding f of G with boundary K into $\Re^3$.
Step 2: By applying a ham-sandwich cut algorithm to $f(V)$ in $\Re^3$, find a plane H in $\Re^3$ which bisects T_1, T_2, and T_3. Output the bisection $\{V_1, V_2\}$ of V divided by H. □

Since Step 1 (resp., Step 2) takes $O(n^3 \log n)$ time (resp., $O(n^{3/2})$ time) from Section 3 (resp., the observation about ham-sandwich cuts in Section 2.1), the time complexity of BISECT3 is $O(n^3 \log n)$, which proves Theorem 1.

5 Remarks

As for a general d, we can observe from Theorem 2 and Lemma 4 that if an SC-embedding of G into $\Re^d$ exists, then G admits a d-bisection with respect to any family $\{T_1, \ldots, T_d\}$ of resource sets. Lemma 8 implies that if

(a) a given graph G has a subgraph G_1 which plays the role as G' in Definition 3 (i),
(b) $|N_G(v)| \geq d + 1$ holds for each node $v \in V(G) - V(G_1)$, and
(c) we can continue contracting arcs not in $E(G_1)$ while preserving the above (b) until a complete graph with $d + 2$ vertices is obtained,

then an SC-embedding of G with boundary G_1 into $\Re^d$ can be found. Hence, a sufficient condition for G to satisfy the above

(a)–(c) indicates one for G to have a d-bisection. Note that for such a G satisfying (a)–(c), we have $|N_G(X)| \geq d+1$ for every set $X \subseteq V(G) - V(G_1)$, and this condition needs to be preserved during the process of the contraction corresponding to (c). This follows since if some $X \subseteq V(G') - V(G_1)$ satisfies $|N_{G'}(X)| \leq d$ for some graph G' obtained during the contraction procedure, then the degree of the node obtained by contracting X becomes at most d in the subsequent contraction procedure.

On the other hand, for example, in the case of $d = 4$, some 5-connected graphs have no 4-bisection as shown in Figure 1(b), and moreover, even if a given graph is 5-connected and contains K_5, then it does not always enjoy the above (a)–(c). This follows since it was shown in [1] that for any graph G_1 with $|V(G_1)| \geq 5$, there exists an uncontractible 5-connected graph which contains G_1 as its induced subgraph.

Finally, we consider the arc-version of the bisection problem: given an undirected graph $G = (V,E)$ and k pairwise disjoint sets $T_1, \ldots, T_k$ of arcs, where each $|T_i|$ is even, find a partition E_1 and E_2 of E such that the graphs induced by E_1 and E_2 are both connected and $|E_1 \cap T_i| = |E_2 \cap T_i| = |T_i|/2$ holds for each $i = 1, \ldots, k$. By applying results about k-bisection problems to the line graph of a given graph G, we can prove that if G is $(k+1)$-edge-connected, then a feasible partition exists for $k = 1,2,3$. This is possible since if a $(k+1)$-edge-connected graph has exactly $k+1$ arcs, then it has exactly two nodes and it is trivial, otherwise then its line graph is $(k+1)$-connected and has K_{k+1}. Moreover, there exist instances that have no feasible partition unless G is $(k+1)$-edge-connected for $k = 1,2,3$. We omit the details.

References

[1] K. ANDO, A. KANEKO, AND K. KAWARABAYASHI, Some properties of 5-contraction critical graphs, *Graphs and Combinatorics*, to appear.

[2] L. CHI-YUAN, J. MATOUŠEK AND W. STEIGER, Algorithms for ham-sandwich cuts, *Discrete Comput. Geom.*, **11** (1994), 433–452.

[3] J. CHLEÍKOVÁ, Approximating the maximally balanced connected partition problem in graphs, *Information Processing Letters* **60** (1999), 225–230.

[4] M. E. DYER AND A. M. FRIEZE, On the complexity of partitioning graphs into connected subgraphs, *Discrete Applied Mathematics* **10** (1985), 139–153.

[5] H. EDELSBRUNNER, Algorithms in combinatorial geometry, Springer-Verlag, Berlin, 1987.

[6] A. KANEVSKY, R. TAMASSIA, G. DI BATTISTA, AND J. CHEN, On-line maintenance of the four-connected components of a graph, *Proc. 32th IEEE Symp. on Foundations of Computer Science*, (1991), 793–801.

[7] N. MARTINOV, A recursive characterization of the 4-connected graphs, *Discrete Mathematics* **84** (1990), 105–108.

[8] H. NAGAMOCHI AND T. IBARAKI, A linear-time algorithm for finding a sparse k-connected spanning subgraph of a k-connected graph, *Algorithmica*, **7**, (1992), 583–596.

[9] H. NAGAMOCHI, T. JORDÁN, Y. NAKAO, AND T. IBARAKI, Convex embeddings bisecting of 3-connected graphs, *Combinatorica* **22**(4) (2002), 537–554.

[10] H. SUZUKI, N. TAKAHASHI, AND T. NISHIZEKI, A linear algorithm for bipartition of biconnected graphs, *Information Processing Letters* **33** (1990), 227–232.

[11] K. WADA AND K. KAWAGUCHI, Efficient algorithms for tripartitioning triconnected graphs and 3-edge-connected graphs, *Lecture Notes in Comput. Sci.*, 790, Springer, Graph-theoretic concepts in computer science, 1994, 132–143.

Algorithm for Partitioning Graphs of Bounded Tree-Width of Supply and Demand

TAKEHIRO ITO

Graduate School of Information Sciences, Tohoku University, Aoba-yama 6-6-05, Sendai, 980-8579, Japan.
take@nishizeki.ecei.tohoku.ac.jp

XIAO ZHOU

Graduate School of Information Sciences, Tohoku University, Aoba-yama 6-6-05, Sendai, 980-8579, Japan.
zhou@ecei.tohoku.ac.jp

TAKAO NISHIZEKI

Graduate School of Information Sciences, Tohoku University, Aoba-yama 6-6-05, Sendai, 980-8579, Japan.
nishi@ecei.tohoku.ac.jp

Abstract: Assume that each vertex of a graph G is either a supply vertex or a demand vertex and is assigned a positive integer, called a supply or a demand. Each demand vertex can receive "power" from at most one supply vertex. One thus wishes to partition G into connected components by deleting edges from G so that each component C has exactly one supply vertex whose supply is no less than the sum of demands of all demand vertices in C. If G has no such partition, one wishes to partition G into connected components so that each component C either has no supply vertex or has exactly one supply vertex whose supply is no less than the sum of demands in C, and wishes to maximize the sum of demands in all components with supply vertices. Such a maximization problem is NP-hard even for trees, and is strong NP-hard for general graphs. In this paper, we give pseudo-polynomial-time algorithm to solve the problem for series-parallel graphs. The algorithm can be easily extended for partial k-trees, that is, graphs with bounded tree-width.

Keywords: demand, maximum partition problem, partial k-tree, series-parallel graph, supply

1 Introduction

Let $G = (V,E)$ be a graph with vertex set V and edge set E. The set V is partitioned into two sets V_s and V_d. Let $|V| = n$, $|V_s| = n_s$ and $|V_d| = n_d$, and hence $n = n_s + n_d$. Each vertex $u \in V_s$ is called a *supply vertex* and is assigned a positive integer $\sup(u)$, called a *supply of u*, while each vertex $v \in V_d$ is called a *demand vertex* and is assigned a positive integer $\mathrm{dem}(v)$, called a *demand of v*. Each demand vertex can receive "power" from at most one supply vertex through edges in G. One thus wishes to partition G into connected components by deleting edges from G so that each component C has exactly one supply vertex whose supply is no less than the sum of demands of all demand vertices in C. However, such a partition does not always exist. So we wish to partition G into connected components so that each component C either has no supply vertex or has exactly one supply vertex whose supply is no less than the sum of demands of all demand vertices in C, and wish to maximize the "fulfillment," that is, the sum of demands of the demand vertices in all components with supply vertices. We call the problem the *maximum partition problem*. Figure 1 illustrates a solution of the maximum partition problem for a graph G, whose fulfillment is $(2+7)+(8+7)+(3+6)+(4+8) = 45$, where each supply vertex is drawn as a rectangle and each demand vertex as a circle, the supply or demand is written inside, the deleted edges are drawn by thick dotted lines, and each connected component is indicated by a thin dotted line.

The maximum partition problem has some applications to the power supply problem for power delivery networks [4, 5]. Let G be a graph of a power delivery network. Each supply vertex v represents a "feeder," which can supply at most an amount $\sup(v)$ of electrical power. Each demand vertex v represents a "load," which requires an amount $\mathrm{dem}(v)$ of electrical power supplied from exactly one of the feeders through a network. Each edge of G represents a cable segment, which can be "turned off" by a switch. Then the maximum partition problem represents the "power supply switching problem" to maximize the sum of all loads that can be supplied powers in a network "reconfigured" by turning off some cable segments.

The maximum partition problem is NP-hard even for trees [4]. Thus it is very unlikely that the maximum partition problem can be solved even for trees in polynomial time. However, the problem can be solved in pseudo-polynomial time for trees, and there is a fully polynomial-time approximation scheme (FPTAS) for the problem on trees [4]. A strong NP-complete problem called 3-PARTITION [3] can be easily reduced in pseudo-polynomial time to the maximum partition problem for a

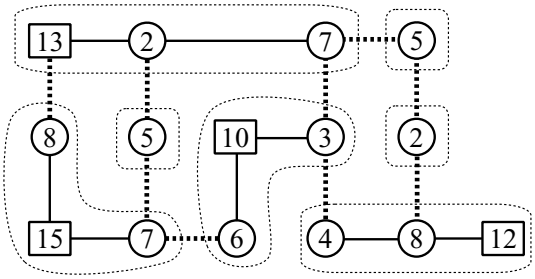


Figure 1: Partition of a graph with maximum fulfillment.

complete bipartite graph, and hence the maximum partition problem is strong NP-hard for general graphs. Therefore, there is no pseudo-polynomial-time algorithm for the problem for general graphs unless P = NP. Hence it is expected to obtain a pseudo-polynomial-time algorithm for a class of graphs, larger than the class of trees.

In this paper we give a pseudo-polynomial-time algorithm to solve the maximum partition problem for series-parallel graphs. (A series-parallel graph will be defined in Section 2.) It takes time $O(m_s^4 n)$, where m_s is the maximum supply, that is, $m_s = \max\{\sup(u) \mid u \in V_s\}$. Thus the algorithm takes polynomial time if m_s is bounded by a polynomial in n.

2 Terminology and Definitions

In this section we give some definitions.

A (*two-terminal*) *series-parallel graph* is defined recursively as follows [6]:

(1) A graph G with a single edge is a series-parallel graph. The ends of the edge are called the *terminals* of G and denoted by $v_s(G)$ and $v_t(G)$. (See Fig. 2(a).)

(2) Let G_1 be a series-parallel graph with terminals $v_s(G_1)$ and $v_t(G_1)$, and let G_2 be a series-parallel graph with terminals $v_s(G_2)$ and $v_t(G_2)$.
   (a) A graph G obtained from G_1 and G_2 by identifying vertex $v_t(G_1)$ with vertex $v_s(G_2)$ is a series-parallel graph, whose terminals are $v_s(G) = v_s(G_1)$ and $v_t(G) = v_t(G_2)$. Such a connection is called a *series connection*, and G is denoted by $G = G_1 \bullet G_2$. (See Fig. 2(b).)
   (b) A graph G obtained from G_1 and G_2 by identifying $v_s(G_1)$ with $v_s(G_2)$ and identifying $v_t(G_1)$ with $v_t(G_2)$ is a series-parallel graph, whose terminals are $v_s(G) = v_s(G_1) = v_s(G_2)$ and $v_t(G) = v_t(G_1) = v_t(G_2)$. Such a connection is called a *parallel connection*, and G is denoted by $G = G_1 \parallel G_2$. (See Fig. 2(c).)

The terminals $v_s(G)$ and $v_t(G)$ of G are often denoted simply by v_s and v_t, respectively. Since we deal with the maximum partition problem, we may assume without loss of generality that G is a simple graph and hence G has no multiple edges.

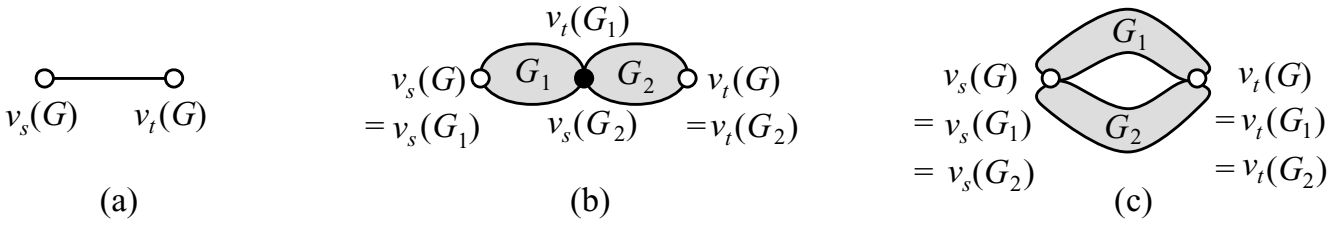


Figure 2: (a) A series-parallel graph with a single edge, (b) series connection, and (c) parallel connection.

A series-parallel graph G can be represented by a "binary decomposition tree" [6]. Figure 3(a) illustrates a series-parallel graph G, and Fig. 3(b) depicts a binary decomposition tree T of G. Labels s and p attached to internal nodes in T indicate series and parallel connections, respectively. Nodes labeled s and p are called *s-* and *p-nodes*, respectively. Every leaf of T represents a subgraph of G induced by a single edge. Each node u of T corresponds to a subgraph G_u of G induced by all edges represented by the leaves that are descendants of u in T. Figure 3(c) depicts G_u for the left child u of the root r of T in Fig. 3(b). G_u is a series-parallel graph for each node u of T, and $G = G_r$ for the root r of T. Since a binary decomposition tree of a given series-parallel graph G can be found in linear time [6], we may assume that a series-parallel graph G and its binary decomposition tree T are given.

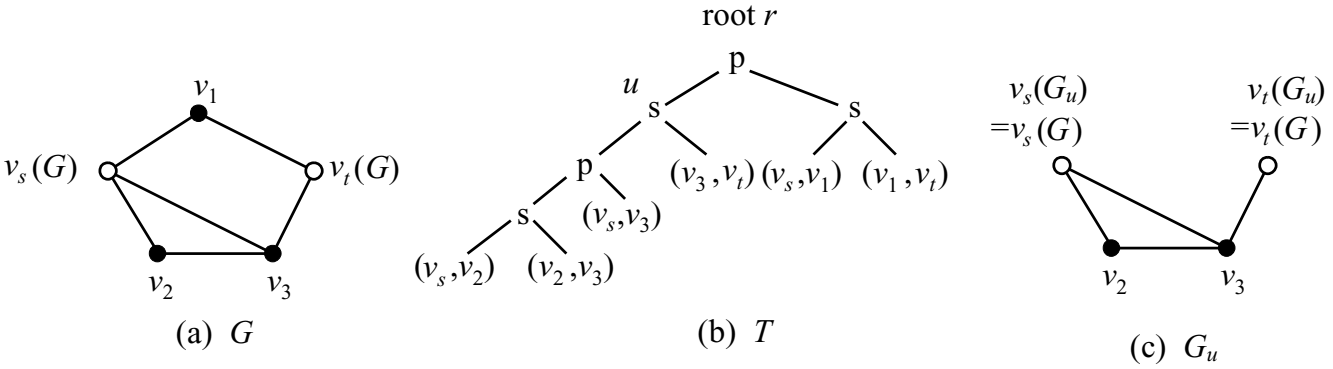


Figure 3: (a) A series-parallel graph G, (b) its binary decomposition tree T, and (c) a subgraph G_u.

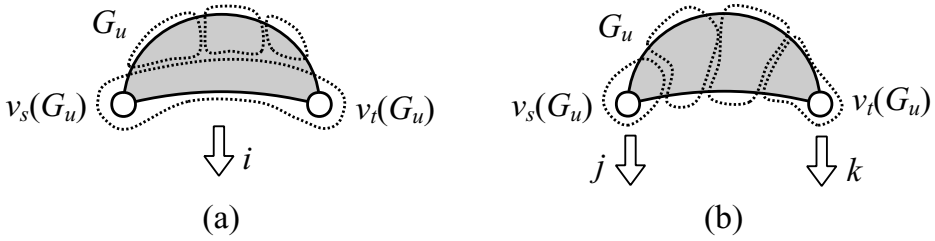


Figure 4: (a) A connected partition, and (b) a separated partition.

3 Algorithm for the Maximum Partition Problem

The main result of the paper is the following theorem.

Theorem 1 *The maximum partition problem can be solved for any series-parallel graph G in time $O(m_s^4 n)$, where n is the number of vertices in G and m_s is the maximum supply.*

In the remainder of this section we give an algorithm to solve the maximum partition problem in time $O(m_s^4 n)$ as a proof of Theorem 1.

We partition a series-parallel graph G into connected components by deleting edges from G so that

(a) each component contains at most one supply vertex; and

(b) if a component C contains a supply vertex, then the supply is no less than the sum of demands of all demand vertices in C.

Such a partition P is called a *partition* of G. The *fulfillment $f(P)$ of a partition P* is the sum of demands of all demand vertices in components with supply vertices. Thus $f(P)$ corresponds to the maximum sum of all loads that are supplied electrical power from feeders through a network reconfigured by cutting off some edges. The *maximum partition problem* is to find a partition of G with the maximum fulfillment. The *maximum fulfillment $f(G)$ of a graph G* is the maximum fulfillment $f(P)$ among all partitions P of G. For the graph G in Fig. 1 the partition P indicated by thin dotted lines has the maximum fulfillment, and hence $f(G) = f(P) = 45$.

Every partition P of a series-parallel graph G naturally induces a partition P' of its subgraph G_u for a node u of a binary decomposition tree T of G. The induced partition P' can be classified into two types of partitions, called a "connected partition" and a "separated partition," which are illustrated in Fig. 4 and will be formally defined later. If a component of P' with a terminal contains a supply vertex, then the component may have the "marginal" power, the amount of which is no greater than m_s. Otherwise, the component may have the "deficient" power, the amount of which should be no greater than m_s. Thus we later introduce two functions $g : (\mathscr{G}, \mathbb{Z}_{m_s}) \to \mathbb{Z}^+$ and $h : (\mathscr{G}, \mathbb{Z}_{m_s}, \mathbb{Z}_{m_s}) \to \mathbb{Z}^+$, where $\mathscr{G}$ denotes the set of all series-parallel graphs, $\mathbb{Z}^+$ denotes the set of all nonnegative integers, and $\mathbb{Z}_{m_s}$ denotes the set of all integers whose absolute values are no greater than m_s. For $G_u \in \mathscr{G}$ and $i, j, k \in \mathbb{Z}_{m_s}$, the values $g(G_u, i)$ and $h(G_u, j, k)$ represent the maximum fulfillment of G_u in a connected partition and in a separated partition of G_u, respectively, and i, j and k represent the amount of "marginal" or "deficient" power in a component with a terminal. Our idea is to compute $g(G_u, i)$ and $h(G_u, j, k)$ from the leaves of T to the root r of T by means of dynamic programming.

We now formally define the notion of connected and separated partitions of a series-parallel graph G. Let P be a partition of a subgraph G_u of G for a node u of a binary decomposition tree T of G, and let $v_s = v_s(G_u)$ and $v_t = v_t(G_u)$. Let $C(P, v_s)$

be the set of all vertices in the component containing v_s, and let $C(P,v_t)$ be the set of all vertices in the component containing v_t. If $C(P,v_s)=C(P,v_t)$, that is, the two terminals v_s and v_t are contained in the same component of P, then we call P a *connected partition* of G_u. (See Fig. 4(a).) If $C(P,v_s)\neq C(P,v_t)$, that is, the two terminals v_s and v_t are contained in the different components of P, we call P a *separated partition* of G_u. (See Fig. 4(b).)

We then classify both connected partitions and separated partitions further into several classes. The "power flow" around a terminal depends on whether the terminal is a supply vertex or a demand vertex. Since we want to deal with the two cases uniformly, we introduce a graph G_u^* for a subgraph G_u of G; let G_u^* be a graph obtained from G_u by regarding each of the two terminals $v_s(G_u)$ and $v_t(G_u)$ as a demand vertex whose demand is zero. It should be noted that a partition of G_u^* is not always a partition of G_u. Let G_u^{in} be the graph obtained from G_u by deleting the two terminals $v_s(G_u)$ and $v_t(G_u)$. Let G_u^{out} be the graph obtained from G by deleting all the vertices of G_u except $v_s(G_u)$ and $v_t(G_u)$.

If P is a connected partition of G_u^*, then $C(P,v_s)=C(P,v_t)$ and we denote it simply by $C(P)$. For each integer $i\in\mathbb{Z}_{m_s}$, we call P an *i-connected partition* of G_u^* if P satisfies the following two conditions (a) and (b):

(a) if $i>0$, then $C(P)$ contains a supply vertex w and $i+\sum_{x\in C(P)-\{w\}}\mathrm{dem}(x)\leq\sup(w)$; and

(b) if $i\leq 0$, then $C(P)$ contains no supply vertex and $\sum_{x\in C(P)}\mathrm{dem}(x)\leq|i|=-i$.

Note that P is an $(i-1)$-connected partition if P is an i-connected partition unless $i=1$. An i-connected partition P of G_u^* with $i>0$ corresponds to a partition of the whole graph G in which all demand vertices in $C(P)$ are supplied power from a supply vertex w in G_u^{in}; an amount i of the remaining power of w can be delivered to G_u^{out} through $v_s(G_u)$ and $v_t(G_u)$; and hence the "margin" of $C(P)$ is i. An i-connected partition P of G_u^* with $i\leq 0$ corresponds to a partition of G in which all (demand) vertices in $C(P)$ are supplied power from a supply vertex in G_u^{out}; an amount $|i|$ of power must be delivered to G_u^{in} through either $v_s(G_u)$ or $v_t(G_u)$, and hence the "deficiency" of $C(P)$ is $|i|$. For an i-connected partition P of G_u^*, let

$$f(P,i)=\begin{cases} f(P) & \text{if } 0<i\leq m_s,\\ f(P)+\sum_{x\in C(P)}\mathrm{dem}(x) & \text{if } -m_s\leq i\leq 0.\end{cases}$$

Thus $f(P,i)$ with $-m_s\leq i\leq 0$ represents the fulfillment of P when an amount $|i|$ of power is delivered to G_u^{in} from a supply vertex in G_u^{out}.

For each pair of integers j and k in $\mathbb{Z}_{m_s}$, we call a separated partition P of G_u^* a *(j,k)-separated partition* if P satisfies the following four conditions (a), (b), (c) and (d):

(a) if $j>0$, then $C(P,v_s)$ contains a supply vertex w and $j+\sum_{x\in C(P,v_s)-\{w\}}\mathrm{dem}(x)\leq\sup(w)$;

(b) if $j\leq 0$, then $C(P,v_s)$ contains no supply vertex and $\sum_{x\in C(P,v_s)}\mathrm{dem}(x)\leq -j$;

(c) if $k>0$, then $C(P,v_t)$ contains a supply vertex w and $k+\sum_{x\in C(P,v_t)-\{w\}}\mathrm{dem}(x)\leq\sup(w)$; and

(d) if $k\leq 0$, then $C(P,v_t)$ contains no supply vertex and $\sum_{x\in C(P,v_t)}\mathrm{dem}(x)\leq -k$.

A (j,k)-separated partition P of G_u^* with $j>0$ corresponds to a partition of the whole graph G in which all demand vertices in $C(P,v_s)$ are supplied power from the supply vertex w in G_u^{in}; an amount j of the remaining power of w can be delivered to G_u^{out} through $v_s(G_u)$, and hence the margin of $C(P,v_s)$ is j. A (j,k)-separated partition P of G_u^* with $j\leq 0$ corresponds to a partition of G in which all (demand) vertices in $C(P,v_s)$ are supplied power from a supply vertex in G_u^{out}; an amount $|j|$ of power must be delivered to G_u^{in} through $v_s(G_u)$, and hence the deficiency of $C(P,v_s)$ is $|j|$. Similarly, a (j,k)-separated partition P of G_u^* with $k>0$ corresponds to a partition of G in which all demand vertices in $C(P,v_t)$ are supplied power from the supply vertex w in G_u^{in}; an amount k of the remaining power of w can be delivered to G_u^{out} through $v_t(G_u)$, and hence the margin of $C(P,v_t)$ is k. A (j,k)-separated partition P of G_u^* with $k\leq 0$ corresponds to a partition of G in which all (demand) vertices in $C(P,v_t)$ are supplied power from a supply vertex in G_u^{out}; an amount $|k|$ of power must be delivered to G_u^{in} through $v_t(G_u)$, and hence the deficiency of $C(P,v_t)$ is k. For a (j,k)-separated partition P of G_u^*, let

$$f(P,j,k)=\begin{cases} f(P) & \text{if } 0<j,k\leq m_s,\\ f(P)+\sum_{x\in C(P,v_t)}\mathrm{dem}(x) & \text{if } 0<j\leq m_s \text{ and } -m_s\leq k\leq 0,\\ f(P)+\sum_{x\in C(P,v_s)}\mathrm{dem}(x) & \text{if } -m_s\leq j\leq 0 \text{ and } 0<k\leq m_s,\\ f(P)+\sum_{x\in C(P,v_s)\cup C(P,v_t)}\mathrm{dem}(x) & \text{if } -m_s\leq j,k\leq 0.\end{cases}$$

Thus $f(P,j,k)$ with nonpositive j or k represents the fulfillment of P when an amount $|j|$ or $|k|$ of power is delivered to G_u^{in} from a supply vertex in G_u^{out} through v_s or v_t.

We now formally define a function $g:(\mathscr{G},\mathbb{Z}_{m_s})\to\mathbb{Z}^+$ for a series-parallel graph $G_u^*\in\mathscr{G}$ and an integer $i\in\mathbb{Z}_{m_s}$, as follows:

$$g(G_u^*,i) \quad = \quad \max\{f(P,i)\mid G_u^* \text{ has an } i\text{-connected partition } P\}.$$

If G_u^* has no i-connected partition, then let $g(G_u^*,i)=-\infty$. We then formally define a function $h:(\mathscr{G},\mathbb{Z}_{m_s},\mathbb{Z}_{m_s})\to\mathbb{Z}^+$ for a series-parallel graph $G_u^*\in\mathscr{G}$ and a pair of integers j and k in $\mathbb{Z}_{m_s}$, as follows:

$$h(G_u^*,j,k) \quad = \quad \max\{f(P,j,k)\mid G_u^* \text{ has a } (j,k)\text{-separated partition } P\}.$$

If G_u^* has no (j,k)-separated partition, then let $h(G_u^*,j,k)=-\infty$.

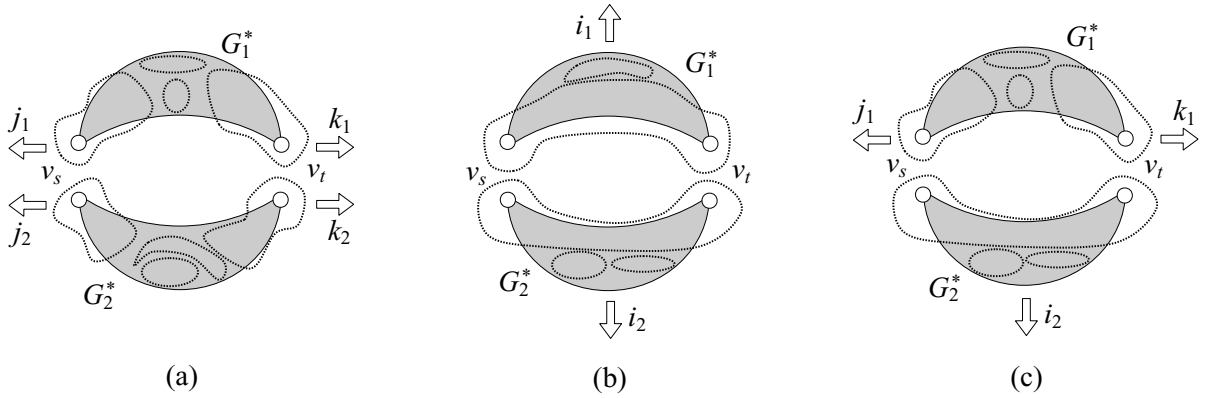


Figure 5: The combinations of a partition P_1 of G_1^* and a partition P_2 of G_2^* for a partition P of $G_u^* = G_1^* \parallel G_2^*$.

Our algorithm computes $g(G_u^*, i)$ and $h(G_u^*, j, k)$ for each node u of a binary decomposition tree T of a given series-parallel graph G from the leaves to the root r of T by means of a dynamic programming. Since $G = G_r$, one can easily compute the maximum fulfillment $f(G)$ from $g(G^*, i)$ and $h(G^*, j, k)$ in time $O(1)$.

We first compute $g(G_u^*, i)$ and $h(G_u^*, j, k)$ for each leaf u of T, for which G_u^* contains exactly one edge. Since the two terminals of G_u^* are demand vertices of demands zero,

$$g(G_u^*, i) = \begin{cases} 0 & \text{if } -m_s \le i \le 0; \\ -\infty & \text{otherwise.} \end{cases} \tag{1}$$

Similarly

$$h(G_u^*, j, k) = \begin{cases} 0 & \text{if } -m_s \le j, k \le 0; \\ -\infty & \text{otherwise.} \end{cases} \tag{2}$$

For each leaf u of T and all integers i, j and k, by Eqs. (1) and (2) one can easily compute $g(G_u^*, i)$ and $h(G_u^*, j, k)$ in time $O(m_s)$ and $O(m_s^2)$, respectively. Since G is a simple series-parallel graph, G has at most $2n-3$ edges and hence T has at most $2n-3$ leaves. One can thus compute $g(G_u^*, i)$ and $h(G_u^*, j, k)$ for all leaves u of T in time $O(m_s^2 n)$.

We next compute $g(G_u^*, i)$ and $h(G_u^*, j, k)$ for each internal node u of T from the counterparts of the two children of u in T.

We first consider a parallel connection.

[Parallel connection]

Let $G_u = G_1 \parallel G_2$, and let $v_s = v_s(G_u^*)$ and $v_t = v_t(G_u^*)$. (See Figs. 2(c) and 5.)

We first compute $h(G_u^*, j, k)$. Every separated partition P of G_u^* can be obtained by combining a separated partition P_1 of G_1^* with a separated partition P_2 of G_2^*, as illustrated in Fig. 5(a). We can thus know that, for each pair (j, k), $h(G_u^*, j, k)$ can be computed as follows:

$$h(G_u^*, j, k) = \max_{j_1, j_2, k_1, k_2} \{h(G_1^*, j_1, k_1) + h(G_2^*, j_2, k_2)\} \tag{3}$$

where the maximum is taken over all integers j_1, j_2, k_1 and k_2 such that

(a) $j_1, j_2, k_1, k_2 \in \mathbb{Z}_{m_s}$;
(b) $j_1 + j_2 = j$ and $k_1 + k_2 = k$;
(c) if $j \le 0$, then $j_1, j_2 \le 0$;
(d) if $j > 0$, then exactly one of the two integers j_1 and j_2 is positive;
(e) if $k \le 0$, then $k_1, k_2 \le 0$; and
(f) if $k > 0$, then exactly one of the two integers k_1 and k_2 is positive.

We next compute $g(G_u^*, i)$. Every connected partition P of G_u^* can be obtained by combining a partition P_1 of G_1^* with a partition P_2 of G_2^*, as illustrated in Figs. 5(b) and (c). There are the following two Cases (a) and (b) to consider, and we define the two functions $g^a(G_u^*, i)$ and $g^b(G_u^*, i)$ for the two cases, respectively.

Case (a): *both P_1 and P_2 are connected partitions.* (See Fig. 5(b).)

We define $g^a(G_u^*, i)$ for each integer $i \in \mathbb{Z}_{m_s}$, as follows:

$$g^a(G_u^*, i) = \max_{i_1, i_2} \{g(G_1^*, i_1) + g(G_2^*, i_2)\} \tag{4}$$

where the maximum is taken over all integers i_1 and i_2 such that

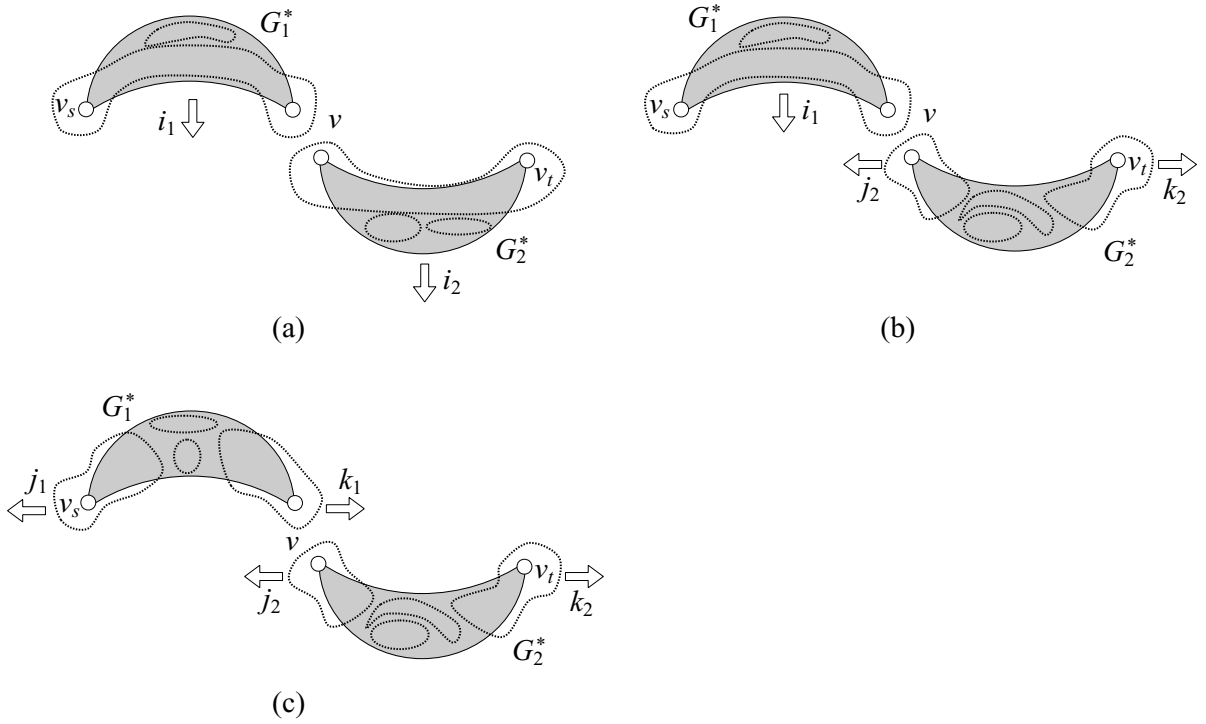


Figure 6: The combinations of a partition P_1 of G_1^* and a partition P_2 of G_2^* for a partition P of G_u^*, where $G_u = G_1 \bullet G_2$.

(a) $i_1, i_2 \in \mathbb{Z}_{m_s}$;
(b) $i_1 + i_2 = i$;
(c) if $i \leq 0$, then $i_1, i_2 \leq 0$; and
(d) if $i > 0$, then exactly one of the two integers i_1 and i_2 is positive.

Case (b): *one of P_1 and P_2 is a separated partition and the other is a connected partition.*

One may assume without loss of generality that P_1 is a separated partition and P_2 is a connected partition. (See Fig. 5(c).) We define $g^b(G_u^*, i)$ for each integer $i \in \mathbb{Z}_{m_s}$, as follows:

$$g^b(G_u^*, i) \quad = \quad \max_{j_1, k_1, i_2} \{h(G_1^*, j_1, k_1) + g(G_2^*, i_2)\} \tag{5}$$

where the maximum is taken over all integers j_1, k_1 and i_2 such that
(a) $j_1, k_1, i_2 \in \mathbb{Z}_{m_s}$;
(b) $j_1 + k_1 + i_2 = i$;
(c) if $i \leq 0$, then $j_1, k_1, i_2 \leq 0$; and
(d) if $i > 0$, then exactly one of the three integers j_1, k_1 and i_2 is positive.

From g^a and g^b above, one can compute $g(G_u^*, i)$ as follows:

$$g(G_u^*, i) \quad = \quad \max\{g^a(G_u^*, i),\ g^b(G_u^*, i)\}. \tag{6}$$

We next consider a series connection.

[Series connection]

Let $G_u = G_1 \bullet G_2$, and let v be the vertex of G identified by the series connection, that is, $v = v_t(G_1) = v_s(G_2)$. (See Figs. 2(b) and 6.) We define $sd(v)$ as follows:

$$sd(v) = \begin{cases} \sup(v) & \text{if } v \text{ is a supply vertex,} \\ -\mathrm{dem}(v) & \text{if } v \text{ is a demand vertex} \end{cases}$$

For convenience' sake, we define $\mathrm{dem}(w) = 0$ for each supply vertex w in G.

We first compute $g(G_u^*, i)$. Every connected partition P of G_u^* can be obtained by combining a connected partition P_1 of G_1^* with a connected partition P_2 of G_2^*, as illustrated in Fig. 6(a). Therefore $g(G_u^*, i)$ can be computed for each integer $i \in \mathbb{Z}_{m_s}$, as follows:

$$g(G_u^*, i) \quad = \quad \max_{i_1, i_2}\{g(G_1^*, i_1) + g(G_2^*, i_2) + \mathrm{dem}(v)\} \tag{7}$$

where the maximum is taken over all integers i_1 and i_2 such that

(a) $i_1, i_2 \in \mathbb{Z}_{m_s}$;
(b) $i_1 + i_2 + sd(v) = i$;
(c) if $i \leq 0$, then v is a demand vertex and $i_1, i_2 \leq 0$; and
(d) if $i > 0$, then exactly one of the three integers i_1, i_2 and $sd(v)$ is positive.
If such integers i_1 and i_2 do not exist, then we let $g(G_u^*, i) = -\infty$.

We next compute $h(G_u^*, j, k)$. Every separated partition P of G_u^* can be obtained by combining a partition P_1 of G_1^* with a partition P_2 of G_2^*, as illustrated in Figs. 6(b) and (c). There are the following two Cases (a) and (b) to consider, and we define the two functions $h^a(G_u^*, j, k)$ and $h^b(G_u^*, j, k)$ for the two cases, respectively.
Case (a): *one of P_1 and P_2 is a connected partition and the other is a separated partition.*

One may assume without loss of generality that P_1 is a connected partition and P_2 is a separated partition. (See Fig. 6(b).) We define $h^a(G_u^*, j, k)$ for each pair (j, k), as follows:

$$h^a(G_u^*, j, k) \quad = \quad \max_{i_1, j_2} \{g(G_1^*, i_1) + h(G_2^*, j_2, k) + \mathrm{dem}(v)\} \tag{8}$$

where the maximum is taken over all integers i_1 and j_2 such that
(a) $i_1, j_2 \in \mathbb{Z}_{m_s}$;
(b) $i_1 + j_2 + sd(v) = j$;
(c) if $j \leq 0$, then v is a demand vertex and $i_1, j_2 \leq 0$; and
(d) if $j > 0$, then exactly one of the three integers i_1, j_2 and $sd(v)$ is positive.
If such integers i_1 and j_2 do not exist, then we define $h^a(G_u^*, j, k) = -\infty$.
Case (b): *both P_1 and P_2 are separated partitions.* (See Fig. 6(c).)

In this case, either (i) all demand vertices in $C(P_1, v) \cup C(P_2, v)$ are supplied power or (ii) none of them is supplied power. For the first case (i), we define $h^{\mathrm{i}}(G_u^*, j, k)$ for each pair (j, k) as follows:

$$h^{\mathrm{i}}(G_u^*, j, k) \quad = \quad \max_{k_1, j_2} \{h(G_1^*, j, k_1) + h(G_2^*, j_2, k) + \mathrm{dem}(v)\} \tag{9}$$

where the maximum is taken over all integers k_1 and j_2 such that
(a) $k_1, j_2 \in \mathbb{Z}_{m_s}$;
(b) $k_1 + j_2 + sd(v) \geq 0$; and
(c) exactly one of the three integers k_1, j_2 and $sd(v)$ is positive.
If such integers k_1 and j_2 do not exist, then we define $h^{\mathrm{i}}(G_u^*, j, k) = -\infty$.

For the second case (ii), we define $h^{\mathrm{ii}}(G_u^*, j, k)$ for each pair (j, k) as follows:

$$h^{\mathrm{ii}}(G_u^*, j, k) \quad = \quad h(G_1^*, j, 0) + h(G_2^*, 0, k). \tag{10}$$

We then define $h^b(G_u^*, j, k)$ for each pair (j, k) as follows:

$$h^b(G_u^*, j, k) \quad = \quad \max\{h^{\mathrm{i}}(G_u^*, j, k), h^{\mathrm{ii}}(G_u^*, j, k)\}. \tag{11}$$

From h^a and h^b above, one can compute $h(G_u^*, j, k)$ as follows:

$$h(G_u^*, j, k) \quad = \quad \max\{h^a(G_u^*, j, k),\ h^b(G_u^*, j, k)\}. \tag{12}$$

For each p-node u of T and all integers i, j and k in $\mathbb{Z}_{m_s}$ by Eqs. (3)–(6) one can compute $g(G_u^*, i)$ and $h(G_u^*, j, k)$ in time $O(m_s^3)$ and $O(m_s^4)$, respectively. For each s-node u of T and all integers i, j and k in $\mathbb{Z}_{m_s}$, by Eq. (7)–(12) one can compute $g(G_u^*, i)$ and $h(G_u^*, j, k)$ in time $O(m_s^2)$ and $O(m_s^4)$, respectively. In this way one can compute $g(G_u^*, i)$ and $h(G_u^*, j, k)$ for each internal node u of T in time $O(m_s^4)$ regardless of whether u is a p-node or an s-node. Since T is a binary tree and has at most $2n-3$ leaves, T has at most $2n-4$ internal node. One can thus compute $g(G^*, i)$ and $h(G^*, j, k)$ in time $O(m_s^4 n)$ since $G = G_r$ for the root r of T.

Thus the maximum partition problem can be solved in time $O(m_s^4 n)$. This completes a proof of Theorem 1.

4 Conclusions

In this paper we obtained a pseudo-polynomial-time algorithm to compute the maximum fulfillment $f(G)$ of a given series-parallel graph G. The algorithm takes time $O(m_s^4 n)$, and hence takes polynomial time if m_s is bounded by a polynomial in n. It is easy to modify the algorithm so that it actually finds a partition of a series-parallel graph.

Our algorithm for series-parallel graphs can be easily extended for partial k-trees, that is, graphs with bounded tree-width [1, 2]. The extended algorithm takes time $O(m_s^{2(k+1)} n)$.

References

[1] S. ARNBORG, J. LAGERGREN AND D. SEESE, Easy problems for tree-decomposable graphs, *J. Algorithms* (1991) **12** 308–340.

[2] H. L. BODLAENDER, Polynomial algorithms for graph isomorphism and chromatic index on partial *k*-trees, *J. Algorithms* (1990) **11** 631–643.

[3] M. R. GAREY AND D. S. JOHNSON, Computers and Intractability: A Guide to the Theory of NP-Completeness, *Freeman, San Francisco, CA* (1979).

[4] T. ITO, X. ZHOU AND T. NISHIZEKI, Partitioning trees of supply and demand, *Proc. ISAAC'02, Lecture Notes in Computer Science* (2002) **2518** 612–623, also *International J. of Foundations of Computer Science*, to appear.

[5] A. B. MORTON AND I. M. Y. MAREELS, An efficient brute-force solution to the network reconfiguration problem, *IEEE Trans. on Power Delivery* (2000) **15** 996–1000.

[6] K. TAKAMIZAWA, T. NISHIZEKI AND N. SAITO, Linear-time computability of combinatorial problems on series-parallel graphs, *J. ACM* (1982) **29** 623–641.

New classes of facets of cut polytope and tightness of I_{mm22} Bell inequalities

DAVID AVIS

School of Computer Science
McGill University
3480 University St.
Montreal, Quebec
Canada H3A 2A7
avis@cs.mcgill.ca

TSUYOSHI ITO

Dept. of Computer Science
Grad. School of Info. Sci. and Tech.
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo
113-0033 Japan
tsuyoshi@is.s.u-tokyo.ac.jp

Abstract: The Grishukhin inequality Gr_7 is a facet of $\mathrm{CUT}_7^{\square}$, the cut polytope on seven points, which is "sporadic" in the sense that its proper generalization has not been known. In this paper, we extend Gr_7 to an inequality $I(G,H)$ valid for $\mathrm{CUT}_{n+1}^{\square}$ where G and H are graphs with n nodes satisfying certain conditions, and prove a necessary and sufficient condition for $I(G,H)$ to be a facet. This result combined with the triangular elimination theorem of Avis, Imai, Ito and Sasaki settles Collins and Gisin's conjecture in quantum theory affirmatively: the I_{mm22} Bell inequality is a facet of the correlation polytope $\mathrm{COR}^{\square}(\mathrm{K}_{m,m})$ of the complete bipartite graph $\mathrm{K}_{m,m}$ for all $m \geq 1$. We also extend the Gr_8 facet inequality of $\mathrm{CUT}_8^{\square}$ to an inequality $I'(G,H,C)$ valid for $\mathrm{CUT}_{n+2}^{\square}$, and provide a sufficient condition for $I'(G,H,C)$ to be a facet.

Keywords: cut polytope, Grishukhin inequality, I_{mm22} Bell inequality, correlation polytope

1 Introduction

Cut polytopes are convex polytopes which arise in many different fields [6, 7, 8]. Since testing membership in cut polytopes is NP-complete [1], it is unlikely that there exists a concise and complete description of their facial structure in general. Much efforts has been devoted to identifying classes of inequalities which are valid for cut polytopes and have good properties. Hypermetric, clique-web and parachute inequalities are examples of classes of valid inequalities for which important subclasses are facet inducing. For $N \leq 6$, all facets of $\mathrm{CUT}_N^{\square}$, the cut polytope of complete graph K_N, are hypermetric. However, $\mathrm{CUT}_7^{\square}$ has a facet called the *Grishukhin inequality* Gr_7 which is not known to belong to any such general class. Efforts have been made to relate Gr_7 to other inequalities. As a result, De Simone, Deza and Laurent [5] showed that Gr_7 is a collapse of a pure facet inequality Gr_8 of $\mathrm{CUT}_8^{\square}$.

The cut polytope $\mathrm{CUT}^{\square}(\mathrm{K}_{1,m,m})$ of the complete tripartite graph $\mathrm{K}_{1,m,m}$ is linearly isomorphic to the correlation polytope $\mathrm{COR}^{\square}(\mathrm{K}_{m,m})$ of the complete bipartite graph $\mathrm{K}_{m,m}$. In quantum theory, the correlation polytope $\mathrm{COR}^{\square}(\mathrm{K}_{m,m})$ is seen as the set of possible results of a series of Bell experiments with a non-entangled (separable) quantum state shared by two distant parties, where each party has m choices of measurements. In this context, a valid inequality of $\mathrm{COR}^{\square}(\mathrm{K}_{m,m})$ is called a *Bell inequality* and if facet inducing, a *tight Bell inequality*. Readers are referred to [11] for further information about Bell inequalities. Collins and Gisin [4] found a class of I_{mm22} inequalities valid for $\mathrm{COR}^{\square}(\mathrm{K}_{m,m})$ for general m and conjectured that for all $m \geq 1$, I_{mm22} inequality is tight, or equivalently, that it is a facet of $\mathrm{COR}^{\square}(\mathrm{K}_{m,m})$.

Avis, Imai, Ito and Sasaki [2] introduced an operation called *triangular elimination* to convert a facet of $\mathrm{CUT}_N^{\square}$ to a facet of $\mathrm{CUT}^{\square}(\mathrm{K}_{1,m,m})$ for appropriate m. By using this operation, the tightness of the I_{3322} and I_{4422} Bell inequalities follows from the fact that the pure pentagonal and the Grishukhin inequalities are facets of $\mathrm{CUT}_5^{\square}$ and $\mathrm{CUT}_7^{\square}$, respectively. This suggests that some natural extensions of the pure pentagonal and the Grishukhin inequalities may give facets of $\mathrm{CUT}_{2m-1}^{\square}$ for $m \geq 3$. We will prove that it is the case and that hence the conjecture by Collins and Gisin is true. More specifically, we will introduce inequalities $I(G,H)$ valid for $\mathrm{CUT}_{n+1}^{\square}$ where G and H are graphs with n nodes which satisfy certain conditions described later, and prove a necessary and sufficient condition for $I(G,H)$ to be a facet.

As further extensions, we apply to $I(G,H)$ an operation similar to the one used to construct Gr_8 from Gr_7. Actually this operation gives inequalities $I'(G,H,C)$ valid for $\mathrm{CUT}_{n+2}^{\square}$ where C is a cycle of length four in G. We will give a sufficient condition for $I'(G,H,C)$ to be a facet, generalizing the fact that Gr_8 is a facet of $\mathrm{CUT}_8^{\square}$.

The rest of the paper is organized as follows. In Section 2, we review the tools used later. In Section 3, we introduce the inequality $I(G,H)$ valid for the cut polytope, which is a generalization of the Gr_7 inequality, and we prove a necessary and sufficient condition for it to be a facet. Section 4 defines the valid inequality $I'(G,H,C)$, which is a generalization of the Gr_8 inequality, and we provide a sufficient condition for it to be a facet. In Section 5, we prove the tightness of I_{mm22} Bell inequalities.

2 Preliminaries

2.1 Cut polytopes

Here we review the definition of and results on cut polytopes only briefly. Readers are referred to the book by Deza and Laurent [8] for details.

Definition The *cut polytope* $\mathrm{CUT}^{\square}(G)$ of a graph $G=(V,E)$ is a convex polytope in the vector space $\mathbf{R}^E$ defined as the convex hull of the $2^{|V|-1}$ different cut vectors $\boldsymbol{\delta}_G(S)$ for $S \subseteq V$. The cut vector $\boldsymbol{\delta}_G(S) \in \mathbf{R}^E$ is a 0/1 vector defined by $\delta_{uv}(S)=1$ if and only if exactly one of u and v belongs to S, where uv denotes the edge connecting two nodes u and v. The cut polytope $\mathrm{CUT}^{\square}(\mathrm{K}_N)$ of the complete graph K_N is denoted by $\mathrm{CUT}^{\square}_N$.

Similarly, the *correlation polytope* $\mathrm{COR}^{\square}(G)$ is a convex polytope in $\mathbf{R}^{V\cup E}$ defined as the convex hull of the $2^{|V|}$ correlation vectors $\boldsymbol{p}_G(S)$ for $S \subseteq V$. The correlation vector $\boldsymbol{p}_G(S) \in \mathbf{R}^{V\cup E}$ is a 0/1 vector defined by $p_u(S)=1$ if and only if $u \in S$ and $p_{uv}(S)=1$ if and only if $\{u,v\} \subseteq S$.

The correlation polytope $\mathrm{COR}^{\square}(G)$ of a graph $G=(V,E)$ is linearly isomorphic to $\mathrm{CUT}^{\square}(\nabla G)$, where ∇G is the *suspension graph* of G: the graph obtained by adding to G a new node Z adjacent to all the nodes of G. The linear isomorphism between them is called the *covariance mapping*: $p_u = x_{\mathrm{Z}u}$ for $u \in V$ and $p_{uv} = \frac{1}{2}(x_{\mathrm{Z}u} + x_{\mathrm{Z}v} - x_{uv})$ for $u,v \in V$, $u \neq v$.

Hypermetric inequalities Let $N \geq 3$ be an integer and $\boldsymbol{b} \in \mathbf{Z}^N$ an integer vector with $\sum_{i=1}^N b_i = 1$. The inequality

$$\sum_{1\leq i<j\leq N} b_i b_j x_{ij} \leq 0$$

is valid for $\mathrm{CUT}^{\square}_N$ and called the *hypermetric inequality* defined by the vector $\boldsymbol{b}$.

While an exact characterization of when a hypermetric inequality becomes a facet of $\mathrm{CUT}^{\square}_N$ is not known, many sufficient conditions are known. We review here some of them which we use later.

Theorem 1 (Corollary 28.2.5 (i) in [8]) *Let $s \geq 1$ be an integer, and $\boldsymbol{b} \in \mathbf{Z}^N$ be an integer vector with $s+1$ entries equal to 1, s entries equal to -1 and the other $N-(2s+1)$ entries equal to 0. Then the hypermetric inequality defined by $\boldsymbol{b}$ is a facet of $\mathrm{CUT}^{\square}_N$. This inequality is called a* pure $(2s+1)$-gonal inequality, *or if $s=1$, simply a* triangle inequality.

We define $T(u,v;w) = x_{uv} - x_{uw} - x_{vw}$. By using this notation, a triangle inequality is written as $T(u,v;w) \leq 0$.

Theorem 2 ("If" part of Theorem 28.2.4 (iiib) in [8]) *The hypermetric inequality defined by $\boldsymbol{b}$ with $b_1 = \cdots = b_{N-2} = 1$, $b_{N-1} = -1$ and $b_N = -N+4$ is a facet of $\mathrm{CUT}^{\square}_N$.*

Switching of inequality We mention three operations on inequalities valid for cut polytopes. One is the switching operation. Let $G=(V,E)$ be a graph, $\boldsymbol{a} \in \mathbf{R}^E$ and $a_0 \in \mathbf{R}$. The *switching* of the inequality $\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} \leq a_0$ by the cut $S \subseteq V$ is an inequality $\boldsymbol{b}^{\mathrm{T}}\boldsymbol{x} \leq b_0$ with $b_{ij} = (-1)^{\delta_{ij}(S)} \cdot a_{ij}$ and $b_0 = a_0 - \boldsymbol{a}^{\mathrm{T}}\boldsymbol{\delta}_G(S)$.

Switching is an automorphism of the cut polytope $\mathrm{CUT}^{\square}(G)$. Therefore $\boldsymbol{b}^{\mathrm{T}}\boldsymbol{x} \leq b_0$ is valid (resp. a facet) if and only if $\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} \leq a_0$ is valid (resp. a facet).

Collapsing and lifting of inequality The other two operations are collapsing and lifting. Let $G=(V,E)$ be a complete graph on node set V and $uv \in E$. Let $G'=(V',E')$ be the complete graph on node set $V' = (V \setminus \{u,v\}) \cup \{w\}$ with a new node w.

The (u,v)-*collapse* of a vector $\boldsymbol{a} \in \mathbf{R}^E$ is a vector $\boldsymbol{a}^{u,v} \in \mathbf{R}^{E'}$ defined by

$$\begin{aligned} a^{u,v}_{ij} &= a_{ij} && \text{for } i,j \in V \setminus \{u,v\},\ i \neq j, \\ a^{u,v}_{wi} &= a_{ui} + a_{vi} && \text{for } i \in V \setminus \{u,v\}. \end{aligned}$$

For $\boldsymbol{a} \in \mathbf{R}^E$ and $a_0 \in \mathbf{R}$, an inequality $(\boldsymbol{a}^{u,v})^{\mathrm{T}}\boldsymbol{x} \leq a_0$ is said to be the (u,v)-*collapse* of the inequality $\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} \leq a_0$.

If the inequality $\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} \leq a_0$ is valid for $\mathrm{CUT}^{\square}(G)$, its collapse $(\boldsymbol{a}^{u,v})^{\mathrm{T}}\boldsymbol{x} \leq a_0$ is valid for $\mathrm{CUT}^{\square}(G')$.

The opposite operation of collapsing is called *lifting*. The following lemma provides a sufficient condition for lifting to preserve a facet. The proof of the lemma is given below Lemma 26.5.3 in the book [8].

Lemma 3 (Lifting lemma [8]) *Let $\boldsymbol{a} \in \mathbf{R}^E$. The inequality $\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} \leq 0$ is a facet of $\mathrm{CUT}^{\square}(G)$ if the following conditions are satisfied.*

(i) The inequality $\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} \leq 0$ is valid for $\mathrm{CUT}^{\square}(G)$, and its (u,v)-collapse $(\boldsymbol{a}^{u,v})^{\mathrm{T}}\boldsymbol{x} \leq 0$ is a facet of $\mathrm{CUT}^{\square}(G')$.

(ii) There exist $|V|-1$ subsets T_j of V with $u \notin T_j$ and $v \in T_j$ such that the cut vectors $\boldsymbol{\delta}_G(T_j)$ are roots (vertices lying on the face) of $\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} \leq 0$ and the incidence vectors of T_j are linearly independent.

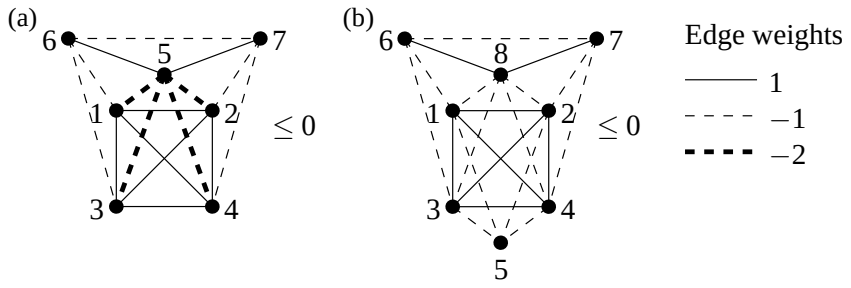


Figure 1: (a) The Grishukhin inequality Gr_7, which is a facet of $\mathrm{CUT}_7^{\square}$. (b) The Gr_8 inequality, which is a facet of $\mathrm{CUT}_8^{\square}$.

Grishukhin inequality The cut polytope $\mathrm{CUT}_7^{\square}$ has 11 inequivalent facets under permutation and switching symmetries [9, 5]. All but one of them belong to at least one of three general classes of valid inequalities: hypermetric, clique-web and parachute inequalities. The remaining facet is not known to belong to any classes that are as general as these classes. This "sporadic" facet is called the *Grishukhin inequality* Gr_7. The Grishukhin inequality looks like $\sum_{1\le i<j\le 4} x_{ij} + x_{56} + x_{57} - x_{67} - x_{16} - x_{36} - x_{27} - x_{47} - 2\sum_{1\le i\le 4} x_{5i} \le 0$ and illustrated in Figure 1 (a).

De Simone, Deza and Laurent [5] found a facet of $\mathrm{CUT}_8^{\square}$ which is pure (all the coefficients are 0 or ± 1) and is a lifting of Gr_7. This facet is called Gr_8 in [8] and illustrated in Figure 1 (b).

2.2 Bell inequalities

I_{mm22} Bell inequalities Collins and Gisin [4] showed that the I_{mm22} inequalities:

$$-p_{\mathrm{A}_1} - \sum_{1\le j\le m}(m-j)p_{\mathrm{B}_j} - \sum_{\substack{2\le i,j\le m\\ i+j=m+2}} p_{\mathrm{A}_i\mathrm{B}_j} + \sum_{\substack{1\le i,j\le m\\ i+j\le m+1}} p_{\mathrm{A}_i\mathrm{B}_j} \le 0, \tag{1}$$

are valid for $\mathrm{COR}^{\square}(\mathrm{K}_{m,m})$ for all $m \ge 1$, generalizing CHSH inequality [3] for $m = 2$ which is a facet of $\mathrm{COR}^{\square}(\mathrm{K}_{2,2})$. They conjectured that for any $m \ge 1$, the I_{mm22} inequality is a facet of $\mathrm{COR}^{\square}(\mathrm{K}_{m,m})$, and showed that the conjecture is true for $m \le 7$.

Triangular elimination Avis, Imai, Ito and Sasaki [2] proposed *triangular elimination* operation to convert any facet inequality of $\mathrm{CUT}_n^{\square}$ other than the triangle inequality to a facet of $\mathrm{CUT}^{\square}(\mathrm{K}_{1,m,m})$ for appropriate m. A basic step in this conversion is described in the following theorem.

Theorem 4 ([2]) *Let $G = (V,E)$ be a graph and $uu' \in E$ an edge of G. Let $W \subseteq \mathrm{N}_G(u) \cap \mathrm{N}_G(u')$ be a set of nodes that are adjacent to both u and u'. We define a graph $G^+ = (V^+, E^+)$, the* detour extension *of G, as follows. We add a new node v to G in the middle of the edge uu', dividing uu' into two edges uv and $u'v$, and add new edges vw for each $w \in W$.*

Let $\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} \le a_0$ be a facet inequality of $\mathrm{CUT}^{\square}(G)$. Define $\boldsymbol{b}^{\mathrm{T}}\boldsymbol{x} \le a_0$, the triangular elimination *of $\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} \le a_0$, to be the inequality obtained by combining the triangle inequality $-a_{uu'}x_{uu'} + a_{uu'}x_{uv} - |a_{uu'}|x_{u'v} \le 0$ with $\boldsymbol{a}^{\mathrm{T}}\boldsymbol{p} \le a_0$.*

If there exists an edge $e \in E \setminus (\{uu'\} \cup \{uw, u'w \mid w \in W\})$ such that $a_e \ne 0$, then the inequality $\boldsymbol{b}^{\mathrm{T}}\boldsymbol{x} \le a_0$ is a facet of $\mathrm{CUT}^{\square}(G^+)$.

3 Inequality $I(G,H)$: A generalization of Gr_7

In this section, we define the inequality $I(G,H)$ valid for the cut polytope, and give a necessary and sufficient condition for $I(G,H)$ to be a facet.

First we define the inequality. Let $n \ge 1$ be an integer, and $G = (V,E)$ and $H = (V,F)$ be two graphs with n nodes. We require that the edges of H are node-disjoint. Let $t = |F|$ and $k = n - t$, and we denote the connected component decomposition of H by $V = V_1 \cup \cdots \cup V_k$. Note that the size of any connected component V_i is one or two. Finally we require that E contains exactly $\binom{k}{2}$ edges: for each $1 \le i < j \le k$ there is an edge e_{ij} connecting a node in V_i and a node in V_j. We consider the following inequality which we denote as $I(G,H)$:

$$\sum_{uv\in E} T(u,v;n+1) - \sum_{uv\in F} T(u,v;n+1) + 2\sum_{V_i=\{u\}} x_{u,n+1} \le 2. \tag{2}$$

For example, $I(\mathrm{K}_2, \overline{\mathrm{K}}_2)$ is identical to the triangle inequality and $I(\mathrm{K}_4, \overline{\mathrm{K}}_4)$ to the pure pentagonal inequality, where K_n is the complete graph on n nodes, and $\overline{\mathrm{K}}_n$ is its complement.

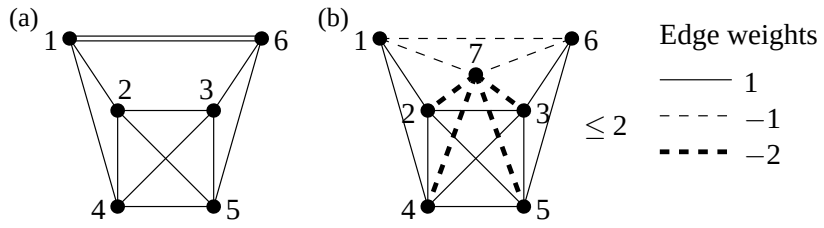


Figure 2: (a) A graph $G_6 = (V,E)$ (edges drawn as single lines) and a graph $H_{5,1} = (V,F)$ (an edge drawn as a double line). (b) The inequality $I(G_6, H_{5,1})$, which is a switching of the Gr_7 inequality.

It is sometimes convenient to relabel the nodes in V so that H is in a restricted form. For $k \geq 1$ and $0 \leq t \leq k$, let $H_{k,t} = (V,E)$ be a graph with node set $V = \{1,\ldots,k+t\}$ and edge set $E = \{(i,k+i) \mid 1 \leq i \leq t\}$. Then any graph H with $n = k+t$ nodes and t node-disjoint edges can be relabelled to $H_{k,t}$, and therefore we can restrict $I(G,H)$ to $I(G,H_{k,t})$ without loss of generality.

We check that the Gr_7 inequality is a switching of an inequality of this kind. Let $G_6 = (V,E)$ and $H_{5,1} = (V,F)$ be the graphs with six nodes shown in Figure 2 (a). Then the inequality $I(G_6, H_{5,1})$ is as shown in Figure 2 (b). We switch $I(G_6, H_{5,1})$ by the cut $\{1,6\}$ and change the labels of nodes $1,2,3,4,5,6,7$ to $6,1,2,3,4,7,5$, respectively. Then the resulting inequality is identical to Gr_7.

Now we prove the validity of $I(G,H)$.

Proposition 5 *The inequality $I(G,H)$ is valid for $\mathrm{CUT}^{\square}_{n+1}$. In addition, the cut vector $\boldsymbol{\delta}(S)$ with $S \subseteq V$ is a root of $I(G,H)$ if and only if one of the following conditions is satisfied.*

(i) There exists a unique i such that $V_i \subseteq S$, and no edge of G is contained in S.

(ii) There exist exactly two values of i (let them be i_1 and i_2) such that $V_i \subseteq S$. In addition, $e_{i_1 i_2}$ is the only edge of G that is contained in S.

PROOF: We show that the cut vector $\boldsymbol{\delta}(S)$ defined by any subset S of V satisfies the inequality $I(G,H)$. Note that with $\boldsymbol{x} = \boldsymbol{\delta}(S)$, each term evaluates to either to zero or two.

Let $A = \{i \mid V_i \subseteq S\}$ and $B = \{ij \mid e_{ij} \subseteq S\}$. The left hand side of $I(G,H)$ evaluated with $\boldsymbol{x} = \boldsymbol{\delta}(S)$ is equal to $2|A| - 2|B|$. Now $|B| \geq \binom{|A|}{2}$, since for each of the $\binom{|A|}{2}$ pairs ij of elements of A, there is an edge e_{ij} with both endpoints in S. Therefore we have $2|A| - 2|B| \leq 3|A| - |A|^2 = 2 - (|A|-1)(|A|-2) \leq 2$. So (2) is valid.

The condition for roots is obtained from the fact that this inequality is satisfied with equality if and only if $|A|$ is one or two and $|B| = \binom{|A|}{2}$. $\square$

Now we consider when the inequality $I(G,H)$ becomes a facet of $\mathrm{CUT}^{\square}_{n+1}$.

Theorem 6 *Assume $k \geq 3$. Then the inequality $I(G,H)$ is a facet of $\mathrm{CUT}^{\square}_{n+1}$ if and only if all nodes in G have degree at least two.*

PROOF: As mentioned above, we can assume $H = H_{k,t}$ without loss of generality.

First we prove the "only if" part. Let u be a node whose degree in G is at most one. In this case $H_{k,t}$ has an edge incident to node u. Without loss of generality, we assume $u = k+t$. If the degree of node $k+t$ in G is one, then let v be the only node that is adjacent to node $k+t$ in G. Otherwise let $v = n+1$. In both cases, $I(G,H_{k,t})$ is the sum of a triangle inequality $T(k+t,v;t) \leq 0$ and the inequality $I(G/(t,k+t), H_{k,t-1})$, where $G/(t,k+t)$ is a graph obtained from G by identifying two nodes t and $k+t$ into a node t. Therefore, $I(G,H_{k,t})$ is not a facet of $\mathrm{CUT}^{\square}_{n+1}$.

Now we prove the "if" part. The proof is by induction on t.

First we consider the case $t = 0$. In this case, $H_{k,0}$ has no edges and G is the complete graph K_n. Switching the inequality $I(\mathrm{K}_n, \overline{\mathrm{K}}_n)$ by the cut $\{1\}$ gives a hypermetric inequality defined by an integer vector $\boldsymbol{b}$ with $b_{n+1} = -(k-3)$, $b_1 = -1$ and $b_2 = \cdots = b_n = 1$. This hypermetric inequality is a facet of $\mathrm{CUT}^{\square}_{n+1}$ by Theorem 2.

Now we consider the case $t \geq 1$. Note that contracting the edge $(t,t+k)$ in $H_{k,t}$ gives $H_{k,t-1}$. Key facts are that the inequality $I(G,H_{k,t})$ is obtained by lifting $I(G/(t,t+k), H_{k,t-1})$, and that $I(G/(t,t+k), H_{k,t-1})$ is a facet of $\mathrm{CUT}^{\square}_{n}$ by the induction hypothesis.

Let $V_i = V_{i+k} = \{i, i+k\}$ for $1 \leq i \leq t$ and $V_i = \{i\}$ for $t+1 \leq i \leq k$. We define n subsets of V as follows.

- Let p and p' be two distinct nodes adjacent to node $t+k$ in G. Then define $T^{(1)} = \{t\} \cup V_p \cup V_{p'}$.
- Let q and q' be two distinct nodes adjacent to node t in G. Then define $T^{(2)} = \{t+k\} \cup V_q \cup V_{q'}$.

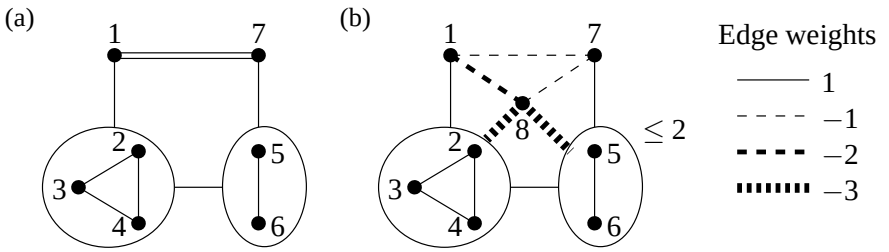


Figure 3: (a) A graph G (edges drawn as single lines) and $H_{6,1}$ (an edge drawn as a double line). (b) The inequality $I(G,H_{6,1})$, which is proved to be a facet of $\mathrm{CUT}_8^{\square}$ by Theorem 6. A line connected to a circle enclosing nodes 1, 2 and 3 represents 3 edges with identical weights each connected to the nodes 1, 2 and 3. Similar for lines connected to the other circles.

- For each $1 \le i \le k$ with $i \ne t$, we define $T_i^{(3)}$. If e_{it} has an endpoint $t+k$, then $T_i^{(3)} = \{t\} \cup V_i$. Otherwise, $T_i^{(3)} = \{t+k\} \cup V_i$.
- For each $1 \le i \le t-1$, we define a subset $T_i^{(4)}$. Let u be either i or $i+k$ that is an endpoint of the edge e_{it}, and $\bar{u}$ be either i or $i+k$ that is different from u. Let v be any node in $\mathrm{N}_G(u) \setminus V_t$ and choose j so that $V_j \ni v$. Let $\bar{w}$ be either t or $t+k$ that is not an endpoint of the edge e_{jt}. Then define $T_i^{(4)} = \{\bar{u}, \bar{w}\} \cup_j V$

It is easy to check that each of these subsets is a root of $I(G,H_{k,t})$ and contains exactly one of t and $t+k$. Note that none of them contains node $n+1$.

The following claim can be proved in a straightforward way. A proof is omitted due to space limination.

Claim 7 *The n incident vectors of $T^{(1)}$, $T^{(2)}$, $T_i^{(3)}$ $(i \ne t)$ and $T_i^{(4)}$ $(1 \le i \le t-1)$ are linearly independent.*

From now on, we refer to the n sets $T^{(1)}$, $T^{(2)}$, $T_i^{(3)}$ $(i \ne t)$ and $T_i^{(4)}$ $(1 \le i \le t-1)$ as $T_1, \ldots, T_n$.

Let $\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} \le 0$ be the switching of $I(G,H_{k,t})$ by its root $\{t,t+k\}$.

The $(t,t+k)$-collapse $(\boldsymbol{a}^{t,t+k})^{\mathrm{T}}\boldsymbol{x} \le 0$ of $\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} \le 0$ is the switching by the cut $\{t\}$ of $I(G/(t,t+k),H_{k,t-1})$, which is a facet of $\mathrm{CUT}_n^{\square}$ by induction hypothesis.

For $1 \le i \le n$, let T_i' be $T_i \triangle \{t,t+k\}$ if $t \in T_i$, and $(V \cup \{n+1\}) \setminus (T_i \triangle \{t,t+k\})$ otherwise, where $\triangle$ means the symmetric difference of two sets. Then T_i' is a root of the inequality $\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} \le 0$ and contains $t+k$ but does not contain t. In addition, the n vectors $T_1', \ldots, T_n'$ are also linearly independent. From Lemma 3, the inequality $\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} \le 0$ is a facet of $\mathrm{CUT}_{n+1}^{\square}$, which means $I(G,H_{k,t})$ is also a facet of $\mathrm{CUT}_{n+1}^{\square}$. $\square$

For example, let us consider the graphs $G=(V,E)$ and $H_{6,1}=(V,F)$ shown in Figure 3 (a). In this case the inequality $I(G,H_{6,1})$, illustrated in Figure 3 (b), is a facet of $\mathrm{CUT}_8^{\square}$ by Theorem 6.

4 Inequality $I'(G,H,C)$: A generalization of Gr_8

Let $G=(V,E)$, $H=(V,F)$, $n=|V|$, $t=|F|$, $k=n-t$ and $V=V_1 \cup \cdots \cup V_k$ be as defined in Section 3. In this section we require an additional condition that G has a cycle C of length four (this condition implies $k \ge 4$). Let V_C be the set of the four nodes of C. Then we consider an inequality for the cut polytope on $n+2$ nodes:

$$\sum_{uv \in E} T(u,v;n+1) - \sum_{uv \in F} T(u,v;n+1) + 2\sum_{V_i=\{u\}} x_{u,n+1} + \sum_{u \in V_C}(x_{u,n+1} - x_{u,n+2}) \le 2. \tag{3}$$

We refer to inequality (3) by $I'(G,H,C)$. Note that the $(n+1,n+2)$-collapsing of $I'(G,H,C)$ is identical to $I(G,H)$.

As an example, we show that the Gr_8 inequality is a switching of an inequality of this kind. Consider again the graphs $G_6=(V,E)$ and $H_{5,1}=(V,F)$ shown in Figure 2 (a). Note that G_6 contains a cycle $C=\{23,34,45,52\}$ of length four. Then the inequality $I'(G,H,C)$ is as shown in Figure 4 (a), and switching it by the cut $\{1,6\}$ and relabelling nodes appropriately gives the Gr_8 inequality.

Proposition 8 *The inequality $I'(G,H,C)$ is valid for $\mathrm{CUT}_{n+2}^{\square}$.*

PROOF: Let M be a set of two node-disjoint edges in the cycle C. Note that there are two choices of M. No matter which set we choose as M, the inequality $I'(G,H,C)$ can be written as

$$\sum_{uv \in E \setminus M} T(u,v;n+1) + \sum_{uv \in M} T(u,v;n+2) - \sum_{uv \in F} T(u,v;n+1) + 2\sum_{V_i=\{u\}} x_{u,n+1} \le 2. \tag{4}$$

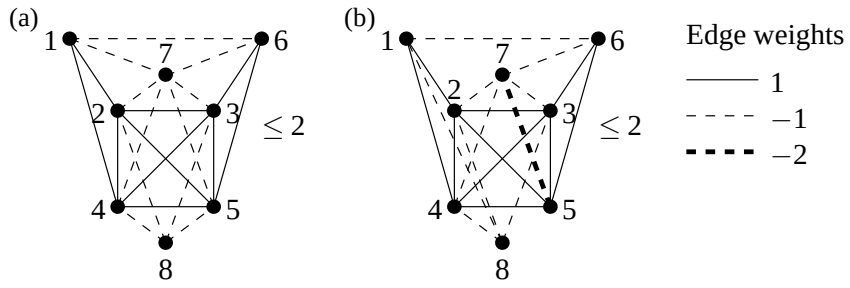


Figure 4: Two inequalities $I'(G_6, H_{5,1}, C)$ with different C. Both are proved to be facets of $\mathrm{CUT}_8^{\square}$ by Theorem 9. (a) Case of $C = \{23, 34, 45, 52\}$. The inequality is a switching of the Gr_8 inequality. (b) Case of $C = \{12, 23, 34, 41\}$.

We show that the cut vector $\boldsymbol{\delta}(S)$ defined by any subset S of $V \cup \{n+2\}$ satisfies (4). Let $A = \{i \mid V_i \subseteq S\}$ and $B = \{ij \mid e_{ij} \subseteq S,\ e_{ij} \in E \setminus M\}$. Now $|B| \geq \binom{|A|}{2} - \lfloor |A|/2 \rfloor$, since for each $ij \in B$ there is an edge e_{ij} with both endpoints in A, except for up to $\lfloor |A|/2 \rfloor$ edges that may be part of M. The left hand side of (3) evaluated with $\boldsymbol{x} = \boldsymbol{\delta}(S)$ is at most $2|A| - 2|B|$. Combining inequalities we have $2|A| - 2|B| \leq 3|A| + 2\lfloor |A|/2 \rfloor - |A|^2 \leq 2$ except when $|A| = 2$. So (4) is valid for all these cases. Suppose $|A| = 2$.

Case 1: The two nodes in A do not form an edge in M.
In this case $|B| = 1$, the LHS of (4) is at most $2|A| - 2|B| = 2$ and the inequality is valid.

Case 2: The two nodes in A form an edge in M.
In this case we replace M by $C \setminus M$. This does not change the LHS of (4), and the inequality is valid by Case 1. □

Before we state a sufficient condition for $I'(G, H, C)$ to be a facet of $\mathrm{CUT}_{n+2}^{\square}$, we assume some conditions on H and C without loss of generality. We assume $H = H_{k,t}$, where $H_{k,t}$ is the same as that defined in the previous section, and we also assume that indices of the four nodes of C are at most k. We say that node i in C is *free* if $1 \leq i \leq t$ and $i + k$ is incident to edge e_{ij} where j is the unique node in C that is not adjacent to i in C. The following theorem gives a sufficient condition for $I'(G, H_{k,t}, C)$ to be a facet.

Theorem 9 *The inequality $I'(G, H_{k,t}, C)$ is a facet of $\mathrm{CUT}_{n+2}^{\square}$ if all of the following conditions are satisfied:*

(i) All nodes in G have at least two neighbors.

(ii) For each $t+1 \leq i \leq k$ except for nodes in C, there exists a free node j in C such that e_{ij} is incident to $j+k$.

(iii) For each $1 \leq i \leq t$ except for nodes in C, either:

- *Nodes i and $i+k$ are incident to exactly two out of four edges e_{ij} with $j \in V_C$, or*
- *There exists a free node j in C such that e_{ij} is incident to $j+k$.*

Since $I'(G, H, C)$ is a lifting of $I(G, H)$, we may prove Theorem 9 by combining the lifting lemma (Lemma 3) with Theorem 6. The proof is omitted due to space limitation. As an example of the theorem, consider the graphs G_6 and $H_{5,1}$ shown in Figure 2 (a), but this time let $C = \{12, 23, 34, 41\}$. In this case the inequality $I'(G, H, C)$, shown in Figure 4 (b), is a facet of $\mathrm{CUT}_8^{\square}$ by Theorem 9.

Unlike $I(\mathrm{K}_k, \overline{\mathrm{K}}_k)$, which is always a facet of $\mathrm{CUT}_{k+1}^{\square}$, the face of $\mathrm{CUT}_{k+2}^{\square}$ supported by the inequality $I'(\mathrm{K}_k, \overline{\mathrm{K}}_k, C)$ with $k \geq 5$ and $C = \{12, 23, 34, 41\}$ is contained in a triangle facet $x_{5,k+2} - x_{5,k+1} - x_{k+1,k+2} \leq 0$ and never supports a facet.

5 Tightness of the I_{mm22} Bell inequalities

In this section, we prove that for any m, the I_{mm22} inequality is a facet of $\mathrm{COR}^{\square}(\mathrm{K}_{m,m})$, or in other words, a tight Bell inequality. Since the proof does not depend on the proof of validity given in [4], our proof also serves as another way to prove the validity of the I_{mm22} inequality.

Let $\mathrm{K}_{1,m,m} = (V_{1,m,m}, E_{1,m,m})$ be a complete tripartite graph with node set $V_{1,m,m} = \{\mathrm{Z}, \mathrm{A}_1, \ldots, \mathrm{A}_m, \mathrm{B}_1, \ldots, \mathrm{B}_m\}$ and edge set $E_{1,m,m} = \{\mathrm{ZA}_i \mid 1 \leq i \leq m\} \cup \{\mathrm{ZB}_j \mid 1 \leq j \leq m\} \cup \{\mathrm{A}_i\mathrm{B}_j \mid 1 \leq i, j \leq m\}$. We rewrite the I_{mm22} inequality to an inequality for $\mathrm{CUT}^{\square}(\mathrm{K}_{1,m,m})$ by using the covariance mapping. We switch this inequality by the cut $\{\mathrm{A}_1, \ldots, \mathrm{A}_m\}$. After that, we change the labels of the m nodes $\mathrm{B}_1, \mathrm{B}_2, \ldots, \mathrm{B}_m$ to $\mathrm{B}_{m+1}, \mathrm{B}_m, \ldots, \mathrm{B}_2$, respectively, both in the inequality and the graph $\mathrm{K}_{1,m,m}$. Let us denote the resulting complete tripartite graph by $\mathrm{K}'_{1,m,m}$. Then the inequality becomes

$$-(m-2)x_{\mathrm{ZA}_1} - \sum_{2 \leq i \leq m}(m-i)x_{\mathrm{ZA}_i} - (m-2)x_{\mathrm{ZB}_{m+1}} - \sum_{2 \leq j \leq m}(j-2)x_{\mathrm{ZB}_j} - \sum_{2 \leq i \leq m} x_{\mathrm{A}_i\mathrm{B}_i} + \sum_{1 \leq i < j \leq m+1} x_{\mathrm{A}_i\mathrm{B}_j} \leq 2. \tag{5}$$

It is easy to check that the inequality (5) is identical to the inequality $I(G,H)$ with $G=(V,E)$ and $H=(V,F)$, where $V=\{A_1,\ldots,A_m,B_2,\ldots,B_{m+1}\}$, $E=\{A_iB_j \mid 1\le i<j\le m+1\}$, and $F=\{A_iB_i \mid 2\le i\le m\}$. Therefore the I_{mm22} inequality is a tight Bell inequality if and only if the inequality $I(G,H)$ is a facet of $\mathrm{CUT}^{\square}(K'_{1,m,m})$.

Note that we cannot use Theorem 6 directly to prove that $I(G,H)$ is a facet, since the graph G does not satisfy the condition of Theorem 6. However, if we assume $m\ge 3$, the inequality $I(G,H)$ is the triangular elimination of another inequality $I(G',H')$, where G' (resp. H') is the graph obtained from G (resp. H) by identifying node B_2 to A_2 and A_m to B_m. The inequality $I(G',H')$ is proved to be a facet of $\mathrm{CUT}^{\square}_{2m-1}$ by Theorem 6. Now, as was pointed out in [2] and [10], we can apply triangular elimination twice to the facet inequality $I(G',H')$ to obtain $I(G,H)$. The first application is done with $uu'=A_1A_2$, $v=B_2$ and $W=\{Z,A_3,\ldots,A_{m-1},B_m,B_{m+1}\}$. The second application is done with $uu'=B_mB_{m+1}$, $v=A_m$ and $W=\{Z,B_2,\ldots,B_{m-1}\}$. Therefore, from Theorem 4, $I(G,H)$ is a facet of $\mathrm{CUT}^{\square}(K'_{1,m,m})$.

Since it is easy to check the cases $m=1$ and 2, we obtain the following theorem.

Theorem 10 *For any $m\ge 1$, the I_{mm22} inequality* (1) *is a tight Bell inequality.*

Acknowledgements

We thank Hiroshi Imai and Yuuya Sasaki for useful discussions.

References

[1] D. AVIS AND M. DEZA, The cut cone, L^1 embeddability, complexity and multicommodity flows, *Networks* (1991) **21**:595–617.

[2] D. AVIS, H. IMAI, T. ITO, AND Y. SASAKI, Deriving tight Bell inequalities for 2 parties with many 2-valued observables from facets of cut polytopes, arXiv:quant-ph/0404014, (2004).

[3] J. F. CLAUSER, M. A. HORNE, A. SHIMONY, AND R. A. HOLT, Proposed experiment to test local hidden-variable theories, *Physical Review Letters* (1969) **23**(15):880–884.

[4] D. COLLINS AND N. GISIN, A relevant two qubit Bell inequality inequivalent to the CHSH inequality, *Journal of Physics A: Mathematical and General* (2004) **37**(5):1775–1787, arXiv:quant-ph/0306129.

[5] C. DE SIMONE, M. DEZA, AND M. LAURENT, Collapsing and lifting for the cut cone, *Discrete Mathematics* (1994) **127**(1–3):105–130.

[6] M. DEZA AND M. LAURENT, Applications of cut polyhedra I, *Journal of Computational and Applied Mathematics* (1994) **55**(2):191–216.

[7] M. DEZA AND M. LAURENT, Applications of cut polyhedra II, *Journal of Computational and Applied Mathematics* (1994) **55**(2):217–247.

[8] M. M. DEZA AND M. LAURENT, *Geometry of Cuts and Metrics*, volume 15 of *Algorithms and Combinatorics*, Springer, (1997).

[9] V. P. GRISHUKHIN, All facets of the cut cone C_n for $n=7$ are known, *European Journal of Combinatorics* (1990) **11**:115–117.

[10] T. ITO, Y. SASAKI, H. IMAI, AND D. AVIS, Families of tight Bell inequalities derived from classes of facets of cut polytopes, In *Proceedings of ERATO conference on Quantum Information Science (EQIS'04)*, pages 78–79, (2004).

[11] R. F. WERNER AND M. M. WOLF, Bell inequalities and entanglement, *Quantum Information & Computation* (2001) **1**(3):1–25, arXiv:quant-ph/0107093.

Linking Systems and Matroid Pencils

SATORU IWATA*

Department of Mathematical Informatics
University of Tokyo
Hongo 7-3-1, Tokyo 113-8656, Japan
iwata@mist.i.u-tokyo.ac.jp

Abstract: A matroid pencil is a pair of linking systems having the same ground sets in common. It provides a combinatorial abstraction of matrix pencils. This paper investigates the properties of matroid pencils analogous to the theory of Kronecker canonical form. As an application, we give a simple alternative proof for a theorem of Murota on power products of linking systems.

Keywords: matroid, linking system, matrix pencil, Kronecker canonical form

1 Introduction

Linking systems (or bimatroids) were introduced by Kung [2] and Schrijver [6] as a combinatorial abstraction of matrices. They naturally provide combinatorial counterparts of linear algebraic notions such as multiplications of matrices. In this paper, we introduce matroid pencils as a combinatorial abstraction of matrix pencils.

A matrix pencil is a pair of matrices of the same size. It is often treated as a polynomial matrix whose nonzero entries are of degree at most one. Based on the theory of elementary divisors, Weierstrass established a criterion for strict equivalence, as well as a canonical form, of regular matrix pencils. Somewhat later, Kronecker investigated singular pencils to obtain a canonical form for matrix pencils in general under strict equivalence transformations, which is now called the Kronecker canonical form. The Kronecker canonical form of matrix pencils plays fundamental roles in application areas such as differential algebraic equations and control theory.

The Kronecker canonical form is characterized by the structural indices determined by the ranks of expanded matrices (Theorems 1 and 2). For matroid pencils, we define associated linking systems corresponding to the expanded matrices. Then we show that the ranks of these linking systems have the same properties as the expanded matrices (Lemmas 5–15), which enables us to define "structural indices" of matroid pencils. In particular, we will reveal that the ranks of a certain type of the associated linking systems are determined by some periodic structure (Theorem 17). This result in turn brings about an alternative proof of a theorem of Murota [3] on power products of linking systems.

The outline of this paper is as follows. Section 2 provides a brief description of the Kronecker canonical form of matrix pencils. Section 3 is devoted to a preliminary on linking systems. In Section 4, we introduce matroid pencils and describe their properties. Section 5 investigates the periodic structure. Finally, in Section 6, we present an alternative proof for the theorem on power products of linking systems.

2 The Kronecker Canonical Form of Matrix Pencils

Let $D(s) = sA + B$ be an $m \times n$ matrix pencil of rank r. A matrix pencil $\bar{D}(s)$ is said to be strictly equivalent to $D(s)$ if there exists a pair of nonsingular constant matrices U and V such that $\bar{D}(s) = UD(s)V$. A matrix pencil $D(s) = sA + B$ is said to be regular if $\det D(s) \neq 0$ as a polynomial in s. It is strictly regular if both A and B are nonsingular matrices.

For a positive integer μ, we consider $\mu \times \mu$ matrix pencils N_μ and K_μ defined by

$$N_\mu = \begin{pmatrix} 1 & s & 0 & \cdots & 0 \\ 0 & 1 & s & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & 1 & s \\ 0 & \cdots & \cdots & 0 & 1 \end{pmatrix}, \qquad K_\mu = \begin{pmatrix} s & 1 & 0 & \cdots & 0 \\ 0 & s & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & s & 1 \\ 0 & \cdots & \cdots & 0 & s \end{pmatrix}.$$

*Research is supported in part by a Grant-in-Aid for Scientific Research of the Ministry of Education, Science, Sports, and Culture of Japan.

For a positive integer ε, we further denote by L_ε an $\varepsilon \times (\varepsilon+1)$ matrix pencil

$$L_\varepsilon = \begin{pmatrix} s & 1 & 0 & \cdots & 0 \\ 0 & s & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & s & 1 \end{pmatrix}.$$

We also denote by $L_\eta^\top$ the transpose matrix of L_η.

The following theorem establishes the Kronecker canonical form of matrix pencils under strict equivalence transformations. See [4, Section 5.1.3] for its proof.

Theorem 1 (Kronecker, Weierstrass) *For any matrix pencil $D(s)$, there exists a pair of nonsingular constant matrices U and V such that $\bar{D}(s) = UD(s)V$ is in a block-diagonal form*

$$\bar{D}(s) = \text{block-diag}(H_\nu, K_{\rho_1}, \cdots, K_{\rho_c}, N_{\mu_1}, \cdots, N_{\mu_d}, L_{\varepsilon_1}, \cdots, L_{\varepsilon_p}, L_{\eta_1}^\top, \cdots, L_{\eta_q}^\top, O),$$

where $\rho_1 \geq \cdots \geq \rho_c > 0$, $\mu_1 \geq \cdots \geq \mu_d > 0$, $\varepsilon_1 \geq \cdots \geq \varepsilon_p > 0$, $\eta_1 \geq \cdots \geq \eta_q > 0$, and H_ν is a strictly regular matrix pencil of size ν. The numbers c, d, p, q, ν, $\rho_1, \cdots, \rho_c$, $\mu_1, \ldots, \mu_d$, $\varepsilon_1, \cdots, \varepsilon_p$, $\eta_1, \cdots, \eta_q$ are uniquely determined.

The block-diagonal matrix pencil $\bar{D}(s)$ in Theorem 1 is often referred to as the Kronecker canonical form of $D(s)$. The numbers $\mu_1, \ldots, \mu_d$ are called the indices of nilpotency. The numbers $\varepsilon_1, \cdots, \varepsilon_p$ and $\eta_1, \cdots, \eta_q$ are the minimal column and row indices, respectively. These numbers together with $\nu, \rho_1, \cdots, \rho_c$ are collectively called the structural indices of $D(s)$.

For an $m \times n$ matrix pencil $D(s) = sA + B$, we construct a $(k+1)m \times kn$ matrix $\Psi_k(D)$ and a $km \times (k+1)n$ matrix $\Phi_k(D)$ defined by

$$\Psi_k(D) = \begin{pmatrix} A & O & \cdots & O \\ B & A & \ddots & \vdots \\ O & B & \ddots & O \\ \vdots & \ddots & \ddots & A \\ O & \cdots & O & B \end{pmatrix}, \qquad \Phi_k(D) = \begin{pmatrix} B & A & O & \cdots & O \\ O & B & A & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & O \\ O & \cdots & O & B & A \end{pmatrix}.$$

We denote $\psi_k(D) = \operatorname{rank} \Psi_k(D)$ and $\varphi_k(D) = \operatorname{rank} \Phi_k(D)$. We also construct a pair of $km \times kn$ matrices $\Theta_k(D)$ and $\Omega_k(D)$ defined by

$$\Theta_k(D) = \begin{pmatrix} A & O & \cdots & \cdots & O \\ B & A & \ddots & & \vdots \\ O & B & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & A & O \\ O & \cdots & O & B & A \end{pmatrix}, \qquad \Omega_k(D) = \begin{pmatrix} B & A & O & \cdots & O \\ O & B & A & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & O \\ \vdots & & \ddots & B & A \\ O & \cdots & \cdots & O & B \end{pmatrix}.$$

We denote $\theta_k(D) = \operatorname{rank} \Theta_k(D)$ and $\omega_k(D) = \operatorname{rank} \Omega_k(D)$. Then it is easy to see that the ranks of these expanded matrices are expressed by the structural indices as follows.

Theorem 2 *Let $(\nu, \rho_1, \cdots, \rho_c, \mu_1, \ldots, \mu_d, \varepsilon_1, \ldots, \varepsilon_p, \eta_1, \ldots, \eta_q)$ be structural indices of a matrix pencil $D(s)$. Then we have*

$$\psi_k(D) = rk + \sum_{i=1}^{p} \min\{k, \varepsilon_i\}, \qquad \varphi_k(D) = rk + \sum_{i=1}^{q} \min\{k, \eta_i\},$$

$$\theta_k(D) = rk - \sum_{i=1}^{d} \min\{k, \mu_i\}, \qquad \omega_k(D) = rk - \sum_{i=1}^{c} \min\{k, \rho_i\},$$

where r is the rank of $D(s)$.

3 Linking Systems

Let S and T be a pair of finite sets. Let Λ be a nonempty collection of pairs of subsets of S and T. Then the triple (S, T, Λ) is a linking system if it satisfies the following axioms.

(L1) If $(X, Y) \in \Lambda$, then $|X| = |Y|$.

(L2) If $(X,Y) \in \Lambda$ and $x \in X$, then there exists $y \in Y$ such that $(X \setminus \{x\}, Y \setminus \{y\}) \in \Lambda$.

(L3) If $(X,Y) \in \Lambda$ and $y \in Y$, then there exists $x \in X$ such that $(X \setminus \{x\}, Y \setminus \{y\}) \in \Lambda$.

(L4) If $(X,Y) \in \Lambda$ and $(X',Y') \in \Lambda$, then there exists $(X^\circ, Y^\circ) \in \Lambda$ such that $X \subseteq X^\circ \subseteq X \cup Y$ and $Y' \subseteq Y^\circ \subseteq X' \cup Y'$.

A member of Λ is called a linked pair. The sets S and T are respectively called the row set and the column set of Λ.

The rank function $\lambda : 2^S \times 2^T \to \mathbf{Z}$ of $\mathbf{L} = (S,T,\Lambda)$ defined by

$$\lambda(X,Y) = \max\{|W| \mid (W,Z) \in \Lambda, W \subseteq X, Z \subseteq Y\} \qquad (X \subseteq S, Y \subseteq T)$$

satisfies the following properties.

(R1) $0 \leq \lambda(X,Y) \leq \min\{|X|,|Y|\}$ for any $X \subseteq S$ and $Y \subseteq T$.

(R2) $\lambda(X,Y) \leq \lambda(X',Y')$ for any $X \subseteq X' \subseteq S$ and $Y \subseteq Y' \subseteq T$.

(R3) $\lambda(X,Y) + \lambda(X',Y') \geq \lambda(X \cup X', Y \cap Y') + \lambda(X \cap X', Y \cup Y')$ for any $X, X' \subseteq S$ and $Y, Y' \subseteq T$.

In particular, (R3) is referred to as linking bisubmodularity. The rank of $\mathbf{L}$, denoted by $r(\mathbf{L})$, is the maximum size $|X|$ of a linked pair $(X,Y) \in \Lambda$, i.e., $r(\mathbf{L}) = \lambda(S,T)$.

Alternatively, we may define linking systems in terms of rank functions satisfying the above (R1)–(R3). Then the family Λ of linked pairs is determined by

$$\Lambda = \{(X,Y) \mid \lambda(X,Y) = |X| = |Y|, X \subseteq S, Y \subseteq T\}.$$

A principal example of linking systems comes from matrices. Let A be a matrix with row set S and column set T. For a pair of $X \subseteq S$ and $Y \subseteq T$, we denote by $A[X,Y]$ the submatrix of A indexed by X and Y. Then $\mathbf{L}(A) = (S,T,\Lambda(A))$ is a linking system, where

$$\Lambda(A) = \{(X,Y) \mid \mathrm{rank} A[X,Y] = |X| = |Y|, X \subseteq S, Y \subseteq T\}.$$

The rank function λ of $\mathbf{L}(A)$ is given by

$$\lambda(X,Y) = \mathrm{rank} A[X,Y].$$

The function λ defined by $\lambda(X,Y) = \mathrm{rank} A[X,Y]$ satisfies (R1)–(R3). Thus
is a linking system.

For a pair of linking systems $\mathbf{L} = (S,T,\Lambda)$ and $\mathbf{L}' = (S',T',\Lambda')$, the union $\mathbf{L} \vee \mathbf{L}' = (S \cup S', T \cup T', \Lambda \vee \Lambda')$ defined by

$$\Lambda \vee \Lambda' = \{(X \cup X', Y \cup Y') \mid X \cap X' = \emptyset, Y \cap Y' = \emptyset, (X,Y) \in \Lambda, (X',Y') \in \Lambda'\}$$

is a linking system. Note that $S \cap S'$ and $T \cap T'$ can be nonempty.

Lemma 3 *Let λ and λ' be the rank functions of $\mathbf{L} = (S,T,\Lambda)$ and $\mathbf{L}' = (S',T',\Lambda')$. Then the rank function $\lambda \vee \lambda'$ of $\mathbf{L} \vee \mathbf{L}'$ is given by*

$$(\lambda \vee \lambda')(X,Y) = \min_{W \subseteq X, Z \subseteq Y} \{\lambda(W \cap S, Z \cap T) + \lambda'(W \cap S', Z \cap T') + |X \setminus W| + |Y \setminus Z|\}.$$

The union of linking systems is analogous to the addition of matrices. Similarly, multiplication of linking systems is defined as follows. For a pair of linking systems $\mathbf{A} = (R,S,\Lambda)$ and $\mathbf{B} = (S,T,\Xi)$, the multiplication is defined by $\mathbf{A} * \mathbf{B} = (R,T,\Lambda * \Xi)$ with

$$\Lambda * \Xi = \{(W,Y) \mid \exists X \subseteq S, (W,X) \in \Lambda, (X,Y) \in \Xi\}.$$

Let $\mathbf{I} = (S,S,\Delta)$ denote the diagonal linking system with $\Delta = \{(X,X) \mid X \subseteq S\}$. Then we have the following lemma.

Lemma 4 *The rank of $\mathbf{A} * \mathbf{B}$ satisfies*

$$r(\mathbf{A} * \mathbf{B}) = r(\mathbf{A} \vee \mathbf{I} \vee \mathbf{B}) - |S|.$$

4 Matroid Pencils

A matroid pencil is a pair of linking systems having the row/column sets in common. Consider a matroid pencil $(\mathbf{A},\mathbf{B})$ with $\mathbf{A}=(S,T,\Lambda)$ and $\mathbf{B}=(S,T,\Xi)$. The rank of $(\mathbf{A},\mathbf{B})$ is defined by the rank of $\mathbf{A}\vee\mathbf{B}$, which we denote by r throughout this section.

We now introduce combinatorial counterparts of expanded matrices. For a positive integer j, let S_j and T_j be distinct copies of S and T, respectively. Furthermore, let $\mathbf{A}_j=(S_j,T_j,\Lambda_j)$ and $\mathbf{B}_j=(S_{j+1},T_j,\Xi_j)$ be the copies of $\mathbf{A}$ and $\mathbf{B}$, respectively.

For each positive integer k, consider the unions:

$$\begin{aligned}
\Psi_k(\mathbf{A},\mathbf{B}) &= \mathbf{A}_1\vee\mathbf{B}_1\vee\mathbf{A}_2\vee\cdots\vee\mathbf{A}_k\vee\mathbf{B}_k,\\
\Phi_k(\mathbf{A},\mathbf{B}) &= \mathbf{B}_1\vee\mathbf{A}_2\vee\mathbf{B}_2\vee\cdots\vee\mathbf{B}_k\vee\mathbf{A}_{k+1},\\
\Theta_k(\mathbf{A},\mathbf{B}) &= \mathbf{A}_1\vee\mathbf{B}_1\vee\mathbf{A}_2\vee\cdots\vee\mathbf{B}_{k-1}\vee\mathbf{A}_k,\\
\Omega_k(\mathbf{A},\mathbf{B}) &= \mathbf{B}_1\vee\mathbf{A}_2\vee\mathbf{B}_2\vee\cdots\vee\mathbf{A}_k\vee\mathbf{B}_k.
\end{aligned}$$

We denote the ranks of by $\Psi_k(\mathbf{A},\mathbf{B})$, $\Phi_k(\mathbf{A},\mathbf{B})$, $\Theta_k(\mathbf{A},\mathbf{B})$, and $\Omega_k(\mathbf{A},\mathbf{B})$ by ψ_k, φ_k, θ_k, and ω_k, respectively. Note that φ_k is equal to the rank of $\Psi_k(\mathbf{B},\mathbf{A})$ and ω_k is the rank of $\Theta_k(\mathbf{B},\mathbf{A})$. For $k=0$, we set $\psi_0=\varphi_0=\theta_0=\omega_0=0$. Obviously, these four sequences are monotone nondecreasing in k. The following lemmas show that ψ_k and φ_k are concave in k while θ_k and ω_k are convex in k.

Lemma 5 *For any $k>0$, we have $2\psi_k\geq\psi_{k-1}+\psi_{k+1}$ and $2\varphi_k\geq\varphi_{k-1}+\varphi_{k+1}$.*

PROOF: Let σ be the rank function of $\Psi_{k+1}(\mathbf{A},\mathbf{B})$. Let S^* and T^* denote the row and column sets of $\Psi_{k+1}(\mathbf{A},\mathbf{B})$, respectively. For $Z=T_2\cup\cdots\cup T_k$, we have

$$\sigma(S^*,T_1\cup Z)+\sigma(S^*,Z\cup T_{k+1})\geq\sigma(S^*,T^*)+\sigma(S^*,Z)$$

by the linking bisubmodularity of σ. Note that $\psi_k=\sigma(S^*,T_1\cup Z)=\sigma(S^*,Z\cup T_{k+1})$, $\psi_{k-1}=\sigma(S^*,Z)$ and $\psi_{k+1}=\sigma(S^*,T^*)$. Thus we obtain $2\psi_k\geq\psi_{k-1}+\psi_{k+1}$. By interchanging the roles of $\mathbf{A}$ and $\mathbf{B}$, we also obtain $2\varphi_k\geq\varphi_{k-1}+\varphi_{k+1}$. □

Lemma 6 *For any $k>0$, we have $2\theta_k\leq\theta_{k-1}+\theta_{k+1}$ and $2\omega_k\leq\omega_{k-1}+\omega_{k+1}$.*

PROOF: Let σ denote the rank function of Θ_{k+1}. Let S^* and T^* denote the row and column sets of Θ_{k+1}. For $X=S_1\cup\cdots\cup S_k$ and $Y=T_2\cup\cdots\cup T_{k+1}$, we have

$$\sigma(X,Y)+\sigma(S^*,T^*)\geq\sigma(X,T^*)+\sigma(S^*,Y)$$

by the linking bisubmodularity of σ. Note that $\theta_{k-1}=\sigma(X,Y)$, $\theta_{k+1}=\sigma(S^*,T^*)$ and $\theta_k=\sigma(X,T^*)=\sigma(S^*,Y)$. Thus we obtain $2\theta_k\leq\theta_{k-1}+\theta_{k+1}$. By interchanging the roles of $\mathbf{A}$ and $\mathbf{B}$, we obtain $2\omega_k\leq\omega_{k-1}+\omega_{k+1}$. □

Let λ and ξ be the rank functions of $\mathbf{A}=(S,T,\Lambda)$ and $\mathbf{B}=(S,T,\Xi)$, respectively. Then the rank r of $(\mathbf{A},\mathbf{B})$ is given by

$$r=\min_{W\subseteq S,Z\subseteq T}\{\lambda(W,Z)+\xi(W,Z)+|S\setminus W|+|T\setminus Z|\}.$$

A pair (W,Z) that attains the minimum in the right hand side is called a minimum cover of $\mathbf{A}\vee\mathbf{B}$. A pair of $(X,Y)\in\Lambda$ and $(X',Y')\in\Xi$ is called a maximum linking if it satisfies $X\cap X'=\emptyset$, $Y\cap Y'=\emptyset$ and $|X|+|X'|=r$.

Lemma 7 *Let (W,Z) be a minimum cover of $\mathbf{A}\vee\mathbf{B}$. Then we have $\psi_k\leq rk+|S\setminus W|$ and $\varphi_k\leq rk+|T\setminus Z|$.*

PROOF: Let S^* and T^* denote the row and column sets of $\Psi_k(\mathbf{A},\mathbf{B})$. That is, $S^*=S_1\cup\cdots\cup S_{k+1}$ and $T^*=T_1\cup\cdots\cup T_k$. Let $W_j\subseteq S_j$ be the copies of W for $j=1,\ldots,k+1$ and $Z_j\subseteq T_j$ the copies of Z for $j=1,\ldots,k$. Put $W^*=W_1\cup\cdots\cup W_{k+1}$ and $Z^*=Z_1\cup\cdots\cup Z_k$. Then we have

$$\begin{aligned}
\psi_k &\leq (\lambda_1\vee\cdots\vee\lambda_k)(W^*,Z^*)+(\xi_1\vee\cdots\vee\xi_k)(W^*,Z^*)+|S^*\setminus W^*|+|T^*\setminus Z^*|\\
&= k\lambda(W,Z)+k\xi(W,Z)+(k+1)|S\setminus W|+k|T\setminus Z|=rk+|S\setminus W|.
\end{aligned}$$

By interchanging the roles of $\mathbf{A}$ and $\mathbf{B}$, we obtain $\varphi_k\leq rk+|T\setminus Z|$. □

Lemma 8 *Let (W,Z) be a minimum cover of $\mathbf{A}\vee\mathbf{B}$. Then we have $\theta_k\leq rk-\xi(W,Z)$ and $\omega_k\leq rk-\lambda(W,Z)$ for any k.*

PROOF: Let S^* and T^* denote the row and column sets of $\Theta_k(\mathbf{A},\mathbf{B})$. That is, $S^* = S_1 \cup \cdots \cup S_k$ and $T^* = T_1 \cup \cdots \cup T_k$. Let $W_j \subseteq S_j$ and $Z_j \subseteq T_j$ be the copies of W and Z, respectively. Then $W^* = W_1 \cup \cdots \cup W_k$ and $Z^* = Z_1 \cup \cdots \cup Z_k$ satisfy

$$\begin{aligned} \theta_k &\leq (\lambda_1 \vee \cdots \vee \lambda_k)(W^*,Z^*) + (\xi_1 \vee \cdots \vee \xi_{k-1})(W^*,Z^*) + |S^* \setminus W^*| + |T^* \setminus Z^*| \\ &= k\lambda(W,Z) + (k-1)\xi(W,Z) + k|S \setminus W| + k|T \setminus Z| = rk - \xi(W,Z). \end{aligned}$$

By interchanging the roles of $\mathbf{A}$ and $\mathbf{B}$, we obtain $\omega_k \leq rk - \lambda(W,Z)$. □

Lemma 9 *Let $(X,Y) \in \Lambda$ and $(X',Y') \in \Xi$ be a maximum linking. Then we have $\psi_k \geq rk$ and $\varphi_k \geq rk$ for any k.*

PROOF: Let $X_j, X'_j \subseteq S_j$ be the copies of $X, X' \subseteq S$ for $j = 1,\ldots,k+1$ and $Y_j, Y'_j \subseteq T_j$ the copies of $Y, Y' \subseteq T$ for $j = 1,\ldots,k$. Put $X^* = X_1 \cup \cdots \cup X_k \cup X'_2 \cup \cdots \cup X'_{k+1}$ and $Y^* = Y_1 \cup \cdots \cup Y_k \cup Y'_1 \cup \cdots \cup Y'_k$. Then (X^*,Y^*) is a linked pair in $\Psi_k(\mathbf{A},\mathbf{B})$. Hence we have $\psi_k \geq k|X| + k|X'| = rk$. By interchanging the roles of $\mathbf{A}$ and $\mathbf{B}$, we obtain $\varphi_k \geq rk$. □

Lemma 10 *Let $(X,Y) \in \Lambda$ and $(X',Y') \in \Xi$ be a maximum linking. Then we have $\theta_k \geq rk - |X'|$ and $\omega_k \geq rk - |X|$.*

PROOF: Let $X_j, X'_j \subseteq S_j$ be the copies of $X, X' \subseteq S$ and $Y_j, Y'_j \subseteq T_j$ the copies of $Y, Y' \subseteq T$ for $j = 1,\ldots,k$. Put $X^* = X_1 \cup \cdots \cup X_k \cup X'_2 \cup \cdots \cup X'_k$ and $Y^* = Y_1 \cup \cdots \cup Y_k \cup Y'_1 \cup \cdots \cup Y'_{k-1}$. Then (X^*,Y^*) is a linked pair in $\Theta_k(\mathbf{A},\mathbf{B})$. Hence we have $\theta_k \geq k|X| + (k-1)|X'| = rk - |X'|$. By interchanging the roles of $\mathbf{A}$ and $\mathbf{B}$, we obtain $\omega_k \geq rk - |X|$. □

Lemma 11 *For any k, we have $\theta_{k+1} - \theta_k \leq r$ and $\omega_{k+1} - \omega_k \leq r$.*

PROOF: This is immediate from Lemmas 6 and 8. □

Lemma 12 *For any k, we have $\psi_{k+1} - \psi_k \geq r$ and $\varphi_{k+1} - \varphi_k \geq r$.*

PROOF: This is immediate from Lemmas 5 and 9. □

Lemma 13 *If $k \geq r$, we have $\psi_{k+1} - \psi_k = \varphi_{k+1} - \varphi_k = r$.*

PROOF: Since ψ_k is concave in k by Lemma 5, it follows from Lemmas 7 and 12 that there exists an integer h such that $\psi_{k+1} - \psi_k = r$ holds for any $k \geq h$. Let ℓ be the smallest such h. Then by Lemma 5, we have $\psi_\ell \geq (r+1)\ell$. On the other hand, a minimum cover (W,Z) of $\mathbf{A} \vee \mathbf{B}$ satisfies $\psi_\ell \leq r\ell + |S \setminus W|$ by Lemma 7. Therefore, we have $\ell \leq |S \setminus W| \leq r$. Thus we obtain $\psi_{k+1} - \psi_k = r$ for $k \geq r$. Similarly, we also obtain $\varphi_{k+1} - \varphi_k = r$ for $k \geq r$. □

Lemma 14 *If $k \geq r$, we have $\theta_{k+1} - \theta_k = \omega_{k+1} - \omega_k = r$.*

PROOF: Since θ_k is convex in k by Lemma 6, it follows from Lemmas 10 and 11 that there exists an integer h such that $\theta_{k+1} - \theta_k = r$ holds for any $k \geq h$. Let ℓ be the smallest such h. Then by Lemma 6, we have $\theta_\ell \leq (r-1)\ell$. On the other hand, for a maximum linking $(X,Y) \in \Lambda$ and $(X',Y') \in \Xi$, we have $\theta_\ell \geq r\ell - |X'|$ by Lemma 10. Therefore, we have $\ell \leq |X| \leq r$. Thus we obtain $\theta_{k+1} - \theta_k = r$ for $k \geq r$. Similarly, we also obtain $\omega_{k+1} - \omega_k = r$ for $k \geq r$. □

Lemma 15 *For any k, we have $\psi_k + \varphi_k - \theta_k - \omega_k \leq r$.*

PROOF: Let S^* and T^* denote the row and column sets of $\Theta_{k+1}(\mathbf{A},\mathbf{B})$. That is, $S^* = S_1 \cup \cdots \cup S_{k+1}$ and $T^* = T_1 \cup \cdots \cup T_{k+1}$. We also denote $S^\circ = S_2 \cup \cdots \cup S_{k+1}$ and $T^\circ = T_1 \cup \cdots \cup T_k$. By the linking bisubmodularity of the rank function σ of $\Theta_{k+1}(\mathbf{A},\mathbf{B})$, we have

$$\sigma(S^*,T^*) + \sigma(S^\circ,T^\circ) \geq \sigma(S^*,T^\circ) + \sigma(S^\circ,T^*).$$

Since $\theta_{k+1} = \sigma(S^*,T^*)$, $\omega_k = \sigma(S^\circ,T^\circ)$, $\psi_k = \sigma(S^*,T^\circ)$ and $\varphi_k = \sigma(S^\circ,T^*)$, this can be rewritten as $\theta_{k+1} + \omega_k \geq \psi_k + \varphi_k$. Therefore, we have $\psi_k + \varphi_k - \theta_k - \omega_k \leq \theta_{k+1} - \theta_k \leq r$ by Lemma 11. □

Lemma 15 leads us to the definition of $\nu(\mathbf{A},\mathbf{B}) = r - \psi_r - \varphi_r + \theta_r + \omega_r \geq 0$, which is analogous to the size of the strictly regular block in the Kronecker canonical form. For a matrix pencil $D(s) = sA + B$, consider a matroid pencil $(\mathbf{L}(A),\mathbf{L}(B))$. It is not always true that $\nu(\mathbf{A},\mathbf{B})$ is equal to the size of the strictly regular block in the Kronecker canonical form $\bar{D}(s)$ of $D(s)$. A recent result in [1] implies that the equality holds if $D(s)$ is a generic matrix pencils, i.e., if the nonzero entries in A and B are independent parameters.

5 Periodic Linking

In this section, we investigate a periodic structure of $\Theta_k(\mathbf{A},\mathbf{B})$. Recall that a linked pair (X^*,Y^*) in $\Theta_k(\mathbf{A},\mathbf{B})$ consists of disjoint sums $X^* = X_1 \cup \cdots \cup X_k \cup X'_2 \cup \cdots \cup X'_k$ and $Y^* = Y_1 \cup \cdots \cup Y_k \cup Y'_1 \cup \cdots \cup Y'_{k-1}$ such that $(X_j,Y_j) \in \Lambda_j$ for $j = 1,\ldots,k$ and $(X'_{j+1},Y'_j) \in \Xi_j$ for $j = 1,\ldots,k-1$. Then a linked pair (X^*,Y^*) with such a decomposition is said to be a periodic linking if (X_j,Y_j) are the copies of the same $(X,Y) \in \Lambda$ for $j = 1,\ldots,k$ and (X'_{j+1},Y'_j) are the copies of the same $(X',Y') \in \Xi$ for $j = 1,\ldots,k-1$. This section is to show that a maximum size $|X^*| = |Y^*|$ of a periodic linking (X^*,Y^*) in Θ_k is equal to the rank θ_k.

Let $(X \cup X', Y \cup Y')$ be a linked pair of $\mathbf{A} \vee \mathbf{B}$ such that $(X,Y) \in \Lambda$ and $(X',Y') \in \Xi$. Then the periodic linking (X^*,Y^*) determined by (X,Y) and (X',Y') is of size $k|X| + (k-1)|X'|$.

Given a linked pair $(X,Y) \in \Lambda$, we construct an auxiliary directed graph $G_{\mathbf{A}}(X,Y) = (S \cup T, E)$ with vertex set $S \cup T$ and arc set $E = E_S \cup E_T \cup E_+ \cup E_-$ defined by

$$\begin{aligned}
E_S &= \{(u,v) \mid u \in S \setminus X, v \in X, (X \cup \{u\} \setminus \{v\}, Y) \in \Lambda\},\\
E_T &= \{(u,v) \mid u \in Y, v \in T \setminus Y, (X, Y \cup \{v\} \setminus \{u\}) \in \Lambda\},\\
E_+ &= \{(u,v) \mid u \in S \setminus X, v \in T \setminus Y, (X \cup \{u\}, Y \cup \{v\}) \in \Lambda\},\\
E_- &= \{(u,v) \mid u \in Y, v \in X, (X \setminus \{v\}, Y \setminus \{u\}) \in \Lambda\}.
\end{aligned}$$

Similarly, for a linked pair $(X',Y') \in \Xi$, we also construct an auxiliary directed graph $G_{\mathbf{B}}(X',Y') = (S \cup T, F)$. Furthermore, the auxiliary directed graph for a linked pair $(X \cup X', Y \cup Y')$ in $\mathbf{A} \vee \mathbf{B}$ is the superposition of $G_{\mathbf{A}}(X,Y)$ and $G_{\mathbf{B}}(X',Y')$. For simplicity we denote this graph by $G_{\mathbf{A}\vee\mathbf{B}} = (S \cup T, E \cup F)$.

The linked pair $(X \cup X', Y \cup Y')$ in $\mathbf{A} \vee \mathbf{B}$ determines a periodic linking (X^*,Y^*). Let S^* and T^* denote the row and column sets of $\Theta_k(\mathbf{A},\mathbf{B})$. That is, $S^* = S_1 \cup \cdots \cup S_k$ and $T^* = T_1 \cup \cdots \cup T_k$. For $j = 1,\ldots,k$, let E_j denote the edge set of $G_{\mathbf{A}_j}(X_j,Y_j)$. For $j = 1,\ldots,k-1$, let F_j denote the edge set of $G_{\mathbf{B}_j}(X_{j+1},Y_j)$. The auxiliary directed graph $G_{\Theta_k(\mathbf{A},\mathbf{B})} = (S^* \cup T^*, E^* \cup F^*)$ for (X^*,Y^*) is given by $E^* = E_1 \cup \cdots \cup E_k$ and $F^* = F_1 \cup \cdots \cup F_{k-1}$.

Lemma 16 *Suppose $(X,Y) \in \Lambda$ and $(X',Y') \in \Xi$ form a linking in $\mathbf{A} \vee \mathbf{B}$ that maximizes $k|X| + (k-1)|X'|$. Then the periodic linking (X^*,Y^*) that consists of the copies of (X,Y) and (X',Y') is a maximum linking in $\Theta_k(\mathbf{A},\mathbf{B})$.*

PROOF: If (X^*,Y^*) is not a maximum linking, there exists a directed path from $S^* \setminus X^*$ to $T^* \setminus Y^*$ in $G_{\Theta_k(\mathbf{A},\mathbf{B})}$. Let P^* be such a path with minimum number of arcs. The corresponding set of arcs in $G_{\mathbf{A},\mathbf{B}}$ forms a directed path from $S \setminus X$ to $T \setminus Y$. Note that P does not include a cycle. Let $S(P)$ and $T(P)$ denote the sets of vertices in S and T, respectively, along P. Then the symmetric differences $X_P = X \triangle S(P)$, $Y_P = Y \triangle T(P)$, $X'_P = X' \triangle S(P)$ and $Y'_P = Y' \triangle T(P)$ form new linked pairs $(X_P,Y_P) \in \Lambda$ and $(X'_P,Y'_P) \in \Xi$.

Let s and t be the initial and terminal vertices of P. Suppose that P^* starts from S_h and terminates in T_ℓ. Then we have $k|X_P| + (k-1)|X'_P| = k|X| + (k-1)|X'| + k + \ell - h$. If $s \notin X'$ and $t \notin Y'$, then $X_P \cap X'_P = \emptyset$ and $Y_P \cap Y'_P = \emptyset$, which implies that (X_P,Y_P) and (X'_P,Y'_P) form a linking in $\mathbf{A} \vee \mathbf{B}$. Since $k + \ell - h > 0$, this contradicts the choice of (X,Y) and (X',Y'). If $s \in X'$ and $t \notin Y'$, we have $X_P \cap X'_P = \{s\}$, $Y_P \cap Y'_P = \emptyset$, and $h = 1$. By (L2), there exists $v \in Y'_P$ such that $(X'_P \setminus \{s\}, Y'_P \setminus \{v\}) \in \Xi$. Thus (X_P,Y_P) and $(X'_P \setminus \{s\}, Y'_P \setminus \{v\})$ form a linking in $\mathbf{A} \vee \mathbf{B}$ with $k|X_P| + (k-1)|X'_P \setminus \{s\}| = k|X| + (k-1)|X'| + \ell$, which contradicts the choice of (X,Y) and (X',Y'). Similarly, if $s \notin X'$ and $t \in Y'$, we have $X_P \cap X'_P = \emptyset$, $Y_P \cap Y'_P = \{t\}$, and $\ell = k$. By (L3), there exists $u \in X'$ such that $(X'_P \setminus \{u\}, Y'_P \setminus \{t\}) \in \Xi$. Thus (X_P,Y_P) and $(X'_P \setminus \{u\}, Y'_P \setminus \{t\})$ form a linking in $\mathbf{A} \vee \mathbf{B}$ with $k|X_P| + (k-1)|X'_P \setminus \{u\}| = k|X| + (k-1)|X'| + k + 1 - h$, which contradicts the choice of (X,Y) and (X',Y'). Finally, if $s \notin X'$ and $t \in Y'$, we have $X_P \cap X'_P = \emptyset$, $Y_P \cap Y'_P = \{t\}$, $h = 1$ and $\ell = k$. It follows from (L2) and (L3) that $(X'_P \setminus \{s\}, Y'_P \setminus \{t\}) \in \Xi$ or there exist $u \in X'$ and $v \in Y'$ such that $(X'_P \setminus \{s,u\}, Y'_P \setminus \{t,v\}) \in \Xi$. In the former case, (X_P,Y_P) and $(X'_P \setminus \{s\}, Y'_P \setminus \{t\})$ form a linking in $\mathbf{A} \vee \mathbf{B}$ with $k|X_P| + (k-1)|X'_P \setminus \{s\}| = k|X| + (k-1)|X'| + k$. In the latter case, (X_P,Y_P) and $(X'_P \setminus \{s,u\}, Y'_P \setminus \{t,v\})$ form a linking in $\mathbf{A} \vee \mathbf{B}$ with $k|X_P| + (k-1)|X'_P \setminus \{s,u\}| = k|X| + (k-1)|X'| + 1$. In either case, we have contradiction to the choice of (X,Y) and (X',Y'). Thus we may conclude that (X^*,Y^*) is a maximum size linking in $\Theta_k(\mathbf{A},\mathbf{B})$. □

Theorem 17 *For a matroid pencil $(\mathbf{A},\mathbf{B})$, we have*

$$\theta_k(\mathbf{A},\mathbf{B}) = \max\{k|X| + (k-1)|X'| \mid (X,Y) \in \Lambda, (X',Y') \in \Xi, X \cap X' = \emptyset, Y \cap Y' = \emptyset\}.$$

6 Eigensets and Power Products

In this section, we give an alternative proof to a theorem of Murota [3] on maximum eigensets and the ranks of power products of linking systems.

Let $\mathbf{A} = (S,S,\Lambda)$ be a linking system whose row set and column set are identical. Murota [3] introduced the concept of eigenset of such a linking system and investigated its connection to power products. A subset $X \subseteq S$ is called an eigenset if

$(X,X) \in \Lambda$. Let $\mathbf{A}^k$ denote the k-th power product $\mathbf{A} * \cdots * \mathbf{A}$ of $\mathbf{A}$. Then $r(\mathbf{A}^k)$ is monotone nonincreasing and convex in k. Hence there exists $\ell \leq |S|$ such that $r(\mathbf{A}^k) = r(\mathbf{A}^{k+1})$ holds for $k \geq \ell$. We denote this rank by $r(\mathbf{A}^\infty)$. The following theorem characterizes $r(\mathbf{A}^\infty)$ in terms of eigensets.

Theorem 18 (Murota [3]) *For a linking system $\mathbf{A} = (S,S,\Lambda)$, we have*

$$r(\mathbf{A}^\infty) = \max\{|X| \mid (X,X) \in \Lambda\}.$$

We now present an alternative proof of this result using Theorem 17. Consider a matroid pencil $(\mathbf{A},\mathbf{I})$ with diagonal linking system $\mathbf{I} = (S,S,\Delta)$ and denote the rank of $\Theta_k(\mathbf{A},\mathbf{I})$ by $\theta_k(\mathbf{A},\mathbf{I})$. Then it follows from Lemma 4 that

$$r(\mathbf{A}^k) = \theta_k(\mathbf{A},\mathbf{I}) - (k-1)\,|S|.$$

Therefore, the following lemma completes the proof of Theorem 18.

Lemma 19 *For $k \geq |S|$, we have*

$$\theta_k(\mathbf{A},\mathbf{I}) = (k-1)\,|S| + \max\{|X| \mid (X,X) \in \Lambda\}.$$

PROOF: Applying Theorem 17 to $(\mathbf{A},\mathbf{I})$, we obtain

$$\theta_k(\mathbf{A},\mathbf{I}) = \max\{k\,|X| + (k-1)\,|Z| \mid (X,Y) \in \Lambda, X \cap Z = \emptyset, Y \cap Z = \emptyset\}.$$

Taking $(X,Y) = (\emptyset,\emptyset)$ and $Z = S$ in the right hand side, we observe $\theta_k \geq (k-1)\,|S|$. If $|X| + |Z| < |S|$, we have $k\,|X| + (k-1)\,|Z| \leq (k-1)\,|S| - (k-1) + |X| \leq (k-1)\,|S|$, where the last inequality follows from $|X| < |S| \leq k$. This implies that the maximum of the right hand side must be attained by some $X \subseteq S$ and $Z = S \setminus X$. Thus we obtain

$$\theta_k(\mathbf{A},\mathbf{I}) = \max\{k\,|X| + (k-1)\,|S \setminus X| \mid (X,X) \in \Lambda\},$$

which is obviously equivalent to the desired formula. □

For a square matrix A, consider a linking system $\mathbf{A} = \mathbf{L}(A)$. It should be noted that $\mathbf{A}^k$ can be different from $\mathbf{L}(A^k)$. A theorem of Poljak [5], however, shows that $\operatorname{rank} A^k = r(\mathbf{A}^k)$ holds if A is a generic matrix, i.e., if the nonzero entries of A are independent parameters. An alternative proof for this theorem is also described in [1].

References

[1] S. IWATA AND R. SHIMIZU, Combinatorial analysis of generic matrix pencils, *Proc. IPCO XI*, to appear.

[2] J. P. S. KUNG, Bimatroids and invariants, *Adv. Math.*, 30 (1978), pp. 238–249.

[3] K. MUROTA, Eigensets and power products of bimatroids, *Adv. Math.*, 80 (1990), pp. 78–91.

[4] K. MUROTA, *Matrices and Matroids for Systems Analysis*, Springer-Verlag, 2000.

[5] S. POLJAK, Maximum rank of powers of a matrix of a given pattern, *Proc. Amer. Math. Soc.*, 106 (1989), pp.1137–1144.

[6] A. SCHRIJVER, *Matroids and linking systems*, J. Combin. Theory, B26 (1979), pp. 349–369.

On Resource Constrained Optimization Problems

ALPÁR JÜTTNER*

Department of Operations Research and Egerváry
Research Group, Eötvös University, Pázmány Péter
Sétány 1/C, Budapest, Hungary, H-1117
alpar@cs.elte.hu

Abstract: This paper shows that a method that has long been used to solve Resource Constrained Optimization Problems and found extremely effective in practice, is effective in the theoretical sense as well, it is proved to be strongly polynomial. In the special case of Resource Constrained Shortest Path Problem a better running time estimation is also presented.

Keywords: Resource constrained Optimization, Lagrangian relaxation, strongly polynomial algorithms

1 Introduction

In order to define the *resource constrained optimization problem* in general, first let us consider an underlying set E, a *cost function* $c : E \longrightarrow \mathbb{R}_+$ and an abstract optimization problem

$$\min\{\sum_{e\in P} c(e) : P \in \mathscr{P}\}, \tag{1}$$

where $\mathscr{P} \subseteq 2^E$ denotes the set of the feasible solutions. We refer this problem *basic problem* in this paper.

The corresponding *constrained optimization problem* is the following. Let $d : E \longrightarrow \mathbb{R}_+$ be another given weighting called *delay*, and $\Delta \in \mathbb{R}_+$ a given constant called *delay constraint*. With these notations we are looking for the value

$$\min\{\sum_{e\in P} c(e) : P \in \mathscr{P}, \quad \sum_{e\in P} d(e) \leq \Delta\}. \tag{2}$$

An important example for this is the *Constrained Shortest Path Problem*. Assume that a network is given as a directed, connected graph $G = (V,E)$, where V represents the set of nodes, and E represents the set of directed links. Each link $e \in E$ is characterized by two nonnegative values, a cost $c(e)$ and a delay $d(e)$. With a given delay constraint $\Delta \in \mathbb{R}_+$ and two given nodes $s,t \in V$ the task is to find a least cost path P between s and t with the side constraint that the delay of the path is less then Δ.

One can define the *Constrained Minimum Cost Perfect Matching Problem* and the *Constrained Minimum Cost Spanning Tree Problem* in the same way.

Although their unconstrained versions are easy to solve, the three problems mentioned above are $\mathscr{NP}$-hard (see e.g. [3]). A usual way to find near optimal solutions to these problems is to get rid of the additional constrain using Lagrangian relaxation. In this way the constrained problem turns into a maximization of a one dimensional concave function (see Section 2).

A simple way to find the optimum of the relaxed problem is to use binary search, which is polynomial for integer costs and delays [28]

Instead of using binary search another a simple and practically even more effective method — described in Section 3 — has been found and applied independently by several authors. After [29] it is sometimes called *Handler-Zang method.* The same method was used in [11] making some people call it *BM method.* Other papers aim at further improving either on the running time in practical cases [25] or on the quality of the found solutions [26].

Although this method turned to be particularly efficient in practical applications, the worst case running time of this method was unknown for a long time.

Mehlhorn and Ziegelmann showed that for the Constrained Shortest Path problem the Handler-Zang algorithm is polynomial for integer costs and delays. If $c(e) \in [0,\cdots,C]$ and $d(e) \in [0,\cdots,R]$ for each $e \in E$, then it will terminate after $O\big(\log(|V|RC)\big)$ iterations. They also presented examples showing that this running time is tight for small costs and delays (i.e. if $R \leq |V|$ and $C \leq |V|$).

*Research is supported by OTKA T37547

One may observe that this problem can be transformed to an extension of the LCFO (Least Cost Fractional Optimization) problem discussed in [30]. Moreover, the strongly polynomial solution method proposed in [30] turns out to be equivalent with the Handler-Zang algorithm in this case, showing that the number of the iterations made by the Handler-Zang algorithm does not depend on the range of the cost and the delay functions, so this method is actually *strongly polynomial* if the basic problem can be solved in strongly polynomial time. This also shows that Mehlhorn's and Ziegelmann's bound on the running time is not tight for large costs or delays.

By a more careful application of the technique proposed in [19], in Section 4.1 we prove that the Handler-Zang algorithm takes $O\big(|E|^2\log(|E|)\big)$ iterations for arbitrary constrained optimization problem. This improves the bound can be obtained from [30] by a factor $O(\log(|E|))$.

For the Constrained Shortest Path problem an even better bound is shown in Section 4.2, the number of iterations is proved to be $O\big(|E|\log^2(|E|)\big)$ in this case.

2 Lagrange Relaxation

Lagrange Relaxation [16, 17] is a common technique for calculating lower bounds, and finding good solutions to hard optimization problem. This section shows how it can be applied to resource constrained optimization problems.

From now on we assume that there exists an algorithm $\mathscr{A}(c)$ which runs in time T and solves Problem (1) for any given cost function c.

Intuitively the presented Lagrangian relaxation method is based on the heuristic of minimizing $c_\lambda := c + \lambda \cdot d$ modified cost function over $\mathscr{P}$ for an appropriate λ. For a given (fixed) λ we can easily calculate the minimal solution $p_\lambda \in \mathscr{P}$ of the basic problem. If $\lambda = 0$ and $d(p_\lambda) \le \Delta$ we found an optimal solution for the constrained problem as well. If $d(p_\lambda) > \Delta$ we must increase λ in order to increase the dominance of delay in the modified cost function until the optimal solution with respect to c_λ suits the delay requirements.

Now, we show how the Lagrangian relaxation helps us to find the value of λ that gives the best result. Moreover, the algorithm will give an upper bound on the badness of the solution as a byproduct, based on the following well known Claim.

Claim 1 *Let*

$$L(\lambda) := \min\{c_\lambda(p) : p \in \mathscr{P}\} - \lambda\Delta. \tag{3}$$

Then $L(\lambda)$ is a lower bound *to problem (2) for any $\lambda \ge 0$.*

Proof. Let p^* denote an optimal solution of (2). Then

$$\begin{aligned} L(\lambda) &= \min\{c_\lambda(p) : p \in \mathscr{P}\} - \lambda\Delta \\ &\le c_\lambda(p^*) - \lambda\Delta \\ &= c(p^*) + \lambda(d(p^*) - \Delta) \le c(p^*) \end{aligned}$$

proves the claim. □

To obtain the best lower bound we need to maximize the function $L(\lambda)$, that is we are looking for the value

$$L^* := \max_{\lambda \ge 0} L(\lambda), \tag{4}$$

and the maximizing λ^*. Now, some more properties of the function $L(\lambda)$ are given.

Claim 2 *L is a concave piecewise linear function, namely the minimum of the linear functions $c(p) + \lambda(d(p) - \Delta)$ for all $p \in \mathscr{P}$.* □

Claim 3 *For any $\lambda \ge 0$ and c_λ-minimal solution $p_\lambda \in \mathscr{P}$, $d(p_\lambda) - \Delta$ is a subgradient of L in the point λ.*

Proof. We have to show that $L(x) \le L(\lambda) + (x - \lambda)(d(p_\lambda) - \Delta)$ holds for any x, which is proved by the following calculation. $L(\lambda) + (x - \lambda)(d(p_\lambda) - \Delta) = c(p_\lambda) + \lambda(d(p_\lambda) - \Delta) + (x - \lambda)(d(p_\lambda) - \Delta) = c(p_\lambda) + x(d(p_\lambda) - \Delta) \ge L(x)$. □

This gives us the following

Claim 4 *Whenever $\lambda < \lambda^*$, then $d(p_\lambda) \ge \Delta$ and if $\lambda > \lambda^*$, then $d(p_\lambda) \le \Delta$ for each c_λ-minimal solution p_λ.* □

Claim 5 *A value λ maximizes the function $L(\lambda)$* if and only if *there are solutions p_c and p_d which are both c_{λ^*}-minimal and for which $d(p_c) \ge \Delta$ and $d(p_d) \le \Delta$. (p_c and p_d can be the same, in this case $d(p_d) = d(p_c) = \Delta$.)* □

The Handler-Zang algorithm will give these solutions along with λ^*.

Claim 6 *Let $0 \le \lambda_1 < \lambda_2$, and $p_{\lambda_1}, p_{\lambda_2} \in P$ λ_1-minimal and λ_2-minimal solutions. Then $c(p_{\lambda_1}) \le c(p_{\lambda_2})$ and $d(p_{\lambda_1}) \ge d(p_{\lambda_2})$.*

The inequality for d immediately follows from Claim 2 and Claim 3. The following calculation proves the inequality for c.

$$\begin{aligned} c(p_{\lambda_1}) &= L(\lambda_1) - \lambda_1(d(p_{\lambda_1}) - \Delta) \\ &\leq L(\lambda_2) + (\lambda_1 - \lambda_2)(d(p_{\lambda_2}) - \Delta) - \lambda_1(d(p_{\lambda_1}) - \Delta) \\ &= c(p_{\lambda_2}) + \lambda_2(d(p_{\lambda_2}) - \Delta) + (\lambda_1 - \lambda_2)(d(p_{\lambda_2}) - \Delta) - \lambda_1(d(p_{\lambda_1}) - \Delta) \\ &= c(p_{\lambda_2}) + \lambda_1(d(p_{\lambda_2}) - d(p_{\lambda_1})) \leq c(p_{\lambda_2}) \end{aligned} \tag{5}$$

□

These two latter Claims together means that the λ^* that maximizes the function $L(\lambda)$ gives the best modified cost function, that is λ^* is the smallest value for which there exists a c_{λ^*}-minimal solution p_d which satisfies the delay constraint.

3 The Handler-Zang method

In this section the Handler-Zang algorithm is described (see Figure 1 for the pseudocode).

1. In the first step the algorithm sets $\lambda = 0$. It calculates an optimal solution with respect to the modified cost function c_λ. It means that the algorithm finds the c-minimal solution. If this solution meets the delay requirement Δ, it is an optimal solution of (2), and the algorithm stops.

2. Otherwise the algorithm stores the solution as the best solution that does not satisfy the delay requirement Δ (it is denoted by p_c in the following), and checks whether an appropriate solution exists or not: Calculates the d-minimal solution. If the obtained solution suits the delay requirement, a proper solution exists, so the algorithm stores it as the best feasible solution found till now (denoted by p_d). Otherwise there is no solution that fulfills the delay requirement, so the algorithm stops.

   In the further steps we obtain the optimal λ, through updating p_c and p_d repeatedly with new solutions.

3. Let's see the current solutions p_c and p_d. If for a certain λ, both p_c and p_d are c_λ-minimal, then using Claim 5, this λ maximizes $L(\lambda)$. But it can be true only if $c_\lambda(p_c) = c_\lambda(p_d)$, from which we get that the only possible λ is

$$\lambda := \frac{c(p_c) - c(p_d)}{d(p_d) - d(p_c)}. \tag{6}$$

   So, we set it as the new candidate for the optimal solution. Then we find a c_λ-minimal solution r. If $c_\lambda(r) = c_\lambda(p_c)$ then p_c and p_d are also c_λ-minimal, so we are done by setting $\lambda^* = \lambda$ and returning p_d. If $c_\lambda(r) < c_\lambda(p_c)$ then we replace either p_c or p_d with r according to whether it fails or fulfills the delay constraint, and repeat this step.

4 Running time of the algorithm

In the following we refer to the solutions and values computed by the algorithm according to Figure 1.

First of all, obviously

$$d(p_c^1) \geq d(p_c^2) \geq d(p_c^3) \geq \cdots > \Delta \tag{7}$$

and

$$d(p_d^1) \leq d(p_d^2) \leq d(p_d^3) \leq \cdots \leq \Delta, \tag{8}$$

and either $d(p_c^i) > d(p_c^{i+1})$ or $d(p_d^i) < d(p_d^{i+1})$ for any i. Since there are only finite number of different solutions, the algorithm finds the optimal λ in finite number of steps.

In this section the following stronger result will be proved.

Theorem 7 *The Handler-Zang algorithm terminates after $O(|E|^2 \log |E|)$ iterations, so the running time of the algorithm is $O(T|E|^2 \log |E|)$.* □

The presented proof below is based on the idea of the proof of the strong polynomiality of the so-called Newton-method for the fractional optimization problem[19].

procedure HanglerZang(s,t,c,d,Δ)
  $p_c^1 := \mathscr{A}(s,t,c)$
  if $d(p_c^1) \le \Delta$ **then return** p_c^1
  $p_d^1 := \mathscr{A}(s,t,d)$
  if $d(p_d^1) > \Delta$
    then return "There is no solution"
  $\lambda_c^1 := 0; \quad \lambda_d^1 := \infty$
  $i := 1$
  repeat
    $\lambda^{i+1} := \frac{c(p_c^i)-c(p_d^i)}{d(p_d^i)-d(p_c^i)}$
    $r := \mathscr{A}(s,t,c_{\lambda^{i+1}})$
    if $c_{\lambda^{i+1}}(r) = c_{\lambda^{i+1}}(p_c)$ **then return** p_d^i
      else if $d(r) \le \Delta$ **then** $p_d^{i+1} := r; \quad p_c^{i+1} := p_c^i; \quad \lambda_d^{i+1} := \lambda^{i+1}; \quad \lambda_c^{i+1} := \lambda_c^i$
                else $p_c^{i+1} := r; \quad p_d^{i+1} := p_d^i; \quad \lambda_c^{i+1} := \lambda^{i+1}; \quad \lambda_d^{i+1} := \lambda_d^i$
  end repeat
end procedure,

where $\mathscr{A}(c)$ returns a c-minimal solution to the basic problem.

Figure 1: The Handler-Zang algorithm

4.1 Strong Polynomiality

The bound on the running time is based on the following theorem due to Goemans' stating that a geometrically decreasing sequence of numbers constructed in a certain restricted way cannot be exponentionally long.

Theorem 8 *Let* $\mathbf{c} = (c_1, c_2, \cdots, c_n)$ *be an n dimensional vector with real components, and let* $\mathbf{y_1}, \mathbf{y_2}, \cdots, \mathbf{y_q}$ *be vectors from* $\{-1,0,1\}^n$. *If for all* $i = 1,2,\cdots,q-1$

$$0 < \mathbf{y_{i+1}c} \le \frac{1}{2}\mathbf{y_i c},$$

then $q = O(n \log n)$. □

For the proof the interested reader is referred to e.g. [19].

For the sake of simplicity let us use the following notations. $c_c^i := c(p_c^i)$, $c_d^i := c(p_d^i)$, $d_c^i := d(p_c^i)$, $d_d^i := d(p_d^i)$, $L_c^i := c(p_c^i) + \lambda_c^i(d(p_c^i) - \Delta)$ and $L_d^i := c(p_d^i) + \lambda_d^i(d(p_d^i) - \Delta)$.

Claim 9 *Suppose that* $p_c^{i+1} \neq p_c^i$. *Then*

$$\frac{c_d^i - c_c^i}{d_c^i - d_d^i} \ge \frac{L^* - c_c^i}{d_c^i - \Delta}. \tag{9}$$

Proof. From the definition of the function $L(\lambda)$ we get that for any $\lambda \ge 0$,

$$L(\lambda) \le c_c^i + \lambda(d_c^i - \Delta) \tag{10}$$

and

$$L(\lambda) \le c_d^i + \lambda(d_d^i - \Delta). \tag{11}$$

By maximizing the value "$L(\lambda)$" subject to the above two conditions we get that

$$L^* = \max_{\lambda \ge 0} L(\lambda) \le c_c^i + \frac{c_d^i - c_c^i}{d_c^i - d_d^i}(d_c^i - \Delta). \tag{12}$$

Substituting this in the right hand side of (9) we get the claim. □

Lemma 10 *Suppose that* $p_c^{i+1} \neq p_c^i$. *Then*

$$\frac{L^* - L_c^{i+1}}{L^* - L_c^i} + \frac{d_c^{i+1} - \Delta}{d_c^i - \Delta} \le 1. \tag{13}$$

Proof.

$$\begin{aligned}
L_c^i &= c_c^i + \lambda_c^i(d_c^i - \Delta) \\
&\le c_c^{i+1} + \lambda_c^i(d_c^{i+1} - \Delta) \\
&= L_c^{i+1} - \lambda_c^{i+1}(d_c^{i+1} - \Delta) + \lambda_c^i(d_c^{i+1} - \Delta) \\
&= L_c^{i+1} + (\lambda_c^i - \lambda_c^{i+1})(d_c^{i+1} - \Delta) \\
&= L_c^{i+1} + (\frac{L_c^i - c_c^i}{d_c^i - \Delta} - \frac{c_d^i - c_c^i}{d_c^i - d_d^i})(d_c^{i+1} - \Delta) \\
&\le L_c^{i+1} + (\frac{L_c^i - c_c^i}{d_c^i - \Delta} - \frac{L^* - c_c^i}{d_c^i - \Delta})(d_c^{i+1} - \Delta) \\
&= L_c^{i+1} + (\frac{L_c^i - L^*}{d_c^i - \Delta})(d_c^{i+1} - \Delta) \\
&= L_c^{i+1} + (L_c^i - L^*)\frac{d_c^{i+1} - \Delta}{d_c^i - \Delta},
\end{aligned}$$

followed by $L^* - L_c^i \ge (L^* - L_c^{i+1}) + (L^* - L_c^i)\frac{d_c^{i+1} - \Delta}{d_c^i - \Delta}$, which is equivalent with the claim.

□

Corollary 11 *Suppose that $p_c^{i+1} \ne p_c^i$. Then*

$$\frac{(L^* - L_c^{i+1})(d_c^{i+1} - \Delta)}{(L^* - L_c^i)(d_c^i - \Delta)} \le \frac{1}{4}. \tag{14}$$

□

By the same token, the following can also be proved.

Corollary 12 *Suppose that $p_d^{i+1} \ne p_d^i$. Then*

$$\frac{(L^* - L_d^{i+1})(\Delta - d_d^{i+1})}{(L^* - L_d^i)(\Delta - d_d^i)} \le \frac{1}{4}. \tag{15}$$

□

From Corollary 11, we get that

Corollary 13 *Suppose that $p_c^{i+1} \ne p_c^i$. Then at least one of the followings are satisfied.*

(a) $L^* - L_c^{i+1} \le \frac{1}{2}(L^* - L_c^i)$,

(b) $d_c^{i+1} - \Delta \le \frac{1}{2}(d_c^i - \Delta)$.

□

First note that both sequences $d_c^i - \Delta$ and $L^* - L_c^i$ are monotonically decreasing. So, from Lemma 8 we get that (b) can be true at most $O(|E|\log|E|)$ times. To estimate how much times (a) can be true, let

$$s_c^i := (L^* - L_c^i)(d_c^{i-1} - d_d^{i-1}) = (L^* - c_c^i)(d_c^{i-1} - d_d^{i-1}) - (c_d^{i-1} - c_c^{i-1})(d_c^i - \Delta) > 0, \tag{16}$$

where $\gamma^i := c_d^i - c_c^i$ and $\delta^i := d_c^i - d_d^i$.

It can be seen from the definition, that s_c^i is monotonically decreasing and if (a) satisfies, then

$$s_c^{i+1} \le \frac{1}{2}s_c^i. \tag{17}$$

The second form of s_c^i shows that there exist vectors $\mathbf{c} \in \mathbb{R}^{2(|E|+1)^2}$ and $\mathbf{y_i} \in \{-1,0,1\}^{2(|E|+1)^2}$ such that $s_c^i = \mathbf{y_i}\mathbf{c}$ for all i. Indeed, $\mathbf{c}$ is an appropriate choice if it consists of the following constants.

$$\Big\{\alpha\cdot\beta\cdot\zeta : \quad \alpha\in\{1,2\}, \quad \beta\in\{c(e):e\in E\}\cup\{L^*\}, \quad \zeta\in\{d(e):e\in E\}\cup\{\Delta\}\Big\}$$

Thus, using Lemma 8 we get that (a) can be true at most $O(|E|^2\log|E|)$ times. So, the sequence $p_c^1, p_c^2, p_c^3, \cdots$ consists of at most $O(|E|^2\log|E|)$ different solutions.

The same can be proved for the sequence $p_d^1, p_d^2, p_d^3, \cdots$, so we obtained that

Theorem 14 *The Handler-Zang algorithm terminates after $O(|E|^2\log|E|)$ iterations.* □□

4.2 Better Bound in Case of Constrained Shortest Path Problem

This section shows that in the case of Constrained Shortest Path Problem a better bound can be actually proved.

In this case we are given a directed graph $G=(V,E)$, two predefined nodes $s,t\in V$ and $\mathscr{P}$ denotes the set of s—t paths. Let $n:=|V|$ and $m:=|E|$.

We use the notations of the previous section with the exception that from now on we consider only the subsequences of $p_c^1,p_c^2,\cdots$ and $p_d^1,p_d^2,\cdots$ that contain each different path exactly ones. In the hope that it will not be confusing, the same notations are used for these subsequences and also for the corresponding values $c_c^i,c_d^i,d_c^i,d_c^i,\lambda_c^i,\lambda_d^i,L_c^i$ and L_d^i. The monotonicity of these latter sequences still hold true of course, however one has to be careful that λ_c^{i+1} is not necessarily equal to $\frac{c_c^i-c_d^i}{d_d^i-d_c^i}$. With these new notations the following version of Claim 11 and Claim 12 is true.

Claim 15 *For each i,*

$$\frac{(L^*-L_c^{i+1})(d_c^{i+1}-\Delta)}{(L^*-L_c^i)(d_c^i-\Delta)}\leq\frac{1}{4} \tag{18}$$

and

$$\frac{(L^*-L_d^{i+1})(\Delta-d_d^{i+1})}{(L^*-L_d^i)(\Delta-d_d^i)}\leq\frac{1}{4}. \tag{19}$$

□

The following claim can be easily seen.

Claim 16 *For any i and* $\lambda\geq\lambda_c^{i+2}$,

$$(c_c^i-\lambda(d_c^i-\Delta))-L^*\geq L^*-L_c^{i+1}$$

or equivalently

$$c_c^i-\lambda(d_c^i-\Delta)\geq 2L^*-L_c^{i+1}$$

□

Definition 17 *Let edge e be called i-*essential *if* $e\in p_c^i\cup p_c^{i+1}\cup p_c^{i+2}\cup\cdots$.

Lemma 18 *Let* $k:=\lceil\frac{\log_2 n+1}{2}\rceil$. *Then for any i at least one of the followings hold.*

(a) $(d_c^{i+k}-\Delta)\leq\frac{1}{2}(d_c^i-\Delta)$,

(b) there exists an i-essential edge e that is not $(i+k)$*-essential.*

Proof. Let us suppose that (a) does not hold, i.e.

$$(d_c^{i+k}-\Delta)>\frac{1}{2}(d_c^i-\Delta)>\frac{1}{2}(d_c^{i+1}-\Delta).$$

Since

$$(L^*-L_c^{i+k})(d_c^{i+k}-\Delta)\leq\frac{1}{2n}(L^*-L_c^{i+1})(d_c^{i+1}-\Delta),$$

we get that

$$(L^*-L_c^{i+k})\leq\frac{1}{n}(L^*-L_c^{i+1}).$$

Now, let us define the following auxiliary cost function on the edges.

$$\widetilde{c}(\vec{uv}):=c_{\lambda_c^{i+k}}(\vec{uv})-\ell(v)+\ell(v),$$

where

$$\ell(u):=\min\{c_{\lambda_c^{i+k}}(p):p\in\mathscr{P}_{s,u}\}.$$

Note that $\widetilde{c}(e)\geq 0$ for each edge e and $\ell(t)=L_c^{i+k}+\lambda_c^{i+k}\Delta$. Moreover, $\widetilde{c}(p)=c_{\lambda_c^{i+k}}(p)-\ell(v)$ for any $p\in\mathscr{P}_{s,v}$.

Using Claim 16 we get that

$$\begin{aligned}\widetilde{c}(p_c^i)&=c_{\lambda_c^{i+k}}(p_c^i)-\ell(t)=c(p_c^i)+\lambda_c^{i+k}d(p_c^i)-\ell(t)\\&=c_c^i+\lambda_c^{i+k}(d_c^i-\Delta)+\lambda_c^{i+k}\Delta-\ell(t)\\&\geq 2L^*-L_c^{i+1}+\lambda_c^{i+k}\Delta-\ell(t)\\&>n(L^*-L_c^{i+k})+L^*+\lambda_c^{i+k}\Delta-\ell(t)\\&=n(L^*-L_c^{i+k})+L^*-L_c^{i+k}=(n+1)(L^*-L_c^{i+k}).\end{aligned}$$

Thus, there exists an edge $e \in p_c^i$ such that $\widetilde{c}(e) > L^* - L_c^{i+k}$. This particular edge e is of course i-essential. To finish the proof, we show that e is not $(i+k)$-essential. To the contrary, suppose that $e \in p_c^j$ for some $j \geq i+k$. Then

$$\begin{aligned} L^* &> c_{\lambda_c^j}(p_c^j) - \lambda_c^j \Delta \\ &\geq c_{\lambda_c^{i+k}}(p_c^j) - \lambda_c^{i+k}\Delta \\ &= \widetilde{c}(p_c^j) + \ell(t) - \lambda_c^{i+k}\Delta \\ &\geq \widetilde{c}(e) + \ell(t) - \lambda_c^{i+k}\Delta \\ &= \widetilde{c}(e) + L_c^{i+k} > L^* \end{aligned}$$

shows the contradiction. □

Corollary 19 *The length of sequence $p_c^1, p_c^2, p_c^3, \cdots$ is $O(m \log^2 m)$.*

Proof. Let $k := \lceil \frac{\log_2 n+1}{2} \rceil$ and let us consider the sequence $i = k, 2k, 3k, \cdots$. According to the previous lemma, at these iteration, either the value $d_c^i - \Delta$ at least halves or the number of the essential edges decreases. Since $d_c^i - \Delta$ is monotonically decreasing, Lemma 8 shows that it can be halved at most $O(m \log m)$ times. On the other hand, we have at most m essential edges at the beginning, so the number of these steps is $O(m \log m + m) = O(m \log m)$. From this the statement follows, because $k = O(\log m)$. □

Again in the same way we get

Lemma 20 *Let $k := \lceil \frac{\log_2 n+1}{2} \rceil$. Then for any i at least one of the followings hold.*

(a) $(\Delta - d_d^{i+k}) \leq \frac{1}{2}(\Delta - d_d^i)$,

(b) there exists an i-d-essential edge e that is not $(i+k)$-essential,

*where an edge e is called i-d-*essential *if $e \in p_d^i \cup p_d^{i+1} \cup p_d^{i+2} \cup \cdots$.* □

and

Corollary 21 *The length of sequence $p_d^1, p_d^2, p_d^3, \cdots$ is $O(m \log^2 m)$.* □

To sum up, Corollary 19 and Corollary 21 together gives the following. (The running time of the Dijkstra algorithm for finding shortest path in a directed graph is $O(m + n \log n)$.)

Theorem 22 *In case of Constrained Shortest Path Problem, the Handler-Zang algorithm terminates after $O(m \log^2 m)$ iterations, thus the full running time of the algorithm is $O(m^2 \log^2 m + mn \log^3 m)$.* □□

Acknowledgments

The author would like to acknowledge the valuable suggestions of Rolf Möhring, Thomasz Radzik, Áron Szentesi and Balázs Szviatovszki.

References

[1] Zheng Wang and Jon Crowcroft, *'Bandwidth-Delay based routing algorithms'*, IEEE GlobeCom'95, Singapore, November 1995.

[2] Funda Ergun, Rakesh Sinha, Lisa Zhang, *'QoS Routing with Performance-Dependent Costs'* IEEE INFOCOM'2000 pp., March, 2000.

[3] M. S. Garey, D.S. Johnson, *'Computers and Intractability: Guide to the Theory of NP-Completeness'*, W. H. Freeman, New York, 1979.

[4] Hussein F. Salama, Douglas S. Reeves and Yannis Viniotis, *'A Distributed Algorithm for Delay-Constrained Unicast Routing'*, IEEE Infocom'97, Kobe, Japan, April 1997.

[5] R. Widyono, *'The design and evaluation of routing algorithms for real-time channels'*, Technical Report TR-94-024, University of California at Berkeley, June 1994.

[6] W.C.Lee, et al. *'Multi-Criteria Routing subject to Resource and Performance Constraints'*, ATM Forum 94-0280, March 1994.

[7] C. Pornavalai, G. Chakraborty and N. Shiratori, *'Routing with multiple QoS requirements for supporting multimedia applications'*, Journal of High Speed Networks, 1998.

[8] Kenji Ishida and Kitsutaro Amano, *'A Delay-Constrained Least-Cost Path Routing Protocol and the Synthesis Method'*, IEEE 1998.

[9] Shigang Cheng and Klara Nahrstedt, *'On finding multi-constrained paths'* ICC'98, Atlanta, Georgia, 1998.

[10] Jeffrey Jaffe, *'Algorithms for finding path with multiple constraints'*, Networks, vol. 14, pp.95-116, 1984.

[11] David Blokh and George Gutin, *'An approximation algorithm for combinatorial optimization problems with two parameters'*, Ausralasian J. Combin. 14 (1996) 157-164

[12] Ravindra K. Ahuja, Tomas L. Magnanti and James B. Orlin, *'Network Flows'* PRENTICE HALL, Upper Saddle River, New Jersey 07458, 1993.

[13] H. de Neve, P. van Mieghem, *'A multiple quality of service routing algorithm for PNNI'*, IEEE ATM'98 Workshop, pp.306-314, Fairfax, Virginia, May 1998.

[14] Liang Guo and Ibrahim Matta, *'Search Space Reduction in QoS Routing'* Technical Report NU-CCS-98-09, October 1998.

[15] Atsushi Iwata, et al. *'ATM Routing Algorithms with Multiple QoS Requirements for Multimedia Internetworking'*, IEICE Trans. Commun., vol.E79-B, pp.999-1007, August 1996

[16] M. Held and R. Karp, *'The traveling salesman problem and minimum spanning trees'*, Operations Research 18, pp.1138-1162, 1970.

[17] M. Held and R. Karp, *'The traveling salesman problem and minimum spanning trees, Part II.'*, Mathematical Programming 6, pp.62-88, 1971.

[18] Balázs Gábor Józsa and Dániel Orincsay, *'Random Graph Generator (for telecommunication networks)'*, Technical Report, April 1999.

[19] T. Radzik, *'Fractional combinatorial optimization'* , In *Handbook of Combinatorial Optimization*, editors Ding Zhu Du and Panos Pardalos, vol. 1, Kluwer Academic Publishers, December 1998.

[20] R. Hassin, *'Approximation schemes for the restricted shortest path problem'*, Mathematics of Operations Research, 17(1):36–42, Feb 1992.

[21] D.H. Lorenz and A. Orda, *'QoS routing in networks with uncertain parameters'*, IEEE/ACM Transactions on Networking, 6(6):768–778, Dec 1998.

[22] Danny Raz and Yuval Shavitt, *'Optimal Partition of QoS requirements with Discrete Cost Functions'*, INFOCOM 2000

[23] Dean H. Lorenz, Ariel Orda, Danny Raz, and Yuval Shavitt, *'Efficient QoS Partition and Routing of Unicast and Multicast'*, IWQoS 2000, June 2000.

[24] Ashish Goel, K.G. Ramakrishnan, Deepak Kataria, Dimitris Logothetis, *Efficient Computation of Delay-sensitive Routes from One Source to All Destinations* Infocom 2001.

[25] A. Jüttner, B. Szviatovszki, Ildikó Mécs, Zs. Rajkó, *'Lagrange Relaxation Based Method for the QoS Routing Problem'*, Infocom 2001.

[26] K. Mehlhorn, M. Ziegelmann, *Resource constrained shortest paths*, 8th Annual European Symposium on Algorithms (ESA), LNCS 1879, 326-337, 2000.

[27] G. Xue, *Primal-dual algorithms for computing weight-constrained shortest paths and weight-constrained minimum spanning trees*, 19th IEEE International Performance, Computing and Communications Conference (IPCCC).

[28] M. Ziegelmann, *Constrained Shortest Paths and Related Problems*, PhD Thesis 2001

[29] G. Handler, I. Zang, *A dual algorithm for the constrained shortest path problem*, Networks **10**, 293-310 (1980)

[30] T. Radzik, *Parametric flows, weighted means of cuts, and fractional combinatorial optimization* Complexity in Numerical Optimization, ed. P. Pardalos, World Scientific 1993, pp. 351-386.

Uniquely Localizable Networks with Few Anchors

ZSOLT FEKETE*

Department of Operations Research
Eötvös University
Pázmány Péter sétány 1/C, 1117 Budapest, Hungary
fezso@cs.elte.hu

TIBOR JORDÁN†

Department of Operations Research
Eötvös University
Pázmány Péter sétány 1/C, 1117 Budapest, Hungary
jordan@cs.elte.hu

Abstract: We consider 'source location problems' in undirected graphs motivated by localization problems in sensor networks. In such a network the fundamental problem is to determine the locations of the sensors in the plane from a subset of pairwise distances. To achieve unique localizability it is necessary to designate a set of sensors, called anchors, for which the exact location is known. We consider the problem of finding a smallest set of anchors which make the network uniquely localizable, provided that the coordinates are 'generic'. We give polynomial time algorithms for two relaxations of the problem. By combining these algorithms we obtain a 2-approximation algorithm for the anchor minimization problem.

Keywords: localization in sensor networks, globally rigid graphs

1 Introduction

The localization problem in two-dimensional sensor networks can be stated as follows: given a set S of nodes (sensors), and the distance d_{ij} between some pairs of nodes s_i, s_j, find a map $p : S \to \mathbb{R}^2$, assigning coordinates $p_i \in \mathbb{R}^2$ to each node s_i such that $||p_i - p_j|| = d_{ij}$ holds for all pairs i, j for which d_{ij} is given. This is one of the fundamental algorithmic problems in the theory of wireless sensor networks and it has been in the focus of a number of recent research articles, see e.g. [2, 11]. A closely related problem is the question of unique localizablity: given a feasible solution for the localization problem, is it unique? If only distance information is available, feasible solutions for the localization problem can never be unique, since we may obtain other solutions by rotations, translations, or reflections of the plane. To exclude such isometries of the plane it is necessary to collect exact location information for some nodes. The nodes for which the exact location is known will be called *anchors*. The previous observation implies that a uniquely localizable network must have at least three anchors.

For some networks we may not expect unique solutions even if we have at least three anchors. This is the case, for example, if the network has a separating pair s_i, s_j of nodes for which some component C of the network obtained by deleting s_i and s_j does not contain anchor nodes. In this case we may obtain another feasible solution by reflecting all nodes of C through the line containing p_i, p_j. Another necessary condition for unique localizablity is *rigidity*: if we think of the network as a two dimensional bar-and-joint framework, in which the anchor nodes are pinned down, and this framework has a continuous deformation, then no solution can be unique. Thus two natural questions emerge: which networks (with anchors) are uniquely localizable? How can we find a smallest set of anchors which make the network uniquely localizable?

In what follows we shall assume that the network is *generic*, i.e. the set of coordinates of the nodes is algebraically independent over $\mathbb{Q}$. With this non-degeneracy assumption it is known that the answer to both questions depends only on the *graph of the network*. In the graph $G = (V, E)$ of the network the vertices $v_i \in V$ correspond to sensor nodes $s_i \in S$, and edges correspond to known distances. Thus G has an edge $v_i v_j$ whenever either d_{ij} is given or s_i, s_j are both anchor nodes (since the distance between two anchor nodes is implicitly given by their locations). As it was observed in [2, 11], the network is uniquely localizable if and only if it has at least three anchors and the graph G of the network is *globally rigid*. We say that a graph H is globally rigid if for all generic networks on H the locations of the nodes are uniquely determined up to isometries of the plane. Globally rigid graphs have been characterized in [5]: a graph is globally rigid if and only if either G is a complete graph on at most three vertices of G is 3-connected and redundantly rigid. (A graph $G = (V, E)$ is *redundantly rigid* if $G - e$ is rigid for all $e \in E$.) Thus we obtain a necessary and sufficient condition for unique localizability.

Theorem 1 *[5, 2, 11] A sensor network (with at least four nodes and with at least three anchors) is uniquely localizable if and only if its graph is* 3*-connected and redundantly rigid.*

*Supported by the MTA-ELTE Egerv´ary Research Group on Combinatorial Optimization and Communication Networks Laboratory, P´azm´any P. s. 1/A, Budapest, Hungary, 1117 and OTKA grants No. T037547, TS049788.

†Supported by the MTA-ELTE Egerv´ary Research Group on Combinatorial Optimization and the Hungarian Scientifi c Research Fund grant No. T049671, T037547.

In this paper we consider the second question (which was also posed in [11]). In light of Theorem 1 the anchor minimization problem can be formulated as follows: given a graph $G=(V,E)$, find a smallest set $P \subseteq V$, $|P| \geq 3$, for which $G+K(P)$, the graph obtained from G by adding a complete graph on vertex set P, is 3-connected and redundantly rigid.

The complexity of the this minimization problem is not known. The goal of this paper is to show that if we replace '$G+K(P)$ is 3-connected and redundantly rigid' by the weaker condition that '$G+K(P)$ is M-connected' then we obtain a tractable problem, which can also be used to give a 2-approximation algorithm for the anchor minimization problem. (A graph is *M-connected* if its rigidity matroid is connected. See Section 3 for more details.)

We shall present polynomial time algorithms for finding optimal solutions for the following relaxed problems: given G, find a smallest set P for which (i) $G+K(P)$ is M-connected, and, assuming that the input graph G is 2-connected, find a smallest set P for which (ii) $G+K(P)$ is 3-connected, or (iii) $G+K(P)$ is 3-connected and rigid.

Since M-connected graphs are 2-connected, we can easily obtain a 2-approximation algorithm for our anchor minimization problem by first solving (i) for the input graph G and then solving (ii) for the augmented graph $G+K(P)$.

Our algorithm for problem (iii) relies on a recent solution of the first author [3] for the problem of finding a smallest set P for which $G+K(P)$ is rigid (the so called 'pinning problem'). We shall not discuss (iii) in more detail in this extended abstract.

2 Rigid graphs

We shall apply graph and matroid theoretical methods. We need the following basic results from combinatorial rigidity (see also [4, 5, 13, 15] for more details). For a graph $G=(V,E)$ and a subset $X \subseteq V$ let $E_G(X)$ and $i_G(X)$ (or simply $E(X)$ or $i(X)$ when it is obvious to which graph we are referring) denote the set and the number of edges in the subgraph induced by X in G. For a set $F \subseteq E$ we shall use $i_F(X)$ to denote the number of those edges in F which are induced by X.

Let $G=(V,E)$ be a graph and let F be a non-empty subset of E. We say that F is *independent* if

$$i_F(X) \leq 2|X|-3 \text{ for all } X \subseteq V \text{ with } |X| \geq 2. \tag{1}$$

The empty set is also defined to be independent. The *rigidity matroid* $\mathscr{M}(G)=(E,\mathscr{I})$ is defined on the edge set of G by

$$\mathscr{I}=\{F \subseteq E : F \text{ is independent in } G\}.$$

A *cover* of $G=(V,E)$ is a collection of subsets $\mathscr{X}=\{X_1,X_2,\ldots,X_t\}$ of V, each of size at least two, such that $\{E_G(X_1),E_G(X_2),\ldots,E_G(X_t)\}$ partitions E. The cover is *non-trivial* if $t \geq 2$. The *value* of the cover is equal to $val(\mathscr{X})=\sum_{i=1}^{t}(2|X_i|-3)$.

Theorem 2 *[10] Let $G=(V,E)$ be a graph. Then $\mathscr{M}(G)$ is a matroid, in which the rank of a non-empty set $E' \subseteq E$ of edges is given by*

$$r(E')=\min val(\mathscr{X}),$$

where the minimum is taken over covers $\mathscr{X}$ of (V,E').

The matroid $\mathscr{M}(G)$ is called the *rigidity matroid* of G. The following fundamental theorem of Laman gives a combinatorial characterization of rigidity in $\mathbb{R}^2$.

Theorem 3 *[6] A graph $G=(V,E)$ is rigid in $\mathbb{R}^2$ if and only if $r(E)=2|V|-3$.*

3 M-connected graphs

Given a matroid $\mathscr{M}=(E,\mathscr{I})$, we define a relation on E by saying that $e,f \in E$ are related if $e=f$ or if there is a circuit C in $\mathscr{M}$ with $e,f \in C$. It is well-known that this is an equivalence relation. The equivalence classes are called the *components* of $\mathscr{M}$. If $\mathscr{M}$ has at least two elements and only one component then $\mathscr{M}$ is said to be *connected*.

The following lemma summarizes two simple facts in matroid theory.

Lemma 4 *Let $\mathscr{M}$ be a matroid with components $E_1,E_2,\ldots,E_t$. Then*
(i) $r(\mathscr{M})=\sum_1^t r(E_i)$, and
(ii) if $r(\mathscr{M})=\sum_1^q r(F_i)$ for some partition $F_1,F_2,\ldots,F_q$ of E then for all components E_i, $1 \leq i \leq t$, there exists a set F_j with $E_i \subseteq F_j$.

We say that a graph $G=(V,E)$ is *M-connected* if $\mathscr{M}(G)$ is connected. It is easy to verify that M-connected graphs are rigid (in fact, redundantly rigid), and hence they are 2-connected [5]. For example, $K_{3,m}$ is M-connected for all $m \geq 4$. The *M-connected components* of G are the subgraphs of G induced by the components of $\mathscr{M}(G)$. It is easy to see that the M-connected components of G are pairwise edge-disjoint induced subgraphs. In the context of global rigidity, M-connected graphs play a central role:

Theorem 5 *[5] Let G be a 3-connected graph. Then G is redundantly rigid if and only if G is M-connected.*

A simple corollary of Lemma 4 is the following.

Lemma 6 *$G=(V,E)$ is M-connected if and only if $val(\mathscr{X})\geq 2|V|-2$ for all non-trivial covers $\mathscr{X}$ of G.*

Let $G=(V,E)$ be a graph and let $\mathscr{H}=\{H_1,H_2,...,H_t\}$ be its M-connected components.

Lemma 7 *Let $G=(V,E)$ be a graph and $P\subseteq V$ with $|P|\geq 4$. Then $G+K(P)$ is M-connected if and only if*

$$2|V|-2\leq 2|Z|-3+\sum_{H_i\in\mathscr{H}:V(H_i)\cap(V-Z)\neq\emptyset} 2|V(H_i)|-3 \tag{2}$$

holds for all $Z\subset V$ with $P\subseteq Z$, $Z\neq V$.

PROOF: First suppose that $G+K(P)$ is M-connected. Since $\mathscr{H}$ is a cover of G and $P\subseteq Z$, $Z\cup\{H_i\in\mathscr{H}:V(H_i)\cap(V-Z)\neq\emptyset\}$ is a cover of $G+K(P)$. This cover is non-trivial, since $Z\neq V$. Thus (2) follows from Lemma 6.

To prove the other direction suppose, for a contradiction, that (2) holds but $G'=G+K(P)$ is not M-connected. Let $\mathscr{H}'=\{H'_1,H'_2,...,H'_q\}$ denote the M-connected components of $G+K(P)$. Since $|P|\geq 4$, $G'[P]$ is M-connected, and hence there is an M-connected component, say H'_1, for which $P\subseteq V(H'_1)$ holds.

Claim 8 *Let H be a graph on vertex set V. Then $H=H'_j$ for some M-connected component H'_j of G' with $2\leq j\leq q$, if and only if $H=H_i$ for some M-connected component H_i of G with $V(H_i)\cap(V-V(H'_1))\neq\emptyset$.*

PROOF: Consider an M-connected component $H'_j\in\mathscr{H}'$ with $j\geq 2$. Since $E_{G'}(H'_1)\cap E_{G'}(H'_j)=\emptyset$, it follows that $G[V(H'_j)]$ is M-connected. Thus, since G' is a supergraph of G and H'_1 is an induced subgraph, we must have $H'_j=H_i$ for some $H_i\in\mathscr{H}$ with $V(H_i)\cap(V-V(H'_1))\neq\emptyset$. Now let $H_i\in\mathscr{H}$ with $V(H_i)\cap(V-V(H'_1))\neq\emptyset$. Since H'_1 is M-connected in G' and M-connected components are edge-disjoint, it follows that $V(H_i)$ induces a maximal M-connected subgraph in G'. □

By Lemma 4(i), Claim 8, and by applying (2) with $Z=V(H'_1)$, we obtain $2|V|-3\geq r(G')=2|V(H'_1)|-3+\sum_{H_i\in\mathscr{H}:V(H_i)\cap(V-V(H'_1))\neq\emptyset}(2|V(H_i)|-3)\geq 2|V|-2$, a contradiction. □

Let $\widetilde{\mathscr{H}}=(V,\mathscr{E})$ be the hypergraph obtained from $\mathscr{H}$ by replacing each set H_i by $2|V(H_i)|-3$ copies of $V(H_i)$, $1\leq i\leq t$. (It follows from Lemma 4(i) that $|\mathscr{E}|\leq 2|V|-3$.) For some $X\subseteq V$ let $e_{\widetilde{\mathscr{H}}}(X)$ denote the number of hyperedges $e\in\mathscr{E}$ with $e\cap X\neq\emptyset$. By taking the complement of P in Lemma 7 and using the above definitions we obtain the following corollary.

Corollary 9 *Let $G=(V,E)$ be a graph and $T\subseteq V$ with $|V-T|\geq 4$. Then $G+K(V-T)$ is M-connected if and only if*

$$e_{\widetilde{\mathscr{H}}}(S)\geq 2|S|+1 \tag{3}$$

holds for all $S\subseteq V$ with $\emptyset\neq S\subseteq T$.

Thus finding a smallest set P for which $|P|\geq 4$ and $G+K(P)$ is M-connected is equivalent to finding a largest set T for which (3) holds. The crucial observation is that this problem can be formulated as a matroid matching problem [9] as follows.

We say that a hypergraph $\mathscr{H}'=(V,\mathscr{E}')$ satisfies the *strong Hall condition* (or $\mathscr{H}'$ is a *hyperforest*) if $|\cup\mathscr{F}|\geq|\mathscr{F}|+1$ for all $\emptyset\neq\mathscr{F}\subseteq\mathscr{E}'$. Lorea [7] proved that in a hypergraph $\mathscr{H}'=(V,\mathscr{E}')$ the subhypergraphs of $\mathscr{H}'$ which are hyperforests form a family of independent sets of a matroid on ground set $\mathscr{E}'$. This matroid is the *hypergraphic matroid* $\mathscr{M}_{\mathscr{H}'}$ of $\mathscr{H}'$. In the bipartite graph $G_{\mathscr{H}'}=(U,V;E)$ of a hypergraph $\mathscr{H}'=(V,\mathscr{E}')$ colour class U corresponds to $\mathscr{E}'$, and for $e\in U$ and $v\in V$, $ev\in E$ if and only if $v\in e$. Thus the strong Hall condition for $\mathscr{H}'$ corresponds to the strong Hall condition for all non-empty subsets of U in $G_{\mathscr{H}'}$.

Now let us consider the bipartite graph G^* obtained from the bipartite graph $G_{\widetilde{\mathscr{H}}}$ of $\widetilde{\mathscr{H}}$ by replacing each vertex $v\in V$ by two vertices v_1,v_2 (which are connected to the neighbours of v in $G_{\widetilde{\mathscr{H}}}$). Let $\mathscr{L}$ be the hypergraph whose bipartite graph $G_{\mathscr{L}}$ is obtained from G^* by interchanging the two colour classes. The pairs v_1,v_2, $v\in V$, define a pairing of the hyperedges of $\mathscr{L}$. It is not difficult to see that finding a largest set T for which (3) holds is equivalent to finding a largest independent set in the hypergraphic matroid $\mathscr{M}_{\mathscr{L}}$ of $\mathscr{L}$ which consists of pairs.

The matroid matching problem in hypergraphic matroids has been shown to be polynomially solvable in a recent paper of Makai [12]. By using this result, the fact that $|\mathscr{E}|\leq 2|V|-3$, and that the M-connected components can be found in polynomial time [1], we obtain that problem (i) is polynomially solvable. Since the algorithm of [12] is rather complicated, we shall also discuss a somewhat simpler randomized version in the next section, which is based on the fact that hypergraphic matroids are linear.

4 Generic representation of hypergraphic matroids

Lovász [9] proved that the matroid matching problem is solvable in polynomial time if the matroid is linear, and a linear representation is given. Hypergraphic matroids are known to be linear, but it is not known how to find a suitable linear representation. In a randomized setting this difficulty can be overcome by using a *generic* representation, i.e. a matrix whose entries are variables. In this section we show how to obtain such a generic representation for hypergraphic matroids.

Let $\mathscr{H} = (V, \mathscr{E})$ be a hypergraph and let $\mathscr{M}_{\mathscr{H}}$ be its hypergraphic matroid.

Theorem 10 *[8] $\mathscr{F} \subseteq \mathscr{E}$ is independent in $\mathscr{M}_{\mathscr{H}}$ if and only if there exists a function $f : \mathscr{F} \to \binom{V}{2}$ such that $f(e) \subseteq e$ for every $e \in \mathscr{E}$ and $\{f(e) : e \in \mathscr{F}\}$ is a forest.*

For every $e \in \mathscr{E}$ we introduce $|e|-1$ variables $x^e_1, \ldots, x^e_{|e|-1}$ and define a vector $R_e \in \mathbb{R}^{|V|}$ as follows. Let us fix an enumeration of V: $V = \{v_1, \ldots, v_{|V|}\}$. If $e = \{v_{i_1}, \ldots, v_{i_k}\}$ where $e \in \mathscr{E}$ and $1 \le i_1 < i_2 < \cdots < i_k \le |V|$, then let

$$R_e(v) := \begin{cases} x^e_j & \text{if } v = v_{i_j} \text{ and } 1 \le j \le k-1, \\ -\sum_{i=1}^{k-1} x^e_j & \text{if } v = v_{i_k}, \\ 0 & \text{otherwise.} \end{cases}$$

Note that if $|e| = 1$, then $R_e = 0 \in \mathbb{R}^{|V|}$. Let $a_v \in \mathbb{R}^{|V|}$ denote the unit vector of coordinate v ($a_v(v) = 1$ and $a_v(u) = 0$ for $u \in V - v$) and let $a \in \mathbb{R}^{|V|}$ the all-one vector ($a(u) = 1$ for all $u \in V$). Observe that $\sum_{v \in V} R_e(v) = 0$, that is, R_e is orthogonal to a.

Claim 11 *Let $\mathscr{F} \subseteq \mathscr{E}$. The set $\mathscr{F}$ is independent in $\mathscr{M}_{\mathscr{H}}$ if and only if the vectors $R_e,\ e \in \mathscr{F}$ are linearly independent.*

PROOF: Suppose that the vectors $R_e,\ e \in \mathscr{F}$ are linearly independent. Let $\emptyset \ne X \subseteq V$. The vectors $a_v, v \in V - X$ and a are orthogonal to $R_e, e \in \mathscr{F}$, thus $\dim(\{R_e \in \mathbb{R}^{|V|} : e \in \mathscr{F}, e \subseteq X\}) \le |X| - 1$. By the independence of vectors R_e we get $i_{\mathscr{F}}(X) = |\{e \in \mathscr{F} : e \subseteq X\}| = \dim(\{R_e \in \mathbb{R}^{|V|} : e \in \mathscr{F}, e \subseteq X\})$. Thus $i_{\mathscr{F}}(X) \le |X| - 1$.

Suppose that set $\mathscr{F} \subseteq \mathscr{E}$ is independent in the hypergraphic matroid $\mathscr{M}_{\mathscr{H}}$. By Theorem 10 there exists a function $f : \mathscr{F} \to \binom{V}{2}$ such that $f(e) \subseteq e$ for every $e \in \mathscr{E}$ and $\{f(e) : e \in \mathscr{F}\}$ is a forest. It is easy to see that if $f(e) = \{u, v\}$ holds then we can assign values to variables x^e_j so that $R_e(u) = 1, R_e(v) = -1$ and $R_e(w) = 0$ holds for all $w \in V - \{u, v\}$. With these values let M denote the $|\mathscr{F}| \times |V|$ matrix formed by the vectors $R_e, e \in \mathscr{F}$ as rows. This matrix M is the oriented incidence matrix of the forest $\{f(e) : e \in F\}$. This implies that the rows of M are linearly independent. □

Corollary 12 *The vectors $R_e,\ e \in \mathscr{E}$ represent $\mathscr{M}_{\mathscr{H}}$.*

By using a lemma of Schwartz [14] it can be verified that there is an integer N of the order $2^{O(|V|)}$ such that if we assign random integers from the interval $[0, N]$ to the variables of the above generic representation then we obtain a matrix whose matroid will be isomorphic to $\mathscr{M}_{\mathscr{H}}$ with probability at least $1/2$. Based on this fact, and using the matroid matching algorithm of Lovász [8], we obtain an efficient randomized algorithm for problem (i).

5 An algorithm for the anchor minimization problem

First we show how to solve problem (ii). Let $H = (V, E)$ be a graph. For some $X \subseteq V$ let $N(X)$ denote the set of neighbours of X and let $S(X) = X \cup N(X)$. We say that $X \subset V$ is *tight* if $|N(X)| = 2$ and $S(X) \ne V$. The following lemmas are easy to prove.

Lemma 13 *Let $H = (V, E)$ be 2-connected and let $X, Y \subset V$ be distinct minimal tight sets in G. Then $X \cap Y = \emptyset$. Furthermore, if $N(X) \cap Y \ne \emptyset$ then $|X| = |Y| = 1$.*

Lemma 14 *Let $H = (V, E)$ be 2-connected and let $P \subseteq V$. Then $H + K(P)$ is 3-connected if and only if $P \cap X \ne \emptyset$ for all minimal tight sets X of H.*

It follows from Lemmas 13 and 14 that every inclusionwise minimal subset P for which $H + K(P)$ is 3-connected is an optimal solution for problem (ii). This gives rise to simple polynomial algorithm for (ii).

Finally we give a sketch of the polynomial 2-approximation algorithm for the anchor minimization problem. First we check whether there exists a set $P \subset V$ in the input graph $G = (V, E)$ for which $|P| = 3$ and $G + K(P)$ is M-connected. If there is no such set, we apply the matroid matching based algorithm to find a smallest set P for which $G + K(P)$ is M-connected. Note that $H = G + K(P)$ is 2-connected. Then we find a smallest set P' for which $H + K(P')$ is 3-connected. It is easy to see that $P \cup P'$ will be a feasible solution whose size is not more than twice the size of an optimal solution.

References

[1] A.R. BERG AND T. JORDÁN, Algorithms for graph rigidity and scene analysis, Proc. 11th Annual European Symposium on Algorithms (ESA) 2003, (G. Di Battista, U. Zwick, eds) Springer Lecture Notes in Computer Science 2832, pp. 78-89, 2003.

[2] T. EREN, D.K. GOLDENBERG, W. WHITELEY, Y.R. YANG, A.S. MORSE, B.D.O. ANDERSON, AND P.N. BELHUMEUR, Rigidity, Computation, and Randomization in Network Localization, Proceedings of the International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), March 2004.

[3] Z. FEKETE, Source location problems with rigidity and tree packing requirements, Proceedings of the 4th Japanese-Hungarian Symposium on Discrete Mathematics and Its Applications, Budapest, June 2005.

[4] J. GRAVER, B. SERVATIUS, AND H. SERVATIUS, Combinatorial Rigidity, AMS Graduate Studies in Mathematics Vol. 2, 1993.

[5] B. JACKSON AND T. JORDÁN, Connected rigidity matroids and unique realizations of graphs, *J. Combinatorial Theory, Ser. B.*, Vol. 94, 1-29, 2005.

[6] G. LAMAN, On graphs and rigidity of plane skeletal structures, *J. Engineering Math.* 4 (1970), 331-340.

[7] M. LOREA, Hypergraphes et matroides, *Cahiers Centre Etud. Rech. Oper.* 17 (1975) pp. 289-291.

[8] L. LOVÁSZ, A generalization of Kőnig's theorem, *Acta. Math. Acad. Sci. Hungar.* 21 (1970), 443–446.

[9] L. LOVÁSZ, The matroid matching problem, in: *Algebraic methods in graph theory*, Proc. Conf. Szeged (1978).

[10] L. LOVÁSZ AND Y. YEMINI, On generic rigidity in the plane, *SIAM J. Algebraic Discrete Methods* 3 (1982), no. 1, 91–98.

[11] A. MAN-CHO SO AND Y. YE, Theory of semidefinite programming for sensor network localization, Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA) 2005.

[12] M. MAKAI, Rigid graphs from edge-pairs, Proceedings of the 4th Japanese-Hungarian Symposium on Discrete Mathematics and Its Applications, Budapest, June 2005.

[13] A. RECSKI, *Matroid theory and its applications in electric network theory and in statics*, Akadémiai Kiadó, Budapest, 1989.

[14] J.T. SCHWARTZ, Fast probabilistic algorithms for verification of polynomial identities, *J. ACM* 27, 701-717, 1980.

[15] W. WHITELEY, Some matroids from discrete applied geometry, in Matroid theory (J.E. Bonin, J.G. Oxley and B. Servatius eds., Seattle, WA, 1995), Contemp. Math., 197, Amer. Math. Soc., Providence, RI, 1996, 171–311.

On the Rank Function of the 3-Dimensional Rigidity Matroid

BILL JACKSON*

School of Mathematical Sciences
Queen Mary, University of London
Mile End Road, London E1 4NS, England
B.Jackson@qmul.ac.uk

TIBOR JORDÁN†

Department of Operations Research
Eötvös University
Pázmány sétány 1/C, 1117 Budapest, Hungary
jordan@cs.elte.hu

Abstract: A major open problem in combinatorial rigidity is to find a good characterization for independence in the 3-dimensional rigidity matroid, or more generally, to give a min-max formula for the rank function. We give a new upper bound on the rank and conjecture that our bound is tight.

Keywords: rigid graphs, rigidity matroid, rank function

1 Introduction

A *framework* (G,p) in d-space is a graph $G=(V,E)$ and a map $p:V\to\mathbb{R}^d$. The *rigidity matrix* of the framework is the matrix $R(G,p)$ of size $|E|\times d|V|$, where, for each edge $v_iv_j\in E$, in the row corresponding to v_iv_j, the entries in the d columns corresponding to vertices i and j contain the d coordinates of $(p(v_i)-p(v_j))$ and $(p(v_j)-p(v_i))$, respectively, and the remaining entries are zeros. See [14] for more details. The rigidity matrix of (G,p) defines the *rigidity matroid* of (G,p) on the ground set E by linear independence of rows of the rigidity matrix. A *framework* (G,p) is *generic* if the set of coordinates of the points $p(v)$, $v\in V$, is algebraically independent over the rationals. Any two generic frameworks (G,p) and (G,p') have the same rigidity matroid. We call this the d-dimensional *rigidity matroid* $\mathscr{R}_d(G)=(E,r_d)$ of the graph G. We denote the rank of $\mathscr{R}_d(G)$ by $r_d(G)$.

Lemma 1 *[14, Lemma 11.1.3] Let (G,p) be a framework in $\mathbb{R}^d$. Then rank$R(G,p)\le S(n,d)$, where $n=|V(G)|$ and*

$$S(n,d)=\begin{cases} nd-\binom{d+1}{2} & \text{if } n\ge d+2\\ \binom{n}{2} & \text{if } n\le d+1.\end{cases}$$

We say that a graph $G=(V,E)$ is *rigid* in $\mathbb{R}^d$ if $r_d(G)=S(n,d)$. (This definition is motivated by the fact that if G is rigid and (G,p) is a generic framework on G, then every continuous deformation of (G,p) which preserves the edge lengths $||p(u)-p(v)||$ for all $uv\in E$, must preserve the distances $||p(w)-p(x)||$ for all $w,x\in V$, see [14].) We say that G is *M-independent* in $\mathbb{R}^d$ if E is independent in $\mathscr{R}_d(G)$. For $X\subseteq V$, let $E_G(X)$ denote the set, and $i_G(X)$ the number, of edges in $G[X]$, that is, in the subgraph induced by X in G. We use $E(X)$ or $i(X)$ when the graph G is clear from the context. A *cover* of G is a collection $\mathscr{X}$ of pairwise incomparable subsets of V, each of size at least two, such that $\cup_{X\in\mathscr{X}}E(X)=E$.

Lemma 1 implies the following necessary condition for G to be M-independent.

Lemma 2 *If $G=(V,E)$ is M-independent in $\mathbb{R}^d$ then $i(X)\le S(|X|,d)$ for all $X\subseteq V$.*

It also gives the following upper bound on the rank function.

Lemma 3 *If $G=(V,E)$ is a graph then*

$$r_d(G)\le\min_{\mathscr{X}}\sum_{X\in\mathscr{X}}S(|X|,d)$$

where the minimum is taken over all covers $\mathscr{X}$ of G.

The converse of Lemma 2 also holds for $d=1,2$. The case $d=1$ follows from the fact that the 1-dimensional rigidity matroid of G is the same as the cycle matroid of G, see [3, Theorem 2.1.1]. The case $d=2$ is a result of Laman [8]. Similarly, the inequality given in Lemma 3 holds with equality when $d=1,2$. The case $d=2$ is a result of Lovász and Yemini [9]. Neither of these statements hold for $d\ge 3$. Indeed, it remains an open problem to find good characterizations for independence or, more generally, the rank function in the d-dimensional rigidity matroid of a graph when $d\ge 3$.

*Supported by the Royal Society/Hungarian Academy of Sciences Exchange Programme and the Finite Structures project of the R´enyi Institute of Mathematics, Budapest within the 6th Framework Programme of the EU.

†Supported by the MTA-ELTE Egerv´ary Research Group on Combinatorial Optimization and the Hungarian Scientifi c Research Fund grant no. T037547, T049671.

2 Independent covers

In the rest of the paper we shall assume that $d = 3$. Let $G = (V,E)$ be a graph. A cover $\mathscr{X} = \{X_1, X_2, \ldots, X_m\}$ of G is *t-thin* if $|X_i \cap X_j| \leq t$ for all $1 \leq i \leq m$. Let $\mathscr{X}$ be a 2-thin cover of G. For $X_i \in \mathscr{X}$ let $f(X_i) = 1$ if $|X_i| = 2$ and $f(X_i) = 3|X_i| - 6$ if $|X_i| \geq 3$. (Thus $f(X_i) = S(|X_i|, 3)$.) Let $H(\mathscr{X})$ be the set of all pairs of vertices uv such that $X_i \cap X_j = \{u,v\}$ for some $1 \leq i < j \leq m$. For each $uv \in H(\mathscr{X})$ let $d(uv)$ be the number of sets X_i in $\mathscr{X}$ such that $\{u,v\} \subseteq X_i$ and put

$$val(\mathscr{X}) = \sum_{X \in \mathscr{X}} f(X) - \sum_{uv \in H(\mathscr{X})} (d(uv) - 1).$$

In 1983, Dress, Drieding and Haegi [2, equation (39)], [13, Conjecture 3] conjectured that 2-thin covers could be used to determine the rank function of $\mathscr{R}(G)$: if $G = (V,E)$ is a graph and $E' \subseteq E$ then the rank $r(E')$ is equal to

$$\min_{\mathscr{X}} \{val(\mathscr{X})\}, \tag{1}$$

where the minimum is taken over all 2-thin covers $\mathscr{X}$ of $G[E']$. This conjecture, which would have provided a good characterization for the rank function of $\mathscr{R}(G)$, was recently disproved in [6].

At a conference on rigidity held in Montreal in 1987, Dress conjectured that equality is obtained in (1) for the special 2-thin cover defined as follows. For $u, v \in V$, the edge uv is an *implied edge* of G if $uv \notin E$ and $r(E + uv) = r(E)$. The closure $\widehat{G}$ of G is the graph obtained by adding all the implied edges to G. A *rigid cluster* of G is a set of vertices which induce a maximal complete subgraph of $\widehat{G}$. It is not difficult to see that any two rigid clusters of G intersect in at most two vertices. Thus the set of rigid clusters of G is a 2-thin cover of G.

Conjecture 4 *(see [1],[3, Conjecture 5.6.1], and [11, Conjecture 2.3]) Let $G = (V,E)$ be a graph and $\mathscr{X}$ be the set of rigid clusters of G. Then*

$$r(E) = val(\mathscr{X}). \tag{2}$$

This conjecture is still open. Note however, that even if Conjecture 4 was shown to be true, it would not provide a good characterization for the rank function.

It is conceivable that Conjecture 4 is true because of the special intersection properties of rigid clusters. If so, then it may be possible to resurrect the first conjecture of Dress et al. by only considering 2-thin covers whose intersection properties reflect those of rigid clusters. Note that for graphs of bounded maximum degree the rank function has been determined in [7].

We say that a 2-thin cover $\mathscr{X}$ of a graph $G = (V,E)$ is *independent* if the subgraphs of $(V, H(\mathscr{X}))$ induced by the sets $X_i \in \mathscr{X}$ are M-independent. The cover is *closed* if $(V, H(\mathscr{X}))$ is a subgraph of G. The following lemma shows that independent 2-thin covers of G can be used to give an upper bound on $r(G)$ (c.f. [6, Lemma 3.4]). (The hypothesis that the cover is closed is not crucial since an independent 2-thin cover of G is an independent closed 2-thin cover of a supergraph of G.)

Lemma 5 *Let $G = (V,E)$ be a graph, and $\mathscr{X}$ be an independent closed 2-thin cover of G. Then $r(E) \leq \sum_{X_i \in \mathscr{X}} r(G[X_i]) - \sum_{uv \in H(\mathscr{X})} (d(uv) - 1)$.*

PROOF: Let $H = H(\mathscr{X})$ and for each $X_i \in \mathscr{X}$ let $S_i = E(X) \cap H$. Since $\mathscr{X}$ is independent and closed, (X_i, S_i) is an M-independent subgraph of $G[X_i]$ and hence S_i can be extended to a basis B_i for the rigidity matroid of $G[X_i]$. Let $S = \cup_{X_i \in \mathscr{X}} B_i$. Then S spans E since, if $e \in E$ then $e \in E(X_i)$ for some $X_i \in \mathscr{X}$ and hence e is spanned by $B_i \subseteq S$. Thus $r(E) \leq |S|$. Furthermore, $|B_i| = r(G[X_i])$ for all $X_i \in \mathscr{X}$. Since S covers each $uv \in S - H$ exactly once and covers each $uv \in H$ exactly $d(uv)$ times, we have

$$|S| = \sum_{X_i \in \mathscr{X}} |B_i| - \sum_{uv \in H} (d(uv) - 1) \leq \sum_{X_i \in \mathscr{X}} r(G[X_i]) - \sum_{uv \in H(\mathscr{X})} (d(uv) - 1),$$

as claimed. □

3 Iterated covers

Definition 6 *An iterated 2-thin cover of G of depth m is a rooted tree $\mathscr{C}$ of depth m whose nodes are induced subgraphs of G and is such that*

(i) the root of $\mathscr{C}$ is G,

(ii) each leaf of $\mathscr{C}$ is at distance m from the root,

(iii) for each node W of $\mathscr{C}$ which is not a leaf, the vertex sets of the children of W is an independent closed 2-thin cover $\mathscr{X}_W$ of W.

Example Let G be obtained from two disjoint K_5's by identifying an edge uv. This is the well-known 'double banana' graph with an extra edge connecting the two vertices of the 2-separator. An iterated cover of depth one has G as its root, and the two K_5's on the first level.

The next proposition follows from the definition.

Proposition 7 *Any two subgraphs at the same level of $\mathscr{C}$ have at most two vertices in common.*

Note that the vertex sets of the subgraphs at the same level do not necessarily form a cover of G since subgraphs of size two may be repeated or contained in other subgraphs.

Let $\mathscr{C}^j$ be the family of covers at level j in $\mathscr{C}$. We must have $\mathscr{C}^0 = \{\{G\}\}$ by definition. For each graph W belonging to a cover at level i, $0 \le i \le m-1$, there exists a unique element $\mathscr{X}_W \in \mathscr{C}^{i+1}$ such that $\mathscr{X}_W$ is a cover of W.

For $\mathscr{X} \in \mathscr{C}^j$ let

$$\gamma(\mathscr{X}) = \sum_{(u,v)\in H(\mathscr{X})} (d(u,v)-1),$$

$$\gamma_j = \sum_{\mathscr{X}\in\mathscr{C}^j} \gamma(\mathscr{X}),$$

and

$$\gamma(\mathscr{C}) = \sum_{j=0}^{m} \gamma_j.$$

Put

$$val(\mathscr{C}) = \sum_{\mathscr{X}\in\mathscr{C}^m} \sum_{W\in\mathscr{X}} f(|V(W)|) - \gamma(\mathscr{C}).$$

Lemma 8 *$r(G) \le val(\mathscr{C})$ for all iterated 2-thin covers $\mathscr{C}$ of G.*

PROOF: Let $\mathscr{C}$ have depth m. We use induction on m. If $m = 0$ then $\mathscr{C} = \mathscr{C}^0 = \{\{G\}\}$ and $val(\mathscr{C}) = f(|V(G)|) \ge r(G)$ by Lemma 2. So suppose $m \ge 1$. We have $\mathscr{C}^1 = \{\mathscr{X}\}$, where $\mathscr{X}$ is a closed, independent 2-thin cover of G. By Lemma 5

$$r(G) \le \sum_{W\in\mathscr{X}} r(W) - \gamma(\mathscr{X}).$$

For each $W \in \mathscr{X}$ let $\mathscr{C}_W$ be the iterated cover of W induced by W in $\mathscr{C}$ (we take the subtree of $\mathscr{C}$ roted at W). Then $\mathscr{C}_W$ has depth $m-1$ and by induction

$$r(W) \le val(\mathscr{C}_W).$$

Furthermore $\gamma(\mathscr{C}) = \gamma(\mathscr{X}) + \sum_{W\in\mathscr{X}} \gamma(\mathscr{C}_W)$ and

$$\sum_{\mathscr{Y}\in\mathscr{C}^m} \sum_{F\in\mathscr{Y}} f(|V(F)|) = \sum_{W\in\mathscr{X}} \sum_{\mathscr{Y}\in\mathscr{C}_W^{m-1}} \sum_{F\in\mathscr{Y}} f(|V(F)|),$$

which implies the lemma. □

Corollary 9 *Let G be a graph. Then $r(G) \le \min\{val(\mathscr{C}) : \mathscr{C}$ is an iterated 2-thin cover of K for some supergraph K of G on vertex set $V(G)\}$.*

Example The double banana graph shows that we can obtain a better upper bound by considering supergraphs. This is because Definition 6(iii) requires independent 2-thin covers of $\mathscr{C}$ to be closed.

Conjecture 10 *Let G be a graph. Then $r(G) = \min\{val(\mathscr{C}) : \mathscr{C}$ is an iterated 2-thin cover of K for some supergraph K of G on vertex set $V(G)\}$.*

Example The following construction due to Tay [12] shows that we may need iterated covers of depth at least two to verify the rank of G. Let $G_0 = (V_0, E_0)$ be a complete graph on five vertices with $V_0 = \{v_i : 1 \le i \le 5\}$. For $1 \le i < j \le 5$ let $G_{i,j} = (V_{i,j}, E_{i,j})$ be a complete graph on five vertices with $V_{i,j} \cap V_0 = \{v_i, v_j\}$ and $E_{i,j} \cap E_0 = \{v_i v_j\}$ for $1 \le i < j \le 5$. Let

$$G = \big(G_0 \cup (\cup_{1\le i<j\le 5} G_{i,j})\big) - E_0.$$

It can be seen that $r(G) = |E(G)| - 1 = 89$. On the other hand, we believe that $\min_{\mathscr{X}} val(\mathscr{X})$ over all independent 2-thin covers $\mathscr{X}$ of G is 90. Note that the set of implied edges of G is E_0, and hence the rigid clusters of G are V_0 and the sets $V_{i,j}$ for $1 \le i < j \le 5$. Hence, if $\mathscr{X}$ is the set of rigid clusters of G, then we have $H(\mathscr{X}) = E_0$ and $val(\mathscr{X}) = 89$. Thus Conjecture 4 holds for G. To see that Conjecture 10 holds for G take $K = G + E_0$ and define an iterated cover $\mathscr{C}$ of depth 2 by $\mathscr{C}^0 = \{\{K\}\}$

and such that $\mathscr{C}^1$ consists of $W = K[V_0 \cup V_{1,2}]$ and the remaining nine K_5's on $V_{i,j}$, $1 \leq i < j \leq 5$, $(i,j) \neq (1,2)$. For $\mathscr{C}^2$ let $\mathscr{X}_W$ be $\{K[V_0], K[V_{1,2}]\}$ and let $\mathscr{X}_{G[V_{i,j}]} = \{G[V_{i,j}]\}$ for the remaining K_5's. Now

$$\sum_{\mathscr{X} \in \mathscr{C}^2} \sum_{W \in \mathscr{X}} f(|V(W)|) - \gamma(\mathscr{C}) = 99 - 10 = 89.$$

Example The following example is due to Tay [12]. Let $G = G_1 \cup G_2 \cup \ldots \cup G_m$ where $m = 7$, $V(G_i) \cap V(G_{i+1}) = \{x_i, y_i\}$, $E(G_i) \cap V(E(G_{i+1}) = \emptyset$, $G_i = K_5 - \{x_{i-1}y_{i-1}, x_i y_i\}$. Then G is M-independent, the rigid clusters of G are $V(G_i)$. Let $H_1 = G + x_1 y_1$. Then $|E(H_1)| = 3|V(H_1)| - 6$ but H_1 is M-dependent. This follows by taking an iterated cover of depth one, where the subgraphs on level one are the seven K_5's. Note that this is a 4-connected graph satisfying the necessary condition of Lemma 2 but it is not rigid.

Lemma 11 *Let K be a graph on n vertices and let $\mathscr{C}$ be an iterated 2-thin cover of K. Then*
(a) the number of subgraphs at level i containing some edge uv is at most $i(n-2)+1$,
(b) there exists an iterated cover $\mathscr{C}'$ of K of depth at most $n-1$ on at most $(n-1)^3\binom{n}{2}$ nodes such that $val(\mathscr{C}') \leq val(\mathscr{C})$.

PROOF: First we prove (a) by induction on i. Since $\mathscr{C}^0 = \{\{K\}\}$, the statement is trivially true for $i = 0$. Now suppose that part (a) is true for levels up to i and let $H_1, H_2, ..., H_t$ be the subgraphs containing edge uv at level i and let $n_j = |H_j|$ for $1 \leq j \leq t$. Then Proposition 7 implies that $\sum_{j=1}^t n_j \leq n - 2 + 2t$. By using this inequality, the induction hypothesis, and that $\mathscr{X}_{H_j}$ is a 2-thin cover of H_j, we can deduce that the number of subgraphs containing uv on level $i+1$ is at most $\sum_{j=1}^t (n_j - 1) \leq n - 2 + t \leq n - 2 + i(n-2) + 1 \leq (i+1)(n-2) + 1$, as required.

To prove (b) choose $\mathscr{C}'$ such that $val(\mathscr{C}') \leq val(\mathscr{C})$, $depth(\mathscr{C}')$ is as small as possible, and subject to this condition, the number of nodes of $\mathscr{C}'$ is as large as possible. Suppose there exists a node W_i on level i in $\mathscr{C}'$ such that W has exactly one child W_{i+1} on level $i+1$ and then at least two children on level $i+2$. Construct $\mathscr{C}''$ from $\mathscr{C}'$ by contracting the edge $W_i W_{i+1}$ and adding a new leaf to each leaf in the subtree rooted at W_{i+1}. Then $val(\mathscr{C}'') = val(\mathscr{C}')$, $depth(\mathscr{C}'') = depth(\mathscr{C}')$, and $\mathscr{C}''$ has more nodes than $\mathscr{C}'$, a contradiction. Let $depth(\mathscr{C}') = m$. If each node on level $m-1$ has exactly one child then we construct $\mathscr{C}''$ by deleting the leaves of $\mathscr{C}'$. Hence $\mathscr{C}^{m-1}$ has at least one node W with at least two children. Each node on the path from G to W has at least two children. Hence each node is a proper subgraph of the parent. It follows that $m \leq n-1$. The upper bound on the number of nodes now follows from (a). □

Lemma 11(b) implies that Conjecture 10 would give a good characterization for $r(G)$. This follows from the polynomial upper bound on the number of nodes of the iterated 2-thin cover of K and the fact that M-independence of some graph H can be verified in polynomial time by providing a framework on H whose rigidity matrix has sufficiently high rank (a lemma of Schwartz [10] implies that there is always such a matrix with small enough entries). (There exist combinatorial certificates for M-independence by using Henneberg sequences [4]. These methods work for many graphs, but the proof of the existence of such a certificate for all graphs is still missing.)

Example Let G be obtained from K_r, $r \geq 3$, by taking a '2-sum' with K_5 along each edge of K_r and then adding back all edges of K_r. (The graph $G + E_0$ in Example 3 corresponds to the case $r = 5$.) We have $r(G) = 3r - 6 + 8\binom{r}{2}$. We conjecture that any iterated 2-thin cover of G with $val(\mathscr{C}) = r(G)$ has depth $f(r)$ where $f(r)$ tends to infinity. Note that $\widehat{G} = G$ so $val(\mathscr{C}') \geq r(K) > r(G)$ for any iterated 2-thin cover $\mathscr{C}'$ of a proper supergraph K of G.

Our upper bound may be used as a certificate of correctness for algorithms for 3-dimensional rigidity. For instance, to get structural information on molecules, biologists and physicists have developed heuristic algorithms for computing the rank of a graph of the molecule, see e.g. [5].

References

[1] H. CRAPO, A. DRESS AND T.-S. TAY, Problem 4.2, in Matroid Theory (J.E. Bonin, J.G. Oxley and B. Servatius eds., Seattle, WA, 1995), Contemp. Math., 197, Amer. Math. Soc., Providence, RI, 1996, 414.

[2] A. DRESS, A. DRIEDING AND H. HAEGI, Classification of mobile molecules by category theory, *in* Symmetries and properties of non-rigid molecules: A comprehensive study, (J. Maruana and J. Serre, eds.) Elsevier, Amsterdam, 1983, 39-58.

[3] J. GRAVER, B. SERVATIUS, AND H. SERVATIUS, Combinatorial Rigidity, AMS Graduate Studies in Mathematics Vol. 2, 1993.

[4] L. HENNEBERG, Die graphische Statik der starren Systeme, Leipzig 1911.

[5] D.J. JACOBS, A.J. RADER, L.A. KUHN, AND M.F. THORPE, Protein flexibility predictions using graph theory, *Proteins: Structure, Function, and Genetics* 44:150-165 (2001).

[6] B. JACKSON AND T. JORDÁN, The Dress conjectures on rank in the 3-dimensional rigidity matroid, to appear in *Advances in Applied Mathematics*. (See also EGRES TR-2003-04. www.cs.elte.hu/egres/.)

[7] B. JACKSON AND T. JORDÁN, The d-dimensional rigidity matroid of sparse graphs, to appear in *J. Combinatorial Theory, Ser. B.* (See also EGRES TR-2003-06. www.cs.elte.hu/egres/.)

[8] G. LAMAN, On graphs and rigidity of plane skeletal structures, *J. Engineering Math.* 4 (1970), 331-340.

[9] L. LOVÁSZ AND Y. YEMINI, On generic rigidity in the plane, *SIAM J. Algebraic Discrete Methods* 3 (1982), no. 1, 91–98.

[10] J.T. SCHWARTZ, Fast probabilistic algorithms for verification of polynomial identities, *J. ACM* 27, 701-717, 1980.

[11] T.-S. TAY, On the generic rigidity of bar frameworks, *Advances in Applied Mathematics*, 23, (1999), 14-28.

[12] T.-S. TAY, On generically dependent bar frameworks in space, *Structural Topology*, 20, (1993), 27-48.

[13] T.-S. TAY AND W. WHITELEY, Recent advances in the generic rigidity of structures, *Structural Topology* 9, 1984, pp. 31-38.

[14] W. WHITELEY, Some matroids from discrete applied geometry, in Matroid theory (J.E. Bonin, J.G. Oxley and B. Servatius eds., Seattle, WA, 1995), Contemp. Math., 197, Amer. Math. Soc., Providence, RI, 1996, 171–311.

Sign-Solvable Linear Programs

NAONORI KAKIMURA

Department of Mathematical Informatics
The University of Tokyo
Tokyo, 113-8656, Japan.
naonori_kakimura@mist.i.u-tokyo.ac.jp

SATORU IWATA

Department of Mathematical Informatics
The University of Tokyo
Tokyo, 113-8656, Japan.
iwata@mist.i.u-tokyo.ac.jp

In this paper, we are concerned with obtaining the sign of an optimal solution by the sign patterns of a linear program. We give a sufficient condition of the sign patterns of linear programs for which the simplex method works, that is, for which the pivoting operation can be computed by the sign patterns. For this purpose, we introduce a class of matrices, called totally sign-nonsingular matrices. Furthermore, linear programs satisfying this condition can be solved in strongly polynomial time.

Keywords: qualitative matrix theory, linear programming, sign-nonsingular matrices

1 Introduction

Consider a linear program described by $\mathrm{LP}(A,b,c) = \max\{cx \mid Ax = b,\ x \geq 0\}$. In this paper, we are concerned with solving the linear program qualitatively. Namely, we aim at classifying the properties of the set of optimal solutions determined uniquely by the sign patterns of A, b, and c.

The *sign* of a real number a is defined by $\operatorname{sgn} a = +1$ for $a > 0$, $\operatorname{sgn} a = 0$ for $a = 0$, and $\operatorname{sgn} a = -1$ for $a < 0$. The *sign pattern* of a real matrix A is the $\{+1, 0, -1\}$-matrix obtained from A by replacing each entry by its sign. For a matrix A, we denote by $\mathcal{Q}(A)$ the set of all matrices having the same sign pattern as A, which is called the *qualitative class* of A. For a vector b, the qualitative class $\mathcal{Q}(b)$ is defined in a similar way. Our purpose of this paper is to decide whether the set of the sign patterns of optimal solutions to $\mathrm{LP}(\widetilde{A}, \widetilde{b}, \widetilde{c})$ for each $\widetilde{A} \in \mathcal{Q}(A)$, $\widetilde{b} \in \mathcal{Q}(b)$ and $\widetilde{c} \in \mathcal{Q}(c)$ is the same or not, and, if it is, to compute the sign pattern of an optimal solution by given sign patterns. If we can obtain the sign pattern of an optimal basic solution, its exact value can easily be computed.

For a linear system $Ax = b$ with a real matrix A and a real vector b, the properties of the sign patterns of A and b have been investigated. Klee, Ladner and Manber [8] examined the linear system which can be solved independently of the absolute values of the entries of A and b, called a *sign-solvable* linear system. They showed that it is NP-complete to decide whether a rectangular matrix is not row full-rank independently of the absolute values. The study of sign-solvable linear systems is compiled in the book by Brualdi and Shader [2].

To solve a linear program, the simplex method is the most prominent algorithm (see [3]). The idea of the simplex method is to obtain an optimal basic solution by repeating pivoting operations. It is known that there are some pivoting rules which take care of the sign patterns in each iterations. Such pivoting rules are called *combinatorial pivoting rules*. Some of combinatorial pivoting rules, such as Bland's minimal index rule [1] and the criss-cross method [12], make the simplex method terminate in finite number of iterations. See Terlaky and Zhang [11] for a survey on pivoting rules.

In this paper, we give a sufficient condition of the sign pattern of linear programs for which the simplex method works, that is, the sign pattern in each iteration of the simplex method is determined uniquely. For this purpose, we introduce totally sign-nonsingular matrices. An $m \times n$ *totally sign-nonsingular* matrix with $m \leq n$ is a matrix satisfying that the sign of the determinant of each submatrix with size m is determined uniquely by the sign pattern. We prove that it can be tested in polynomial time to decide whether a given matrix is totally sign-nonsingular or not. If a linear program satisfies this condition, the sign pattern of optimal solutions can be obtained in strongly polynomial time by the ellipsoid methods [6], or the interior point method [5].

This paper is organized as follows. In Section 2, we give some definitions needed later, term-nonsingular, sign-nonsingular, totally sign-nonsingular, and so on. Then we show that the total sign-nonsingularity can be recognized in polynomial time. In section 3, we give a sufficient condition for the sign pattern of an optimal solution to be determined uniquely by given sign patterns. Finally, in Section 4, we design a combinatorial pivoting algorithm for totally sign-nonsingular matrices.

2 Totally Sign-Nonsingular Matrices

For a matrix A, the *row index set* and the *column index set* are denoted by R and C, i.e., $A = (a_{ij} \mid i \in R, j \in C)$. For $I \subseteq R$ and $J \subseteq C$, $A[I,J] = (a_{ij} \mid i \in I, j \in J)$ means the submatrix of A with row set I and column set J. A submatrix $A[R,J]$ is simply

denoted by $A[J]$. For a square matrix A of order n, the *determinant* of A is defined by

$$\det A = \sum_{\pi \in \mathscr{S}_n} \operatorname{sgn} \pi \prod_{i=1}^{n} a_{i\pi(i)},$$

where $\mathscr{S}_n$ denotes the set of all the permutations of order n, and $\operatorname{sgn} \pi \in \{+1, -1\}$ is the signature of the permutation $\pi \in \mathscr{S}_n$. A square matrix is said to be *nonsingular* if its determinant is distinct from zero. A matrix A is said to be *term-nonsingular* if $\det A$ contains at least one nonvanishing term, that is, if $a_{i\pi(i)} \neq 0$ $(\forall i \in R)$ for some permutation $\pi \in \mathscr{S}_n$. We say a matrix A is *term-singular* if A is not term-nonsingular. Obviously, nonsingularity implies term-nonsingularity, since one of expansion terms of $\det A$ is distinct from zero only if the summation contains a nonzero term. The *term-rank* of A is equal to the maximum size of a term-nonsingular submatrix of A, which is denote by t-rank$A$. It is easily deduced that we have t-rank$A \geq$ rank$A$.

A square matrix A is said to be *sign-nonsingular* if each $\widetilde{A} \in \mathscr{Q}(A)$ is nonsingular. We say that a matrix A has the *equisignum* determinant if every expansion term of $\det A$ has the same sign. A matrix A is sign-nonsingular if and only if A has the equisignum determinant. It is known that it can be tested in $O(n^3)$ time whether a given square matrix of order n is sign-nonsingular or not [9, 10] (see also Section 4).

We say that an $m \times n$ matrix A is *totally sign-nonsingular* if every term-nonsingular submatrix of order m is sign-nonsingular. An $m \times n$ totally sign-nonsingular matrix is said to have a *signed mth compound* by Brualdi and Shader [2]. Kim and Shader [7] proved that a matrix A is totally sign-nonsingular if and only if the set of sign patterns obtained from the kernel of $\widetilde{A}$ is the same as that of A for each $\widetilde{A} \in \mathscr{Q}(A)$.

We show that the computational complexity of the problem to decide whether a given $m \times n$ matrix is totally sign-nonsingular or not is equivalent to the decision problem of recognizing sign-nonsingular matrices.

Theorem 1 *It can be tested in* $O(n^3)$ *time whether a given* $m \times n$ *matrix* A *is totally sign-nonsingular or not.*

PROOF: Let $\widetilde{A}$ and $\widehat{A}$ be distinct matrices in $\mathscr{Q}(A)$. Consider the following square matrix P of order $m+n$:

$$P = \begin{pmatrix} O & \widetilde{A} \\ \widehat{A}^{\top} & I \end{pmatrix},$$

where I is the identity matrix of order n. The column of P is indexed by $R \cup C$. We will show that A is totally sign-nonsingular if and only if P is sign-nonsingular. Indeed, we have

$$\det P = -\det(\widetilde{A}\widehat{A}^{\top}) = -\sum_{\substack{J \subseteq C, \\ |J|=m}} (\det \widetilde{A}[J])(\det \widehat{A}[J]). \tag{1}$$

It follows from (1) that if A is totally sign-nonsingular, P is sign-nonsingular, since every expansion term of $\det P$ occurs on the right-hand side of (1) (Take entries of the identity matrix for the remaining indices $C \setminus J$). Assume that A is not totally sign-nonsingular. Then, since A does not have the equisignum determinant, there exists $J \subseteq C$ such that $\det A[J]$ has at least two expansion terms with different signs. Let π and π' be bijections from R onto J corresponding these terms, that is, $\operatorname{sgn} \pi \prod_{i \in R} \operatorname{sgn} a_{i\pi(i)} \neq \operatorname{sgn} \pi' \prod_{i \in R} \operatorname{sgn} a_{i\pi'(i)}$. Permutation σ over index set of P is defined by $\sigma(i) = \pi(i)$ for $i \in R$, $\sigma(i) = \pi^{-1}(i)$ for $i \in J$, and $\sigma(i) = i$ for $i \in C \setminus J$. Permutation σ' is also defined by $\sigma'(i) = \pi'(i)$ for $i \in J$, and $\sigma'(i) = \sigma(i)$ for the other indices. Then, by $\operatorname{sgn} \pi \prod_{i \in R} \operatorname{sgn} a_{i\pi(i)} \neq \operatorname{sgn} \pi' \prod_{i \in R} \operatorname{sgn} a_{i\pi'(i)}$, we have $\operatorname{sgn} \sigma \prod \operatorname{sgn} a_{i\sigma(i)} \neq \operatorname{sgn} \sigma' \prod \operatorname{sgn} a_{i\sigma'(i)}$. Therefore, P is not sign-nonsingular. □

3 Linear Programs with Sign Patterns

Consider a linear program $\mathrm{LP}(A, b, c)$ described as follows:

$$\begin{aligned} \mathrm{LP}(A, b, c) \quad & \text{maximize} && cx \\ & \text{sub. to} && Ax = b, \\ & && x \geq 0, \end{aligned} \tag{2}$$

where A is an $m \times n$ real matrix with row index set R and column index set C, b is a column vector of order m and c is a row vector of order n. If A, b, c are implicit, we simply denote this by LP. In this section, if there is no ambiguity, we also use $A_J = A[J]$ and $x_J = x[J]$. We say that x is a *feasible solution* of LP if x satisfies all the constraints in LP. We say x' is an *optimal solution* if $cx' = \max\{cx \mid x : \text{ feasible}\}$, and the objective value is called the *optimal value* of LP. We say LP is *solvable* if LP has a feasible solution and bounded. A *basis* B is the index subset such that $|B| = m$, and A_B is nonsingular. In this paper, we assume that there exists at least one basis in $\mathrm{LP}(A, b, c)$.

For any basis B, we can transform $\mathrm{LP}(A,b,c)$ to

$$\begin{array}{ll}\text{maximize} & c_B A_B^{-1} b + (c_N - c_B A_B^{-1} A_N) x_N \\ \text{sub. to} & x_B = A_B^{-1} b - A_B^{-1} A_N x_N, \\ & x \geq 0,\end{array}$$

where $N := C \setminus B$, the *non-basis*. This linear program is referred to as a *dictionary*. With two distinct indices f, g, the *coefficient* matrix $\overline{A} = (\overline{a}_{ij})$ of a dictionary is defined by the matrix with row set $B \cup \{f\}$ and column set $N \cup \{g\}$ as follows:

$$\overline{a}_{ij} = \begin{cases} -c_B A_B^{-1} b & \text{if } i = f, j = g, \\ -(c_N - c_B A_B^{-1} A_N)_j & \text{if } i = f, j \in N, \\ -(A_B^{-1} b)_i & \text{if } i \in B, j = g, \\ (A_B^{-1} A_N)_{ij} & \text{if } i \in B, j \in N. \end{cases} \tag{3}$$

We often identify a dictionary with the tableau shown in Figures 1 and 2. For a dictionary, the *basic* solution x is defined by $x_i = 0$ for $i \in N$ and $x_i = -\overline{a}_{ig}$ for $i \in B$. A basic solution satisfies $Ax = b$, and the objective value is $-\overline{a}_{fg}$.

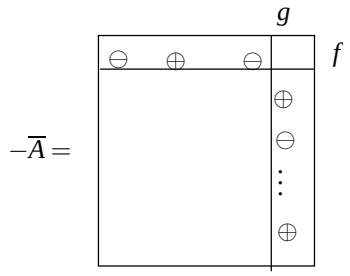


Figure 1: Representation of a dictionary.

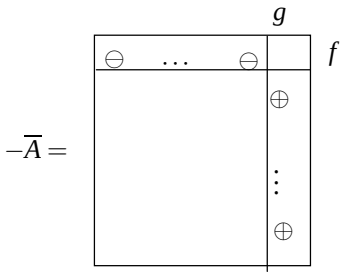


Figure 2: An optimal dictionary.

A dictionary is said to be *optimal* if it satisfies $-\overline{a}_{ig} \geq 0$ for all $i \in B$ and $-\overline{a}_{fj} \leq 0$ for all $j \in N$. If a dictionary is optimal, then the basic solution is an optimal basic solution. A dictionary is said to be *primal infeasible* if there exists an index $r \in B$ satisfying the condition that $-\overline{a}_{rg} < 0$ and $-\overline{a}_{rj} \leq 0$ for all $j \in N$. A dictionary is said to be *dual infeasible* if there exists an index $s \in N$ satisfying the condition that $-\overline{a}_{fs} > 0$ and $-\overline{a}_{is} \geq 0$ for all $i \in B$. If a dictionary is primal or dual infeasible, the linear program is infeasible or unbounded, respectively.

We say a linear program is *sign-solvable* if it is solvable and the set of the sign patterns of optimal solutions of $\mathrm{LP}(\widetilde{A}, \widetilde{b}, \widetilde{c})$ is the same as that of $\mathrm{LP}(A,b,c)$ for every $\widetilde{A} \in \mathcal{Q}(A)$, $\widetilde{b} \in \mathcal{Q}(b)$ and $\widetilde{c} \in \mathcal{Q}(c)$. We give an example of a sign-solvable linear program.

Example 2 *Consider the following linear program of three linear equations in six unknowns:*

$$\begin{array}{ll}\text{maximize} & \begin{pmatrix} 0 & 0 & c_1 & 0 & 0 & -c_2 \end{pmatrix} \quad x \\ \text{sub. to} & \begin{pmatrix} a_1 & 0 & 0 & a_2 & 0 & 0 \\ -a_3 & a_4 & 0 & a_5 & a_6 & 0 \\ 0 & a_7 & -a_8 & 0 & -a_9 & -a_{10} \end{pmatrix} x = \begin{pmatrix} b_1 \\ 0 \\ 0 \end{pmatrix}, \\ & x \geq 0,\end{array} \tag{4}$$

where $c_i\,(i = 1,2)$, $a_j\,(j = 1,\ldots,10)$ and b_1 are positive constants. Let $B = \{1,2,3\}$ be a basis. Then the sign pattern of the dictionary is determined uniquely by the sign patterns:

$$-\overline{A} = \begin{pmatrix} - & - & - & + \\ - & 0 & 0 & + \\ - & - & 0 & + \\ - & - & - & + \end{pmatrix}.$$

The sign pattern of this coefficient matrix implies that the dictionary is optimal. Hence, the sign of an optimal solution is

$$x = (+\ +\ +\ 0\ 0\ 0).$$

The sign pattern of the optimal basic solution can be obtained independently of the absolute values of the matrix entries. Thus, this linear program is sign-solvable. Moreover, in this case, the sign of the optimal value is positive, determined uniquely by the sign pattern.

We examine when the sign pattern of any dictionary in LP is determined uniquely by the sign pattern of LP, which is a sufficient condition for LP to be sign-solvable. We say that a dictionary is *sign-admissible* if the signs of $\overline{a}_{fj}$, $\overline{a}_{gi}$ and $\overline{a}_{ij}$ for all $i \in N$ and $j \in B$ are determined uniquely by the sign patterns of A, b and c. Then we obtain the following theorem.

Theorem 3 *Any dictionary of LP(A,b,c) is sign-admissible if and only if both $(A\ b)$ and $\begin{pmatrix} c \\ A \end{pmatrix}$ are totally sign-nonsingular. Hence, if both $(A\ b)$ and $\begin{pmatrix} c \\ A \end{pmatrix}$ are totally sign-nonsingular, LP is sign-solvable.*

PROOF: We first show that if both $(A\ b)$ and $\begin{pmatrix} c \\ A \end{pmatrix}$ are totally sign-nonsingular, any dictionary is determined uniquely by the sign patterns.

Let B be a basis, and $\overline{A} = (\overline{a}_{ij} \mid i \in B \cup \{f\}, j \in N \cup \{g\})$ be the coefficient matrix. Because of $\overline{a}_{ig} = (A_B^{-1}b)_i$ and $\overline{a}_{ij} = (A_B^{-1}a_j)_i$ for each $i \in B$ and $j \in N$ (see (3)), $\overline{a}_{ig}$ and $\overline{a}_{ij}$ are regarded as the solution of the following linear equations, respectively:

$$A_B y = b, \text{ and } A_B y = a_j, \ \forall j \in N.$$

Then, by Cramer's rule, we have

$$\overline{a}_{ig} = \frac{\det A[B-i+g]}{\det A[B]}, \ \forall i \in B, \tag{5}$$

$$\overline{a}_{ij} = \frac{\det A[B-i+j]}{\det A[B]}, \ \forall i \in B, \forall j \in N, \tag{6}$$

where $B-i+j$ means $B \setminus \{i\} \cup \{j\}$ in which the index j is put at the position of the index i in B, and b is indexed by g. Since $(A\ b)$ is totally sign-nonsingular, the sign of each entry $\overline{a}_{ij}$ for $i \in B$, $j \in N \cup \{g\}$ is determined uniquely by the sign patterns of A and b. We next consider the signs of $\overline{a}_{fj}$ for all $j \in N$. Since we have $\overline{a}_{fj} = c_N - c_B A_B^{-1} A_N$ for all $j \in N$ by (3), it follows from Schur complement that the sign of $\overline{a}_{fj}$ is equal to the sign of

$$\det \begin{pmatrix} c_j & c_B \\ a_j & A_B \end{pmatrix} = \det \begin{pmatrix} c_j - c_B A_B^{-1} a_j & 0 \\ 0 & I \end{pmatrix}, \tag{7}$$

where I is the identity matrix. Since $\begin{pmatrix} c \\ A \end{pmatrix}$ is totally sign-nonsingular, (7) is sign-nonsingular, or term-singular. Thus, if both $(A\ b)$ and $\begin{pmatrix} c \\ A \end{pmatrix}$ are totally sign-nonsingular, the sign pattern of the coefficient matrix except $\overline{a}_{fg}$ is determined uniquely by the sign pattens of A, b and c.

Conversely, assume that $(A\ b)$ or $\begin{pmatrix} c \\ A \end{pmatrix}$ is not totally sign-nonsingular. We first suppose $(A\ b)$ is not totally sign-nonsingular. Then there exists $J \subseteq C \cup \{g\}$ such that $A[J]$ is term-nonsingular, but not sign-nonsingular. If $g \notin J$, then the set of bases in A can be changed by the magnitude of the entries. Hence the dictionary with respect to J can be changed. If $g \in J$, since we assume A has at least one basis B, the coefficient matrix with respect to B has the entry $\det A[J]/\det A[B]$ by (5). Hence, the sign pattern of the coefficient matrix with respect to B can be changed. Next assume that $(A\ b)$ is totally sign-nonsingular, but $\begin{pmatrix} c \\ A \end{pmatrix}$ is not. Notice that, by the total sign-nonsingularity of $(A\ b)$, A is also totally sign-nonsingular. By the assumption, there exists $J \subseteq C$ such that $\begin{pmatrix} c \\ A \end{pmatrix}[J]$ is term-nonsingular, but not sign-nonsingular. Then, term-nonsingularity implies that A has a basis $B \subseteq J$, and hence the coefficient matrix with respect to B has the entry $\det \begin{pmatrix} c \\ A \end{pmatrix}[J]$ by (7). Thus there exists a dictionary such that it is not sign-admissible.

Furthermore, since an optimal solution is a convex combination of optimal basic solutions which are nonnegative, if both $(A\ b)$ and $\begin{pmatrix} c \\ A \end{pmatrix}$ are totally sign-nonsingular, the sign of every optimal solution is determined uniquely by the sign patterns. Thus, LP is sign-solvable. □

Notice that the sign of the optimal value may not be determined uniquely by the sign patterns, even if both $(A\ b)$ and $\begin{pmatrix} c \\ A \end{pmatrix}$ are totally sign-nonsingular. Indeed, let B^* be an optimal basis, and $N^* = C \setminus B^*$. Then the optimal value is determined uniquely by the sign pattern if and only if

$$\begin{pmatrix} 0 & c_{N^*} \\ b_{B^*} & A_{B^*} \end{pmatrix}$$

is sign-nonsingular.

Theorem 3 implies that if both $(A\ b)$ and $\begin{pmatrix} c \\ A \end{pmatrix}$ are totally sign-nonsingular, we can solve LP by the simplex method using combinatorial pivoting rules that make the simplex method terminate in finite number of iterations. Moreover, in each iteration, we can design a combinatorial pivoting algorithm. We will discuss this algorithm in Section 4.

Furthermore, the linear program satisfying the condition of Theorem 3 can be solved in strongly polynomial time. By Theorem 3, it is sufficient to solve LP($\widetilde{A},\widetilde{b},\widetilde{c}$) for some $\widetilde{A} \in \mathscr{Q}(A)$, $\widetilde{b} \in \mathscr{Q}(b)$ and $\widetilde{c} \in \mathscr{Q}(c)$. Then take all the nonzero entries in $\widetilde{A}$, $\widetilde{b}$, and $\widetilde{c}$ as 1 or -1. Since the size of the numbers in $\widetilde{A}$, $\widetilde{b}$, and $\widetilde{c}$ is constant, it can be solved in strongly polynomial time by the ellipsoid or interior point method.

Corollary 4 *For a linear program* LP(A,b,c), *if both $(A\ b)$ and $\begin{pmatrix} c \\ A \end{pmatrix}$ are totally sign-nonsingular, an optimal solution can be obtained in strongly polynomial time.*

4 Pivoting Algorithm for Totally Sign-Nonsingular Matrices

In this section, we discuss pivoting operation for totally sign-nonsingular matrices. We show that the pivoting operation is equivalent to finding an alternating path in a bipartite graph. Most totally sign-nonsingular matrices are sparse, and hence pivoting can be computed more efficiently than Gaussian elimination. For that purpose, we first associate a matrix with a bipartite graph.

Let $G = (U,V;E)$ be a bipartite graph with vertex sets U,V and edge set $E \subseteq U \times V$. A *path* $P \subseteq E$ is a sequence of consecutive edges in a graph. A *circuit* $C \subseteq E$ is a path which ends at the vertex it begins. For an edge subset $F \subseteq E$, we denote by ∂F the set of all the end-vertices of edges in F, i.e., $\partial F := \{u,v \mid (u,v) \in F\}$. An edge subset M in G is called a *matching* if $2|M| = |\partial M|$, and a matching M is said to be a *perfect matching* if $\partial M = U \cup V$. With a matching M, we say a path or cycle P of G is *M-alternating* if the elements of P alternate between elements of M and elements of $E \setminus M$ along the path or cycle. A vertex v is said to be *covered* (with respect to M) if $v \in \partial M$. A vertex is *exposed* if the vertex is not covered. For edge subsets F_1 and F_2 of G, we denote by $F_1 \triangle F_2$ the symmetric difference between F_1 and F_2. Notice that for a matching M and an M-alternating path P, $M \triangle P$ is also a matching in G.

With a matrix A, we associate the bipartite graph $G(A) = (U,V;E)$ with vertex sets $U := \{u_i \mid i \in R\}$ and $V := \{v_j \mid j \in C\}$. The edge set E is given by $E := \{(u_i,v_j) \mid a_{ij} \neq 0, u_i \in U, v_j \in V\}$, that is, an edge of $G(A)$ represents a nonvanishing entry of A. Then a perfect matching in $G(A)$ corresponds to one nonvanishing term of $\det A$. Therefore, A is term-nonsingular if and only if $G(A)$ has a perfect matching. Furthermore, the term-rank of A is equal to the maximum size of a matching in $G(A)$.

We will associate sign-nonsingularity of a matrix A with a bipartite graph. Let $G = (U,V;E)$ be a bipartite graph. An edge subset $F \subseteq E$ is said to be *central* if the subgraph obtained from G by deleting the vertices ∂F has a perfect matching. For an orientation D of G, a circuit of even length is said to be *oddly oriented* (*evenly oriented*) in D if in traversing the circuit, an odd (even) number of its edges is directed in the direction of traversal. We say that an orientation D of G is a *Pfaffian orientation* if every central circuit of even length is oddly oriented. The directed graph D is simply called *Pfaffian*. It can be tested in polynomial time whether a given directed bipartite graph is Pfaffian or not [9, 10]. For a matrix A, the directed bipartite graph $D(A) = (U,V;E)$ is defined by the orientation of $G(A)$ such that each edge of $D(A)$ is oriented according to the sign pattern of A, that is, an edge $e = (u_i,v_j)$ is oriented from u_i to v_j for $a_{ij} > 0$ and from v_j to u_i for $a_{ij} < 0$. Then it is known that a square matrix A is sign-nonsingular if and only if $D(A)$ is Pfaffian.

Let A be a totally sign-nonsingular matrix A, B be a basis, and N be the non-basis. For $r \in B$ and $s \in N$, $B' = B \setminus \{r\} \cup \{s\}$ is a basis if and only if $\overline{a}_{rs} \neq 0$. We say that the operation to construct such B' from B is *pivoting on* (r,s).

Let $D(A) = (U,V;E)$ be the directed bipartite graph associated with a totally sign-nonsingular matrix A. For an index set $J \subseteq C$, the subgraph associated with the submatrix $A[J]$ is simply denoted by $D[J]$. Since A is totally sign-nonsingular, $B \subseteq C$ is a basis if and only if $D[B]$ is Pfaffian. Note that $D[B]$ has a perfect matching. Namely, $D[J]$ is either Pfaffian or has no perfect matching for each $J \subseteq C$ with $|J| = |U|$.

Lemma 5 *Let $D(A) = (U,V;E)$ be the directed bipartite graph associated with an $m \times n$ totally sign-nonsingular matrix A. Let B be a basis, and M be a perfect matching in $D[B]$. Then, $B \setminus \{s\} \cup \{t\}$ is a basis if and only if there exists an M-alternating path P from v_s to v_t.*

PROOF: Assume that $B \setminus \{s\} \cup \{t\}$ is a basis. Then there exists a perfect matching M' in $D[B \setminus \{s\} \cup \{t\}]$, and $M \cup M'$ consists of M-alternating paths and circuits. Since both M and M' cover all vertices in $\partial(M \cup M') \setminus \{v_s, v_t\}$ and $v_s \in \partial M$ and $v_t \in \partial M'$, $M \cup M'$ contains an M-alternating path from v_s to v_t.

Conversely, assume that there exists an M-alternating path P. Then consider the symmetric difference $M \triangle P$. The edge subset $M \triangle P$ is also a matching with size $|M|$, and covers the vertices indexed by $B \setminus \{s\} \cup \{t\}$. Hence $D[B \setminus \{s\} \cup \{t\}]$ has a matching with size $|U|$, and this implies $B \setminus \{s\} \cup \{t\}$ is a basis. □

Furthermore, the following theorem implies that the difference of the signs of $\det A[B]$ and $\det A[B-s+t]$ is obtained by the orientation of the alternating path, where $B-s+t$ means $B \setminus \{s\} \cup \{t\}$ in which the index t is put at the position of the s in B. We say that a path P is *central* if the subgraph obtained from D by deleting the vertices ∂P has a matching with size $|U \setminus \partial P|$. Clearly, alternating paths are central. We say that a central path of even length is *oddly* (*evenly*) *oriented* in the same way as an oddly (evenly) oriented circuit.

Theorem 6 *Let $D(A)$ be the directed bipartite graph associated with an $m \times n$ totally sign-nonsingular matrix A. Let B be a basis, M be a perfect matching in $D[B]$, and P be an M-alternating path from v_s to v_t ($s \in B, t \in N$). Then, P is oddly oriented if and only if* $\operatorname{sgn} \det A[B] = \operatorname{sgn} \det A[B-s+t]$.

PROOF: By Lemma 5, since $M' = M \triangle P$ is a perfect matching in $G[B-s+t]$, $B-s+t$ is also a basis in A. By the total sign-nonsingularity of A, to compare the signs of $\det A[B]$ and $\det A[B-s+t]$, it is sufficient to compare the signs of M and M'.

Let $2p$ be the length of P with an integer $p \geq 1$. Let the vertices $\partial P \cap U$ be indexed by $s_1, s_2, \ldots, s_p$ along P. Let $\pi : R \to \partial M \cap V$ be the bijection corresponding to M. Then M' corresponds to the bijection $\pi' : R \to \partial M' \cap V$ defined by $\pi'(s_k) = \pi(s_{k+1})$ for $k = 1, \ldots, p-1$, and $\pi'(i) = \pi(i)$ for the other indices $i \in R \setminus \partial M$. Since π' is a product of π and a cyclic permutation of length $p+1$, if p is even, the signs of π and π' are different, and if p is odd, the signs are the same. Examples of the M-alternating path P and the associated submatrix are described as Figures 3 and 4.

In traversing P from v_s to v_t, the number of edges in the direction of traversal is the sum of the number of negative entries in a_{ij} for $(u_i, v_j) \in M \cap P$ and positive entries in a_{ij} for $(u_i, v_j) \in M' \cap P$. Hence, if p is even, P is oddly oriented if and only if $\prod_{(u_i,v_j)\in M\cap P} a_{ij} = \prod_{(u_i,v_j)\in M'\cap P} a_{ij}$, and if p is odd, P is oddly oriented if and only if $\prod_{(u_i,v_j)\in M\cap P} a_{ij} = -\prod_{(u_i,v_j)\in M'\cap P} a_{ij}$.

Therefore, by $\operatorname{sgn} \det A[B] = \operatorname{sgn} \pi \prod_{i\in R} \operatorname{sgn} a_{i\pi(i)}$, it follows that $\operatorname{sgn} \det A[B] = \operatorname{sgn} \det A[B-s+t]$ if and only if P is oddly oriented. □

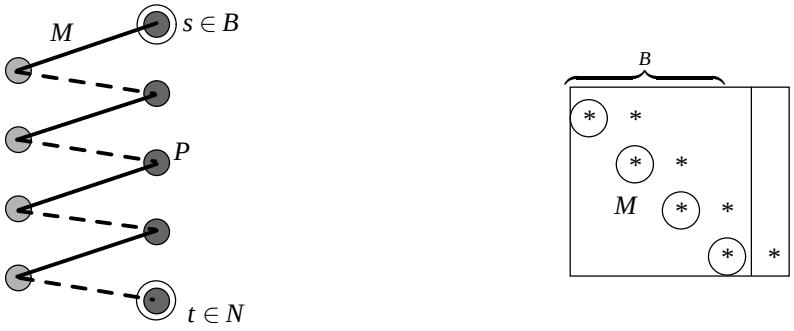


Figure 3: An M-alternating path of length $2p$ with an even integer $p \geq 2$.

Figure 4: An M-alternating path of length $2p$ with an odd integer $p \geq 1$.

We now return to solving $\mathrm{LP}(A, b, c)$. A matrix L is defined by

$$L = \begin{pmatrix} 1 & -c & 0 \\ 0 & A & -b \end{pmatrix}.$$

The column vector $\binom{1}{0}$ is indexed by f, and $\binom{0}{b}$ is indexed by g. For a basis B of A, $B \cup \{f\}$ is also a basis of L. Note that a linear program $\mathrm{LP}(A, b, c)$ is equivalent to

$$\begin{array}{ll} \text{maximize} & x_f \\ \text{sub. to} & Lx' = 0, \\ & x_g = 1, \\ & x_i \geq 0, \ \forall i \in C, \end{array}$$

where $x' = (x_f, x, x_g)^\top$.

Corollary 7 *Consider a linear program $LP(A, b, c)$ such that both $(A\ b)$ and $\binom{C}{A}$ are totally sign-nonsingular. Let B be a basis of the matrix L ($f \in B, g \notin B$). Let $D(L)$ be the directed bipartite graph associated with L, and M be a perfect matching in $D[B]$. Then, the coefficient matrix $\overline{A} = (\overline{a}_{ij})$ is obtained by*

$$\operatorname{sgn} \overline{a}_{ij} = \begin{cases} + & \text{if } P_{ij} \text{ is oddly oriented,} \\ - & \text{if } P_{ij} \text{ is evenly oriented,} \\ 0 & \text{if there is no } M\text{-alternating path } P_{ij}, \end{cases}$$

for $i, j \in C \cup \{f, g\}$, where P_{ij} is an M-alternating path from v_i to v_j.

PROOF: This easily follows from Theorem 6 and (5)–(7). □

By Corollary 7, we can obtain a dictionary in $O(m\gamma)$ time for an $m \times n$ matrix L with γ nonzero entries. Indeed, for a basis B, we can obtain a perfect matching M in $D[B \cup \{f\}]$ in $O(\sqrt{m}\gamma)$ time [4]. Since we can find an M-alternating path from v_i ($i \in B$) to all vertices in N by breadth first search, it requires $O(\gamma)$ time to obtain the signs of $\overline{a}_{it}$ for all $t \in N$. Hence it requires $O(m\gamma)$ time to obtain the signs of all entries of $\overline{A}$.

However, in each iteration of the simplex method, we do not need to compute a new matching and to obtain all signs in $\overline{A}$. It is sufficient to obtain the signs of $\overline{a}_{fj}$ for all $j \in N$, $\overline{a}_{ig}$ for all $i \in B$, and $\overline{a}_{rj}$ for all $j \in N$ or $\overline{a}_{is}$ for all $i \in B$ for pivoting on (r,s). Then a matching with respect to $B \setminus \{r\} \cup \{s\}$ can be constructed by the symmetric difference. Therefore, each iteration of the simplex method requires $O(\gamma)$ time. Since most totally sign-nonsingular matrices are sparse, if both $(A\ b)$ and $\binom{c}{A}$ are totally sign-nonsingular, each iteration of the simplex method can be computed more efficiently than Gaussian elimination.

We finally give a necessary and sufficient condition for sign-solvability.

Corollary 8 *Consider a linear program $LP(A,b,c)$ satisfying that both $(A\ b)$ and $\binom{c}{A}$ are totally sign-nonsingular. Let $D(L)$ be the directed bipartite graph associated with the matrix L. Then LP is sign-solvable if and only if there exists a basis B in L ($f \in B, g \notin B$) such that a perfect matching M in $D[B]$ satisfies the following conditions:*

- *For any $j \in N$, if there exists an M-alternating path from v_f to v_j, it is oddly oriented.*
- *For any $i \in B$, if there exists an M-alternating path from v_i to v_g, it is evenly oriented.*

PROOF: This follows from the sign pattern of an optimal dictionary. □

References

[1] R. G. Bland: New finite pivoting rules for the simplex method, *Mathematics of Operations Research*, 2, pp.103–107, 1977.

[2] R. A. Brualdi and B. L. Shader: *Matrices of Sign-solvable Linear Systems*, Cambridge University Press, Cambridge, 1995.

[3] G. B. Dantzig: *Linear Programming and Extensions*. Princeton University Press, Princeton N.J., 1963.

[4] J. E. Hopcroft and R. M. Karp: An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs, *SIAM Journal on Computing*, 2, pp. 225–231, 1973.

[5] N. Karmarkar: A new polynomial-time algorithm in linear programming, *Combinatorica*, 4, pp.373–395, 1984.

[6] L. G. Khachiyan: A polynomial time algorithm in linear programming, *Doklady Akademii Nauk SSSR*, 244, pp.1093–1096, 1979 (English translation: *Soviet Mathematics Doklady*, 20, pp.191–194, 1979).

[7] S.-J. Kim and B. L. Shader: Linear systems with signed solutions, *Linear Algebra and Its Applications*, 313, pp. 21–40, 2000.

[8] V. Klee, R. Ladner and R. Manber: Sign-solvability revisited, *Linear Algebra and Its Applications*, 59, pp. 131–158, 1984.

[9] W. McCuaig: Pólya's permanent problem, *The Electronic Journal of Combinatorics*, 11, # R79, 2004.

[10] N. Robertson, P. D. Seymour and R. Thomas: Permanents, Pfaffian orientations, and even directed circuits, *Annals of Mathematics*, 150, No. 3, pp. 929–975, 1999.

[11] T. Terlaky and S. Zhang: Pivot rules for linear programming — a survey, *Annals of Operations Research*, 46, pp. 203–233, 1993.

[12] S. Zionts: The criss-cross method for solving linear programming problems, *Management Science*, 15, pp.426–445, 1969.

Improved YBLM for Sperner families

JERROLD R. GRIGGS

Department of Mathematics, University of South Carolina,
Columbia SC 29208
griggs@math.sc.edu

GYULA O.H. KATONA

Alfréd Rényi Institute of Mathematics, HAS,
Budapest P.O.B. 127 H-1364 HUNGARY
ohkatona@renyi.hu

1 Introduction

Let $[n] = \{1,2,\dots,n\}$ and let $2^{[n]}$ denote the family of all subsets of $[n]$. If $\mathscr{F} \subseteq 2^{[n]}$ contains no distict members $F,G \in \mathscr{F}$ such that $F \subset G$, then $\mathscr{F}$ is called a *Sperner family*. It is well-known [5] that in this case $|\mathscr{F}| \le \binom{n}{\lfloor n/2 \rfloor}$. However a stronger statement is true, too.

Theorem 1 ([6], [2], [4], [3]) *If $\mathscr{F} \subseteq 2^{[n]}$ is a Sperner family then*

$$\sum_{F\in\mathscr{F}} \frac{1}{\binom{n}{|F|}} \le 1. \tag{1}$$

It is easy to see that (1) implies $|\mathscr{F}| \le \binom{n}{\lfloor n/2 \rfloor}$.

If $\mathscr{F}$ is an arbitrary family, let $\mathscr{F}_i$ denote the i-element members of $\mathscr{F}$, that is, $\mathscr{F}_i = \mathscr{F} \cap \binom{[n]}{i}$. Moreover, let $p_i = p_i(\mathscr{F}) = |\mathscr{F}_i|$. Then (1) can be written in the following form:

$$\sum_{i=0}^{n} \frac{p_i}{\binom{n}{i}} \le 1. \tag{2}$$

The $n+1$-dimensional vector $(p_0, p_1, \dots p_n)$ is called the *profile-vector* of $\mathscr{F}$. (2) expresses that the profile-vector of a Sperner family is "below" the hyperplane

$$\sum_{i=0}^{n} \frac{p_i}{\binom{n}{i}} = 1. \tag{3}$$

Christian Bey [1] has improved Theorem 1.1 in the following way.

Theorem 2 [1] *If $\mathscr{F} \subseteq 2^{[n]}$ is a Sperner family then*

$$\sum_{i=0}^{n} \frac{p_i}{\binom{n}{i}} + \sum_{1\le i_1<\dots<i_s\le n-1, s\ge 2} \left(\prod_{j=1}^{s-1} \frac{n(i_{j+1}-i_j)}{i_j(n-i_{j+1})}\right)\left(\prod_{j=1}^{s} \frac{p_{i_j}}{\binom{n}{i_j}}\right) \le 1. \tag{4}$$

The goal of the present paper is to give a sharper estimate on the profile-vectors of Sperner families. The exact formulation is rather complicated, we will give it only later. However let us sketch the way, our result will be described. We will determine many profile-vectors which are "the worst ones", that is, which are the "closest" to the hyperplane (4). These will be connected by hyperplanes and this surface will be the upper bound for all profile-vectors of Seprner families.

2 Tool: a new form of the shadow theorem

Let $\binom{[n]}{k}$ denote the family of all k-element subsets of $[n]$. If $\mathscr{F} \subseteq \binom{[n]}{k}$, $1 \le \ell < k$ then the *ℓ-shadow* of $\mathscr{F}$ is

$$\sigma_\ell(\mathscr{F}) = \{G:\ |G| = \ell \ \text{ and } \ G \subset F \ \text{ holds for some member } F \text{ of } \mathscr{F}\}.$$

Theorem 3 (Shadow Theorem). *If*

$$|\mathscr{F}| = \binom{a_k}{k} + \binom{a_{k-1}}{k-1} + \dots + \binom{a_t}{t} > 0$$

where $a_k > a_{k-1} > \dots > a_t \ge t \ge 1$ are integers (this expansion is known to exist and is unique), then

$$|\sigma_\ell(\mathscr{F})| \ge \binom{a_k}{\ell} + \binom{a_{k-1}}{\ell-1} + \dots + \binom{a_t}{t-k+\ell}. \tag{5}$$

This bound is sharp.

Theorem 4 (Lovász' Version). *Write $|\mathscr{F}|$ in the form of $\binom{x}{k}$ where $k \le x$ is a real number. Then*

$$|\sigma_\ell(\mathscr{F})| \ge \binom{x}{\ell}.$$

This estimate is sharp only for integer x's. On the other hand it is much easier to use.

Introducing the notation $L_{k,\ell}\left(\binom{x}{k}\right) = \binom{x}{\ell}$, the Lovász' version can be formulated in the following way.

$$|\sigma_\ell(\mathscr{F})| \ge L_{k,\ell}(|\mathscr{F}|). \tag{6}$$

In the interval $\binom{n-1}{\ell} < |\mathscr{F}| < \binom{n}{\ell}$ there is no equality in these estimates. If $n/2 < k$ then this interval is longer than the half of the total range. One would like to have an estimate which has an easy form, but gives equality at some places in this interval. The following theorem satifies these wishes.

Theorem 5 *Let $\mathscr{F} \subseteq \binom{[n]}{k}$, $1 \le \ell < k$. Then*

$$|\sigma_\ell(\mathscr{F})| \ge \binom{n}{\ell} - L_{n-k,n-\ell}\left(\binom{n}{k} - |\mathscr{F}|\right) = M_{k,\ell}(|\mathscr{F}|), \tag{7}$$

with equality only when

$$\binom{n}{k} - |\mathscr{F}| = \binom{y}{n-k}$$

holds with some integer y, that is when

$$|\mathscr{F}| = \binom{n-1}{k} + \binom{n-2}{k-1} + \ldots + \binom{n-i}{k-i+1}$$

holds for some $1 \le i = n - y \le k$.

These functions are much nicer than the exact shadow function, but they are still difficult to handle. Especially when multiple applications are needed. This is why we want to exploit the following lemma.

Lemma 6 *If $\ell < k$ then $L_{k,\ell}(u)$ is concave from below in the interval $(1,\infty)$.*

It is easy to see that this lemma implies that $L_{n-k,n-\ell}(u)$ is convex from below (since $n-k < n-\ell$). Hence $M_{k,\ell}(u)$ is also concave. The function $L_{k,\ell}(u)$ will be used from 1 to $\binom{n-1}{k}$ and $M_{k,\ell}(u)$ starting from this point. More exactly, a broken line defined by some integer points of this combined function.

First we define the function $T_{k,\ell}(u)$ at some values: $T_{k,\ell}(0) = 0$,

$$T_{k,\ell}\left(\binom{i}{k}\right) = \binom{i}{\ell} \ (k \le i \le n-1)$$

,

$$T_{k,\ell}\left(\binom{n-1}{k} + \binom{n-2}{k-1} + \ldots + \binom{n-i}{k-i+1}\right) =$$

$$\binom{n-1}{\ell} + \binom{n-2}{\ell-1} + \ldots + \binom{n-i}{\ell-i+1} \ (1 \le i \le k),$$

$$T_{k,\ell}\left(\binom{n}{k}\right) = \binom{n}{\ell}.$$

The graph of the function $T_{k,\ell}$ is obtained by connecting the neighboring points defined above by strait line segments. The so obtained broken line gives a fairly good and at the same time easily treatable estimate on the shadow.

Theorem 7

$$|\sigma(\mathscr{F})| \ge T_{k,\ell}(|\mathscr{F}|). \tag{8}$$

3 The result

In this section first we define the *corner points*. They are such profile-vectors which are the "closest" to the hyperplane (3) in a precisely undefined way. They will be formally given by their coordinates. The first type of corner points are given in the following way.

$$p_n = p_{n-1} = \ldots = p_{k+1} = 0,$$

$$p_k = \binom{a_k}{k},$$

$$p_{k-1} = \binom{a_k}{k-2} + \binom{a_k+1}{k-2} + \binom{a_k+2}{k-2} + \ldots + \binom{a_{k-1}-1}{k-2},$$

$$p_{k-2} = \binom{a_{k-1}}{k-3} + \binom{a_{k-1}+1}{k-3} + \binom{a_{k-1}+2}{k-3} + \ldots + \binom{a_{k-2}-1}{k-3},$$

$$\vdots$$

$$p_\ell = \binom{a_{\ell+1}}{\ell-1} + \binom{a_{\ell+1}+1}{\ell-1} + \ldots + \binom{n-2}{\ell-1} + \binom{n-2}{\ell-1} + \binom{n-3}{\ell-2} + \ldots + \binom{n-b_\ell}{\ell-b_\ell+1},$$

$$p_{\ell+1} = \binom{n-b_{\ell+1}-1}{\ell-b_{\ell+1}} + \binom{n-b_{\ell+1}-2}{\ell-b_{\ell+1}-1} + \ldots + \binom{n-b_{\ell+2}}{\ell-b_{\ell+2}+1}$$

$$\vdots$$

$$p_{m-1} = \binom{n-b_m-1}{\ell-b_m} + \binom{n-b_m-2}{\ell-b_m-1} + \ldots + \binom{n-b_{m+1}}{\ell-b_{m+1}+1}$$

$$p_m = \binom{n-b_{m+1}}{\ell-b_{m+1}}$$

$$p_{m+1} = \ldots = p_1 = p_0 = 0,$$

where $0 \le m < \ell < k \le n, k \le a_k \le a_{k-1} \le \ldots \le a_{\ell+1} \le n-2, 2 \le b_\ell \le b_{\ell+1} \le \ldots \le b_{m+1} \le \ell$. These horrible formulas can be understood easier with some helping remarks. Observe that $a_k, a_k+1, \ldots, a_{k-1}-1, a_{k-1}, a_{k-1}+1, \ldots, a_{\ell+1}, a_{\ell+1}+1, \ldots, n-2$ are all integers from a_k to $n-2$, in increasing order. This sequence is broken into parts by the a's; the lower parts of the binomial coefficients are chosen according to the parts. This sequece of binomial coefficients ends in the middle of p_ℓ. Here a new sequence starts: $n-2, n-3, \ldots, n-b_\ell, n-b_\ell-1, \ldots, n-b_{m+1}$. These are just the integers from $n-2$ to $n-b_{m+1}$ in decreasing order. This is broken into segments by the b's. The other two types of corner points are "special cases" of the first type. They are obtained when one of these two parts are missing.

Therefore the corner points of type 2 are given by the following formulas.

$$p_n = p_{n-1} = \ldots = p_{k+1} = 0,$$

$$p_k = \binom{a_k}{k},$$

$$p_{k-1} = \binom{a_k}{k-2} + \binom{a_k+1}{k-2} + \binom{a_k+2}{k-2} + \ldots + \binom{a_{k-1}-1}{k-2},$$

$$p_{k-2} = \binom{a_{k-1}}{k-3} + \binom{a_{k-1}+1}{k-3} + \binom{a_{k-1}+2}{k-3} + \ldots + \binom{a_{k-2}-1}{k-3},$$

$$\vdots$$

$$p_\ell = \binom{a_{\ell+1}}{\ell-1} + \binom{a_{\ell+1}+1}{\ell-1} + \ldots + \binom{n-2}{\ell-1} + \binom{n-1}{\ell-1},$$

$$p_{\ell+1} = \ldots = p_1 = p_0 = 0,$$

where $1 \le \ell < k \le n, k \le a_k \le a_{k-1} \le \ldots \le a_{\ell+1} \le n-2$.

The corner points of type 3 are given by the following formulas.

$$p_n = p_{n-1} = \ldots = p_{\ell-1} = 0,$$

$$p_\ell = \binom{n-1}{\ell} + \binom{n-2}{\ell-1} + \binom{n-2}{\ell-1} + \binom{n-3}{\ell-2} + \ldots + \binom{n-b_\ell}{\ell-b_\ell+1},$$

$$p_{\ell+1} = \binom{n-b_{\ell+1}-1}{\ell-b_{\ell+1}} + \binom{n-b_{\ell+1}-2}{\ell-b_{\ell+1}-1} + \ldots + \binom{n-b_{\ell+2}}{\ell-b_{\ell+2}+1}$$

$$\vdots$$

$$p_{m-1} = \binom{n-b_m-1}{\ell-b_m} + \binom{n-b_m-2}{\ell-b_m-1} + \ldots + \binom{n-b_{m+1}}{\ell-b_{m+1}+1}$$

$$p_m = \binom{n-b_{m+1}}{\ell-b_{m+1}}$$

$$p_{m+1} = \ldots = p_1 = p_0 = 0,$$

where $0 \le m < \ell < n, 2 \le b_\ell \le b_{\ell+1} \le \ldots \le b_{m+1} \le \ell$.

A certain linear extension of the set of corner points will replace the hyperplane (3) in the upper bound. This linear extension will be defined in the following rows. Take the convex hull of the set of corner points. It is easy to see that his $n+1$-dimensional convex body is bordered by the hyperplane given by (3) and some other ($n+1$-dimensional) facets determined by the corner points. The facet belonging to (3) is the positive part of the hyperplane, it cuts the positive octant into two parts. Therefore the same must be true for the union of the other facets. The one containing the origin is denoted by T_{cp}.

Theorem 8 *The profil-evectors of Sperner families are in T_{cp}.*

The proof is based on Theorem 2.5.

We are working on the more exact determination of the structure of the facets of T_{cp} which allows an algorithmic decision if a vector of integer coordinates is in T_{cp} or not.

References

[1] CH. BEY Polynomial LYM inequalities,

[2] B. BOLLOBÁS, On generalized graphs, *Acta Math. Acad. Sci. Hungar* **16**(1965), 447–452.

[3] D. LUBELL, A short proof of Sperner's lemma, *J. Combin. Theory* **1**(1966), 299.

[4] L.D. MESHALKIN, A generalization of Sperner's theorem on the number of a finite set (in Russian), *Teor. Verojatnost. i Primen.* **8**(1963), 219–220.

[5] E. SPERNER, Ein Satz über Untermegen einer endlichen Menge, *Math. Z.* **27**(1928), 544–548.

[6] K. YAMAMOTO, Logarithmic order of free distributive lattices, *J. Math. Soc. Japan* **6**(1954), 347–357.

Algorithmic aspects of Hadwiger's Conjecture*

KEN-ICHI KAWARABAYASHI*

Graduate School of Information Sciences (GSIS)
Tohoku University
Sendai, 980-8579, Japan
e-mail address :
k_keniti@dais.is.tohoku.ac.jp

Abstract: Hadwiger's Conjecture claims that any graph without K_k as a minor is $(k-1)$-colorable. It is well-known that the case with $k = 5$ is equivalent to

the Four Color Theorem. In 1993, Robertson, Seymour and Thomas [27] proved that the case $k = 6$ is also equivalent to the Four Color Theorem. For every $k \geq 7$, the conjecture is still open. It is not even known if there exists an absolute constant c such that any ck-chromatic graph has K_k as a minor. So far, it is known that there exists a constant c such that any $ck\sqrt{\log k}$-chromatic graph has K_k as a minor. So it would be of great interest to prove that a linear function of the chromatic number is sufficient to force K_k as a minor.

In this paper, we show that from an algorithmic point of view, we can "decide" this problem in polynomial time. Our main result is that for every fixed integer k, there is an algorithm with running time $O(n^3)$ for deciding either that

(1)) a given graph G is $27k$-colorable, or
(2) G contains K_k-minor, or
(3) G contains a minor H of bounded size which contains no K_k-minors and has no $27k$-colorings.

Note that if (3) holds, then H is a counterexample to Hadwiger's conjecture. In fact, this would be a counterexample to the weaker conjecture that any $27k$-chromatic graph has K_k as a minor. If the output is (1), then our algorithm also finds a coloring of G using at most $27k$ colors. In case (3), the corresponding minor H is found.

We also remark recent progress.

Keywords: Hadwiger's Conjecture, Polynomial time algorithm, Robertson-Seymour theory

1 Introduction

In this paper, all graphs are finite and simple. We follow standard graph theory terminology and notation as used, for example, in [7]. A graph H is a *minor* of a graph K if H can be obtained from a subgraph of K by contracting edges.

The theory of graph minors developed by Robertson and Seymour made considerable impact on the theory of algorithms. Tree-decompositions used in that theory and the corresponding notion of the tree-width have been devised independently by Arnborg and Proskurowski (cf. [3]), and have proved to be a useful tool in the design of algorithms.

Yet, some very basic theoretical and algorithmic problems about graph minors remain open. Our research is motivated by Hadwiger's Conjecture from 1943 which suggests a far-reaching generalization of the Four Color Theorem and is one of the most challenging open problems in graph theory.

Conjecture 1 (Hadwiger [10]) *For every $k \geq 1$, every graph with chromatic number at least k contains the complete graph K_k as a minor.*

For $k = 1,2,3$, this is easy to prove, and for $k = 4$, Hadwiger himself [10] and Dirac [8] proved it. For $k = 5$, however, it becomes extremely difficult. In 1937, Wagner [33] proved that the case $k = 5$ is equivalent to the Four Color Theorem. So, assuming the Four Color Theorem [1, 2, 28], the case $k = 5$ of Hadwiger's Conjecture holds. Robertson, Seymour and Thomas [27] proved that a minimal counterexample to the case $k = 6$ is a graph G that has a vertex v such that $G - v$ is planar. By the Four Color Theorem, this implies Hadwiger's Conjecture for $k = 6$. This result is the deepest in this research area. So far, the conjecture is open for every $k \geq 7$. For the case $k = 7$, Kawarabayashi and Toft [18] proved that any 7-chromatic

*This is joint work with Bojan Mohar, and partially joint work with Neil Robertson and Paul Seymour

*Research is supported by the Japan Society for the Promotion of Science for Young Scientists, by Japan Society for the Promotion of Science, Grant-in-Aid for Scientifi c Research, by Sumitomo Foundation and by Inoue Research Award for Young Scientists.

graph has K_7 or $K_{4,4}$ as a minor, and recently, Kawarabayashi [13] proved that any 7-chromatic graph has K_7 or $K_{3,5}$ as a minor.

It is even not known if there exists an absolute constant c such that any ck-chromatic graph has K_k as a minor. So far, it is known that there exists a constant c such that any $ck\sqrt{\log k}$-chromatic graph has K_k as a minor. This follows from the results in [30, 31, 20, 19]. This result was proved 25 years ago, but no one can improve the superlinear order $k\sqrt{\log k}$ of the bound on the chromatic number. So it would be of great interest to prove that a linear function of the chromatic number is sufficient to force K_k as a minor. We refer to [32] for further information on the Hadwiger Conjecture.

Although, it is still open if there exists a constant c such that any ck-chromatic graph contains K_k as a minor, we show in this paper that from an algorithmic point of view, we can "decide" this problem in polynomial time. Actually, the complexity is $O(n^3)$. Our main result is the following.

Theorem 2 *For every fixed k, there is an algorithm with running time $O(n^3)$ for deciding either that*

(1) *a given graph G of order n is $27k$-colorable, or*

(2) *G contains K_k-minor, or*

(3) *G contains a minor H of bounded size which does not contain a K_k-minor and has no $27k$-colorings.*

Let us remark the following:

(a) If (3) holds, then H is a counterexample to Hadwiger's conjecture. In fact, this would be a counterexample to the weaker conjecture that any $27k$-chromatic graph has K_k as a minor. The conclusion of Theorem 2(3) that such a minor has bounded number of vertices is an interesting theoretical outcome of the algorithm.

(b) If (1) holds, we can actually color the graph using at most $27k$ colors. If (3) holds, we can exhibit the minor H by means of a subgraph $\widetilde{H}$ of G whose contraction yields H.

(c) We need the result of [5], Theorem 3, but we do not need any result in Graph Minor series. However, the proof of Theorem 3 given in [5] depends on Robertson and Seymour's deep results, see [5, 24, 25].

To prove Theorem 2, we need the following result from [5].

Theorem 3 *For any k, there exists a constant $N(k)$ such that every $2k$-connected graph with minimum degree at least $\frac{31k}{2}$ and with at least $N(k)$ vertices contains K_k as a minor.*

Actually, the main result proved in [5] is stronger: For any integers k, s and n, there exists a constant $N(k,s,n)$ such that every $(3k+2)$-connected graph of minimum degree at least $\frac{31}{2}(k+1)-3$ and with at least $N(k,s,n)$ vertices either contains $K_{k,sn}$ as a topological minor or a minor isomorphic to s disjoint copies of $K_{k,n}$. It is necessary to include the possibility of having $K_{k,sn}$ as a subdivision since G could be a complete bipartite graph $K_{\frac{31}{2}(k+1)-3,N}$, where N could be arbitrarily large.

Let us conclude with a brief historical overview and background on graph minors. The study of graphs containing a given graph as a minor, or as a topological minor, goes back to the very beginnings of graph theory. Wagner and Mader studied the maximum size of graphs not having K_k as a (topological) minor. Wagner [34] showed that a sufficiently large chromatic number (which depends only on k) guarantees K_k as a minor, and Mader [21] showed that a sufficiently large average degree will do the same.

Later, Kostochka [20, 19] and Thomason [30] independently proved that $\Theta(k\sqrt{\log k})$ is the correct order of the average degree forcing K_k as a minor. Recently, Thomason [31] found the asymptotically best possible value of this "extremal" function.

These results show that if the minimum degree of given graph G is a linear function of k, then G does not necessarily contain a K_k-minor. It does not help even if we add a connectivity condition. Only the connectivity of order $\Theta(k\sqrt{\log k})$ forces the presence of K_k-minors. However, as Thomason [31] pointed out, extremal graphs are more or less exactly vertex disjoint unions of suitable dense random graphs.

Motivated by this question and the results stated above, Böhme et al. [5] proved Theorem 3 and its strengthening mentioned above. The proof in [5] uses the Excluded Minor Theorem from [24] and its improved version from [25]. The Excluded Minor Theorem is one of the deepest results in graph theory, and is used by Robertson and Seymour to prove Wagner's Conjecture [26] which says that for every infinite set of graphs, one is a minor of another.

Theorem 2 is also interesting from the point of approximation algorithms. It is well-known (cf. Feige and Kilian [9] and Hĺstad [11]) that approximating the chromatic number within a factor of $n^{1-\varepsilon}$ cannot be done in polynomial time for any $\varepsilon > 0$, unless coRP = NP. Our result suggests that there may be a fixed constant c such that for any minor closed family M of graphs which does not contain all graphs, there could be a polynomial time approximation algorithm with factor c for the chromatic number of graphs in M. Of course, our algorithm does not give the answer to the constant factor approximation coloring in minor-closed classes of graphs, and a constant factor approximation coloring does not yield our result since we do not know if $O(n)$ colors suffice to color K_n-minor-free graphs.

2 Lemmas

A graph L is said to be *k-linked* if it has at least $2k$ vertices and for any ordered k-tuples $(s_1,\dots,s_k)$ and $(t_1,\dots,t_k)$ of $2k$ distinct vertices of L, there exist pairwise disjoint paths $P_1,\dots,P_k$ such that for $i=1,\dots,k$, the path P_i connects s_i and t_i. Such collection of paths is called a *linkage* from $(s_1,\dots,s_k)$ to $(t_1,\dots,t_k)$.

An important tool is the following theorem due to Thomas and Wollan [29].

Theorem 4 *Every $2k$-connected graph G with at least $5k|V(G)|$ edges is k-linked.*

Theorem 4 implies that every $10k$-connected graph is k-linked. Bollobás and Thomason [6] proved that every $22k$-connected graph is k-linked, and Kawarabayashi, Kostochka and Yu [16] proved that every $12k$-connected graph is k-linked.

Let G be a graph and let A,B be subgraphs of G. We say that the pair (A,B) is a *separation* of G if $A\cup B=G$, $V(A)-V(B)\neq\emptyset$, and $V(B)-V(A)\neq\emptyset$. The *order* of a separation (A,B) is $|V(A)\cap V(B)|$.

The following result is a variation of an old theorem of Mader [22]. Its nonalgorithmic counterpart appeared in [5]. But it is easy to convert the proof into algorithm.

Theorem 5 *Let G be a graph and k an integer such that*

(a) $|V(G)|\geq\frac{5}{2}k$ *and*

(b) $|E(G)|\geq\frac{25}{4}k|V(G)|-\frac{25}{2}k^2$.

Then $|V(G)|\geq 10k+2$ and G contains a $2k$-connected subgraph H with at least $5k|V(H)|$ edges. If G has n vertices, then H can be found in time $O(n^3)$.

We also need the following lemma, Lemma 6. Before we state that, we need some definitions.

Let (A,B) be a separation in a graph G and let $S=V(A)\cap V(B)$. Let $S_1\cup\cdots\cup S_r$ be a partition of S. Then we say that A can be *contracted* to $S_1,\dots,S_r$ if A contains pairwise disjoint connected subgraphs $T_1,\dots,T_r$ such that $S_i\subseteq V(T_i)$ for $i=1,\dots,r$.

The following lemma was originally proved in [14], but for completeness, we include its proof.

Lemma 6 *Let G be a graph with minimum degree d. Let (A,B) be a separation in G of minimum order, let $S=V(A)\cap V(B)$ and $s=|S|$. If $s\leq\lfloor\frac{2d}{27}\rfloor$, then for every partition $S_1\cup\cdots\cup S_r$ of S, A (and B) can be contracted to $S_1,\dots,S_r$. The corresponding disjoint connected subgraphs $T_1,\dots,T_r$ in A (resp. B) can be found in $O(n^3)$ time, where $n=|V(G)|$.*

3 Description of the algorithm

In this section, we will describe the algorithm of Theorem 2. Before that, we need some definitions. A *tree decomposition* of a graph G is a pair (T,Y), where T is a tree and Y is a family $\{Y_t\mid t\in V(T)\}$ of vertex sets $Y_t\subseteq V(G)$, such that the following two properties hold:

(W1) $\bigcup_{t\in V(T)}Y_t=V(G)$, and every edge of G has both ends in some Y_t.

(W2) If $t,t',t''\in V(T)$ and t' lies on the path in T between t and t'', then $Y_t\cap Y_{t''}\subseteq Y_{t'}$.

The *width* of a tree decomposition (T,Y) is $\max_{t\in V(T)}(|Y_t|-1)$.

Now we are ready to describe our algorithm.

Algorithm for Theorem 2

Input: A graph G.

Output: As described in Theorem 2.

Running time: $O(f(k)n^3)$ for some function $f:\mathbb{N}\to\mathbb{N}$.

Description:

Step 1. If G has a vertex of degree at most $27k-1$, then we delete it. We continue this procedure until there are no vertices of degree at most $27k-1$. This can be done in linear time. Let G' be the resulting graph. Proceed to Step 2.

Step 2. Test if the tree-width of G' is small or not, say smaller than some value $g(k)$. For simplicity in later steps, we assume that $g(k)\geq N(k)$, where $N(k)$ is as in Theorem 3. This can be done in linear time by the algorithm of Bodlaender [4]. If the tree-width is at least $g(k)$, then go to Step 3. Otherwise, we use the linear-time algorithm of Arnborg and Proskurowski [3] to color G'. If G' can be colored by at most $27k$ colors, then we color $G-G'$ greedily, and output the coloring of G. If G'

cannot be colored with $27k$ colors, then we check if G' contains a K_k-minor. Again, this can be done by using the algorithm of Arnborg and Proskurowski [3] (or the algorithm of Robertson and Seymour [23]). If G' contains K_k as a minor, then we output that G contains K_k as a minor. If G' does not contain K_k as a minor, then we proceed as argued below. The whole process up to this point can be done in linear time.

Let (T,Y) be the corresponding tree-decomposition found above. The dynamic programming approach of Arnborg and Proskurowski assumes that T is a rooted tree whose edges are directed away from the root. For $tt' \in E(T)$ (where t is closer to the root than t'), define $S(t,t') = Y_t \cap Y_{t'}$ and $G'(t,t')$ be the induced subgraph of G' on vertices $\cup Y_s$, where the union runs over all nodes of T that are in the component of $T - tt'$ that does not contain the root. The algorithm of Arnborg and Proskurowski starts at all leaves of T and computes, for every $tt' \in E(T)$, the set $C(t,t')$ of all $27k$-colorings of $S(t,t')$ which can be extended to the whole $G'(t,t')$. If T has a vertex t of very large degree, then two neighbors t' and t'' have $S(t,t') = S(t,t'')$ and $C(t,t') = C(t,t'')$. Then $G'(t,t')$ can be deleted, and we still have a graph of bounded tree-width without K_k minor and without $27k$-colorings.

If all vertices of T have bounded degree, then T has a long path and there are distinct edges t_1t_1' and t_2t_2' on this path (where the second one is further from the root) such that $|S(t,t')| = |S(t,t'')|$ and $C(t,t') = C(t,t'')$. In the same way as argued in [5], we may assume that there are are $|S(t,t')|$ disjoint paths joining $S(t,t')$ and $S(t,t'')$. By contracting these paths and replacing $G'(t,t')$ with $G'(t,t'')$, we get a minor of G' which is still of bounded tree-width, without K_k minor, and without $27k$-colorings. Repeating this, we eventually end up with the desired minor of G' of bounded size. (The bound is actually a doubly exponential value expressed in terms of k. More details on this part will be given in the full paper.)

Step 3. Test whether G' is $2k$-connected or not. Suppose first that G' is $2k$-connected. By the assumption in Step 2, we have $|G'| \geq g(k) \geq N(k)$. It follows from Theorem 3 that G' contains K_k as a minor. So, we output that G has K_k as a minor.

If G' is not $2k$-connected, then go to Step 4.

Step 4. G' is not $2k$-connected; detect a minimal separation (A,B). This can be done in polynomial time by standard methods. The best known algorithm is that of Henzinger, Rao, and Gabow [12] which needs $O(n^2)$ time for this task.

Let $S = V(A) \cap V(B)$. Then $|S| < 2k$. Let A_1' be a component of $G' - S$ and $A_2' = G' - A_1' - S$. Let S_1 be a maximal independent set in the subgraph $G'(S)$ of G' induced on S. Let S_i be a maximal independent set in $G'(S) - \bigcup_{l=1}^{i-1} S_l$, for $i = 2,3,\ldots$. Maximal independent sets can be found greedily or by means of any other method in constant time (since $|S| < 2k$). Next, we identify every nonempty set S_i into one vertex s_i ($i = 1,\ldots,r$). Then the resulting graph on $S' = \{s_1,\ldots,s_r\}$ is a clique. Let A_1, A_2 be the corresponding graphs obtained from A_1', A_2' by adding the clique S' and the corresponding edges between A_i' and S'.

Finally, we test A_1, A_2 (recursively), starting from Step 1. If both graphs A_1, A_2 have $27k$-colorings, they give rise to a $27k$-coloring of their union since S' is a clique. Since vertex sets S_i ($i = 1,\ldots,r$), that were identified into single vertices s_i in S', are independent in G', this coloring gives rise to a coloring of G'. Of course, we can extend this coloring of G' to G.

If one of the graphs, say A_1, contains K_k-minor, then we obtain a K_k-minor in G' (after contracting A_2 onto S') by using Lemma 6 with $d = 27k$. Similarly, if outcome (3) appears for A_1, we get the same outcome for G' by using a contraction of A_2 onto S'.

This algorithm stops when either the tree-width of the current graph is small or the current graph is $2k$-connected with minimum degree at least $27k$.

Now we shall estimate time complexity of the algorithm. All steps except the application of Lemma 6 can be done in time proportional to n^2. Another factor of n pops up because of applying the recursion in Step 4. Finally, Lemma 6 is applied only when we backtrack from the recursion. If we apply it on the graph A_1 of order n_1, we spend $O(n_1^3)$ time, but we never use it again on the same vertices. Therefore applications of Lemma 6 use only $O(n^3)$ time all together. This completes the proof of the correctness and of the stated time complexity of the algorithm. □

4 Conclusion

In this paper, we give a polynomial time algorithm for deciding whether linear lower bound on the chromatic number in terms of a parameter k is enough to force a K_k minor. Recently, Kawarabayashi and Mohar [17] proved that Theorem 3 can be improved as follows: For any k, there exists a constant $N(k)$ such that every $2k$-connected graph with minimum degree at least $9k$ and with at least $N(k)$ vertices has a K_k-minor. Also, if the tree-width is large (in a sense that we can apply Robertson and Seymour's result in [25] to G, see a detailed description in [5, 17]), then the minimum degree condition can be improved to $\frac{15a}{2}$. Also, Kawarabayashi proved in [15] that the order of the separation (A,B) in Lemma 6 can be reduced to $\frac{1}{6}k$. Together with these results, our algorithm implies that the chromatic number in Theorem 2 can be improved from $27k$ to $12k$.

Furthermore, Robertson and Seymour (private communication) have the following unpublished result, which would give rise to a polynomial-time algorithm for k-coloring K_k-minor free graphs if the Hadwiger Conjecture is true.

Theorem 7 (Robertson and Seymour, unpublished) *For every fixed k, there is a polynomial-time algorithm for deciding either that*

(1) *a given graph G is k-colorable, or*

(2) *G contains K_{k+1}-minor, or*

(3) *G contains a minor H without K_{k+1}-minors, of order at most $N(k)$, and with no k-coloring.*

Neil Robertson (private communication) pointed out that in order to prove the above theorem, Robertson and Seymour used the following lemma, which is of independent interest, and perhaps would be the strongest result in this direction.

Lemma 8 *Let $k \geq 4$ be an integer. For any graph G with no K_{k+1}-minor, one of the followings holds:*

(1) *There exists an integer $f(k)$ such that G has tree-width at most $f(k)$.*

(2) *G contains a vertex of degree at most k.*

(3) *G contains a vertex v of degree $k+1$ whose neighbors include three mutually nonadjacent vertices.*

(4) *G has a separation (A,B) of order at most k with $V(A) \neq V(G)$ such that A can be contracted to a clique on $A \cap B$ such that each vertex of $A \cap B$ is contained in the different node of this clique minor.*

(5) *G has a vertex set X, $|X| \leq k-4$, such that $G-X$ is planar.*

Note that (2), (3) and (4) cannot happen in minimal counterexamples to Hadwiger's conjecture, and (5) is no longer counterexample, assuming the Four Color Theorem [1, 2, 28]. The proof is complicated and uses the graph minor structure theory (cf., e.g., [24, 25]) heavily (but does not use the well-quasi-ordering result). Paul Seymour also pointed out that outcome (3) can be eliminatedon the expense of a considerably longer proof.

Acknowledgement

We would like to thank Neil Robertson and Paul Seymour for their helpful remarks.

References

[1] K. Appel and W. Haken, Every planar map is four colorable, Part I. Discharging, *Illinois J. Math.* **21** (1977), 429–490.

[2] K. Appel, W. Haken and J. Koch, Every planar map is four colorable, Part II. Reducibility, *Illinois J. Math.* **21** (1977), 491–567.

[3] S. Arnborg and A. Proskurowski, Linear time algorithms for NP-hard problems restricted to partial *k*-trees, *Discrete Appl. Math.* **23** (1989), 11–24.

[4] H.L. Bodlaender, A linear-time algorithm for finding tree-decomposition of small treewidth, *SIAM J. Comput.* **25** (1996) 1305–1317.

[5] T. Böhme, K. Kawarabayashi, J. Maharry and B. Mohar, Linear connectivity forces large complete bipartite minors, *submitted.*

[6] B. Bollobás and A. Thomason, Highly linked graphs, *Combinatorica* **16** (1996), 313–320.

[7] R. Diestel, Graph Theory, 2nd Edition, Springer, 2000.

[8] G. A. Dirac, A property of 4-chromatic graphs and some remarks on critical graphs, *J. London Math. Soc.* **27** (1952), 85–92.

[9] U. Feige, J. Kilian, Zero-knowledge and the chromatic number, *J. Comput. System Sci.* **57** (1998), 187–199.

[10] H. Hadwiger, Über eine Klassifikation der Streckenkomplexe, *Vierteljahrsschr. naturforsch. Ges. Zürich* **88** (1943), 133–142.

[11] J. HÍstad, Clique is hard to approximate within $n^{1-\varepsilon}$, *Acta Math.* **182** (1999), 105–142.

[12] M. R. Henzinger, S. Rao, H. N. Gabow, Computing vertex connectivity: New bounds from old techniques, *J. Algorithms* **34** (2000), 222–250.

[13] K. Kawarabayashi, Minors in 7-chromatic graphs, *preprint.*

[14] K. Kawarabayashi, On the connectivity of minimal counterexamples to Hadwiger's conjecture, *preprint*.

[15] K. Kawarabayashi, Linkage problem and its application to Hadwiger's conjecture, *preprint.*

[16] K. Kawarabayashi, A. Kostochka and G. Yu, On sufficient degree conditions for a graph to be *k*-linked, to appear in *Combin. Probab. Comput.*

[17] K. Kawarabayashi and B. Mohar, Improved connectivity bound on K_k-minors in large graphs, *preprint.*

[18] K. Kawarabayashi and B. Toft, Any 7-chromatic graph has K_7 or $K_{4,4}$ as a minor, *Combinatorica*, in press.

[19] A. Kostochka, The minimum Hadwiger number for graphs with a given mean degree of vertices (in Russian), *Metody Diskret. Analiz.* **38** (1982), 37–58.

[20] A. Kostochka, Lower bound of the Hadwiger number of graphs by their average degree, *Combinatorica* **4** (1984), 307–316.

[21] W. Mader, Homomorphiesätze für Graphen, *Math. Ann.* **178** (1968), 154–168.

[22] W. Mader, Existenz *n*-fach zusammenhängender Teilgraphen in Graphen genügend grosser Kantendichte, *Abh. Math. Sem. Univ. Hamburg* **37** (1972), 86–97.

[23] N. Robertson and P. D. Seymour, Graph minors XIII. The disjoint paths problems. *J. Combin. Theory Ser. B* **63** (1995), 65–110.

[24] N. Robertson and P. D. Seymour, Graph minors. XVI. Excluding a non-planar graph, *J. Combin. Theory Ser. B* **89** (2003), 43–76.

[25] N. Robertson and P. D. Seymour, Graph minors. XVII. Taming a vortex, *J. Combin. Theory Ser. B* **77** (1999), 162–210.

[26] N. Robertson and P. D. Seymour, Graph minors. XX. Wagner's conjecture, to appear in *J. Combin. Theory Ser. B.*

[27] N. Robertson, P. D. Seymour and R. Thomas, Hadwiger's conjecture for K_6-free graphs, *Combinatorica* **13** (1993), 279–361.

[28] N. Robertson, D. P. Sanders, P. D. Seymour and R. Thomas, The four-color theorem, *J. Combin. Theory Ser. B* **70** (1997), 2–44.

[29] R. Thomas and P. Wollan, An improved linear edge bound for graph linkages, *Europ. J. Combin.* 26 (2005) 309–324.

[30] A. Thomason, An extremal function for contractions of graphs, *Math. Proc. Cambridge Philos. Soc.* **95** (1984), 261–265.

[31] A. Thomason, The extremal function for complete minors, *J. Combin. Theory Ser. B* **81** (2001), 318–338.

[32] B. Toft, A survey of Hadwiger's conjecture, *Congr. Numer.* **115** (1996), 249–283.

[33] K. Wagner, Über eine Eigenschaft der ebenen Komplexe, *Math. Ann.* **114** (1937), 570–590.

[34] K. Wagner, Beweis einer Abschwächung der Hadwiger-Vermutung, *Math. Ann.* **153** (1964), 139–141.

Orientations with parity and capacity constraints

TAMÁS KIRÁLY*

MTA-ELTE Egerváry Research Group
Eötvös University
1117, Pázmány P. sétány 1/C
Budapest, Hungary
tkiraly@cs.elte.hu

GYULA PAP†

Department of Operations Research
Eötvös University
1117, Pázmány P. sétány 1/C
Budapest, Hungary
gyuszko@cs.elte.hu

Abstract: In [1] Frank, Sebő and Tardos gave a necessary and sufficient condition for a graph to have an orientation with in-degrees satisfying lower and upper bounds and parity constraints. We give a polyhedral description for the convex hull of the in-degrees of such orientations. Our description admits strongly polynomial integer optimization for both the primal and the dual system. As a generalization of the Gallai-Edmonds decomposition for matchings, we show a structural description of optimal orientations.

Keywords: parity, orientations, TDI descriptions, Gallai-Edmonds decomposition

1 Introduction

In [1] Frank, Sebő and Tardos observed that for a graph $G=(V,E)$ the existence of an orientation fulfilling parity and capacity constraints for the in-degree of each node in V can be described by the existence of a perfect matching in an auxiliary graph. In this paper we give a TDI description of the convex hull of the in-degree sequences of these orientations. This polytope is a homomorphic image of a perfect matching polytope, and can only be described using inequalities with large coefficients. In general no TDI description is known for homomorphic images of perfect matching polytopes; the main obstacle is that large coefficients appear in the description of the facets, so they must appear in any full description. There are only few known examples of classes of polytopes where this is the case and a computationally tractable TDI description still exists.

In this extended abstract we show the sketch of two entirely different proofs for the TDI-ness of the description. The first one is based on the homomorphic mapping from the perfect matching polytope of an auxiliary graph, exploiting a result of F. Barahona and W.H. Cunningham [2] on partial dual integrality of a description. The second proof is a direct proof of the TDI property, based on the transformation of a non-integer optimal dual solution into an integer one.

2 Notions and notations

For a graph $G=(V,E)$ we denote by $d_G(v)$ the degree of a node $v\in V$. For disjoint node-sets S and T, $d_G(S,T)$ denotes the set of edges joining nodes in S and T; for $X\subseteq V$, $d_G(X):=d_G(X,V-X)$. An *orientation* of G is a directed graph on the same node-set obtained by replacing each edge $ab\in E$ by either the arc ab or the arc ba. In a directed graph, if S is a set of nodes then $\rho(S)=\delta(V-S)$ denotes the number of edges ab with $a\notin S\ni b$. If w is a real-valued function on a set V then for a set $U\subseteq V$ let $w(U):=\sum_{u\in U}w(u)$.

Let $G=(V,E)$ be undirected graph, $l,u:V\to\mathbb{N}$ lower and upper node-capacities so that $u(v)\equiv l(v) \pmod 2$ and $0\le l(v)\le u(v)\le d_G(v)$ for each node $v\in V$. Let T be the set of nodes for which $u(v)$ is odd. An orientation $\overrightarrow{G}$ of G is said to be T-*odd* if $\rho(v)=\rho_{\overrightarrow{G}}(v)$ is odd exactly in the nodes of T. An orientation is called an (l,u)-*orientation* if it is a T-odd orientation for which $l\le\rho\le u$.

For a set $X\subseteq V$, $\rho(X)\equiv u(X)+i_G(X)$ is true in every (l,u)-orientation. This means that if we have a partial orientation D of G where each edge in $d_G(X)$ is oriented in D, and $\rho_D(X)\not\equiv u(X)+i_G(X)$, then no matter how we complete D to an orientation of G there will be a node in X with wrong parity. In this case we say that X *has wrong in-degree parity in* D, otherwise it has *correct in-degree parity in* D.

*Research is supported by the Hungarian National Foundation for Scientifi c Research, OTKA T037547, and by European MCRTN Adonet, Contract No. 504438.

†Research is supported by the Hungarian National Foundation for Scientifi c Research, OTKA T037547, and by European MCRTN Adonet, Contract No. 504438.

3 The polytope

Let $G=(V,E)$ be undirected graph, $l,u:V\to\mathbb{N}$ lower and upper node-capacities so that $u(v)\equiv l(v) \pmod 2$ and $0\le l(v)\le u(v)\le d_G(v)$ for each node $v\in V$. Let $\mathscr{P}=\mathscr{P}(G,l,u)$ denote the convex hull of in-degree sequences of (l,u)-orientations. Consider a 'height'-function $h:V\to\mathbb{Z}$, and a laminar family of node sets $\mathscr{F}$ with a multiplicity function $m:\mathscr{F}\to\mathbb{Z}_+$. Let $D=D(h)=(V,E_0\cup E_1)$ denote the partial orientation of E in which an edge ab is oriented towards b if and only $h(a)<h(b)$, and the edges ab with $h(a)=h(b)$ are not oriented; E_1 denotes the set of directed edges, and E_0 the set of undirected edges. When the triple $(h,\mathscr{F},m)$ has the following properties, we call it a *legal triple*.

1. For each edge with $ab\in E$ and $h(a)\le h(b)$ we have $h(b)-h(a)\ge d_{\mathscr{F},m}(ab):=\sum\{m(F):F\in\mathscr{F},ab\in d_G(F)\}$.
2. For each set $F\in\mathscr{F}$ each edge in $d_G(F)$ is oriented in D (i.e. it is in E_1) and F has wrong in-degree parity in D. (in this case we say that F *is h-odd*).

The first property may be told in the following equivalent way: each edge must span at least as much height in h as the number of sets (with multiplicities) it crosses in $(\mathscr{F},m)$. For a height function h let us define $e(h):=\sum_{ab\in E}\max\{h(a),h(b)\}$, which is an extension of the usual set-function $e(\cdot)$, since for a set $X\subseteq V$ we have $e(\chi(X))=e(X)$.

Now we state the main result on the convex hull of in-degree sequences.

Theorem 1 *The polytope $\mathscr{P}$ is described by the inequalities*

$$h\cdot x\le e(h)-m(\mathscr{F}) \qquad (h,\mathscr{F},m)\ \textit{legal} \tag{1}$$
$$-x\ge -l \tag{2}$$
$$x\le u. \tag{3}$$

Moreover, for any integer vector c the dual to $\max cx$ *subject to (1)-(3) has an integer optimal solution with at most one of the inequalities (1) having a non-zero dual entry – and this non-zero entry is 1. In particular, the system (1)-(3) is TDI.*

The following claim implies that the inequalities (1)-(3) are valid for $\mathscr{P}$.

Claim 2 *If $x\in\mathscr{P}$ then $h\cdot x\le e(h)-m(\mathscr{F})$ for any legal triple $(h,\mathscr{F},m)$.*

PROOF: We need to show the inequality if $x:=\rho_{\overrightarrow{G}}$ for some (l,u)-orientation $\overrightarrow{G}$. Consider the partial orientation $D=D(h)$, then $h\cdot\rho_{D'}=e(h)$ for any completion D' of D, since an edge ab with $h(a)=h(b)$ contributes to $h\cdot\rho_{D'}$ with $h(a)=h(b)$ independently of its orientation in D'.

The orientation of edges in E_0 in $\overrightarrow{G}$ does not make a difference in $h\cdot x$, we need to concentrate on edges in E_1. We compare the orientations of edges E_1 in D and the orientations of these edges in $\overrightarrow{G}$: let M denote the set of edges in E_1 which have a different orientation in D and $\overrightarrow{G}$. It is easy to see that

$$h\cdot x=e(h)+\sum_{\substack{ab\in M\\ h(a)\le h(b)}}(h(a)-h(b)). \tag{4}$$

As each set $F\in\mathscr{F}$ has wrong in-degree parity in D, in any (l,u)-orientation $\overrightarrow{G}$ there must be an edge in $d(F)$ oriented in the other direction as in D, i.e. $M\cap d(F)\neq\emptyset$. By the definition of legality

$$\sum_{\substack{ab\in M\\ h(a)\le h(b)}}(h(a)-h(b))\le\sum_{ab\in M}\sum_{\substack{F\in\mathscr{F}\\ ab\in d(F)}}-m(F)=$$
$$=\sum_{F\in\mathscr{F}}\sum_{ab\in M\cap d(F)}-m(F)\le\sum_{F\in\mathscr{F}}-m(F)=-m(\mathscr{F}), \tag{5}$$

which completes the proof. □

Our first proof of the theorem is based on the observation (first made in [1]) that the polytope $\mathscr{P}$ arises as a homomorphic image of the perfect matching polytope of an auxiliary graph G'. We apply the following well-known theorem on the perfect matching polytope. Consider an undirected graph $G'=(V',E')$, then the following is a totally dual half-integral description of the perfect matching polytope of G'.

$$\begin{aligned} 2x(E'[S])&\le|S|-1 && \text{if}\quad S\subseteq V',\,3\le|S'|\ \text{odd}\\ x(d(v))&=1 && \text{if}\quad v\in V'\\ x&\ge 0 \end{aligned} \tag{6}$$

Barahona and Cunningham [2] showed the following strengthening of this property.

Theorem 3 (Barahona, Cunningham [2]) *If $w \in \mathbb{Z}^{E'}$ is a weight-function such that $w(C)$ is even for each cycle C in G', then the problem of minimizing wx subject to (6) has an integer optimum dual solution.*

The proof of Theorem 1 consists of transforming the integer optimum dual solution obtained from Theorem 3 into an integer optimum dual solution of the system in Theorem 1. The details are omitted in this extended abstract. This proof method also gives a polynomial algorithm for finding an integer optimum dual solution, since the transformation of the dual solution in Theorem 3 can be done in polynomial time.

4 Sketch of second proof of Theorem 1

The second proof of Theorem 1 is a direct proof of the TDI property. First we show that there is an optimum dual solution where at most one of the inequalities (1) has non-zero dual entry. If there are legal triples $(h_1, \mathscr{F}_1, m_1)$ and $(h_2, \mathscr{F}_2, m_2)$ with dual entries $y_1 \geq y_2 > 0$, then we can find a legal triple $(h_1 + h_2, \mathscr{F}_3, m_3)$ such that by setting $y'_1 := y_1 - y_2$, $y'_2 := 0$, $y'_3 := y_2$, and $y' = y$ on the other entries, we get an optimal dual solution y'. By doing such modifications to the dual solution we eventually get to one where at most one of the inequalities (1) has non-zero dual entry. The pair $(\mathscr{F}_3, m_3)$ can be obtained from $(\mathscr{F}_1, m_1)$ and $(\mathscr{F}_2, m_2)$ by a series of transformations that involve decreasing some values of m_1 and m_2, and replacing some sets $X \in \mathscr{F}_1$ by one of $X \cap Y, X \cup Y, X - Y, Y - X$ for some $Y \in \mathscr{F}_2$, or vice versa. The details are omitted here.

The second part of the proof consists of transforming a non-integer optimal dual solution into an integer one. The following claim implies that such a transformation always exists. A dual solution y is said to be *based* on the legal triple $(h, \mathscr{F}, m)$ if $y(h, \mathscr{F}, m) = 1$ and y is 0 on all other legal triples.

Claim 4 *Suppose that every value of the cost function c is divisible by an integer α, and we have an integer dual optimal solution y that is based on $(h, \mathscr{F}, m)$. Then we can modify y to obtain an integer dual optimal solution y' that is based on $(h', \mathscr{F}', m')$ where $h'(v)$ is divisible by α for every $v \in V$, and $m'(Z)$ is divisible by α for every $Z \in \mathscr{F}'$.*

PROOF: Let $G^= = (V, E^=)$ denote the graph containing the edges $uv \in E$ for which $h(v) - h(u) = d_{\mathscr{F},m}(uv)$ (these are called *tight edges*). It can be assumed that $G^=[X]$ is connected for every $X \in \mathscr{F}$. Suppose that there is a set $X_0 \in \mathscr{F}$ such that every maximal subset X of X_0 in $\mathscr{F}$ has the following properties:

- $h(v)$ is divisible by α for every $v \in X$,
- if $Z \subsetneq X$, $Z \in \mathscr{F}$, then $m(Z)$ is divisible by α.

Let $\mathscr{F}_{X_0}$ denote the maximal subsets of X_0 in $\mathscr{F}$, and let S be the set of nodes $v \in X_0$ with $h(v) \not\equiv 0 \bmod \alpha$. It can be shown that there is a partition $\{\mathscr{F}_1, \mathscr{F}_2\}$ of $\mathscr{F}_{X_0}$ and a partition $\{S_1, S_2\}$ of S such that the following hold.

- There is no tight edge between members of $\mathscr{F}_i$ $(i = 1, 2)$.
- There is no tight edge between nodes of S_1 and S_2.
- If $v \in S_i$, $X \in \mathscr{F}_i$, and there is a tight edge between v and X, then $h(v) > h(w)$ for every $w \in X$.
- If $v \in S_i$, $X \in \mathscr{F}_{3-i}$, and there is a tight edge between v and X, then $h(v) < h(w)$ for every $w \in X$.

Let us consider the dual variables $y_{l(v)}$ and $y_{u(v)}$ for some $v \in S$. Since the cost $c(v)$ is divisible by α but $h(v)$ is not, we have either $y_{l(v)} > 0$ or $y_{u(v)} > 0$. Let S_i^l be the set of nodes in S_i for which $y_{l(v)} > 0$, and S_i^u be the set of nodes in S_i for which $y_{u(v)} > 0$. Consider the following two possible dual changes:

1) Increase: $m(X)$ if $X \in \mathscr{F}_1$; $h(v)$ if $v \in S_1$; $y_{u(v)}$ if $v \in S_1^u$; $y_{l(v)}$ if $v \in S_2^l$.
Decrease: $m(X_0)$; $m(X)$ if $X \in \mathscr{F}_2$; $h(v)$ if $v \in S_2$; $y_{l(v)}$ if $v \in S_1^l$; $y_{u(v)}$ if $v \in S_2^u$.

2) Increase: $m(X)$ if $X \in \mathscr{F}_2$; $h(v)$ if $v \in S_2$; $y_{l(v)}$ if $v \in S_1^l$; $y_{u(v)}$ if $v \in S_2^u$.
Decrease: $m(X_0)$; $m(X)$ if $X \in \mathscr{F}_1$; $h(v)$ if $v \in S_1$; $y_{u(v)}$ if $v \in S_1^u$; $y_{l(v)}$ if $v \in S_2^l$.

From the properties of $\mathscr{F}_1, \mathscr{F}_2, S_1, S_2$ and the fact that every $Z \in \mathscr{F}$ is h-odd we can deduce that one of the above dual changes is feasible and maintains optimality.

If we increase/decrease by the maximum possible amount, then either the number of tight edges induced by X_0 increases, or some $h(v)$ $(v \in S)$ or $m(Z)$ $(Z \in \mathscr{F}_{X_0})$ becomes integer. So by similar dual changes we can eventually reach a solution where $h(v)$ is divisible by α if $v \in X \in \mathscr{F}$, and $m(Z)$ is divisible by α if $Z \in \mathscr{F}$ is not maximal. An argument similar to the above can also be used to treat the maximal members of $\mathscr{F}$ and the nodes not in members of $\mathscr{F}$. □

Claim 4 implies that the system (1)–(3) is TDI. To prove that it corresponds to the convex hull of in-degree vectors of (l,u)-feasible orientations, it suffices to prove that if the legal triple $(h, \mathscr{F}, m)$ defines a face of the polyhedron, then $e(h) - m(\mathscr{F}) \equiv \sum_{v \in V} h(v) l(v) \bmod 2$. This can be shown by parity arguments, using the fact that every component of $G^=[X]$ is h-even since otherwise it could have been added to $\mathscr{F}$.

5 A structural description

It is possible to obtain a structural description of parity-constrained orientations of a given graph by applying the Gallai-Edmonds structure theorem on the auxiliary graph that was introduced by Frank, Sebő and Tardos [1].

Let $G=(V,E)$ be undirected graph, $l,u :\rightarrow \mathbb{N}$ lower and upper node-capacities so that $u(a) \equiv l(a) \pmod 2$ and $0 \leq l(a) \leq u(a) \leq d(a)$ for each node $a \in V$. Let T be the set of nodes for which $u(a)$ is odd.

For a node $a \in V$ let us call the integers in the interval $[l(a),u(a)]$ with the same parity as $u(a)$ *allowed for* a. Consider an arbitrary orientation $\overrightarrow{G}$ of G, then for a node $a \in V$ let $\mathrm{def}(a) = \mathrm{def}_{\overrightarrow{G}}(a)$ denote the distance between $\rho(a) := \rho_{\overrightarrow{G}}(a)$ and the allowed integers for a. That is,

1. if $\rho(a) < l(a)$ then $\mathrm{def}(a) = l(a) - \rho(a)$,
2. if $\rho(a) > u(a)$ then $\mathrm{def}(a) = \rho(a) - u(a)$,
3. if $l(a) \leq \rho(a) \leq u(a)$ and $\rho(a) \not\equiv u(a)$ then $\mathrm{def}(a) = 1$,
4. otherwise $\mathrm{def}(a) = 0$.

Let us define the deficiency of this orientation by $\mathrm{def}(\overrightarrow{G}) = \sum_{a \in V} \mathrm{def}(a)$. An orientation is called *optimal* if it has the smallest possible deficiency, denoted by $\mathrm{def}(G,l,u)$.

If A,B is a pair of disjoint node sets in G, then let $D(A,B)$ denote the partial orientation of G with each arc in $d_G(A)$ oriented out of A, each arc in $d_G(B)$ oriented into B. A component of $G-A-B$ is called *odd/even* whenever it has wrong/correct in-degree in $D(A,B)$. Let $c(G,A,B)$ denote the number of odd components for A,B.

Claim 5 *If A,B are disjoint node sets in G, then*

$$\mathrm{def}(G,u,l) \geq i(A) - u(A) - e(B) + l(B) + c(G,A,B). \tag{7}$$

PROOF: Consider an optimal orientation $\overrightarrow{G}$ and the partial orientation $D = D(A,B)$. Let c denote the number of components of $G-A-B$ which have wrong in-degree parity in $\overrightarrow{G}$. It is easy to see the following:

$$\mathrm{def}(G,u,l) \geq \mathrm{def}(A) + \mathrm{def}(B) + \mathrm{def}(V-A-B) \geq$$
$$\geq (i(A) - u(A) + \rho_{\overrightarrow{G}}(A)) + (-e(B) + l(B) + \delta_{\overrightarrow{G}}(B)) + c. \tag{8}$$

Suppose there are m components X of $G-A-B$ with at least one edge in $d_G(X)$ oriented differently in D and $\overrightarrow{G}$. Then $c \geq c(G,A,B) - m$ and $\rho_{\overrightarrow{G}}(A) + \delta_{\overrightarrow{G}}(B) \geq m$, which by (8) imply (7). □

Theorem 6 *Let A be the set of nodes a for which each optimal orientation has in-degree at least $u(a)$ and there is an optimal orientation with in-degree at least $u(a)+1$. Let B be the set of nodes a for which each optimal orientation has in-degree at most $l(a)$ and there is an optimal orientation with in-degree at most $l(a)-1$. Let C be the set of nodes which have allowed in-degree in each optimal orientation. Let $Q := V-A-B-C$, then*

1. *The components of $G[Q]$ are odd components for A,B.*
2. *The components of $G[C]$ are even components for A,B.*
3. $\mathrm{def}(G,u,l) = i(A) - u(A) - e(B) + l(B) + c(G,A,B)$.

The proof of Theorem 6 consists of showing that the deficiency of matchings in the auxiliary graph and the deficiency of orientations of G are closely related, and that the sets A,B can be constructed from the Gallai-Edmonds decomposition of the auxiliary graph. The details are omitted in this extended abstract.

References

[1] A. FRANK, A. SEBŐ AND É. TARDOS, Covering directed and odd cuts, *Mathematical Programming Study* (1994) **22**

[2] F. BARAHONA AND W.H. CUNNINGHAM, On dual integrality in matching problems, *Operations Research Letters* (1989) **8**

On well-balanced orientations

SATORU IWATA

Department of Mathematical Informatics
Graduate School of Information Science and Technology
University of Tokyo, Japan
e-mail: iwata@mist.i.u-tokyo.ac.jp

TAMÁS KIRÁLY*

Egerváry Research Group, MTA-ELTE
Department of Operations Research
Eötvös University
Pázmány Péter sétány 1/C, Budapest, Hungary
e-mail: tkiraly@cs.elte.hu

ZOLTÁN KIRÁLY†

Department of Computer Science and Communication Networks Laboratory
Eötvös University
Pázmány Péter sétány 1/C, Budapest, Hungary
e-mail: kiraly@cs.elte.hu

ZOLTÁN SZIGETI‡

Equipe Combinatoire et Optimisation
Université Paris 6
75252 Paris, Cedex 05, France
e-mail: szigeti@math.jussieu.fr

Abstract: This note contains some remarks on the well-balanced orientation theorem of Nash-Williams [17]. He announced in [18] an extension of his theorem. We present a proof for a generalization of this extension. We show some new consequences of Nash-Williams' odd vertex pairing theorem. In the second part we consider problems related to the well-balanced orientation theorem and some possible generalizations. Some of them we leave as open questions, for others we give counterexamples; and show some positive results for special cases.

Keywords: directed graphs, orientation, local connectivity

1 Introduction

This paper concerns orientations of undirected graphs. The starting point is Robbins' theorem [19] that states that an undirected graph G has a strongly connected orientation if and only if G is 2-edge-connected. The following generalization was proved by Nash-Williams [17]: an undirected graph G has a k-arc-connected orientation if and only if G is $2k$-edge-connected. Nash-Williams [17] also provided the following extension on local edge-connectivity: for any undirected graph G there exists a **well-balanced** orientation that is an orientation of G such that for every ordered pair of vertices u,v, if the maximum number of edge disjoint $(u,v)-$paths was $\lambda_G(u,v)$ in G then the maximum number of arc disjoint directed $(u,v)-$paths is at least $\lfloor \lambda_G(u,v)/2 \rfloor$ in the resulting directed graph. The well-balanced orientation may also be required to be **smooth**, that is the difference between the in-degree and the out-degree of every vertex is at most one (Theorem 5). A smooth well-balanced orientation is called **best-balanced**. In fact, Nash-Williams proved an even stronger result in [17], the so-called odd vertex pairing theorem (Theorem 7) stating that for every graph there exists a "feasible pairing" (for definition see Section 2). In [18], Nash-Williams announced an extension of his orientation theorem: for an arbitrary subgraph H of an undirected graph G there exists a best-balanced orientation of H that can be extended to a best-balanced orientation of G (Theorem 6). He mentioned that *"Given Theorem 7, the proof of Theorem 6 is not unreasonably difficult. At a certain stage in the proof Theorem 5 ... we had occasion to select an arbitrary di-Eulerian orientation Δ of the finite Eulerian graph $G+P$. ... the proof of Theorem 6 depends essentially on the idea of modifying this step ... by choosing Δ to be, not just any di-Eulerian orientation of $G+P$, but one which satisfies certain additional restrictions."* The first part of the present paper provides a simple proof for a generalization of this result and shows some consequences. This part relies on the work of Király and Szigeti [10], for all proofs missing from this part see [10].

The second part (based on [9]) concerns problems and questions related to this area. The aim of this paper is to help to find a transparent proof for the well-balanced orientation theorem. A possible way could be to find a convenient generalization that has a simple inductive proof. Here we think of results like Theorems 8, 9 and 10. Unfortunately we do not have direct proofs for them, they follow easily from the odd vertex pairing theorem. This result (Theorem 7) is a miracle, it has no generalization, no application (except the well-balanced theorem), no relation to any other result in Graph Theory.

*Research supported by OTKA grant T 037547 and European MCRTN ADONET, Contract Grant no. 504438.
†Research supported by EGRES group (MTA-ELTE) and OTKA grants T 037547 and T046234.
‡This research was partially done when this author visited University of Tokyo supported by the 21 Century COE Program from the Ministry of Education, Culture, Sports, Science and Technology of Japan.

As mentioned we are looking for orientations of graph G satisfying $\lambda_{\vec{G}}(u,v) \geq \mathbf{r}(u,v)$ for some prescription function $\mathbf{r}$. We call the global case of this problem if $\mathbf{r}$ is a constant function, and the local case otherwise.

Splitting off usually helps to prove orientation theorems for the global case. An example of Enni [1] shows that there is no splitting off theorem preserving local arc-connectivities in directed graphs. In Question 20 we provide a smaller example showing that even if $\vec{G}$ is a well-balanced orientation of G there is no splitting off that preserves local arc-connectivities in V.

The original proof of the odd vertex pairing theorem in [17] and Frank's proof [5] as well relies heavily on the skew-submodularity of the function b_G. We show (Question 24) that the existence of a feasible pairing cannot be generalized to arbitrary skew-submodular functions. Skew-submodular functions correspond to local edge-connectivity, while crossing submodular functions can be considered as generalizations of global edge-connectivity. For such a function it is an open problem whether there exists a feasible pairing.

Nash-Williams [17] showed that if M is a feasible pairing of G then for every Eulerian orientation $\vec{G}+\vec{M}$ of $G+M$, $\vec{G}$ is best-balanced. We show (Question 28) that not every best-balanced orientation can be defined by a feasible pairing. On the other hand we prove in Theorem 29 that for the global case it can be.

We also show that the *general subgraph chain property* is not valid, that is the extension of Nash-Williams [18] proved in the first part cannot be further generalized for subgraph chain of length three, neither for the global case (see Question 34).

Frank [3] proved the following *reorientation property* for the k-arc-connected orientations: Given two k-arc-connected orientations of G, there exists a series of k-arc-connected orientations of G (leading from the first to the second given orientation), such that in each step we reverse a directed path or a circuit. For well-balanced (or best-balanced) orientations it is not known whether the reorientation property is valid.

Frank [2] also proved that the *linkage property* is valid for the k-arc-connected orientation problem, namely there exists a k-arc-connected orientation whose in-degree function satisfies lower and upper bounds if and only if there is one satisfying the lower bound and one satisfying the upper bound. É. Tardos [20] showed that the linkage property is not valid for the well-balanced orientation problem. Here we present another example (see Question 31).

By the proof of Frank [3] it is easy to see that the following *matroid property* is valid for smooth k-arc-connected orientations: The family of sets, over smooth k-arc-connected orientations, consisting of vertices whose in-degree is larger than the out-degree, forms the basis of a matroid. We show that this is not true for best-balanced orientations (see Question 32).

We also show that the *polyhedron* of the fractional in-degree vectors of well-balanced orientations is not necessarily integral (see Question 30). On the other hand for the global case it is. For missing proofs of the second part see [9].

2 Notation, definitions

A directed graph is denoted by $\vec{G}=(V,A)$ and an undirected graph by $G=(V,E)$. For a directed graph $\vec{G}$, a set $X \subseteq V$ and $u,v \in V$, let $\delta_{\vec{G}}(X) := |\{uv \in A : u \in X,\ v \notin X\}|$, $\rho_{\vec{G}}(X) := \delta_{\vec{G}}(V-X)$, $f_{\vec{G}}(X) := \rho_{\vec{G}}(X) - \delta_{\vec{G}}(X)$, $\lambda_{\vec{G}}(u,v) := \min\{\delta_{\vec{G}}(Y) : u \in Y,\ v \notin Y\}$, and $\overleftarrow{G} := (V, \{vu : uv \in A\})$. For an undirected graph G, a set $X \subseteq V$ and $u,v \in V$, let $\Delta_G(X) := \{uv \in E : u \in X,\ v \notin X\}$, $d_G(X) := |\Delta_G(X)|$, $d_G(X,Y) := |\{uv \in E(G) : u \in X-Y, v \in Y-X\}|$, $i(X) := |\{uv \in E : u,v \in X\}|$, $\lambda_G(u,v) := \min\{d_G(X) : u \in X,\ v \notin X\}$, $R_G(X) := \max\{\lambda_G(x,y) : x \in X,\ y \notin X\}$, $\widehat{R}_G(X) := 2\lfloor R_G(X)/2 \rfloor$, $b_G(X) := d_G(X) - \widehat{R}_G(X)$ and $T_G := \{v \in V : d_G(v) \text{ is odd}\}$. Observe that $\forall X \subseteq V$,

$$f_{\vec{G}}(X) = \sum_{v \in X} f_{\vec{G}}(v), \tag{1}$$

An undirected graph G (directed graph D) is **connected** (**strongly connected**) if for every (ordered) pair $u,v \in V$ of vertices there is a (u,v)-path in G (resp. D). It is called **k-edge-connected** (**k-arc-connected**) if $G-F$ is connected ($D-F$ is strongly connected resp.) for $\forall F \subseteq E$ ($\forall F \subseteq A$) with $|F| \leq k-1$. For a function $\mathbf{r} : V \times V \to \mathbb{Z}_0^+$, we say that G is **$\mathbf{r}$-edge-connected** (D is **$\mathbf{r}$-arc-connected**) if $\lambda_G(u,v) \geq \mathbf{r}(u,v)$ ($\lambda_{\vec{G}}(u,v) \geq \mathbf{r}(u,v)$ resp.) for every (ordered) pair u,v of vertices. By Menger's Theorem, G is k-edge-connected (k-arc-connected) if and only if it is $\mathbf{r}$-edge-connected ($\mathbf{r}$-arc-connected) for $\mathbf{r}(u,v) := k\ \ \forall u,v \in V$.

An orientation $\vec{G}$ of G is called **well-balanced** if $\vec{G}$ satisfies (2), **smooth** if $\vec{G}$ satisfies (3) and **best-balanced** if it is smooth and well-balanced. Let us denote by $\mathscr{O}_w(G)$ and $\mathscr{O}_b(G)$ the set of well-balanced and best-balanced orientations of G. Note that if $\vec{G}$ is best-balanced then so is $\overleftarrow{G}$.

$$\lambda_{\vec{G}}(x,y) \geq \lfloor \lambda_G(x,y)/2 \rfloor \ \ \forall\, (x,y) \in V \times V, \tag{2}$$

$$|f_{\vec{G}}(v)| \leq 1 \quad \forall\, v \in V. \tag{3}$$

A **pairing** M of G is a new graph on vertex set T_G in which each vertex has degree one. Let M be a pairing of G. An orientation $\vec{M}$ of M that satisfies (4) is called **good**. Pairing M is **well-orientable** if there *exists* a good orientation of M, M is **strong** if *every* orientation of M is good and M is **feasible** if (5) is satisfied. Clearly an oriented pairing $\vec{M}$ is good iff $\overleftarrow{M}$ is good. We say that the orientations $\vec{M}$ and $\vec{G}$ are **compatible** if (6) is satisfied or equivalently if $\vec{G}+\overleftarrow{M}$ is Eulerian.

Let us denote by $\mathscr{P}_f(G)$ the set of feasible pairings of G.

$$f_{\vec{M}}(X) \leq b_G(X) \quad \forall X \subseteq V, \tag{4}$$
$$d_M(X) \leq b_G(X) \quad \forall X \subseteq V, \tag{5}$$
$$f_{\vec{G}}(X) = f_{\vec{M}}(X) \quad \forall X \subseteq V. \tag{6}$$

3 Equivalent forms

Claim 1 *For an orientation $\vec{G}$ of an undirected graph G, the following are equivalent:*

$$\vec{G} \text{ is } well-balanced, \tag{7}$$
$$\delta_{\vec{G}}(X) \geq \lfloor R_G(X)/2 \rfloor \quad \forall X \subseteq V, \tag{8}$$
$$f_{\vec{G}}(X) \leq b_G(X) \quad \forall X \subseteq V. \tag{9}$$

Claim 2 *A pairing M of G is strong if and only if M is feasible.*

Claim 3 *Let G be an undirected graph.*

(a) If $\vec{M}$ is a well-oriented pairing of G, then G has an orientation $\vec{G}$ compatible to $\vec{M}$.

(b) For a pairing M of G, let the orientations $\vec{M}$ and $\vec{G}$ be compatible. Then $\vec{G}$ is smooth. Moreover, $\vec{G}$ is well-balanced if and only if $\vec{M}$ is good.

(c) G has a best-balanced orientation if and only if G has a well-orientable pairing.

(d) For a strong pairing M of G, for every Eulerian orientation $\vec{G}+\vec{M}$ of $G+M$, $\vec{G}$ is best-balanced.

4 Theorems

The following four theorems are due to Nash-Williams [17], [18].

Theorem 4 *A graph G has a k-arc-connected orientation if and only if G is $2k$-edge-connected.*

Theorem 5 *Every graph has a best-balanced orientation.*

Theorem 6 *For every subgraph H of G, there exists a best-balanced orientation of H that can be extended to a best-balanced orientation of G.*

Theorem 7 *Every graph has a feasible pairing.*

We shall show that the above "odd vertex pairing" theorem implies all the other results presented in this section. By Theorem 5 and Claim 3, every graph has a well-orientable pairing. In the following theorem we generalize this result.

Theorem 8 *Every pairing is well-orientable.*

The main results of [10] are the following generalizations of Theorem 6 and Theorem 5.

Theorem 9 *For every partition $\{E_1, E_2, ..., E_k\}$ of $E(G)$, if $G_i = (V, E_i)$ then G has a best-balanced orientation $\vec{G}$, such that the inherited orientation of each G_i is also best-balanced.*

Theorem 10 *For every partition $\{X_1, ..., X_l\}$ of $V = V(G)$, G has an orientation $\vec{G}$ such that $\vec{G}$, $((\vec{G}/X_1)...)/X_l$ and $\vec{G}/(V - X_i)$ $(1 \leq i \leq l)$ are best-balanced orientations of the corresponding graphs.*

5 Proofs

In this section we shall apply Theorem 7 to prove the new results in the previous section. We must emphasize that we do not have a new proof for Theorem 7.

PROOF:[of Theorem 8:] Let M_1 be an arbitrary and M_2 be a strong pairing of G. M_2 exists by Theorem 7. $M_1 \cup M_2$ is an Eulerian graph so it has an Eulerian orientation $\vec{M}_1 \cup \vec{M}_2$. Then $f_{\vec{M}_1}(v) = -f_{\vec{M}_2}(v) = f_{\overleftarrow{M}_2}(v) \ \ \forall v \in V$. Then, by (1) and using that $\overleftarrow{M}_2$ is a good orientation of M_2, $f_{\vec{M}_1}(X) = \sum_{v \in X} f_{\vec{M}_1}(v) = \sum_{v \in X} f_{\overleftarrow{M}_2}(v) = f_{\overleftarrow{M}_2}(X) \leq b_G(X) \ \ \forall X \subseteq V$, so $\vec{M}_1$ is a good orientation of M_1. □

By the above proof, if we know a feasible pairing, then for every pairing we can find the required orientation in linear time.

The following proofs of Theorem 9 and Theorem 10 are the main contribution of the first part. Theorem 6 clearly follows from Theorem 9.

PROOF:[of Theorem 9:] Let M_0 and M_i be strong pairings of G and of G_i for $1 \leq i \leq k$ provided by Theorem 7. Note that for $K := \sum_0^k M_i$, for every $v \in V$, $d_K(v) = \sum_0^k d_{M_i}(v) \equiv d_G(v) + \sum_1^k d_{G_i}(v) = 2d_G(v)$ is even, so K has an Eulerian orientation $\vec{K} = \sum_0^k \vec{M}_i$ that is $\sum_1^k f_{\vec{M}_i}(X) = -f_{\vec{M}_0}(X) \ \ \forall X \subseteq V$. For $1 \leq i \leq k$, $\vec{M}_i$ is a good orientation of M_i so, by Claim 3(a) and (b), G_i has a best-balanced orientation $\vec{G}_i$ compatible to $\vec{M}_i$. Let $\vec{G} := \cup_1^k \vec{G}_i$. Then, by (6), $f_{\vec{G}}(X) = \sum_1^k f_{\vec{G}_i}(X) = \sum_1^k f_{\vec{M}_i}(X) = -f_{\vec{M}_0}(X) = f_{\overleftarrow{M}_0}(X) \ \ \forall X \subseteq V$ so $\vec{G}$ and $\overleftarrow{M}_0$ are compatible. $\overleftarrow{M}_0$ is a good orientation of M_0 thus, by Claim 3(b), $\vec{G}$ is a best-balanced orientation of G. □

PROOF:[of Theorem 10:] Let $G_0 := (((G/X_1)/X_2)/...)/X_l$ and $G_i := G/(V - X_i)$ $(1 \leq i \leq l)$. Let M_i be a strong pairing of G_i $(0 \leq i \leq l)$ provided by Theorem 7. It is easy to see that G has a unique pairing M whose restriction in G_i is M_i. By Theorem 8, M has a good orientation $\vec{M}$. By Claim 3(a) and (b), G has a best-balanced orientation $\vec{G}$ compatible to $\vec{M}$. $\vec{G}$ and $\vec{M}$ define the orientations $\vec{G}_i$ of G_i and $\vec{M}_i$ of M_i for $0 \leq i \leq l$. Then, by construction, $\vec{G}_i$ and $\vec{M}_i$ are compatible. Since $\vec{M}_i$ is a good orientation of M_i, $\vec{G}_i$ is a best-balanced orientation of G_i by Claim 3(b). □

6 Corollaries

Theorem 6 implies the following result for global edge-connectivity.

Corollary 11 *For a subgraph H of G, H has an l-arc-connected orientation that can be extended to a k-arc-connected orientation of G if and only if H and G are $2l$- and $2k$-edge-connected respectively.*

Theorem 6 easily implies the following.

Corollary 12 *If H is an Eulerian subgraph of G, then any Eulerian orientation of H can be extended to a best-balanced orientation of G.*

Let G be a non-Eulerian graph. Then, by a theorem of Lovász [12], for *every* best-balanced orientation $\vec{G}$ *there exists* a pair of vertices x and y with $\lambda_{\vec{G}}(x,y) > \lambda_{\vec{G}}(y,x)$. In the following corollary we show that for *every* pair x,y of vertices with $\lambda_G(x,y)$ odd *there exists* a best-balanced orientation $\vec{G}$ with $\lambda_{\vec{G}}(x,y) > \lambda_{\vec{G}}(y,x)$.

Corollary 13 *Let $x,y \in V(G)$ with $\lambda_G(x,y) = 2k+1$. Then G has a best-balanced orientation $\vec{G}$ such that $\lambda_{\vec{G}}(x,y) = k+1$.*

Let us finish by a related conjecture (see in [5]) and a result on a special case of it.

Conjecture 14 *An undirected graph G has a k-vertex-connected orientation if and only if for every vertex set $X \subseteq V$ with $|X| \leq k$, $G - X$ is $(2k - 2|X|)$-edge-connected.*

Corollary 15 *Let $G = (V,E)$ be undirected graph and $u \in V$. Then G has an orientation such that for every ordered pair of vertices $(x,y) \in (V-u) \times (V-u)$ there are k arc-disjoint $(x,y)-$paths such that at most one of them contains u if and only if G is $2k$-edge-connected and $G-u$ is $(2k-2)$-edge-connected.*

7 Extensions of well-balanced orientation

7.1 r-arc-connected orientations

We can reformulate the well-balanced orientation theorem as follows.

Theorem 16 *Let $\mathbf{r} : V \times V \to \mathbb{Z}_0^+$ be a symmetric function. Then a graph $G = (V,E)$ has an* **r***-arc-connected orientation if and only if G is* 2**r***-edge-connected.*

In light of this form the following problem is a natural generalization of the well-balanced orientation theorem.

Problem 1 Given an undirected graph $G = (V,E)$ and a function $\mathbf{r} : V \times V \to \mathbb{Z}_0^+$, decide if G has an **r**-arc-connected orientation.

If the function **r** is symmetric then, by Theorem 16, there exists an **r**-arc-connected orientation of G if and only if $\lfloor \frac{\lambda_G(x,y)}{2} \rfloor \geq \mathbf{r}(x,y) \ \ \forall\, x,y \in V$. If the function **r** is not symmetric then the problem is difficult.

Theorem 17 *[7] Problem* 1 *is NP-complete.*

7.2 Mixed graphs

A possible way to prove the well-balanced orientation theorem could be to characterize mixed graphs whose undirected edges can be oriented to have a well-balanced orientation of the underlying undirected graph.

Problem 2 Given a mixed graph G', decide whether the undirected edges can be oriented in such a way that the directed graph $\vec{G}$ obtained satisfies $\lambda_{\vec{G}}(x,y) \geq \lfloor \frac{\lambda_G(x,y)}{2} \rfloor \ \ \forall\, (x,y) \in V \times V$, where G is the undirected graph obtained from G' by deleting the orientation of the directed edges.

Question 18 *Is Problem* 2 *NP-complete?*

8 Splitting off

Splitting off theorems are very useful in the proof of the global or local well-balanced orientation theorem. We also mention that Mader's proof [13] for the well-balanced orientation theorem as well as Frank's proof [5] for Theorem 7 uses Mader's splitting off theorem as well.

The odd vertex pairing theorem would be an easy task if the following was true.

Question 19 *For every* 2*-edge-connected graph G there exists a vertex s and a pair of adjacent edges rs,st such that for $G_{rt} := G - \{rs,st\} + rt$ we have:*

$$b_G(X) \geq b_{G_{rt}}(X) \ \ \forall X \subseteq V. \tag{10}$$

Counterexample 19 Let $G = (U,V;E)$ be the complete bipartite graph $K_{3,4}$. Let us denote the vertices as follows: $U := \{a,b,c,d\}$ and $V := \{x,y,z\}$. By symmetry, $\{rs,st\}$ is either $\{xd,dy\}$ or $\{az,zb\}$. In the first case $b_G(z) = 0 < 2 = b_{G_{xy}}(z)$ and in the second case $b_G(\{a,x,y\}) = 3 < 5 = b_{G_{ab}}(\{a,x,y\})$. In both cases (10) is violated. □

Question 20 *If $\vec{G}$ is a best-balanced orientation of $G := (V+s,E)$ and $\rho_{\vec{G}}(s) = \delta_{\vec{G}}(s)$ then there exist $rs,st \in A(\vec{G})$ so that for $\vec{G}_{rt} := \vec{G} - \{rs,st\} + rt$ we have*

$$\lambda_{\vec{G}_{rt}}(x,y) \geq \lambda_{\vec{G}}(x,y) \ \forall (x,y) \in V \times V. \tag{11}$$

The answer for Question 20 is false. For the counterexample see [9]. We note that our example is a smaller counterexample for a conjecture of Jackson than Enni's one [1].

Question 21 *If $\vec{G}$ is a best-balanced orientation of $G := (V+s,E)$ and $\rho_{\vec{G}}(s) = \delta_{\vec{G}}(s)$ then there exist $rs,st \in A(\vec{G})$ so that $\vec{G}_{rt}$ is a best-balanced orientation of G_{rt}.*

Question 21 is an open problem. However the following is true.

Theorem 22 *For every pair rs,st of edges of a graph $G := (V+s,E)$ there exists a best-balanced orientation $\vec{G}$ of G so that $rs,st \in A(\vec{G})$ and $\vec{G}_{rt}$ is a best-balanced orientation of G_{rt}.*

Question 23 *For every graph $G = (V+s,E)$ with $d(s) \geq 4$ there exist $rs,st \in E$ such that for every best-balanced orientation $\vec{G}_{rt}$ of G_{rt}, $\vec{G} := \vec{G}_{rt} - rt + rs + st$ is a best-balanced orientation of G.*

Question 23 is an open problem. If Question 23 has an affirmative answer, it would give us a possible way to prove the best-balanced orientation theorem.

9 Feasible pairing for connectivity functions

A set function $b : V \to \mathbb{R}$ is called **skew-submodular** if for every $X, Y \subseteq V$, at least one of the following two inequalities is satisfied:

$$b(X) + b(Y) \geq b(X \cap Y) + b(X \cup Y), \tag{12}$$

$$b(X) + b(Y) \geq b(X - Y) + b(Y - X). \tag{13}$$

A set function $p(X)$ is called **skew-supermodular** if $-p(X)$ is skew-submodular. We mention that, by [17], $\widehat{R}_G(X)$ is skew-supermodular and hence $b_G(X)$ is skew-submodular. A set function b on V is called **crossing submodular** if (12) is satisfied for every $X, Y \subseteq V$ with $X \cap Y$, $X - Y$, $Y - X$, $V - (X \cup Y) \neq \emptyset$.

Question 24 *Let $b : V \to \mathbb{Z}_0^+$ be a symmetric, skew-submodular function with $b(\emptyset) = 0$ and $b(X) \equiv |X \cap T_b| \bmod 2$, where $T_b = \{v : b(v) \text{ is odd}\}$. Then there exists a pairing M on T_b that satisfies*

$$d_M(X) \leq b(X) \quad \forall\, X \subseteq V. \tag{14}$$

Counterexample 24 Let $b(X)$ be defined on V with $|V| = 6$ as follows $b(X) := 0$ if $X = \emptyset, V$; 1 if $|X|$ is odd and 2 otherwise. It is easy to see that b satisfies all the conditions. Note that $T_b = V$. For any pairing M on T_b, we may choose $X \subset V$ with $d_M(X) = 3$ but then X violates (14). □

Note that, by Theorem 7, the answer for Question 24 is affirmative for $b(X) = b_G(X)$.

The problem corresponding to the global case is the following open problem.

Question 25 *Let $b : V \to \mathbb{Z}_0^+$ be a symmetric crossing submodular function with $b(\emptyset) = 0$ and $b(X) \equiv |X \cap T_b| \bmod 2$. Then there exists a pairing M on T_b that satisfies (14).*

Question 25 is an open problem.

Question 26 *Let $d : V \to \mathbb{Z}_0^+$ be a symmetric function that satisfies $d(\emptyset) = 0$ and $\forall X, Y \subseteq V$*

$$d(X) + d(Y) + d(X \triangle Y) = d(X \cap Y) + d(X \cup Y) + d(X - Y) + d(Y - X), \tag{15}$$

$$d(X) + d(Y) - d(X \cup Y) \text{ is even if } X \cap Y = \emptyset. \tag{16}$$

Let $\widehat{R} : V \to \mathbb{Z}_0^+$ be an even valued, symmetric, skew-supermodular function. Suppose that $\widehat{R}(X) \leq d(X)$ $\forall X \subseteq V$. Then there exists a pairing M on T_d that satisfies

$$d_M(X) \leq d(X) - \widehat{R}(X) \quad \forall\, X \subseteq V. \tag{17}$$

The answer for Question 26 is false. For the counterexample see [9]. Note that, by Theorem 7, Question 26 is true for $d(X) = d_G(X)$ and $\widehat{R}(X) = \widehat{R}_G(X)$.

Question 27 *Let $G = (V, E)$ be a graph and $\widehat{R} : V \to \mathbb{Z}_0^+$ an even valued, symmetric, skew-supermodular function. Suppose that $\widehat{R}(X) \leq d_G(X)$ $\forall X \subseteq V$. Then there exists a pairing M on T_G that satisfies*

$$d_M(X) \leq d_G(X) - \widehat{R}(X) \quad \forall\, X \subseteq V. \tag{18}$$

Question 27 is an open problem. If $\widehat{R}$ satisfies $\widehat{R}(X \cup Y) \leq \max\{\widehat{R}(X), \widehat{R}(Y)\}$ $\forall X, Y \subset V$ then $\widehat{R}(X) = \max\{\mathbf{r}(x, y) : x \in X, y \in V - X\}$ for some $\mathbf{r} : V \times V \to \mathbb{Z}_0^+$ and hence, by Theorem 7, such a pairing exists.

10 Feasible pairing defining a best-balanced orientation

Nash-Williams' original idea was that every feasible pairing provides a best-balanced orientation. In fact Theorem 7 shows that every feasible pairing provides lots of best-balanced orientations. A natural question is whether every best-balanced orientation can be defined by a feasible pairing.

Question 28 *For every best-balanced orientation $\vec{G}$ of G there exists a feasible pairing M and an orientation $\vec{M}$ of M such that $\vec{G} + \vec{M}$ is Eulerian.*

The answer for Question 28 is false. For the counterexample see [9]. The following theorem shows that Question 28 is true for global edge-connectivity.

Theorem 29 *Let $G := (V, E)$ be a $2k$-edge-connected graph and let $\vec{G} := (V, A)$ be a smooth k-arc-connected orientation of G. Then there is a pairing M of G and an orientation $\vec{M}$ of M so that*

$$d_M(X) \leq d_G(X) - 2k \quad \emptyset \neq \forall X \subset V \text{ and} \tag{19}$$

$$\vec{G} + \vec{M} \text{ is Eulerian.} \tag{20}$$

11 The polyhedron

In this section we consider the polyhedron of the fractional in-degree vectors of all orientations, k-arc-connected orientations and best-balanced orientations. Though the first two polyhedra are integral we show that the third one is not necessarily integral.

Let $G=(V,E)$ be a graph. Let us introduce the following polyhedra:

$$P_0 := \{x \in \mathbb{R}^{|V|} : x(X) \geq i(X) \qquad \forall X \subseteq V, \qquad x(V) = |E|\},$$
$$P_1 := \{x \in \mathbb{R}^{|V|} : x(X) \geq i(X)+k \qquad \forall \emptyset \neq X \subset V,\ x(V) = |E|\},$$
$$P_2 := \{x \in \mathbb{R}^{|V|} : x(X) \geq i(X) + \frac{\widehat{R}(X)}{2} \ \ \forall X \subseteq V,\ x(V) = |E|, \left\lfloor \frac{d(v)}{2} \right\rfloor \leq x(v) \leq \left\lceil \frac{d(v)}{2} \right\rceil \ \ \forall v \in V\}.$$

It is known that $m \in \mathbb{Z}^{|V|}$ is the in-degree vector of an orientation, of a k-arc-connected orientation, of a best-balanced orientation of G if and only if m belongs to P_0, P_1, P_2. Since $i(X)$ is supermodular and $i(X)+k$ is crossing supermodular, P_0 and P_1 are base-polyhedra and hence they are integral polyhedra, that is P_0 and P_1 coincide with the convex hull of the indegree vectors of all orientations and of k-arc-connected orientations of G.

Question 30 *The polyhedron P_2 is integral.*

The answer for Question 30 is false. For the counterexample see [9].

12 Matroid property

Question 31 *Let $l,u : V \to \mathbb{Z}_0^+$. Then there exists $\vec{G} \in \mathscr{O}_w(G)$ such that $l(v) \leq \rho_{\vec{G}}(v) \leq u(v)\ \forall v \in V$ if and only if there exist $\vec{G}^1, \vec{G}^2 \in \mathscr{O}_w(G)$ such that $l(v) \leq \rho_{\vec{G}^1}(v)\ \forall v \in V$ and $\rho_{\vec{G}^2}(v) \leq u(v)\ \forall v \in V$.*

The answer for Question 31 is false. For the counterexample see [9]. Question 31 is valid for the global case by Frank [2]. This follows from the facts that the in-degree vectors of k-arc-connected orientations form a base-polyhedron and for such polyhedra the linkage property holds.

Let $\vec{G}$ be an orientation of G. Let $T^+_{\vec{G}} := \{v \in V(G) : \rho_{\vec{G}}(v) > \delta_{\vec{G}}(v)\}$. Note that if $\vec{G}$ is smooth, then $|T^+_{\vec{G}}| = |T_G|/2$.

Question 32 $\mathscr{T} := \{T^+_{\vec{G}} : \vec{G} \in \mathscr{O}_b(G)\}$ *is the base family of a matroid.*

The answer for Question 32 is false. For the counterexample see [9]. Question 32 is true for the global case by the proof of Frank [3].

Question 33 *Let $\vec{G}^a, \vec{G}^b \in \mathscr{O}_w(G)$. Then there exist $\vec{G}^0 = \vec{G}^a, \vec{G}^1, \ldots, \vec{G}^l = \vec{G}^b \in \mathscr{O}_w(G)$ such that $\vec{G}^k$ is obtained from $\vec{G}^{k-1}$ by reversing a directed path or a directed cycle $(1 \leq k \leq l)$.*

Question 33 is an open problem. We mention that, by Frank [3], Question 33 is true for global edge-connectivity.

13 Subgraph chain property

In this section we consider a possible generalization of Theorem 6. The two questions correspond to the global and local cases.

Question 34 *Let G_3 be a subgraph of G_2 and G_2 a subgraph of G_1. Then, for $i=1,2,3$, there exists an orientation $\vec{G}_i$ of G_i such that $\vec{G}_j$ is an extension of $\vec{G}_i$ if $1 \leq i < j \leq 3$ and*

(a) Local case: $\vec{G}_i \in \mathscr{O}_w(G_i)$.

(b) Global case: *$\vec{G}_i$ is a k_i-arc-connected orientation of G_i provided that G_i is $2k_i$-edge-connected.*

The answers for Question 34 are false, even for the (b) part. For the counterexamples see [9].

References

[1] S. Enni, A note on mixed graphs and directed splitting off, *J. Graph Theory* **27** (1998) 312-221.

[2] A. Frank, On the orientations of graphs, *Journal of Comb. Theory/ Series B* **28** (1980) 251-261.

[3] A. Frank, A note on *k*-strongly connected orientations of an undirected graph, *Discrete Mathematics* **39** (1982) 103-104.

[4] A. Frank, On a theorem of Mader, *Discrete Mathematics*, **101** (1992) 49-57.

[5] A. Frank, Applications of submodular functions, *in: Surveys in Combinatorics, London Mathematical Society Lecture Note Series 187*, Cambridge Univ. Press, Ed.: K. Walker, (1993) 85-136.

[6] A. Frank, Connectivity augmentation problems in network design, *in: Mathematical Programming: State of the Art 1994, (Eds: J. R. Birge and K. G. Murty)*, The University of Michigan, Ann Arbor, MI, (1994) 34-63.

[7] A. Frank, T. Király, Z. Király, On the orientation of graphs and hypergraphs, *Discrete Applied Mathematics* **131** (2003) 385-400.

[8] H. N. Gabow, Efficient Splitting Off Algorithms for Graphs, *STOC 94*, (1994) 696-705.

[9] S. Iwata, T. Király, Z. Király, Z. Szigeti, On well-balanced orientations, counterexamples for related problems, *Egres Technical Reports*, **TR-2004-16** `http://www.cs.elte.hu/egres/`, 2004.

[10] Z. Király, Z. Szigeti, Notes on well-balanced orientations, *submitted to Journal of Combinatorial Theory/ Series B, Egres Technical Reports*, **TR-2004-10** `http://www.cs.elte.hu/egres/`, 2004.

[11] L. Lovász, Combinatorial problems and exercises, North-Holland, Amsterdam, (1979).

[12] L. Lovász, Connectivity in Digraphs, *J. Combin. Theory Series B* **15** (1973) 174-177.

[13] W. Mader, A reduction method for edge-connectivity in graphs, *Ann. Discrete Math.* **3** (1978) 145-164.

[14] W. Mader, Konstruction aller *n*-fach kantenzusammenhängenden Digraphen, *European J. Combin* **3** (1982) no. 1, 63-67.

[15] W. Mader, Ecken vom Innen- und Aussengrad *n* in minimal *n*-fach kantenzusammenhängenden Digraphen, *Arch. Math. (Basel)* **25** (1974) 107-112.

[16] K. Menger, Zur allgemeinen Kurventheorie, *Fund. Math.* **10** (1927) 96-115.

[17] C. St. J. A. Nash-Williams, On orientations, connectivity and odd vertex pairing in finite graphs, *Canad. J. Math.* **12** (1960) 555-567.

[18] C. St. J. A. Nash-Williams, Well-balanced orientations of finite graphs and unobtrusive odd-vertex-pairings, *Recent progress in Combinatorics (Proc. Third Waterloo Conf. on Combinatorics, 1968)* Academic Press, (1969) 133-149.

[19] H. E. Robbins, A theorem on graphs with an application to a problem of traffic control, *American Math. Monthly* **46** (1939) 281-283.

[20] personal communication

Enumerating Labeled Chordal Graphs on Complete Graph

MASASHI KIYOMI

National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
masashi@grad.nii.ac.jp

TAKEAKI UNO

National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
uno@nii.jp

Abstract: A chordal graph is a graph such that no its vertex subset induces a cycle composed of at least four edges. The class of chordal graphs contains many famous graph classes such as trees, interval graphs, and split graphs, and is also a large subclass of perfect graphs. In this paper, we address the problem of enumerating all chordal graphs included in the complete graph of n vertices. For this problem, we introduce an efficient enumeration scheme based on reverse search, and propose an algorithm taking constant time for each chordal graph.

Keywords: chordal graph, enumeration, constant time

1 Introduction

A chordal graph is an undirected graph in which every cycle with at least four edges has a chord. Here a chord of a cycle is an edge connecting two vertices of the cycle but not an edge of the cycle. Chordal graphs have been considered to be important in the sense of both theoretical and application aspects. The class of chordal graphs contains many popular graph classes such as trees, interval graphs, and split graphs[6], and many polynomial time algorithms to solve combinatorial problems on chordal graphs are known. Moreover, the class of chordal graphs is a large subclass of the class of perfect graphs[8]. Chordal graphs have many applications, for example, to matrix computation [4] or to relational database[5]. Chordal graphs are sometimes called triangulated graphs since every cycle in them is divided into some triangles. Chordal graphs are used for numerical computation of models, and modeling some systems in computer science and social sciences[19].

Chordal graphs have many good properties. One important property of them is that a chordal graph has at least one *simplicial vertex*, where a vertex is simplicial if its neighbors induce a clique. The removal of a simplicial vertex from a chordal graph results a chordal graph, hence we can iteratively remove simplicial vertices from a chordal graph until the graph has no vertex. The ordering of vertices in the removals is called *perfect elimination ordering*. Given a general graph, a perfect elimination ordering can be found in linear time, if it exists[14]. Chordal graphs are characterized by perfect elimination orderings; a graph is chordal if and only if it has a perfect elimination ordering.

These properties of chordal graphs give many simple and fast algorithms for combinatorial problems. In particular, some combinatorial problems on chordal graphs, such as maximum independent set and minimum vertex coloring, can be solved in polynomial time with simple combinatorial algorithms. However, no polynomial combinatorial algorithm has been known for perfect graphs. This is a reason that the class chordal graphs is an important subclass of perfect graphs.

Recently, because of the increase of computation power, many problems in practice are solved with enumeration. The graphical structures are used to model the systems and objects in many scientific area. Enumeration is used for optimizing, simulate and to find good properties and new knowledges in these models. For example, enumeration algorithms for graph structures such as labeled paths, labeled trees, labeled graphs are used in frequent pattern mining problems [16, 2, 1, 9]. Enumeration itself has theoretical interests, and many studies have been done[7].

Here, we address the problem of enumerating connected chordal subgraphs included in the given complete graph K_n of n vertices. Here we assume that every vertex in K_n has an index (label) different from the other vertices, and identify two graphs of different edge sets even if they are isomorphic. Thereby, the problem is to enumerate all subsets of the edge set of K_n which induce connected chordal graphs. We here call such chordal graphs induced by edge subsets *labeled chordal graphs*. We note that it is easy to extend our algorithm to deal with chordal subgraphs given by a pair of a vertex subset and an edge subset.

In this paper, we propose an efficient algorithm for the problem. The memory complexity of the algorithm is $O(n^2)$, and the time complexity of the algorithm is proportional to the number of labeled chordal graphs in K_n, i.e., the algorithm takes constant time for each labeled chordal graph on average. Our algorithm is based on reverse search. Reverse search was originally developed to enumerate all vertices of polytopes in [3], and used to enumerate many other combinatorial objects such as spanning trees, trees, plane graphs, maximal cliques, etc.[15, 12, 13, 11]. The basic idea of reverse search is to

implicitly construct a search tree on the objects to be enumerated, and perform depth first search on the search tree. We developed a good search tree on the labeled chordal graphs, and an efficient algorithm which performs depth first search taking constant time on average to trace an edge of the search tree.

In the following section, we describe our problems and algorithms in detail. Section 2 describes a framework of reverse search and our enumeration scheme for connected labeled chordal graphs. In section 3, we describe our enumeration algorithm, and analyze the time complexity in Section 4. Finally we conclude the paper in Section 5.

2 Enumeration Scheme

In this section, we explain a basic concept of reverse search, then describe our enumeration scheme for connected labeled chordal graphs.

2.1 Reverse Search

Let $\mathscr{F}$ be the set of object to be enumerated. In our problem, $\mathscr{F}$ is the set of connected labeled chordal graphs included in K_n. We introduce a parent–child relation by defining a parent for each object except for some specified objects called *root objects*. The definition of the parents has to satisfy that no object is a proper ancestor of itself, i.e., by iteratively moving from an object x to the parent of x, to the parent of the parent of x, and so on, we never come to the start object x again. Then, the graph representation of the relation induces a set of disjoint rooted trees spanning all objects, in which paths from all leaves aim to the roots. We illustrate an example of the graph representation in Figure 1. Each object to be enumerated is drawn by a point, and an object and its parent is connected by a directed arrow.

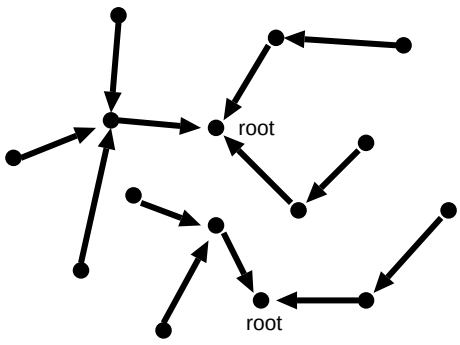


Figure 1: Spanning forest in which paths from all leaves aim to the roots

By tracing each edge in the opposite direction, we can perform depth first search visiting all objects. Thus, we can enumerate all objects by using an algorithm to find all root objects, and an algorithm to find all children of an object. This is a basic concept of reverse search. Since it is very simple and general, many enumeration algorithms use reverse search [3, 12, 13, 11].

2.2 Parent–Child Relation

To construct a reverse search algorithm for connected labeled chordal graphs, we need parent–child relations defined on the connected labeled chordal graphs.

Let K_n be the complete graph with n vertices. Suppose that the vertex set of K_n is $\{1,...,n\}$. Let $G=(V,E)$ be a chordal subgraph of K_n such that G has more than one edges. We define *minimum degree simplicial vertex* of G as the simplicial vertex having the minimum degree, and denote it by $s^*(G)$. If there are more than one such simplicial vertices, we choose the minimum (in vertex indices) as $s^*(G)$, so that $s^*(G)$ is defined uniquely. Note that any chordal graph has at least one simplicial vertex. Then, we define the parent of G by the removal of $s^*(G)$ from G. Since the removal of a simplicial vertex results a chordal graph, the parent of G is also a chordal graph. Moreover, if G is connected, then the parent of G is also connected since any two neighbors of a simplicial vertex are connected by an edge (See Figure 2). The number of edges on the parent chordal graph is always strictly less than that of its child. Therefore, a chordal graph never be an ancestor of itself in this parent–child relation. Thus, we can see that both the parent–child relation defined on all labeled chordal graphs of K_n and that defined on all connected labeled chordal graphs of K_n satisfy the conditions to be used for reverse search. The root connected chordal graphs are the graphs with exactly one edge. We illustrate an example of the parent–child relation of connected labeled chordal graphs in Figure 2.

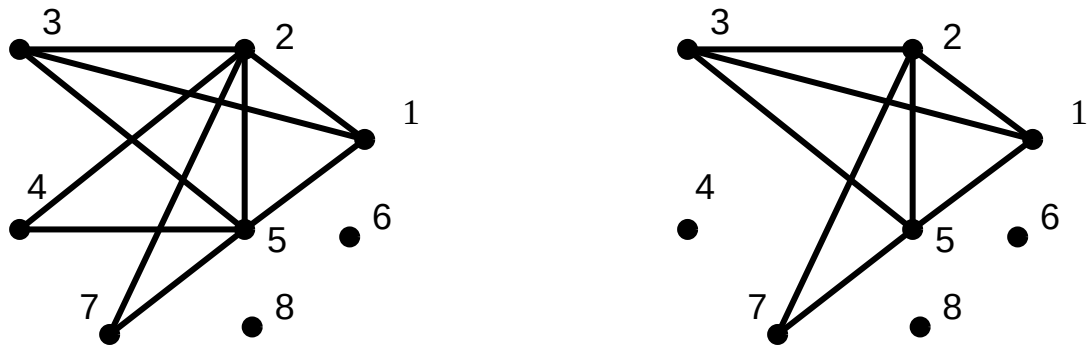


Figure 2: The left chordal graph has simplicial vertices 1, 3, 4 and 7. The simplicial vertices of the minimum degree are 4 and 7. The minimum degree simplicial vertex is 4. The right chordal graph obtained by removing 4 from the left graph is the parent of the left graph.

3 Algorithms for Reverse Search

In this section, we describe an algorithm to enumerate all root connected labeled chordal graphs, and an algorithm to enumerate all children of a given connected labeled chordal graph. The former algorithm is very simple; just generate all subgraphs having exactly one edge. In the following subsection, we describe the latter algorithm.

3.1 Enumerating Children

The parent of a connected labeled chordal graph is obtained by removing a simplicial vertex. Hence, given a connected labeled chordal graph G, any of its child can be obtained by adding a vertex v and edges connecting the vertex v and the other vertices C in G. A necessary condition to obtain a child is that C contains at least one vertex, and is a clique so that v is a simplicial vertex of the child. In the following, we characterize a necessary and sufficient condition to obtain a child.

We first introduce some notations. We denote the set of simplicial vertices in G by $S(G)$. The minimum degree in $S(G)$ is denoted by $k(G)$. Let $S_d(G)$ be the set of simplicial vertices of degree d, and particularly, we denote $S_{k(G)}(G)$ by $S^*(G)$. We denote the minimum vertex in a vertex set X by $\min(X)$. If $X = \emptyset$, we define $\min(X)$ by $+\infty$.

Let $G = (V,E)$, be a connected labeled chordal graph included in K_n. Let v be a vertex of K_n and not in G, and C be a vertex set in G. We denote by $G(v,C)$ the graph obtained by adding v and edges connecting v and all vertices in C. From the observation above, it is necessary that any child H of G satisfies that $H = G(v,C)$ for some v and C, and C is a clique of G.

Lemma 1 *For a vertex $v \notin G$ and a clique C in G, $G(v,C)$ is a child of G if and only if one of the following condition holds.*
(1) $|C| < k(G)$
(2) $|C| = k(G)$ *and* $v < \min(S^*(G) \setminus C)$
(3) $|C| = k(G)+1$, $S^*(G) \subseteq C$, *and* $v < \min(S^*(G) \cup S_{k(G)+1}(G))$.

PROOF: First, we claim the following propositions.

Proposition 2 *Any simplicial vertex $u(\neq v)$ in $G(v,C)$ is simplicial in G.*

PROOF: If u is not adjacent to v, the neighbors of u forms a clique in G. If u is adjacent to v, the removal of v from the neighbors of u also forms a clique in G. Thus, in both cases, u is simplicial in G. □

Proposition 3 *$G(v,C)$ is a chordal graph, and v is simplicial in $G(v,C)$.*

PROOF: Let X be a cycle in $G(v,C)$ having at least four edges. If X does not include v, then X is included in G, hence X has a chord. If X includes v, then X includes at least two neighbors of v. The edge connecting the two neighbors is a chord of X, thus any cycle in $G(v,C)$ of at least four edges has a chord. Since the neighbors of v is C, v is simplicial in $G(v,C)$. □

Now we prove the statement. From these two propositions, we see that $G(v,C)$ is a child of G if and only if $s^*(G(v,C)) = v$. Thus, to prove the statement, we only need to check that the conditions hold or not in the case of (1), (2) and (3). We consider the following four cases according to the size of C.

(a) $|C| < k(G)$ holds: The degree of v in $G(v,C)$ is smaller than any other simplicial vertex in G. From proposition 2, any simplicial vertex in $G(v,C)$ is a simplicial vertex in G, hence v is the unique minimum degree vertex among simplicial vertices in $G(v,C)$. Thus, $s^*(G(v,C)) = v$.

ALGORITHM Enum_Labeled_Chordal ($G = (V,E)$):
1: **output** G;
2: **if** $|V| = n$ **then return;**
3: **if** $k(G) = 1$ **then do**
  for each pair of vertices $v \notin G$ and $u \in G$ such that $v < \min(S^*(G) \setminus \{u\})$
    call Enum_Labeled_Chordal ($G(v,\{u\})$);
  return;
 end;
4: compute $S_d(G)$ and $\min(S_d(G))$ for each d such that $S_d(G) \neq \emptyset$;
5: **for each** clique C of sizes at most $k(G)-1$ in G
  call Enum_Labeled_Chordal ($G(v,C)$) for each $v \notin G$;
6: **for each** pair of vertex $v \notin G$ and clique C of size $k(G)$ such that $v < \min(S^*(G) \setminus C)$
  call Enum_Labeled_Chordal ($G(v,C)$);
7: **for each** pair of vertex $v \notin G$ and clique C of size $k(G)+1$ such that $S^*(G) \subseteq C$ and
      $v < \min(S^*(G) \cup S_{k(G)+1}(G))$
  call Enum_Labeled_Chordal($G(v,C)$);

Figure 3: Algorithm to enumerate connected labeled chordal graphs

(b) $|C| = k(G)$ holds: From the similar observation to the above, v has the minimum degree among simplicial vertices in $G(v,C)$ but not always unique. Since, the degree of vertices of C in $G(v,C)$ is larger than the degree of vertices of C in G, $S^*(G(v,C)) = (S^*(G) \setminus C) \cup \{v\}$. Thus, $s^*(G(v,C)) = v$ if and only if $v < \min(S^*(G) \setminus C)$.

(c) $|C| = k(G)+1$ holds: v has the minimum degree in $S(G(v,C))$ if and only if $S^*(G) \subseteq C$. Thus, $s^*(G(v,C)) = v$ if and only if $S^*(G) \subseteq C$ and $v < \min(S^*(G) \cup S_{k(G)+1}(G))$ hold.

(d) $|C| > k(G)+1$: In this case, $C \cap S^*(G) = \emptyset$ since any vertex in $S^*(G)$ is adjacent to exactly $k(G)$ vertices. Thus, $s^*(G(v,C)) = s^*(G)$.

From these observations, we can see that for a vertex v not in G and a clique C in G,

- if (1), (2) or (3) holds, $G(v,C)$ is a child of G, and
- if (1), (2) and (3) do not hold, i.e., one of
  (2') $|C| = k(G)$ and $v > \min(S^*(G) \setminus C)$,
  (3') $|C| = k(G)+1$ and $S^*(G) \setminus C \neq \emptyset$,
  (3") $|C| = k(G)+1$ and $v > \min(S^*(G))$, or
  (4') $|C| > k(G)+1$ holds,

then $G(v,C)$ is not a child of G. □

From the proof of the lemma, we obtain the following corollary.

Corollary 4 *If $G(v,C)$ is a child of G, $k(G(v,C)) = |C|$.*

Lemma 1 characterizes the children of a connected labeled chordal graph efficiently. Using the lemma, we obtain an algorithm for enumerating the children of a connected labeled chordal graph, described in Figure 3.

To characterize the children for the parent–child relation on general labeled chordal graphs, we just modify the condition "C has to be a clique of G" to "C has to be a clique of G, or a singleton of a vertex not in $G \cup \{v\}$". Then, the statements of the lemma and the corollary stated in this subsection hold by similar observations.

3.2 Enumerating Cliques in Chordal Graphs

The algorithm described in the previous subsection needs an algorithm for enumerating cliques in a chordal graph. Cliques in a general graphs can be enumerated in $O(|V|)$ time for each by simple backtracking algorithms. However, this algorithm cannot be used to obtain constant time enumeration algorithms. Here, we describe a new algorithm for enumerating all cliques in a chordal graph $G = (V,E)$.

Let v be a simplicial vertex of G. We consider a partition of the set of cliques in G; all cliques including v, and all cliques not including v. Since the neighbors of v form a clique, v and its neighbors induce a complete graph. Thus, the cliques including v can be enumerated by generating all subsets of the neighbors of v. This can be done in constant time for each. The cliques not including v can be enumerated recursively by enumerating all cliques in the graph obtained by removing v from G. We choose the vertex v in each level of the recursive calls along a perfect elimination ordering. Hence, we need only

constant time to obtain a simplicial vertex in each level of the recursive calls. All subsets of a set can be generated in certain ordering such as Gray code [17] so that the difference between a subset and the next is constant size on average. Thus, we obtain the following theorem and corollary.

Theorem 5 *All cliques in a chordal graph can be enumerated in constant time for each clique on average so that the size of the difference between an output clique and the previously output clique is constant on average.*

Corollary 6 *All cliques of sizes at most k in a chordal graph can be enumerated in constant time for each clique on average and additional time to obtain a perfect elimination ordering so that the size of the difference between an output clique and the previously output clique is constant on average.*

4 Time Complexity

We evaluate the computation time of an iteration to bound the time complexity of our algorithm. To show the time complexity, we bound the computation time of an iteration. Here we define an iteration of the algorithm by the operation in a vertex in the computation tree, which is a tree representation of the recursive structure of an execution of the algorithm. Thus, an iteration corresponds to the operations in an execution of ALGORITHM Enum_Labeled_Chordal excluding the operations in the recursive call generated by it. Iterations and connected labeled chordal graphs have a one-to-one correspondence, thus we call an iteration inputting a chordal graph G *iteration of* G.

Our goal to the analysis of the time complexity is to bound the computation time of the iteration of any $G = (V,E)$ by linear in the number of children of G. Since the sum of the number of the children over all connected labeled chordal graphs is less than the number of all connected labeled chordal graphs, the statement assures that the computation time of the algorithm is constant for each connected labeled chordal graph on average.

Here we bound the computation time of each step of an iteration one by one. We can see that step 1 and step 2 in figure 3 can be done in constant time. From Corollary 4, we can compute $k(G)$ in constant time, thus the conditional branch in step 3 is performed in constant time.

To execute step 3 quickly in the case $k(G) = 1$, we maintain a sorted list of the vertices in $S_1(G)$ at every iteration. We can delete a vertex from the list in constant time. When the algorithm construct $G(v,C)$ and adds v to $S_1(G)$ in some iterations, v is always smaller than any vertex in $S_1(G)$. Thus, we can add a vertex v to $S_1(G)$ in constant time by attaching v to the head of the list.

To compute $S_d(G)$ and $\min(S_d(G))$ in step 4, we take $O(|V|)$ time. When step 4 is executed, we have $k(G) \geq 2$. Thus, the set of cliqeus of size at most $k(G)-1$ includes the cliques of size one. Thereby G has at least $|V|$ children. Therefore the computation time for step 4 is order of the number of children of G. Step 7 can be also done in $O(|V|)$ time, since at most one clique satisfies the condition of step 7.

The enumeration of the cliques needs a perfect elimination ordering. Since in each iteration the algorithm adds a simplicial vertex to the graph, the ordering of the vertices added to obtain G forms a perfect elimination ordering. Thus, we can keep a perfect elimination ordering of the current operating graph in memory, and update it in constant time at each iteration.

Step 6 takes long time in a straightforward way. To avoid this, we use cliques C' of size $k(G)-1$ found in step 5. We find all vertices $u \in G$ such that there is a vertex $v \notin G$ satisfying $v < \min(S^*(G) \setminus (C' \cup \{u\}))$. To satisfy the condition, $S^*(G) \setminus C'$ includes at most one vertex smaller than the minimum vertex not included in G. This can be checked in $O(|C'|)$ time.

Here we see that steps 5, 6 and the maintenance of the sorted list of the vertices in $S_1(G)$ takes $O(|C|)$ time for each child. The time to compute $G(v,C)$ in step 5 can be reduced by using $G(v,C')$ where C' is the previously obtained clique. By modifying $G(v,C')$ to obtain $G(v,C)$, the computation time is reduced to $O((C \setminus C') \cup (C' \setminus C))$. Thus, from Corollary 6, the reduced computation time is constant time for each on average. By similar observation, we can see that computation time for steps 5, 6 and the maintenance of the sorted list of the vertices in $S_1(G)$ is bounded by constant time for each child on average. Therefore, we obtain the following theorem.

Theorem 7 *For a given complete graph K_n, all connected labeled chordal graphs, which are equivalent to all edge subsets of K_n inducing connected chordal graphs, can be enumerated in constant time for each edge subset, and in $O(n^2)$ memory.*

As a corollary, we have the following statement.

Corollary 8 *For a given complete graph K_n, all labeled chordal graphs, which are equivalent to all edge subsets of K_n inducing chordal graphs, can be enumerated in constant time for each edge subset, and in $O(n^2)$ memory.*

5 Conclusion

In this paper, we addressed the problem of enumerating all connected labeled chordal graphs on a vertex labeled complete graph K_n. We introduced a good search tree on the set of connected labeled chordal graphs, and presented an efficient

algorithm taking constant time on average for each connected chordal graph. As corollaries, we showed that all labeled chordal graphs (not needed to be connected) can be enumerated in constant time for each on average, and all cliques in a chordal graph can be enumerated in constant time for each on average. We leave the following problem as an open problem.

- Is there an output polynomial time algorithm for enumerating all maximal connected labeled chordal subgraph of a given complete graph or of a given general graph?

References

[1] R. Agrawal and R. Srikant, Fast algorithms for mining association rules in large databases, *Proceedings of VLDB '94*, (1994), 487–499.

[2] T. Asai, K. Abe, S. Kawasoe, H. Arimura, H. Sakamoto, S. Arikawa, Efficient Substructure Discovery from Large Semi-structured Data, *Proceedings of Second SIAM International Conference on Data Mining 2002* (SDM'02), (2002) 158–174.

[3] D. Avis and K. Fukuda, Reverse search for enumeration, *Discrete Applied Mathematics*, Vol. 65, (1996) 21–46.

[4] C. Beeri, R. Fagin, D. Maier, and M. Yannakakis, On the Desirability of Acyclic Database Schemes, *Journal of the ACM*, 30, (1983) 479–513.

[5] J. R. S. Blair and B. Peyton, An Introduction to Chordal Graphs and Clique Trees, *Graph Theory and Sparse Matrix Computation*, IMA 56, Springer, (1993) 1–29.

[6] A. Brandstädt, V. B. Le, and J. P. Spinrad. *Graph Classes: A Survey*, SIAM (1999).

[7] L. A. Goldberg, *Efficient algorithms for listing combinatorial structures*, Cambridge University Press, New York, (1993).

[8] M. C. Golumbic, Algorithmic Graph Theory and Perfect Graphs, *Annals of Discrete Mathematics* 57. Elsevier, 2nd edition (2004).

[9] A. Inokuchi, T. Washio, H. Motoda, An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data, *Lecture Notes in Computer Science*, 1910, Springer, (2000).

[10] D. S. Johnson, M. Yanakakis and C. H. Papadimitriou, On generating all maximal independent sets, *Information Proceeding Letters*, Vol. 27, (1998) 119–123.

[11] K. Makino and T. Uno, New Algorithms for Enumerating All Maximal Cliques, *Lecture Notes in Computer Science*, 3111, Springer, (2004) 260–272.

[12] S. Nakano, Enumerating Floorplans with n Rooms, *Lecture Notes in Computer Science*, 2223, Springer, (2001) 107–115.

[13] S. Nakano, Efficient Generation of Triconnected Plane Triangulations, *Computational Geometry Theory and Applications*, Vol. 27(2), (2004) 109–122.

[14] D. J. Rose, R. E. Tarjan, and G. S. Lueker, Algorithmic Aspects of Vertex Elimination on Graphs, *SIAM Journal on Computing*, Vol. 5(2), (1976) 266–283.

[15] A. Shioura, A. Tamura, T. Uno, An Optimal Algorithm for Scanning all Spanning Trees of Undirected Graphs, *SIAM Journal on Computing*, Vol. 26(3), (1997) 678–692.

[16] K. Taniguchi, H. Sakamoto, H. Arimura, S. Shimozono and S. Arikawa, Mining Semi-Structured Data by Path Expressions, *Lecture Notes in Artificial Intelligence*, Springer, 2226, (2001) 378–388.

[17] C. Savage, A Survey of Combinatorial Gray Codes, *SIAM Review*, Vol. 39, (1997) 605-629.

[18] S. Tsukiyama, M. Ide, H. Ariyoshi and I. Shirakawa, A new algorithm for generating all the maximal independent sets, *SIAM Journal on Computing*, Vol. 6 (1977) 505–517.

[19] J. Whittaker, *Graphical Models in Applied Multivariate Statistics*, Chichester: Wiley (1991).

A Generic framework for plagiarism detection in programs

GERGELY LUKÁCSY, PÉTER SZEREDI

Department of Computer Science and Information Theory
Budapest University of Technology and Economics
1117 Budapest, Magyar tudósok kőrútja 2., Hungary
lukacsy@cs.bme.hu, szeredi@cs.bme.hu

Abstract: The paper presents a plagiarism detection framework, which aims to determine the similarity degree of program source codes. The issue of plagiarism detection has been considered earlier for written material, such as student essays. For these, text-based algorithms have been published. We argue that in case of program code comparison, structure based techniques may be much more efficient. The main idea is to transform the source code into mathematical objects, use appropriate reduction and comparison methods on them, and interpret the results appropriately. We have designed a generic program structure comparison framework and implemented it for the Prolog and SML programming languages. We have been using the implementation to successfully detect plagiarism in student assignments since 2000.

Keywords: plagiarism, program source, graph similarity

1 Introduction and motivation

Comparison of essays and other written materials has been in focus in recent years [14]. Detecting plagiarism in written materials is an issue in education as well as in law procedures. World wide public polls show that two-third of university students have used other people's ideas in an impermissible way, at least once during their studies. Law disputes include the famous SCO-IBM debate over the allegedly unauthorized use of portions of the AIX operating system in Linux.

Regrettably, several sites on the Internet provide free (or low cost), quick and efficient access to written materials of many types. Unbelievably, sites such as `CheatHouse`* or `SchoolSucks`† proudly provide tons of essays, dissertations, reports, etc. for students looking for an easy way to have their assignment of some sort fulfilled. We do agree that it is a good idea to get acquainted with the area one is interested in by reading similar materials. However inspiring someone to cheat is a different issue.

In case of programming assignments, duplication of the programs or parts of them is a real issue. During the compulsory course "Declarative Programming" at the BUTE we expect the students to hand in a major programming assignment at the end of the semester. This means mass amount of source code year by year.

Checking these programs by hand seems to be beyond possibility. Having n programs we should check $\frac{n*(n-1)}{2}$ pairs to have all the cases covered. Notice, that we really should check all of the pairs, because the *similar* relation between programs is not transitive. This practically means, that even if we know that source A is similar to source B and source B to source C we cannot draw any direct conclusion about the similarity degree of source A and C.

Luckily, in our particular case several assignments can be excluded from the whole set. For example, we do not care whether two bad solutions (a solution is bad if it does not solve a certain percentage of our given test cases) are similar or not. However we still have $O(n^2)$ pairs to test manually, where the average value of n is greater that 100.

Our aim was to develop methods and tools to assess the similarity of programs in order to narrow down the need for manual testing to an acceptable amount. We have defined the notion of a *similarity degree* specifying how much two programs match. For the methods to be generic and flexible enough we have developed a *multi phase comparison framework*.

The actual comparison is performed between mathematical entities where the meaning of similarity can be formally specified. These entities are generated from the programs to be compared. The procedure may differ in case of different programming languages, so separate front-end modules should be developed for each language. Naturally, the mathematical entities must be generic and powerful enough to be able to capture the semantics of the different languages. We have chosen directed, labelled graphs for this purpose. Now, the comparison of source programs is performed by actually determining the similarity measure of corresponding graphs. Notice that it is thus also possible to determine the similarity degree of two source programs written in different languages.

*`http://www.cheathouse.com`
†`http://www.schoolsucks.com/`

The framework is designed to be parametrizable, so that it can be used efficiently under varying circumstances. For instance, in case of shorter programs a different similarity threshold may be more appropriate than in the case of longer ones. Moreover, it turned out that applying certain well selected simplifying transformations — we call them *reductions* — to the graphs has favorable effects on the efficiency of the approach. Such reductions include dropping specific nodes and edges and thus creating higher level, more abstract views of the programs.

The structure of the paper is as follows. First, we give a brief comparison of our approach with other ongoing research work. Next, the paper gives an overview of the proposed framework and introduces the main concepts. Following this, the paper describes the three components of the framework: the *front-end module*, the *simplifier* and the *comparator*. The next section describes our example implementation of the framework for Prolog programs. Finally, we summarize the conclusions.

2 Related work

While several solutions exist for checking plagiarism in written documents (like iThenticate [10], FindSame [6], CopyCatch [5] or SCAM [13]), the same cannot be said for program sources. The reason for this is that it is widely believed that detecting plagiarism in programs is much easier than in free text. This is because programming languages are formally well defined and unlike the case of free text, it is assumed that people use only a few tricks to hide the fact of plagiarism.

Alan Parker and James Hamblen in [12] explicitly say that copied software is "a program which has been produced from another program with a small number of routine transformations". These routine transformations include modifying the comments, changing the names of the variables or (in the worst case) changing the control structures (using `while` instead of `for`). The suggested technique is the following.

- Get rid of every comment in the source code
- Get rid of every useless new line, white space, etc.
- Use a normal UNIX diff program, which compares the files line by line, for each pair of source codes
- Examine the results

In [7] J. A. Faidhi and S. K. Robinson suggested a diagram which defines the level of plagiarism (L0-L6) based on what kind of modifications the cheater used. For example, we obtain L1 from L0 by modifying the comments, L2 from L1 by further modifying the variable names as well, etc.

Most of the existing software is based on statistical or lexicographic approach where they compare identifiers to identifiers for example to determine how similar they are. Such programs are DUP [3], SIM [8], SIFF [11] or Bandit [15].

Some early ideas suggested the use of structure based approaches. For example in [4] J. M. Bieman and N. C. Debnath proposed building program graphs, while T. J. McCabe assigned a number to each source code according to its complexity (which was based on the number of computation paths available within the program).

Existing structure based programs include the Plague [16], the YAP series [17] and the Moss [1] program. Plague creates so called structure profiles for source codes and compares them. YAP was tested against Plague and found to be more efficient in case the students used more sophisticated tricks. Perhaps the lack of more detailed information about these programs can be attributed to the fact that the more details are known, the easier is for the students to cheat.

In the following we present our framework for detecting plagiarism in source codes.

3 The framework

The proposed framework consists of three main tasks which are handled by independent program modules :

1. source code to model mapping (front-end)
2. model reduction (simplifier)
3. model comparison (comparator)

The *front-end* creates a mathematical entity — which we call a **model** or an **abstract view** — from the source program to be examined. These views can be reduced in many ways by the *simplifier*, creating different **abstraction levels**. We use the *comparator* to compare models on the same level and determine a similarity degree (a number between 0 and 1) on that abstraction level. As the abstraction level becomes higher, the similarity of the abstract views is less and less indicative of the similarity of the original code. Therefore we assign a factor (again a number between 0 and 1) to each abstraction level, with which we multiply the similarity degree obtained.

Figure 1 shows the overview of the proposed framework.

In the following we present the details of the main parts of the framework.

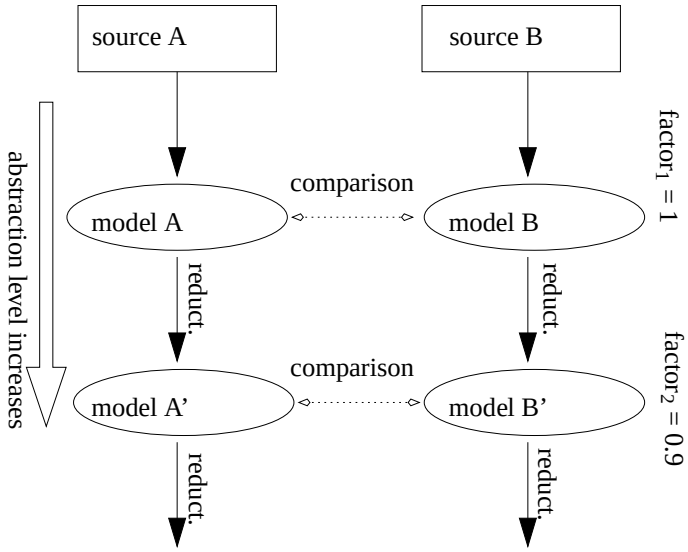


Figure 1: Overview of the proposed framework

3.1 Source code to model mapping

In general, the entity corresponding to a program source code can be chosen arbitrarily. For example, let us consider the size of the program source as an abstract entity characterizing the program, and consider the advantages and disadvantages of this choice. It is true that if we examine two entirely identical programs, then the comparison of their abstract views will signal match (the sizes of the programs will be the same). It also sounds feasible to consider the two program instances suspicious, if their sizes are the same. However, if the programs are similar, but not identical, then the program size abstraction cannot give any hint on their similarity.

A further issue is that of simplifying transformations. When a program is characterized by its size, practically no further simplifications can be applied. The only, very weak option is to make further abstractions by rounding the size, e.g. using 1 kbyte instead of 1324 bytes.

Therefore, the abstract view should be more sophisticated (to allow diverse abstraction levels) and more importantly, it should be possible to draw conclusions on the similarity of the programs from the similarity of the abstract views.

Therefore we suggest the use of directed, labelled graphs as the abstract views characterizing the programs. Here the meaning of nodes and edges may vary from implementation to implementation. For example, the abstraction may be the program call graph, the data-flow graph of an execution, or — in case of object-oriented languages — the graph describing the object structure.

The graph representation is general enough to describe any kind of entity. Even our first example, the “program size” abstraction, can be described as a labelled graph (with a single node whose label is the size).

We now discuss the most significant student tricks which a good abstraction must be resistant to. We use Prolog language examples to illustrate various such cheating attempts. However, we believe that very similar examples can be made for other languages as well.

Changing the names of identifiers and variables is the most common trick. A piece of source code which contains only one letter variable names may look rather complicated and tangled. However, it can be easily transformed into a program which uses talkative names. For humans, sometimes only this is enough to hide plagiarism. A similar trick is to change the natural language in which the program identifiers are formulated: use English names in one program and use another language in the other. It is also possible to change not just the variable, but the function and/or predicate names, too. For example, it is very easy to transform a predicate head

```
solve_the_problem(Input_data, Results)
```

...to the following:

```
do(Input, Output)
```

One can also change the number of arguments (the arity) of the predicates, without affecting the code. For example, one can use dummy parameters, which are set to something irrelevant at call time. If one changes not only the name of a function, but also its arity, it may become really difficult for the human to recognize that it is equivalent to some other function.

Sometimes it is profitable for students to cut the code into several pieces and place them into separate files. Similarly, reordering the sequence of the functions/predicates in a source file or reordering even the body of a predicate is an easy, but effective trick.

Putting useless functions into the code may be also misleading. For example, a student can "borrow" some code from another program which has nothing to do with the current assignment. Automatic methods may find this disturbing, because this technique introduces new variables and functions, changes the size of the file, etc. Sometimes we can recognize this trick by analyzing the source code and detecting that these functions are never called, but this is not true in general.

Consider the following example, where the `calculate/2` predicate will never be called, because the value of variable `X` will always be greater than 35. This predicate can be anything, most likely a piece of some big code, with the only aim to conceal the fact that the original source code was made by some other individual.

```
% homework made by XY
:- use_module(library(lists)).
.
.
.
borrowed_predicate(A, B) :-
    A > 0,
    ...
    X is A + 35,
    ...
    (
      X < 0 ->
      calculate(X, B)
    ;
      true
    ),
    ...
```

Those parts of the program which are never called can only be detected at run time. Unfortunately, even if we detect such code fragments it does not mean that we have found an instance of plagiarism. Sometimes such code is simply the result of programming errors, which even the author of the program is unaware of.

Analogously to placing useless predicates in the program code, we can place useless calls in the body of a predicate without changing its working. In the following example we show two totally useless lines, which always succeed:

```
    ...
    C is 2, B = [1,2], A is 3-C ,
    ...
    memberchk(A, B),
    ...
```

Finally, we show two tricky, but easily implementable types of program transformation. The first is called *call-tunneling*, while the second is *call-grouping*. Call tunneling is based on the idea that instead of letting function `A` to call function `C` directly, we insert an intermediate function `B`. In this new scenario `A` calls `B` and `B` calls `C`. If function `B` returns what it got from `C` without any modification, then the transformed program will be equivalent to the original one. Call-tunneling is very hard to detect, because, for example, function `B` is actually called during the execution, therefore it seems to be an important part of the program.

Call-grouping is a simple technique to significantly modify the structure of a program even if one does not really understand how the code actually works. The main idea is very similar to the one presented in call-tunneling. If there is a predicate which calls several others, we can regroup these calls to produce code with a totally different look. Let us consider the following piece of code:

```
original_predicate(A, B, C) :-
  call1(A, T),
  call2(B, T, Q),
  call3(Q, E),
  call4(A, E, Z),
  call5(Z, C).
```

Using call-grouping one can transform it to the following, equivalent program.

```
original_predicate(A, B, C) :-
  temp1(A, B, E),
  temp2(A, E, C).

temp1(A, B, E) :-
  call1(A, T),
  call2(B, T, Q),
  call3(Q, E).

temp2(A, E, C) :-
  call4(A, E, Z),
  call5(Z, C).
```

3.2 Model reduction techniques - abstraction levels

One can envisage some kind of *perfect* mathematical models, that contain every bit of information present in the program source code. In this case we can be sure that, if two such models are isomorphic, then the corresponding program source code is the same. Of course, such a model is nothing else but the source code itself, in a different representation. From the theoretical point of view, however, it is quite useful to suppose that such **perfect models** exist.

For any programming language and for any specific piece of source code, the lowest abstraction level, which we call level 0, could be considered to contain perfect models only.

At first, one may think that the best one could do is to determine the similarity degree of such perfect models. However, this would require some kind of a very sophisticated comparison algorithm, which is on one hand fast and parametrizable enough for our purposes, and on the other hand resistant to the possible cheating methods mentioned in the previous subsection. In the proposed framework we decided to follow a different approach examining a series of views with increasing abstraction levels.

We thus propose to use several abstraction levels (as shown in Figure 1) and use *relatively simple and fast comparison algorithms* between models on the same level. Higher abstraction levels are built from lower ones using some kind of *reduction* steps. Our task is to transform the initial perfect models to ones which are more and more resistant to specific tricks, and which still represent the original source code as much as possible.

Naturally, reduction steps are destructive operations, with every bit of dropped information we widen the gap between the perfect model and the model in question. Because of this, a perfect match (isomorphism for example) between two models on a high abstraction level "means less" than such a match on a lower level. To handle this, we assign a weight to each abstraction level in question, with which we multiply the similarity degree achieved on that level. These weights are not static values as we will see later.

We continue the simplification up to the point where every model becomes a singleton graph. On this highest abstraction level every pair of models is isomorphic.

Our task is to maximize the expression $B_i * H_i$, where B_i is the weight factor assigned to abstraction level i and H_i is the actual similarity degree obtained on abstraction level i. For this we use a simple iteration over abstraction levels $i = 1, \ldots$, starting with $Best_0 = 0$

1. compare the two models on abstraction level i,
2. in case of isomorphism, exit the algorithm with the output value $max(B_i, Best_{i-1})$
3. in case of similarity calculate $Best_i = max(H_i * B_i, Best_{i-1})$
4. if $Best_i \geq B_{i+1}$ then exit the algorithm with $Best_i$, otherwise continue with abstraction level $i+1$,

3.3 Model comparison algorithms

We argued, in subsection 3.1, that directed, labelled graphs are good mathematical constructs for describing models of programs. Considering this, the concrete comparison algorithms are most likely related to graph theoretical algorithms.

In general, our task is to define how much two graphs are *similar* to each other. Here we first introduce the concept of graph isomorphism algorithm as an extreme case of graph similarity.

The problem of isomorphism is the following. Given two graphs, G and H, we look for a bijection f between the nodes of the graphs, such as (x,y) is an edge in G if and only if $(f(x), f(y))$ is an edge in H.

The graph isomorphism problem belongs to the class of NP problems, but we still do not know if it is NP-complete. However, in special cases we know the exact complexity, or at least we can produce algorithms which run with acceptable speed. Some of these special cases are the following.

- The graphs are labelled, that is, there are red and blue nodes in G and H. In this case the bijection f must not assign red nodes to blue nodes or vice versa. In general we cannot say anything about the complexity, but these constraints sometimes are enough to make the problem polynomial.

  The idea of labelling is very important. Most of the graph isomorphism algorithms are based on the fact that once we are able to identify equivalent classes of the nodes (sets of vertices such that two vertices in the same set share some crucial property) we can consider this as a labelling.

  Examples for these classes may include vertices with the same degree, vertices such that if we take two vertices from the class and calculate for each of them the shortest paths to the rest of the nodes, these sets are the same, etc.

  This decomposition can drastically reduce the size of the problem. For example, if in a graph containing 300 nodes we could identify 10 equivalent classes (of the same size, for the sake of the example), than instead of the naive 300! number of steps we have to only take $(30!)^{10}$.

- Sometimes we are interested in knowing if G contains a subgraph that is isomorphic to H. For example, the problem of finding a Hamiltonian circle in a graph can be formulated in this way. However, depending on what we mean by "contain" there are two types of subgraph isomorphisms. Normally we simply ask whether there is a subset of edges and vertices of G that is isomorphic to H. In contrast with this, in case of induced subgraph isomorphism, we ask whether there is a subset of vertices of G whose deletion leaves a subgraph isomorphic to H. Both problems are known to be NP-complete [2].

- We know polynomial algorithms for planar graphs as well as for graphs where the maximum vertex degree is bounded.

In case of trees, a more straightforward approach is applicable, which can also be used as the basis for the general graph similarity problem. Namely, it is possible to construct a code in *linear time* for trees, which fulfills the following criteria ($T1$ and $T2$ are trees).

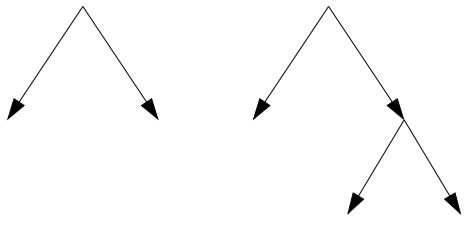
1. if $T1$ is isomorphic to $T2$, then the code of $T1$ equals to the code of $T2$
2. if the code of $T1$ equals to the code of $T2$, then $T1$ is isomorphic to $T2$

We note, that so far, for general graphs nobody has found such a coding scheme which would satisfy both conditions. The Tutte polynomial for example satisfies the former, but it is not true that we could conclude that two graphs are isomorphic because their Tutte polynomials are the same.

Actually the construction for trees is nothing more than a geometrical transformation, which maps a 2D tree to a one dimensional sequence over an alphabet of two characters. One can use the digits 0 and 1 or the parentheses (and) as the elements of the sequence, and accordingly the code of a leaf is 10 or (). Let T be the tree to be encoded, R is the root of the tree and $C(T)$ is the code of the tree T. The following recursive algorithm can be given, which assigns a binary number to any tree T.

1. determine the children of T, let us call them $N_1, N_2, \ldots, N_k$
2. determine the codes for $N_1, N_2, \ldots, N_k$
3. order the codes $C(N_1), C(N_2), \ldots, C(N_k)$ by their values in a monotonic ascending order resulting in a sequence S, and return the sequence $1S0$ as the code assigned to T

Thus the code for the entire tree is $C(R)$. Two examples of such encoding are given in Figure 2.



Code: 110100 Code: 1101101000

Figure 2: Two examples of tree coding

It is important that the above algorithm can be used for DAGs (Directed Acyclic Graphs), which are normal directed graphs, but do not contain directed circles. An example of such a graph and its code can be seen in Figure 3. It should also

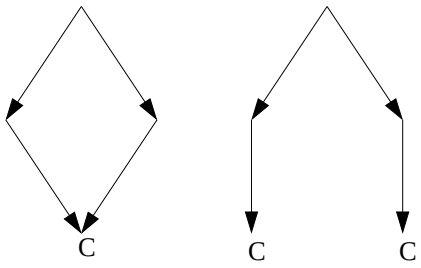


Figure 3: DAG, the corresponding tree and their tree codes

be noted that, in case of a DAG, its code is the same as that of a normal tree produced from the DAG by multiplexing specific nodes, such as node C in Figure 3.

It is important to note that checking isomorphism is usually not enough by itself, no matter what kind of models are used (graphs or trees). The reason is that we cannot expect that the graphs will be totally isomorphic, even with the most sophisticated source code to model mappings and reduction techniques. We really would like to detect if two graphs of hundreds of nodes (which is a real example) at a given abstraction level are *nearly* identical. We will deal with this issue in the next section.

4 Test implementation

In the following, we present our test implementation of the framework for Prolog programs.

4.1 Source code to model mapping

In our test implementation we worked with Prolog and SML source programs as we teach these languages as part of a Declarative Programming course, and major assignments must be written in these languages. Here we only introduce the Prolog part, more about the SML test implementation can be read in [9].

We chose call graphs as the models of Prolog programs. A call graph is a graph where the nodes correspond to the Prolog predicates and the edges to the calls (according to the procedural semantics of Prolog). If in the body of predicate `A` there is a call to predicate `B` then an edge between the corresponding nodes is present in the graph. In case of multiple calls, multiple edges are applied. The graph does not contain the built-in predicates (such as `is/2`), because they do very elementary tasks and would increase the graph size, without adding significantly more precision to the model. The graph includes however all other predicates, such as the library predicates and also takes care of the implicit calls made by meta-predicates, e.g. `findall/3`.

Call graphs are well suited for Prolog programs, because in Prolog the only control structure is predicate invocation (Prolog does not have `while`, `for`, `goto` constructs, nor any other "usual" imperative control elements).

The call graph is built from program source code by using source code analysis. For this, we use the modified version of the `xref` package of SICStus Prolog.

Actually, we made some simplifications to our model. For example, we remove the loops from the call graphs and we also remove some edges in order to avoid circles. Loops are removed because explicit recursion is so common is Prolog, that for us it does not contain valuable information. Breaking circles was a must, as we wanted to work with DAGs instead of general graphs. In our model, breaking the circles actually means to leave out of consideration the so called mutual recursion (for example when `A` calls `B` and vice versa). However, based on our experience, we can surely claim that mutual recursion is not very often used by students, and so neglecting it does not effect the final similarity measure in a significant way.

Furthermore, the graphs do not contain those predicates to which there was no reference in the source code.

4.2 Model reduction techniques

On the call graphs we can apply several reduction techniques to create abstraction levels (we have five levels including the original non-filtered call graph). In the following we describe each of the reduction steps and specify the maximal reachable similarity degree for each abstraction level created by applying the first few of these transformations one after the other.

1. non called predicates: in this reduction step we filter out those predicates which were not called during a test call. The test call is given by the tester. Test calls are usually simple tests which are easily solvable by the programs. The

maximal similarity degree is 95%.

2. library or dynamic predicates: in the reduction step we remove the Prolog library and dynamic predicates from the call graphs. The maximal similarity degree is 90%.

3. multiple edges: we filter out the multiple edges from the call graph and only keep one edge between any given two predicates. The maximal similarity degree is 80%.

4. topological isomorphism: We filter out those vertices in the call graphs which have degree 2 (one edge comes in and one goes out). This helps to detect the call-tunneling trick. The maximal similarity degree is 70%.

The maximal reachable similarity degree may vary depending on exactly what reduction steps we use. For example, when the first, second and fourth steps are used (but multiple edges are not filtered), the value is 80%.

4.3 Model comparison algorithms

We defined a quantitative measure which represents how much two call graphs are similar. Due to the fact that we use DAGs, the comparison can be based on the codes shown in 3.3. We have chosen this approach because several good heuristics can be given specifying how much a specific difference in the codes affects the similarity degree of the original trees.

Trivially, if the codes match, the corresponding models are the same. If one of the codes contains the other as its subsequence, then it can be suspected that one student got the other's program and added some new structure to it (of course the length of the subsequence matters: it is not the same when we find a matching sequence of 99 digits in a code of length 100, or the same matching sequence in a code of length 10000).

Call grouping can also be detected based on the codes. For example, if predicate `A` calls `B` which calls four other predicates, the corresponding code will be `( L ( L L L L ) )`, where `L` represents a leaf. If we apply call grouping, for example `B` will call `C` and `D` which will further call the other 2-2 predicates, the code takes the form `( L ( ( L L ) ( L L ) ) )`. We marked with italics those parentheses which have to be removed in order to get the original code.

Our general approach is to check if it is possible to transform (and if it is, how) one DAG code to another. We use the widely available UNIX diff program to actually enumerate the differences. For this we first create the appropriate files: we put each parenthesis and leaf on a separate line. Then we let the UNIX diff utility to calculate how the files can be made equal, i.e. to produce the instructions on which leaves and nodes should be added or removed to make the call graphs isomorphic.

Analyzing the information given by the diff utility we can determine the similarity degree of codes. For this we use penalties, which define how much we should "punish" certain modifications of the code sequences. We found that the following penalties are appropriate (we always try to modify the bigger graph and check what transformations we can use to obtain the smaller one).

- removal of a leaf: 1%
- addition of a leaf: 3%
- removal of a node: 2%
- addition of a node: 6%

This means that, for example, if we need to remove one leaf from our bigger call graph to make it identical to the smaller one, then the similarity degree is 99%.

Please note, that although diff can be used to produce a transformation which makes a call graph identical to another, abstraction levels are not useless. For example, filtering out multiple edges reduces the maximal similarity by 10%. If we used diff to remove these edges it could easily mean 0% similarity, provided there are sufficient number of multiple edges in the call graph.

4.4 Evaluation

We were lucky enough to have an abundant amount of Prolog source programs to test the prototype on. As an example to discuss, we took the set of major assignments for a semester of the "Declarative Programming" course.

Several students were kind enough (*after* they completed the course and were promised full amnesty) to provide us with some hints on what way were they cheating. So we had the minimal expectation that the prototype should at least mark those assignments as matched pairs.

The 92 source programs were evaluated against 4 different similarity thresholds: 60%, 70%, 80%, 90%. For example, the 90% threshold means that the prototype shows pairs of source codes which have similarity degree at least 90% percent. For every threshold, the system were run with 24 different parameter variations. These include the most useful settings in practical cases. We have 6 base options corresponding to the abstraction levels.

- setting 0: all options are disabled
- setting 1: enabled *test* mode, filtering non called parts
- setting 2: 1 + filtering library/dynamics predicates
- setting 3: $1+2$ + filtering multiple edges
- setting 4: $1+2$ + topological isomorphisms
- setting 5: $1+2+3$ + topological isomorphisms

With every setting we measured the run time, the number of hits and the ratio of hits and the real hits. Here, by real hits we mean that we examined the hits by hand and decided if a given hit was really something that we can call plagiarism. In case of hits, our program also shows the assumed mapping between the two source codes. Namely, we can see which predicate in one program matches which predicate in the other.[‡]

From our results we can draw the following main conclusions:

1. Students do not often use sophisticated tricks. Without the diff algorithm the only abstraction level where we had new hits (even using a 60% threshold) was the one using the topological isomorphism test. Here the efficiency dropped back to 50% from 100% at the 60% threshold with 8 real hits. Without this filtering we had 6 real hits (only 5 at 90% threshold). There was a pair who claimed they worked on the modifications for more than 5 hours, and in spite of this their similarity degree was nearly 90% without diff and topological isomorphism.

2. The diff algorithm and the abstraction levels fit together very well. There were cases when the plagiarism was detected with a high degree of similarity due to the fact that only minor differences were found on the abstraction level 0. Without diff, the same hits appeared on a higher abstraction level and so received a smaller similarity degree. Reverse situations also occurred. We could find programs (real hits) which were isomorphic on an abstraction level with relatively high similarity degree, and although this could be discovered using diff as well, the latter comparison gave a smaller similarity degree.

3. Our approach is very fast. Using 100 programs, the run time for comparing all the pairs varied between 19.3s (no diff and abstraction levels) and 414s (when most of the reduction steps were applied as well as the diff algorithm).

5 Summary and future work

In the paper we presented a plagiarism detection framework, which aims to determine the similarity degree of program source codes. The framework uses directed, labelled graphs to represent the structural information extracted from the source codes. Instead of using sophisticated comparison algorithms on these graphs, we presented an approach where we combined the use of relatively simple comparison techniques and reduction steps. We have implemented the framework for the Prolog and SML programming languages and we have been using this implementation to successfully detect plagiarism in homework assignments for several years.

Our future plans include the integration of the most promising statistical and/or lexicographic approaches into the framework. This way we can use hybrid comparison techniques that may be more efficient than the pure structural approach.

References

[1] ALEX AIKEN, A System for Detecting Software Plagiarism, `http://www.cs.berkeley.edu/ aiken/moss.html`

[2] MIKHAIL J. ATALLAH, ED., Algorithms and Theory of Computation Handbook, *CRC Press LLC* (1999) **page 6-21**

[3] BRENDA S. BAKER, Parametrized Pattern Matching: Algorithms and Applications, *Journal of Computing System Science* (1996) **52**

[4] J.M. BIEMAN AND N.C. DEBNATH, An analysis of software structure using generalized program graph, *In Proceedings of COMPSAC* (1985) **page 254-259**

[5] CFL SOFTWARE DEVELOPMENT, CopyCatch, `http://www.copycatchgold.com/`

[‡] In the final similarity degree we also take into consideration the arity of the mapped predicates. If we can only give such a mapping which binds together predicates with different arities, then this results in a smaller similarity measure.

[6] DIGITAL INTEGRITY, FindSame, http://www.findsame.com/

[7] J.A. FAIDHI AND S. K. ROBINSON, An empirical approach for detecting program similarity and plagiarism within a university programming environment, *Computing in Education* (1987) **vol. 11**

[8] DICK GRUNE, The software and text similarity tester SIM, http://www.cs.vu.nl/ dick/sim.html

[9] DÁVID HANÁK, Computer based support for teaching declarative languages (in Hungarian), *Master Thesis* (2001)

[10] IPARADIGMS, Educators, Cheating Students Rely on Web, *The Washington Post* (November 27, 2003) http://www.ithenticate.com/

[11] UDI MANBER AND BRENDA S. BAKER, Deducing similarities in Java sources from bytecodes, *USENIX Technical Conference* (1998)

[12] ALAN PARKER AND JAMES HAMBLEN, Computer algorithms for Plagiarism Detection, *IEEE Transactions in Education* (1989) **vol. 32, number 2**

[13] NARAYANAN SHIVAKUMAR AND HECTOR GARCIA-MOLINA, The SCAM Approach to Copy Detection in Digital Libraries, *Proceedings of 2nd International Conference in Theory and Practice of Digital Libraries* (1995) http://www-db.stanford.edu/ shiva/SCAM/scamInfo.html

[14] SUAREZ, J. AND MARTIN, A., Internet plagiarism: A teacher's combat guide, *Contemporary Issues in Technology and Teacher Education* (2001) **1 (4)**

[15] ADRIAN WEST, Coping with plagiarism in Computer Science teaching laboratories, *Computers in Teaching Conference* (1995)

[16] G, WHALE, Identification of Program Similarity in Large Populations, *The computer journal* (1990) **vol 33**

[17] MICHAEL J. WISE, Detection of similarities in student programs, *SIGSCI Technical Symposium* (1996)

Rigid graphs from edge-pairs

MÁRTON MAKAI*

Department of Operations Research
Eötvös University
Pázmány Péter sétány 1/C, Budapest
Hungary, H-1117.
marci@cs.elte.hu

Abstract: Let $H=(V,E)$ be a hypergraph and let $k\geq 1$ and $l\geq 0$ be fixed integers. Let $\mathscr{M}$ be the matroid with ground-set E s.t. a set $F\subseteq E$ is independent if and only if each $X\subseteq V$ with $k|X|-l\geq 0$ spans at most $k|X|-l$ hyperedges of F. We prove that if H is dense enough, then $\mathscr{M}$ satisfies the double circuit property, thus the min-max formula of Dress and Lovász on the maximum matroid matching holds for $\mathscr{M}$. Our result implies the Berge-Tutte formula on the maximum matching of graphs ($k=1$, $l=0$), generalizes Lovász' graphic matroid (cycle matroid) matching formula to hypergraphs ($k=l=1$) and gives a min-max formula for the maximum matroid matching in the 2-dimensional rigidity matroid ($k=2$, $l=3$).

Keywords: matroid matching, Dilworth truncation, double circuit property

1 Introduction

The notion of matroid matching is known to be an involved class of combinatorial optimization problems concerning parity. One of its numerous equivalent definitions is as follows. Let $\mathscr{M}$ be a matroid with ground-set E, with rank-function $r_{\mathscr{M}}$, with span-function $\mathrm{sp}_{\mathscr{M}}$ and let $A\subseteq\binom{E}{2}$ be a set of (not necessarily disjoint) pairs of E. For short, if $F\subseteq E$ and $M\subseteq A$, then $r_{\mathscr{M}}(F\cup M)$ denotes $r_{\mathscr{M}}(F\cup\bigcup M)$ and $\mathrm{sp}(F\cup M)=\mathrm{sp}(F\cup\bigcup M)$. A set of pairs $M\subseteq A$ is said to be a *matroid matching* of A w.r.t. $\mathscr{M}$ if $r_{\mathscr{M}}(M)=2|M|$. The *matroid matching problem* is to compute a matroid matching of maximum size, the size of which is denoted by $\nu_{\mathscr{M}}(A)$.

Jensen and Korte [5] and Lovász [9] proved that the matroid matching problem cannot be solved in its full generality. On the other hand, the increasing number of its solvable special subclasses shows its particular importance.

Starting from the very special matching problem of graphs and the matroid intersection problem, good characterization of the maximum matroid matching was obtained by Lovász [8] for linear matroids. He developed also a polynomial algorithm [9] for represented linear matroids. This gives a method to compute the maximum matroid matching in the graphic matroid (cycle matroid), the minimum number of vertices of a graph to pin to obtain a 2-dimensional rigid graph, and the maximum genus of a graph. By a construction of Schrijver [10], also Mader's maximum number of vertex-disjoint $\mathscr{T}$-paths can be computed by this. However, the min-max formula is given in a geometric language, it cannot be translated to a combinatorial one. This gap is partially filled by Lovász' structure theorem for 2-polymatroids [7] which enables us to derive combinatorial min-max formula for some of the above problems.

In the middle of the eighties, Dress and Lovász [3] pointed out that the tractability of the known solvable cases is due to a more general common property of the above matroids. Up to this day, the *double circuit property* is the only general property that assures a method to compute the maximum matroid matching for every $A\subseteq\binom{E}{2}$.

A set $U\subseteq E$ is said to be a *double circuit* of $\mathscr{M}$ if $U=U_1\dot\cup U_2\dot\cup\ldots\dot\cup U_d$ s.t. $C_i=U-U_i$ $(i=1,2,\ldots,d)$ are all the circuits of U. The above defined partition of U is called its *principal partition*. The double circuit is said to be *non-trivial* if its principal partition has at least three classes. The crucial situation where most of the solution approaches to the matroid matching problem can get stuck is the existence of a non-trivial double circuit of a certain distinguished size. (This situation is described more precisely in Lovász' structure theorem.) In this case, the possibility of reducing the problem to a "smaller" one has to be assured by the structural properties of the particular matroid we are dealing with. Lovász proved that in the case of full linear matroids, the modularity of lattice of flats (subspaces) is sufficient for this.

Dress and Lovász said that the matroid $\mathscr{M}$ has the *double circuit property* (*DCP* for short) if

$$r_{\mathscr{M}/Z}\left(\bigcap_{1\leq i\leq d}\mathrm{sp}_{\mathscr{M}/Z}(C_i)\right)>0 \tag{1}$$

*Research is supported by Communication Networks Laboratory, P´azm´any P´eter s´et´any 1/A, Budapest, Hungary, H-1117, by Hungarian National Foundation for Scientifi c Research, OTKA T037547 and TS 049788, by European MCRTN Adonet Contract Grant No. 504438, and by MTA-ELTE Egerv´ary Research Group on Combinatorial Optimization. Part of this work was done when the author was at Laboratoire Leibniz (UFR IMA), Universit´e Joseph Fourier, Grenoble.

holds for each non-trivial double circuit U of each contraction $\mathscr{M}/Z$ of $\mathscr{M}$ (using the above notations). They proved that for every matroid which has the DCP, the following combinatorial min-max formula holds.

Theorem 1 (Dress and Lovász [3]) *If $\mathscr{M}$ is a matroid which has the DCP and $A \subseteq \binom{E}{2}$, then*

$$\nu_{\mathscr{M}}(A) = \min r_{\mathscr{M}}(Z) + \sum_{j=1}^{t} \left\lfloor \frac{r_{\mathscr{M}/Z}(A_j)}{2} \right\rfloor,$$

where the minimum is taken for all $Z \subseteq E$ and for all partitions $A_1, A_2, \ldots, A_t$ of A.

The min-max formula has the same form as in Lovász [8] for linear matroids, but in the case of Dress and Lovász it is stated in a more general setting.

If the matroid $\mathscr{M}$ with ground-set E is embedded into a matroid $\mathscr{M}'$ with larger ground-set $E' \supseteq E$ and $A \subseteq \binom{E}{2}$, then $\nu_{\mathscr{M}}(A) = \nu_{\mathscr{M}'}(A)$. Thus, the natural question which arises is to explore the class of matroids which have the DCP, and matroids with embedding into a larger matroid having the DCP. Dress and Lovász proved that full linear, full algebraic, full transversal, and full graphic matroids have the DCP, for definitions see [3]. In other words, every linear, algebraic, transversal, and graphic matroid has an embedding into a matroid which has the DCP.

Based on lattice theoretic concepts, Björner and Lovász [2] introduced the class of pseudomodular matroids and they have shown that the matroids of the above classes are pseudomodular. Later, Hochstättler and Kern [4] proved that pseudomodular matroids have the DCP.

Although the class of linear matroids, pseudomodular matroids, and matroids having the DCP are rather general, it can be difficult to prove that a particular matroid has an embedding into a matroid from one of these classes. If only linearity is known, then the matroid is embedded into the full linear space and combinatorial min-max formula cannot be expected. Moreover, some of the most important combinatorially defined matroids are linear but not known to be deterministically representable, which would be a requirement for computational results. Pseudomodular matroids form a special class of matroids having the DCP but searching for a pseudomodular embedding is usually not easy for complicated combinatorial matroids. Thus it remains a great challenge to explore combinatorially suggested tractable classes which give a more unified view of the solvable cases.

The rank-functions of matroids which are the most interesting from the application point of view are defined by Dilworth truncation of modular functions or functions defined by graphs or hypergraphs. The main goal of this paper is to take a step in the way of better comprehension of the matching problem of such matroids. This is carried out by considering the matroid matching in the following class of purely combinatorially defined matroids. Note that this may be a class where Dilworth truncation arise in the most simple way, but even this gives a more unified view of some solved cases and also contains previously unsolved problems.

Our class of matroids is defined as follows. Let $k \geq 1$ and $l \geq 0$ be fixed integers and let $H = (V, E)$ be a finite hypergraph. Let us define $b : 2^V \to \mathbb{Z}$ by $b(X) = k|X| - l$ if $k|X| - l \geq 0$ and 0 otherwise. For $X \subseteq V$ and $F \subseteq E$ let $\gamma_F(X) = \{e \in F : e \subseteq X\}$. Finally, let $\mathscr{M}$ be the matroid with ground-set E s.t. $F \subseteq E$ is independent in $\mathscr{M}$ if and only if $|\gamma_F(X)| \leq b(X)$ for each $X \subseteq V$. We may suppose that each hyperedge is of size bigger than $\frac{l}{k}$ since the smaller hyperedges are loops. The hyperedges of size two will be called *graph-edge*s.

The above defined matroids can be shown to be linear, but for getting computational results from the application of Lovász' linear matroid matching theorem, the matroid has to be represented. However, if H contains only graph-edges, $k = 2$ and $l = 3$, then $\mathscr{M}$ is the 2-dimensional rigidity matroid (Laman, [6]) which is not known to be deterministically representable. The other problem which is mentioned again is that Lovász' formula does not show the combinatorial behavior of the problem.

Thus, the matroid will be embedded into a relatively small, combinatorially defined matroid which have the DCP. This embedding is obtained by adding further hyperedges to H. As this operation does not affect $\nu_{\mathscr{M}}(A)$, we may assume for simplicity that these hyperedges are already in H. We have to note that the new hyperedges have no individual importance. In a matroid which have the DCP, the flats have a very special structure. The main goal of adding new hyperedges is to reach this desired structure. The following property put a precise criterion for this.

Property 2 *If $X \subseteq V$, then $r_{\mathscr{M}}(\gamma_E(X)) = b(X)$.*

Based on this property, we can state our first theorem.

Theorem 3 *If Property 2 holds, then the matroid $\mathscr{M}$ has the DCP.*

Using Theorem 1, this immediately implies our main theorem.

Theorem 4 *If Property 2 holds, $A \subseteq \binom{E}{2}$, then for each contraction $\mathscr{N}$ of $\mathscr{M}$,*

$$\nu_{\mathscr{N}}(A) = \min r_{\mathscr{N}}(Z) + \sum_{j=1}^{t} \left\lfloor \frac{r_{\mathscr{N}/Z}(A_j)}{2} \right\rfloor, \tag{2}$$

where the minimum is taken for all $Z \subseteq E$ and for all partitions $A_1, A_2, \ldots, A_t$ of A.

As Property 2 is in the hypothesis of Theorem 4 we should give a short study of some criteria which imply Property 2. First, it is easy to see that if each set $X \subseteq V$ of size bigger than $\frac{l}{k}$ is in E with multiplicity $k|X|-l$, then Property 2 holds. A weaker condition also assures Property 2, as it is described in the following theorem.

Theorem 5 *Let $l = ck+d$ where c,d are integers, $0 \leq c$ and $0 \leq d < k$. Suppose that E contains all the subsets of V of size $c+1$ with multiplicity $k-d$. Suppose moreover that if $\frac{ck}{c+1} < d$, then E contains all the subsets of V of size $c+2$ with multiplicity $cd+d-ck$. Then Property 2 holds.*

Although the conditions of Theorem 5 seem to be artificial, they translate to clear requirements in the case of each particular matroid of our class.

2 Special cases

2.1 Berge-Tutte formula

First, we show that if $k = 1$ and $l = 0$, then Theorem 4 implies the Berge-Tutte formula [1]. Hence, let us consider the undirected graph G with vertex-set V and graph-edge-set $E(G)$. Let E contain each element of V as a singleton hyperedge with multiplicity one. In this case, it can be seen immediately that Property 2 holds. The set of pairs for the matroid matching is $A = \{\{\{u\},\{v\}\} : uv \in E(G)\}$. In addition, observe that M' is a matching of G if and only if $M = \{\{\{u\},\{v\}\} : uv \in M'\}$ is a matroid matching of A w.r.t. $\mathscr{M}$.

Let $Z \subseteq E$ and $A_1, A_2, \ldots, A_t$ give equality in the min-max relation stated in Theorem 4 so that $\mathrm{sp}_{\mathscr{M}}(Z)$ is minimal, and subject to this, t is as small as possible. Clearly, if $F \subseteq E$, then $Y_F = \emptyset$ and $|\mathscr{X}_F| \leq 1$. Let us define X_F by $\mathscr{X}_F = \{X_F\}$ if $\mathscr{X}_F \neq 0$ and let $X_F = \emptyset$ otherwise. If $X_Z = \emptyset$, then $\nu_{\mathscr{M}}(A) = \sum_{j=1}^{t} \left\lfloor \frac{1}{2}|X_{A_j}| \right\rfloor \geq \sum_{C \in \mathscr{C}} \left\lfloor \frac{1}{2}|C| \right\rfloor$, where $\mathscr{C}$ denotes the family of vertex-sets of the components of G. If $X_F \neq \emptyset$, then $X_{A_j} \cap X_{A_{j'}} = X_Z$ for every $1 \leq j < j' \leq t$. Then, $\nu_{\mathscr{M}}(A) = |X_Z| + \sum_{j=1}^{t} \left\lfloor \frac{1}{2}|X_{A_j} - X_Z| \right\rfloor \geq |X_Z| + \sum_{C \in \mathscr{C}} \left\lfloor \frac{1}{2}|C| \right\rfloor$, where $\mathscr{C}$ denotes the family of vertex-sets of the components of $G[V - X_Z]$. This is exactly what we have needed.

2.2 Transversal matroids

One of the usual interpretations of transversal matroids is that we have a hypergraph $H = (V,E)$ and $F \subseteq E$ is independent if and only if $|\gamma_F(X)| \leq |X|$ holds for every $X \subseteq V$. Similarly to the case of Berge-Tutte formula, if each singleton hyperedge is in E with multiplicity one, then Property 2 holds.

We also have to note that the transversal matroid matching can be solved in an easier way. Tong, Lawler and Vazirani [11] showed that even the weighted case can be reduced to the weighted matching problem of graphs.

2.3 Hypergraphic matroid and rigidity matroid

Next, we turn to the case when $1 \leq k \leq l \leq 2k-1$ and

$$A = \{\{e'_1, e''_1\}, \{e'_2, e''_2\}, \ldots, \{e'_p, e''_p\}\}$$

where e'_i and e''_i are all (not necessarily different) graph-edges. According to the choice of k and l, $c = 1$ and $d = l-k$. To satisfy the requirements of Theorem 5, E has to contain $2k-l$ parallel graph-edges on each pair of vertices. If $d > \frac{ck}{c+1} = \frac{k}{2}$ or equivalently $l = k+d > \frac{3k}{2}$, then we have to put $cd+d-ck = 2d-k = 2l-3k$ parallel hyperedges of size three to each triple of vertices. Next we consider the applications of this case.

If $k = l = 1$, then $\mathscr{M}$ is the hypergraphic matroid with ground-set E. As $l \leq \frac{3k}{2}$, then Property 2 holds if $\binom{V}{2} \subseteq E$. If E contains only graph-edges, then Theorem 4 specializes to Lovász' theorem on the maximum graphic matroid matching [7].

If $k = 2$ and $l = 3$, then $F \subseteq E$ is independent in $\mathscr{M}$ if and only if $|\gamma_F(X)| \leq 2|X|-3$ for every $X \in \binom{V}{\geq 2}$. Just as above, Property 2 is satisfied if $\binom{V}{2} \subseteq E$. If E contains only graph-edges, then it is known that the bases of $\mathscr{M}$ are exactly the 2-dimensional minimally rigid graphs on V (Laman, [6]). Let G be a 2-dimensional rigid graph with vertex-set V and let A be a set of (not necessarily disjoint) pairs from $E(G)$. Then the maximum number of graph-edge-pairs from A which are contained in a minimally rigid subgraph of G is $\nu_{\mathscr{M}}(A)$.

2.4 Connectivity augmentation

The problem discussed here was proposed by Zsolt Fekete. Let G be an undirected graph with vertex-set V and edge-set $E(G)$. Let moreover an other edge-set E' on V and a set of packets $\mathscr{P} \subseteq 2^{E'}$ be given. We ask for the minimum cardinality set $\mathscr{P}' \subseteq \mathscr{P}$ s.t. $(V, E(G) \cup \bigcup \mathscr{P}')$ has t edge-disjoint spanning trees. Clearly, if $\mathscr{P}$ is composed by singletons, then this is a minimum cardinality spanning subset problem in a matroid.

Let us consider the problem, when each packet is composed by p parallel edges. Frank observed (personal communication) that if $p = t$ and t is part of the input, then the problem is NP-hard. But for any fixed $t \geq 3$ the complexity is unknown.

Let us sketch what happens in the case $t = 2$. Then, the problem is to add a minimum number of edge-pairs of $\mathscr{P}$ to G s.t. the resulting graph has 2 edge-disjoint spanning trees. Let us fix $k = l = 2$ which defines $\mathscr{M}$ and E. In this language, the goal is to compute a minimum number of edge-pairs which spans E in $\mathscr{M}/E(G)$. By Lovász' theorem [9] on the minimum number of pairs spanning the matroid, this problem reduces to the matroid matching problem of $\mathscr{M}/E(G)$.

3 Preliminaries

It is not difficult to prove that the above defined $\mathscr{M}$ is a matroid. The correctness of the following claims can be seen immediately for the reader who is familiar with matroid theory and Dilworth truncation. We sketch the proof of the equality

$$r_{\mathscr{M}}(F) = \min\left\{ |Y| + \sum_{X\in\mathscr{X}} b(X) : Y \subseteq F,\ \mathscr{X} \subseteq \binom{V}{>\frac{l}{k}},\ Y \cup \bigcup_{X\in\mathscr{X}} \gamma_E(X) \supseteq F \right\}. \tag{3}$$

$\mathscr{X}_1 \subseteq \binom{V}{>\frac{l}{k}}$ is said to be a refinement of $\mathscr{X}_2 \subseteq \binom{V}{>\frac{l}{k}}$ if for each $X_1 \in \mathscr{X}_1$ there exists $X_2 \in \mathscr{X}_2$ s.t. $X_1 \subseteq X_2$.

Claim 6 *(i) The right hand side of (3) is a matroid rank-function.*

(ii) If $F \subseteq E$, then there exists a unique pair $(\mathscr{X}_F, Y_F)$, $\mathscr{X}_F \subseteq \binom{V}{>\frac{l}{k}}$, $Y_F \subseteq F$, s.t. $r_{\mathscr{M}}(F) = |Y_F| + \sum_{X\in\mathscr{X}_F} b(X)$, $Y_F \cup \bigcup_{X\in\mathscr{X}_F} \gamma_E(X) \supseteq F$ and for every $(\mathscr{X}, Y)$ with the same properties, $\mathscr{X}$ is a refinement of $\mathscr{X}_F$ (and $Y_F \subseteq Y$).

(iii) If $F_1 \subseteq F_2 \subseteq E$, then $\mathscr{X}_{F_1}$ refines $\mathscr{X}_{F_2}$.

PROOF: For *(i)*, the right hand side of (3) is monotone increasing and singletons get value at most one. Thus the only non-trivial thing is to prove that the right hand side of (3) is submodular. Let F_1 and F_2 be subsets of E and let resp. $(\mathscr{X}_1, Y_1)$ and $(\mathscr{X}_2, Y_2)$ give the corresponding minimum in (3). In what follows, the word *collection* stands for a family where multiplicities are counted. Algebraic operations with collections are defined by the corresponding operations with the multiplicity functions. Starting from $\mathscr{G}_0 = \mathscr{X}_1 + \mathscr{X}_2$, we apply a simple uncrossing procedure and a sequence of collections $\mathscr{G}_0, \mathscr{G}_1, \ldots, \mathscr{G}_l$ is computed. If for some $i \geq 0$, $\mathscr{G}_i$ has already been defined, $X_1, X_2 \in \mathscr{G}_i$, $|X_1 \cap X_2| \geq \frac{l}{k}$, $X_1 \not\subseteq X_2$ and $X_2 \not\subseteq X_1$, then let us define $\mathscr{G}_{i+1}$ by $\mathscr{G}_i - \{X_1\} - \{X_2\} + \{X_1 \cap X_2\} + \{X_1 \cup X_2\}$. Clearly, $\sum_{X\in\mathscr{G}_i} \chi_{\gamma_E(X)} \leq \sum_{X\in\mathscr{G}_{i+1}} \chi_{\gamma_E(X)}$ and $\sum_{X\in\mathscr{G}_i} b(X) = \sum_{X\in\mathscr{G}_{i+1}} b(X)$. Next, this procedure is finite, since $\sum_{X\in\mathscr{G}_i} |X|^2 < \sum_{X\in\mathscr{G}_{i+1}} |X|^2$ and $\sum_{X\in\mathscr{G}_i} |X|^2 \leq |\mathscr{G}_i||V|^2 = |\mathscr{G}_0||V|^2$. When the uncrossing finishes in $\mathscr{G}_l$, $\{\gamma_E(X) : X \in \mathscr{G}_l\}$ is a laminar family. Let $\mathscr{G}_{\max}$ contain one from each of the maximal members of $\mathscr{G}_l$. Then $(Y_1 \cap Y_2) \cup (Y_1 - F_2) \cup (Y_2 - F_1) \cup \bigcup_{X\in\mathscr{G}_{\max}} \gamma_E(X) \supseteq F_1 \cup F_2$ and $((Y_1 \cup Y_2) \cap F_1 \cap F_2) \cup \bigcup_{X\in\mathscr{G}_l - \mathscr{G}_{\max}} \gamma_E(X) \supseteq F_1 \cap F_2$, thus $r_{\mathscr{M}}(F_1) + r_{\mathscr{M}}(F_2) = |Y_1| + |Y_2| + \sum_{X\in\mathscr{G}_0} b(X) = |(Y_1 \cap Y_2) \cup (Y_1 - F_2) \cup (Y_2 - F_1)| + \sum_{X\in\mathscr{G}_{\max}} b(X) + |(Y_1 \cup Y_2) \cap F_1 \cap F_2| + \sum_{X\in\mathscr{G}_l - \mathscr{G}_{\max}} b(X) \geq r_{\mathscr{M}}(F_1 \cup F_2) + r_{\mathscr{M}}(F_1 \cap F_2)$, which completes the proof.

Next, we prove *(ii)*. Let $(\mathscr{X}_1, Y_1)$ and $(\mathscr{X}_2, Y_2)$ be two different pairs which give the minimum in the right hand side of (3). As there is a finite number of such pairs, it is sufficient to construct a pair $(\mathscr{X}, Y)$ with the same properties s.t. $\mathscr{X}_1$ and $\mathscr{X}_2$ refine $\mathscr{X}$. Let us apply the uncrossing procedure presented in the proof of part *(i)*. Then it constructs families $\mathscr{G}_{\max}$ for $F \cup F$ and $\mathscr{G}_l - \mathscr{G}_{\max}$ for $F \cap F$ s.t. $\mathscr{X}_1$ and $\mathscr{X}_2$ are refinements of $\mathscr{G}_{\max}$. It can be seen easily that $(Y_1 \cap Y_2) \cup \bigcup_{X\in\mathscr{G}_{\max}} \gamma_E(X) \supseteq F$ and $(Y_1 \cup Y_2) \cup \bigcup_{X\in\mathscr{G}_l - \mathscr{G}_{\max}} \gamma_E(X) \supseteq F$, thus $2r_{\mathscr{M}}(F) = |Y_1| + |Y_2| + \sum_{X\in\mathscr{G}_0} b(X) = |Y_1 \cap Y_2| + \sum_{X\in\mathscr{G}_{\max}} b(X) + |Y_1 \cup Y_2| + \sum_{X\in\mathscr{G}_l - \mathscr{G}_{\max}} b(X) \geq 2r_{\mathscr{M}}(F)$. Hence $r_{\mathscr{M}}(F) = |Y_1 \cap Y_2| + \sum_{X\in\mathscr{G}_{\max}} b(X)$, $\mathscr{X}_1$ and $\mathscr{X}_2$ are refinements of $\mathscr{G}_{\max}$ and $(Y_1 \cap Y_2) \cup \bigcup_{X\in\mathscr{G}_{\max}} \gamma_E(X) \supseteq F$. Therefore both $\mathscr{X}_1$ and $\mathscr{X}_2$ refine $\mathscr{X} = \mathscr{G}_{\max}$.

Last we prove *(iii)*. If we apply the above uncrossing procedure for $\mathscr{X}_{F_1}$ and $\mathscr{X}_{F_2}$, then it produces families $\mathscr{G}_{\max}$ for $F_1 \cup F_2 = F_2$ and $\mathscr{G}_l - \mathscr{G}_{\max}$ for $F_1 \cap F_2 = F_1$ s.t. $\mathscr{X}_{F_1}$ and $\mathscr{X}_{F_2}$ are refinements of $\mathscr{G}_{\max}$. We can see that $(Y_1 \cap Y_2) \cup (Y_2 - F_1) \cup \bigcup_{X\in\mathscr{G}_{\max}} \gamma_E(X) \supseteq F_2$ and $((Y_1 \cup Y_2) \cap F_1) \cup \bigcup_{X\in\mathscr{G}_l - \mathscr{G}_{\max}} \gamma_E(X) \supseteq F_1$, thus $r_{\mathscr{M}}(F_1) + r_{\mathscr{M}}(F_2) = |Y_1| + |Y_2| + \sum_{X\in\mathscr{G}_0} b(X) = |(Y_1 \cap Y_2) \cup (Y_2 - F_1)| + \sum_{X\in\mathscr{G}_{\max}} b(X) + |(Y_1 \cup Y_2) \cap F_1| + \sum_{X\in\mathscr{G}_l - \mathscr{G}_{\max}} b(X) \geq r_{\mathscr{M}}(F_2) + r_{\mathscr{M}}(F_1)$. *(ii)* implies $\mathscr{X}_{F_2} = \mathscr{G}_{\max}$, which completes the proof. □

Claim 7 *If $F \subseteq E$, then $\{Y_F\} \cup \{\gamma_E(X) : X \in \mathscr{X}_F\}$ forms a subpartition of E.*

PROOF: If $Y_F \cap \gamma_E(X) \neq \emptyset$ for some $X \in \mathscr{X}_F$, then Y_F could be replaced by $Y_F - \gamma_E(X)$. If $|X_1 \cap X_2| \geq \frac{l}{k}$ for some $X_1, X_2 \in \mathscr{X}_F$, then we could replace $\mathscr{X}_F$ by $\mathscr{X}_F - \{X_1\} - \{X_2\} + \{X_1 \cup X_2\}$. □

It can be proved easily that $\mathscr{M}$ is identical with the matroid defined by the right hand side of (3). In the sense of these Claims, Property 2 can be reformulated s.t. if $X \subseteq V$ and $b(X) > 0$, then $\mathscr{X}_{\gamma_E(X)} = \{X\}$.

4 Structure of double circuits

The key phenomenon in the background of Theorem 3 is the modular structure of double circuits. Let $Z \subseteq E$ be a flat of $\mathscr{M}$, and let U be a non-trivial double circuit of $\mathscr{M}/Z$ with principal partition $U = U_1 \dot\cup U_2 \dot\cup \ldots \dot\cup U_d$. In the following theorem and in its proof sp stands for $\mathrm{sp}_{\mathscr{M}/Z}$. For the positive integer n, let $[n] = \{1,2,\ldots,n\}$, and for $T \subseteq [d]$, let $C(T)$ denote $\bigcap_{t\in T} \mathrm{sp}(C_t)$ (where $C(\emptyset)$ is defined to be $\mathrm{sp}(U)$).

Theorem 8 *If Property 2 holds, U is a double circuit of $\mathscr{M}/Z$ with the above notations, and $T \subseteq [d]$, then*

$$\mathrm{sp}\,(C(T-\{i\}) \cup C(T-\{j\})) = C(T-\{i,j\}), \tag{4}$$

where $i,j \in T$, $i \neq j$, and

$$r_{\mathscr{M}/Z}(C(T)) = |U| - \sum_{t\in T} |U_t| + |T| - 2. \tag{5}$$

PROOF: First we need some technical preliminaries.

Claim 9 *If $X_1, X_2, X_3 \in \binom{V}{>\frac{l}{k}}$ s.t. $b(X_i \cap X_j) > 0$ for every $1 \leq i < j \leq 3$, then*

$$\sum_{1\leq i<j\leq 3} b(X_i \cap X_j) + b(X_1 \cup X_2 \cup X_3) \leq \sum_{i=1}^{3} b(X_i) + b(X_1 \cap X_2 \cap X_3).$$

PROOF: If $k|\bigcap_{1\leq i\leq 3} X_i| - l \geq 0$, then the inequality holds with equality. If $k|\bigcap_{1\leq i\leq 3} X_i| - l < 0$, then the right hand side is greater by $l - k|\bigcap_{1\leq i\leq 3} X_i|$. □

Claim 10 *Let F_1, F_2 and F_3 be flats of $\mathscr{M}/Z$ s.t. $|\mathscr{X}_{F_i\cup Z} - \mathscr{X}_Z| = 1$ and $Y_{F_i\cup Z} \subseteq Y_Z$ for every $1 \leq i \leq 3$. Suppose moreover $r_{\mathscr{M}/Z}(F_i \cap F_j) > 0$ for every $1 \leq i < j \leq 3$. Then*

$$\sum_{1\leq i<j\leq 3} r_{\mathscr{M}/Z}(F_i \cap F_j) + r_{\mathscr{M}/Z}(F_1 \cup F_2 \cup F_3) \leq \sum_{i=1}^{3} r_{\mathscr{M}/Z}(F_i) + r_{\mathscr{M}/Z}(F_1 \cap F_2 \cap F_3).$$

PROOF: As F_1, F_2 and F_3 are flats of $\mathscr{M}/Z$, then $F_i \cap F_j$ (for $1 \leq i < j \leq 3$) and $F_1 \cap F_2 \cap F_3$ are also flats of $\mathscr{M}/Z$. According to the hypothesis, let $\{X_i\} = \mathscr{X}_{F_i\cup Z} - \mathscr{X}_Z$ and let $\mathscr{Z}_i = \mathscr{X}_Z - \mathscr{X}_{F_i\cup Z}$, $Y_i = Y_Z - Y_{F_i\cup Z}$. Then

$$r_{\mathscr{M}/Z}(F_i) = |Y_{F_i\cup Z}| + \sum_{X\in\mathscr{X}_{F_i\cup Z}} b(X) - |Y_Z| - \sum_{X\in\mathscr{X}_Z} b(X) = -|Y_i| + b(X_i) - \sum_{X\in\mathscr{Z}_i} b(X).$$

If $W \in \mathscr{X}_Z$ and $1 \leq i \leq 3$, then either $W \subseteq X_i$ or $|W \cap X_i| < \frac{l}{k}$. Therefore, if $W \in \mathscr{X}_Z$, $X \in \{X_1\cap X_2, X_1\cap X_3, X_2\cap X_3, X_1\cap X_2\cap X_3\}$, then $W \subseteq X$ or $|W \cap X| < \frac{l}{k}$ also holds.

By Claim 6, $\mathscr{X}_Z$ refines $\mathscr{X}_{(F_i\cap F_j)\cup Z}$ which refines both $\mathscr{X}_{F_i\cup Z}$ and $\mathscr{X}_{F_j\cup Z}$. If $b(X_i \cap X_j) = 0$ for some $1 \leq i < j \leq 3$, then $\mathscr{X}_{(F_i\cap F_j)\cup Z}$ would refine $\mathscr{X}_Z$, contradicting $r_{\mathscr{M}/Z}(F_i \cap F_j) > 0$. Thus $b(X_i \cap X_j) > 0$. By Property 2, $\mathscr{X}_{\gamma_E(X_i\cap X_j)} = \{X_i \cap X_j\}$ which refines $\mathscr{X}_{(F_i\cap F_j)\cup Z}$. This together implies

$$\mathscr{X}_{(F_i\cap F_j)\cup Z} = \{X_i \cap X_j\} \cup (\mathscr{X}_Z - (\mathscr{Z}_i \cap \mathscr{Z}_j)),$$

$$r_{\mathscr{M}}((F_i \cap F_j) \cup Z) = b(X_i \cap X_j) + \sum_{X\in\mathscr{X}_Z - (\mathscr{Z}_i\cap\mathscr{Z}_j)} b(X) + |Y_Z - (Y_i \cap Y_j)|,$$

and

$$r_{\mathscr{M}/Z}(F_i \cap F_j) = -|Y_i \cap Y_j| + b(X_i \cap X_j) - \sum_{X\in\mathscr{Z}_i\cap\mathscr{Z}_j} b(X).$$

Similar argument shows that if $b(X_1 \cap X_2 \cap X_3) > 0$, then

$$\mathscr{X}_{(F_1\cap F_2\cap F_3)\cup Z} = \{X_1 \cap X_2 \cap X_3\} \cup (\mathscr{X}_Z - (\mathscr{Z}_1 \cap \mathscr{Z}_2 \cap \mathscr{Z}_3)).$$

If $b(X_1 \cap X_2 \cap X_3) = 0$, then $\mathscr{Z}_1 \cap \mathscr{Z}_2 \cap \mathscr{Z}_3 = \emptyset$ and

$$\mathscr{X}_{(F_1\cap F_2\cap F_3)\cup Z} = \mathscr{X}_Z.$$

In both cases

$$r_{\mathscr{M}/Z}(F_1 \cap F_2 \cap F_3) = -|Y_1 \cap Y_2 \cap Y_3| + b(X_1 \cap X_2 \cap X_3) - \sum_{X \in \mathscr{Z}_1 \cap \mathscr{Z}_2 \cap \mathscr{Z}_3} b(X).$$

Last,

$$F_1 \cup F_2 \cup F_3 \cup Z \subseteq \gamma_E(X_1 \cup X_2 \cup X_3) \cup \bigcup_{X \in \mathscr{X}_Z - (\mathscr{Z}_1 \cup \mathscr{Z}_2 \cup \mathscr{Z}_3)} \gamma_E(X) \cup (Y_Z - (Y_1 \cup Y_2 \cup Y_3)),$$

and

$$r_{\mathscr{M}/Z}(F_1 \cup F_2 \cup F_3) \leq -|Y_1 \cup Y_2 \cup Y_3| + b(X_1 \cup X_2 \cup X_3) - \sum_{X \in \mathscr{Z}_1 \cup \mathscr{Z}_2 \cup \mathscr{Z}_3} b(X).$$

Now we apply Claim 9 and the statement follows.

$$\sum_{1 \leq i < j \leq 3} r_{\mathscr{M}/Z}(F_i \cap F_j) + r_{\mathscr{M}/Z}(F_1 \cup F_2 \cup F_3) \leq$$

$$\sum_{1 \leq i < j \leq 3} \left(-|Y_i \cap Y_j| + b(X_i \cap X_j) - \sum_{X \in \mathscr{Z}_i \cap \mathscr{Z}_j} b(X) \right)$$

$$-|Y_1 \cup Y_2 \cup Y_3| + b(X_1 \cup X_2 \cup X_3) - \sum_{X \in \mathscr{Z}_1 \cup \mathscr{Z}_2 \cup \mathscr{Z}_3} b(X) \leq$$

$$\sum_{i=1}^{3} \left(-|Y_i| + b(X_i) - \sum_{X \in \mathscr{Z}_i} b(X) \right)$$

$$-|Y_1 \cap Y_2 \cap Y_3| + b(X_1 \cap X_2 \cap X_3) - \sum_{X \in \mathscr{Z}_1 \cap \mathscr{Z}_2 \cap \mathscr{Z}_3} b(X) =$$

$$\sum_{i=1}^{3} r_{\mathscr{M}/Z}(F_i) + r_{\mathscr{M}/Z}(F_1 \cap F_2 \cap F_3).$$

□

Claim 11 *If C is a circuit of $\mathscr{M}/Z$, then $Y_{C \cup Z} \subseteq Y_Z$ and $|\mathscr{X}_{C \cup Z} - \mathscr{X}_Z| = 1$.*

PROOF: $\mathscr{X}_Z$ is a refinement of $\mathscr{X}_{C \cup Z}$, thus

$$|C| = |Y_{C \cup Z}| + \sum_{X \in \mathscr{X}_{C \cup Z}} b(X) - |Y_Z| - \sum_{X \in \mathscr{X}_Z} b(X) + 1 =$$

$$|Y_{C \cup Z} - Y_Z| + \sum_{X \in \mathscr{X}_{C \cup Z}} \left(b(X) - |Y_Z \cap \gamma_E(X)| - \sum_{W \in \mathscr{X}_Z, W \subseteq X} b(W) \right) + 1.$$

Since $Y_{C \cup Z} \cap \bigcup_{X \in \mathscr{X}_{C \cup Z}} \gamma_E(X) = \emptyset$ and $\bigcup_{X \in \mathscr{X}_{C \cup Z}} \gamma_E(X) \supseteq \bigcup_{X \in \mathscr{X}_Z} \gamma_E(X)$, then $Y_{C \cup Z} \cap \bigcup_{X \in \mathscr{X}_Z} \gamma_E(X) = \emptyset$. If $Y_{C \cup Z} \not\subseteq Y_Z$, then let $e \in Y_{C \cup Z} - Y_Z$. In this case, $r_{\mathscr{M}/Z}(C - e) = r_{\mathscr{M}/Z}(C) - 1$, contradicting that C is a circuit.

Thus, $|C \cap \gamma_E(X)| \geq b(X) - |Y_Z \cap \gamma_E(X)| - \sum_{W \in \mathscr{X}_Z, W \subseteq X} b(W) + 1$ for some $X \in \mathscr{X}_{C \cup Z}$. If $|C \cap \gamma_E(X)| \geq b(X) - |Y_Z \cap \gamma_E(X)| - \sum_{W \in \mathscr{X}_Z, W \subseteq X} b(W) + 2$ for some $X \in \mathscr{X}_{C \cup Z}$, then a hyperedge could be removed from $C \cap \gamma_E(X)$ and C could not become independent, contradicting that C is a circuit. If there would be different $X_1, X_2 \in \mathscr{X}_{C \cup Z}$ with $|C \cap \gamma_E(X_i)| = b(X_i) - |Y_Z \cap \gamma_E(X_i)| - \sum_{W \in \mathscr{X}_Z, W \subseteq X_i} b(W) + 1$, then $C \cap \gamma_E(X_1)$ could be removed from C without making C independent, contradicting that C is a circuit, and finishing the proof. □

Claim 12 *If $\emptyset \neq T \subseteq [d]$, then $Y_{C(T)} \subseteq Y_Z$ and $|\mathscr{X}_{C(T)} - \mathscr{X}_Z| \leq 1$. If moreover $r_{\mathscr{M}/Z}(C(T)) > 0$, then $|\mathscr{X}_{C(T)} - \mathscr{X}_Z| = 1$.*

PROOF: The statement is proved by induction on $|T|$. If $|T| = 1$, then we are done by Claim 11.

Next, we suppose $|T| \geq 2$, and let $i \in T$. By induction we have $\mathscr{X}_{C(\{i\})} - \mathscr{X}_Z = \{X_i\}$. Similarly, either $\mathscr{X}_{C(T-\{i\})} - \mathscr{X}_Z = \emptyset$ or $\mathscr{X}_{C(T-\{i\})} - \mathscr{X}_Z = \{X_{T-\{i\}}\}$. In the first case, $\mathscr{X}_{C(T)} = \mathscr{X}_Z$. In the second case, $\mathscr{X}_{C(T)} - \mathscr{X}_Z = \{X_i \cap X_{T-\{i\}}\}$ if $b(X_i \cap X_{T-\{i\}}) > 0$ and $\mathscr{X}_{C(T)} - \mathscr{X}_Z = \emptyset$ otherwise.

$Y_{C(T)} \subseteq Y_Z$ can be seen easily, as for each hyperedge $e \in C(\{i\}) \cap C(T - \{i\})$, either $e \in \gamma_E(X_i) \cap \gamma_E(X_{T-\{i\}})$, thus $e \notin Y_{C(T)}$, or $e \in Y_{C(\{i\})} \cup Y_{C(T-\{i\})} \subseteq Y_Z$.

Last, if $\mathscr{X}_{C(T)} - \mathscr{X}_Z = \emptyset$, then $r_{\mathscr{M}/Z}(C(T)) = 0$, which proves the last statement. □

Claim 13 *For $i, j \in [d]$, $i \neq j$,*

$$r_{\mathcal{M}/Z}(\mathrm{sp}(C_i)) = |U| - |U_i| - 1, \tag{6}$$
$$r_{\mathcal{M}/Z}(\mathrm{sp}(C_i \cap C_j)) = |U| - |U_i| - |U_j|, \tag{7}$$
$$r_{\mathcal{M}/Z}(\mathrm{sp}(C_i) \cup \mathrm{sp}(C_j)) = |U| - 2, \tag{8}$$
$$\mathrm{sp}(C_i \cap C_j) = \mathrm{sp}(C_i) \cap \mathrm{sp}(C_j). \tag{9}$$

PROOF: (6) is clear since C_i is a circuit. $C_i \cap C_j$ is independent, hence (7) follows. For (8), $U \subseteq \mathrm{sp}(C_i) \cup \mathrm{sp}(C_j) \subseteq \mathrm{sp}(U)$. For (9), $\mathrm{sp}(C_i \cap C_j) \subseteq \mathrm{sp}(\mathrm{sp}(C_i) \cap \mathrm{sp}(C_j)) = \mathrm{sp}(C_i) \cap \mathrm{sp}(C_j)$ and $r_{\mathcal{M}/Z}(\mathrm{sp}(C_i) \cap \mathrm{sp}(C_j)) \leq r_{\mathcal{M}/Z}(\mathrm{sp}(C_i)) + r_{\mathcal{M}/Z}(\mathrm{sp}(C_j)) - r_{\mathcal{M}/Z}(\mathrm{sp}(C_i) \cup \mathrm{sp}(C_j)) = |U| - |U_i| - |U_j| = r_{\mathcal{M}/Z}(\mathrm{sp}(C_i \cap C_j))$. □ Now we turn to the proof of Theorem 8 by induction on $|T|$. Throughout the proof, the singleton $\{i\}$ is sometimes referred as i. For $|T| = 0$, (5) holds by definition. For $|T| = 1$, (5) only is to be proved, which follows from (6). For $|T| = 2$, (4) follows from (8), and (5) follows from (7) and (9).

So let us assume $|T| \geq 3$ and $T = [|T|]$ for sake of simplicity. First, (4) is proved. It can be seen immediately that

$$C(T - i) \cup C(T - j) \subseteq C(T - \{i, j\}).$$

Using (5), it is known by induction, that

$$r_{\mathcal{M}/Z}(C(T - j)) + r_{\mathcal{M}/Z}(C(\emptyset)) = r_{\mathcal{M}/Z}(C(T - \{i, j\})) + r_{\mathcal{M}/Z}(C(i)).$$

By submodularity,

$$\begin{aligned} r_{\mathcal{M}/Z}(C(T - i) \cup C(T - j)) \geq \\ r_{\mathcal{M}/Z}(C(T - i) \cup C(T - j) \cup C(i)) + r_{\mathcal{M}/Z}((C(T - i) \cup C(T - j)) \cap C(i)) - \\ r_{\mathcal{M}/Z}(C(i)) = \\ r_{\mathcal{M}/Z}(C(\emptyset)) + r_{\mathcal{M}/Z}((C(T - i) \cup C(T - j)) \cap C(i)) - \\ (r_{\mathcal{M}/Z}(C(\emptyset)) + r_{\mathcal{M}/Z}(C(T - j)) - r_{\mathcal{M}/Z}(C(T - \{i, j\}))) = \\ r_{\mathcal{M}/Z}((C(T - i) \cup C(T - j)) \cap C(i)) - r_{\mathcal{M}/Z}(C(T - j)) + r_{\mathcal{M}/Z}(C(T - \{i, j\})) \geq \\ r_{\mathcal{M}/Z}(C(T - \{i, j\})). \end{aligned}$$

$C(T - \{i, j\})$ is a flat, thus

$$\mathrm{sp}(C(T - i) \cup C(T - j)) = C(T - \{i, j\}),$$

proving (4).

For (5), again, we begin with the easier part, using only submodularity and induction:

$$\begin{aligned} r_{\mathcal{M}/Z}(C(T)) = r_{\mathcal{M}/Z}(C(T - \{1\}) \cap C(T - \{2\})) \leq \\ r_{\mathcal{M}/Z}(C(T - \{1\})) + r_{\mathcal{M}/Z}(C(T - \{2\})) - r_{\mathcal{M}/Z}(C(T - \{1\}) \cup C(T - \{2\})) = \\ \left(|U| - \textstyle\sum_{i \in T - \{1\}} |U_i| + |T| - 3\right) + \left(|U| - \sum_{i \in T - \{2\}} |U_i| + |T| - 3\right) - \\ \left(|U| - \textstyle\sum_{i \in T - \{1,2\}} |U_i| + |T| - 4\right) = \\ |U| - \sum_{i \in T} |U_i| + |T| - 2. \end{aligned}$$

For the reverse inequality, we apply Claim 10 for $F_1 = C(T - \{2,3\})$, $F_2 = C(T - \{1,3\})$ and $F_3 = C(T - \{1,2\})$. Then

$$\begin{aligned} r_{\mathcal{M}/Z}(C(T)) = r_{\mathcal{M}/Z}(C(T - \{2,3\}) \cap C(T - \{1,3\}) \cap C(T - \{1,2\})) \geq \\ \sum_{i=1}^{3} r_{\mathcal{M}/Z}(C(T - i)) + r_{\mathcal{M}/Z}(C(T - [3])) - \sum_{\{i,j\} \in \binom{[3]}{2}} r_{\mathcal{M}/Z}(C(T - \{i, j\}) = \\ \sum_{i=1}^{3} \left(|U| - \textstyle\sum_{k \in T - i} |U_k| + |T| - 3\right) + \left(|U| - \sum_{k \in T - [3]} |U_k| + |T| - 5\right) - \\ \sum_{\{i,j\} \in \binom{[3]}{2}} \left(|U| - \textstyle\sum_{k \in T - \{i,j\}} |U_k| + |T| - 4\right) = \\ |U| - \sum_{i \in T} |U_i| + |T| - 2. \end{aligned}$$

□

PROOF:[Proof of Theorem 3] Let $Z \subseteq E$ and let U be a non-trivial (i.e. $d \geq 3$) double circuit of $\mathscr{M}/Z$ with principal partition $U = U_1 \dot\cup U_2 \dot\cup \ldots \dot\cup U_d$. Using the above notations, and by applying (5) to $T = [d]$,

$$r_{\mathscr{M}/Z}(C([d])) = |U| - \sum_{t\in[d]} |U_t| + |[d]| - 2 = d - 2 > 0.$$

□

PROOF:[Proof of Theorem 5] For $X \subseteq V$ with $|X| \leq c$ we have $b(X) = 0$, hence suppose next $|X| = c+1$. Then $b(X) = k(c+1) - (ck+d) = k-d$, and the condition that each hyperedge of size $c+1$ is present with multiplicity $k-d$ gives the proof.

Suppose now that $|X| \geq c+2$, and let $\mathscr{X} \subseteq \binom{V}{>\frac{l}{k}}$, $Y \subseteq E$ s.t. $Y \cup \bigcup_{W\in\mathscr{X}} \gamma_E(W) \supseteq \gamma_E(X)$, $r_{\mathscr{M}}(\gamma_E(X)) = |Y| + \sum_{W\in\mathscr{X}} b(W)$. Let us choose $\mathscr{X}$ and Y s.t. $|Y|$ is minimal and to minimize $|\mathscr{X}|$ with the above primary conditions. If Y contains a hyperedge e of size $c+1$, then it contains all the $k-d$ parallel copies of e. By removing e and its copies from Y and adding e to $\mathscr{X}$, we get a new Y and $\mathscr{X}$ contradicting the extreme choice.

Suppose that for each $X' \subseteq X$, $|X'| = c+2$, there exists $X' \subseteq W' \in \mathscr{X}$. If $|X| = c+2$, then we are done. If $|X| \geq c+3$, then there exists $X', X'' \subseteq X$, $|X'| = |X''| = c+2$, $|X' \cap X''| = c+1$, $X' \subseteq W' \in \mathscr{X}$ and $X'' \subseteq W'' \in \mathscr{X}$. Then $k|W'| - l + k|W''| - l > k|W' \cup W''| - l$, hence we could replace $\mathscr{X}$ by $\mathscr{X} - W' - W'' + \{W' \cup W''\}$.

Thus there exists an X' having no such W'. But the hyperedges of size $c+1$ contained in X' are covered by $\mathscr{X}$ i.e. for each $i \in X'$ there exists $W_i \in \mathscr{X}$, $i \notin W_i$, $X' - i \subseteq W_i$. Then setting $W = \bigcup_{i\in X'} W_i$, $|W| = \sum_{i\in X'} |W_i| - c(c+2)$. If $d \leq \frac{ck}{c+1}$, then $\sum_{i\in X'}(k|W_i| - l) \geq k|W| - l$, thus we could remove each W_i from $\mathscr{X}$ and insert W, contradicting the extreme choice of Y and $\mathscr{X}$. If $d > \frac{ck}{c+1}$, then X' is in E with multiplicity $cd + d - ck$, and $\mathscr{X}$ does not cover these elements, they are included in Y. In this case, $\sum_{i\in X'}(k|W_i| - l) + cd + d - ck \geq k|W| - l$, we could remove each W_i from $\mathscr{X}$, insert W, and remove X' and its parallel copies from Y, which yields again a contradiction. □

5 Algorithmic aspects

The first polynomial matroid matching algorithm to solve problems which are not known to be reduced to the matroid intersection and to the matching problem of graphs was presented to linear matroids by Lovász [9]. Later, Dress and Lovász [3] noticed that this algorithm can be used to the class of matroids having the DCP, provided that we are able to perform some algorithmic manipulations which handle flats and double circuits. If such techniques are available, then Lovász' algorithm can be used for matroids that have the DCP.

In the rest of this section a basic familiarity with Lovász' algorithm is supposed. To apply Lovász' algorithm for a matroid which has the DCP we first need an independence oracle, we have to be able to store and compute flats efficiently, to compute the intersection of two flats, and given a flat, we have to be able to compute an independent set spanning it. The other operations all can be derived from these ones.

First we sketch the oracles for a polynomial algorithm of $|V|$, $|E|$ and $|A|$. (In this case, if k and l are fixed, then $|E|$ is a polynomial of $|V|$, thus a polynomial algorithm of $|V|$ and $|A|$ follows.) As the running time is allowed to depend on $|E|$, we are able to have the whole ground-set E at hand throughout the algorithm. We have to observe that in this case, all the necessary operations can be derived from the independence oracle. The base of this oracle is the following claim.

Claim 14 *$F \subseteq E$ is independent in $\mathscr{M}$ if and only if $|\gamma_{F'}(X)| \leq k|X|$ for every $e \in F$ and $X \subseteq V$, where F' is obtained from F by adding l parallel copies of e.*

The condition stated in Claim 14 can be checked by flow-algorithms in time polynomial of $|V|$ and $|F|$. For a polynomial algorithm of $|V|$, $|E|$ and $|A|$, the oracles computing and handling flats all can be derived easily from this.

If the goal is to construct a polynomial algorithm of $|V|$ and $|A|$, then more refined techniques are needed. It is an important remark that Lovász' algorithm always deals with sets of pairs $M \subseteq A$ and with flats, the rank of which are bounded by a polynomial of $|A|$. Thus the independence of such sets can be checked by flow-algorithms in time polynomial of $|V|$ and $|A|$.

When the algorithm deals with a flat, then a family $\mathscr{X} \subseteq \binom{V}{>\frac{l}{k}}$ and a set $Y \subseteq E$ have to be stored. Each member of $\mathscr{X}$ and Y has a positive contribution to the rank which is bounded by a polynomial of $|A|$, thus the flat can be stored in polynomial space of $|V|$ and $|A|$.

For the computation of $\mathrm{sp}_{\mathscr{M}}(F)$ we have to compute $\mathscr{X}_F$ and Y_F. We may suppose that F is independent since a set and its maximal independent subsets have the same span. The exact way of computing $\mathrm{sp}_{\mathscr{M}}(F)$ is as follows. For each $f \in F$ we will determine whether a set $X \in \mathscr{X}_F$, $e \subseteq X$ exists or not. If does, then it has to be computed. Otherwise we know that $f \in Y_F$.

Consider now the bipartite graph G with color-classes V and F and edge-set $\{ve : v \in V, e \in F, v \in e\}$. A set $C \subseteq V \cup F$ is said to be a vertex-cover of G if $|\{v,e\} \cap C| \geq 1$ for every $ve \in E(G)$. For $f \in F$, let $c_f : V \cup F \to \mathbb{Z}$ be a cost-function defined by

$$c_f(w) = \begin{cases} l+1 & \text{if } w = f \\ 1 & \text{if } w \in E - \{f\} \\ k & \text{if } w \in V. \end{cases}$$

Claim 15 *Let C be a vertex-cover of G minimizing $c_f(C)$. Then either $l = 0$, or at least one of $f \notin C$ and $C \cap V = \emptyset$ holds.*

PROOF: For contradiction, suppose that $l > 0$, $f \in C$ and $C \cap V \neq \emptyset$. Since E is a vertex-cover, then $|F| + l = c_f(E) \geq c_f(C)$. Thus

$$c_f(C) - 2l = k|C \cap V| - l + |C \cap E| = r_{\mathscr{M}}(F) = |F| \geq c_f(C) - l$$

which is a contradiction. □

Using Claim 15, the correctness of the following statement can be checked easily.

Claim 16 *(i) Let $\mathscr{X} \subseteq \binom{V}{>\frac{l}{k}}$ and $Y \subseteq F$ be s.t. $Y \cup \bigcup_{X \in \mathscr{X}} \gamma_E(X) \supseteq F$, $r_{\mathscr{M}}(F) = |Y| + \sum_{X \in \mathscr{X}} b(X)$. Suppose moreover that either $\mathscr{X} = \emptyset$ or $\mathscr{X} = \{X\}$ s.t. $f \subseteq X$. Then $X \cup Y$ is a vertex-cover of cost $c_f(X \cup Y) \leq r_{\mathscr{M}}(F) + l$.*

(ii) If C is a vertex-cover, then let us define $X = C \cap V$ and $Y = C \cap F$. Then $\gamma_E(X) \cup Y \supseteq F$ and $|Y| + b(X) \leq c_f(C) - l$.

For any $f \in F$ we can determine by flow-algorithms whether a vertex-cover C with $C \cap V \supseteq f$ and $c_f(C) - l = r_{\mathscr{M}}(F)$ exists. If the answer is negative, then $f \in Y_F$. If the answer is positive, then $f \in \bigcup_{X \in \mathscr{X}_F} \gamma_E(X)$. In the last case, the goal is to determine the set $X \in \mathscr{X}_F$ s.t. $f \subseteq X$. This is done by computing a vertex-cover C s.t. $c_f(C) - l = r_{\mathscr{M}}(F)$ and $C \cap V$ is maximal. This problem again can be solved by flow-algorithms. Finally, a polynomial algorithm of $|V|$ and $|F|$ can be constructed from the above steps which determines $\mathscr{X}_F$ and Y_F.

Let us turn to the question of constructing the oracle which is able to compute an independent set spanning a given flat. The solution given here probably does not satisfy all expectations of the reader. Clearly, $|E|$ is not given in the input as a set, but in a very implicit way. Moreover, we have mentioned in the introductory part, that it does not count what hyperedges are used to fat up the matroid, the important is that they together give the desired structure of flats. Suppose now that each set $X \subseteq V$ of size bigger than $\frac{l}{k}$ is in E with multiplicity $k|X| - l$. The problem with this can be that the new hyperedges can appear in the min-max formula. This is the point to mention an other important property of the algorithm. Namely, the optimal set $Z \subseteq E$ in the min-max formula (2) is produced in the form $\bigcap_{M \in \mathscr{B}} \mathrm{sp}_{\mathscr{M}}(M)$ where $\mathscr{B}$ is a set of matchings of A. If we state Theorem 4 in the form that Z is a flat of $\mathscr{M}$, then the further hyperedges used in the algorithm have no importance, since Z is given back as a flat.

Now, our task is very easy. Suppose that in the run of the algorithm, we have a flat given by $\mathscr{X} \subseteq \binom{V}{>\frac{l}{k}}$ and $Y \subseteq E$. Then, we take each element of Y with multiplicity one, and each member X of $\mathscr{X}$ with multiplicity $k|X| - l$ as a hyperedge. Then this set is independent, and has the required cardinality.

Acknowledgments

The author wishes to thank Zsolt Fekete, András Frank and Jácint Szabó for useful discussions on the topic.

References

[1] C. Berge. Sur le couplage maximum d'un graphe. *C. R. Acad. Sci. Paris*, 247:258–259, 1958.

[2] A. Björner and L. Lovász. Pseudomodular lattices and continuous matroids. *Acta Sci. Math. (Szeged)*, 51(3-4):295–308, 1987.

[3] A. Dress and L. Lovász. On some combinatorial properties of algebraic matroids. *Combinatorica*, 7(1):39–48, 1987.

[4] W. Hochstättler and W. Kern. Matroid matching in pseudomodular lattices. *Combinatorica*, 9(2):145–152, 1989.

[5] P. M. Jensen and B. Korte. Complexity of matroid property algorithms. *SIAM J. Comput.*, 11(1):184–190, 1982.

[6] G. Laman. On graphs and rigidity of plane skeletal structures. *J. Engrg. Math.*, 4:331–340, 1970.

[7] L. Lovász. Matroid matching and some applications. *J. Combin. Theory Ser. B*, 28(2):208–236, 1980.

[8] L. Lovász. Selecting independent lines from a family of lines in a space. *Acta Sci. Math. (Szeged)*, 42(1-2):121–131, 1980.

[9] L. Lovász. The matroid matching problem. In *Algebraic methods in graph theory, Vol. I, II (Szeged, 1978)*, volume 25 of *Colloq. Math. Soc. János Bolyai*, pages 495–517. North-Holland, Amsterdam, 1981.

[10] Alexander Schrijver. *Combinatorial optimization. Polyhedra and efficiency. Vol. C*, volume 24 of *Algorithms and Combinatorics*. Springer-Verlag, Berlin, 2003. Disjoint paths, hypergraphs, Chapters 70–83.

[11] P. Tong, E. L. Lawler, and V. V. Vazirani. Solving the weighted parity problem for gammoids by reduction to graphic matching. In *Progress in combinatorial optimization (Waterloo, Ont., 1982)*, pages 363–374. Academic Press, Toronto, ON, 1984.

Fast Algorithms for Computing Jones Polynomials of Certain Links

MASAHIKO MURAKAMI

Graduate School of
Integrated Basic Sciences
Nihon University
Setagaya-ku, Tokyo 156-8550, Japan.
masahiko@tani.cs.chs.nihon-u.ac.jp

MASAO HARA[†]

Department of Mathematical Sciences
Tokai University
Hiratsuka-shi, Kanagawa 259-1292, Japan
masao@ss.u-tokai.ac.jp

MAKOTO YAMAMOTO[‡]

Department of Mathematics
Chuo University
Bunkyo-ku, Tokyo 112-8551, Japan
makotoy@math.chuo-u.ac.jp

SEIICHI TANI[§]

Department of
Computer Science and System Analysis
Nihon University
Setagaya-ku, Tokyo 156-8550, Japan
sei-ichi@tani.cs.chs.nihon-u.ac.jp

Abstract: We give fast algorithms for computing Jones polynomials of 2–bridge links and closed 3–braid links from their Tait graphs. Given a Tait graph with n edges, these algorithms run with $\mathcal{O}(n)$ arithmetic operations of polynomials of degree $\mathcal{O}(n)$, namely in $\mathcal{O}(n^2 \log n)$ time.

Keywords: combinatorial algorithm, computational topology, knot theory

1 Introduction

In knot theory, various invariants have been defined and well studied for classifying and characterizing links. The Jones polynomial [3] is powerful for distinguishing link types. L.H. Kauffman [4] gave a combinatorial method for calculating the Jones polynomial by means of the Kauffman bracket polynomial. We denote the number of the crossings of a link diagram $\widetilde{L}$ by $c(\widetilde{L})$. It takes $\mathcal{O}\left(2^{\mathcal{O}(\sqrt{c(\widetilde{L})})}\right)$ arithmetic operations of polynomials of degree $\mathcal{O}(c(\widetilde{L}))$ to compute a Jones polynomial by Kauffman's method. Actually, F. Jaeger, D. L. Vertigan and D. J. A. Welsh showed that computing the Jones polynomial is generally #**P**–hard [2, 10]. It is expected to require exponential time in the worst case. K. Sekine, H. Imai and K. Imai [8] designed an algorithm for computing Jones polynomials in $\mathcal{O}\left(2^{\mathcal{O}(\sqrt{c(\widetilde{L})})}\right)$ time.

Recently, it has been recognized that it is important to compute Jones polynomials for links with reasonable restrictions. J. A. Makowsky [5] showed that Jones polynomials are computed from the Tait graph G of a link diagram $\widetilde{L}$ in polynomial time if the treewidth of G is bounded by a constant. J. Mighton [6] showed that Jones polynomials are computed from the Tait graph G of a link diagram $\widetilde{L}$ with $\mathcal{O}(c(\widetilde{L})^4)$ operations of polynomials of degree $\mathcal{O}(c(\widetilde{L}))$ if the treewidth of G is at most 2. M. Hara, S. Tani and M. Yamamoto [1] showed that Jones polynomials of 2–bridge links are computed from the Tait graph of a link diagram $\widetilde{L}$ with $\mathcal{O}(c(\widetilde{L})^2)$ operations of polynomials of degree $\mathcal{O}(c(\widetilde{L}))$, and Jones polynomials of closed 3–braid links and arborescent links are computed from the Tait graph of a link diagram $\widetilde{L}$ with $\mathcal{O}(c(\widetilde{L})^3)$ operations of polynomials of degree $\mathcal{O}(c(\widetilde{L}))$. T. Utsumi and K. Imai [9] showed that Jones polynomials of pretzel links are computed from the Tait graph of a link diagram $\widetilde{L}$ in $\mathcal{O}(c(\widetilde{L})^2)$ time. We denote that the number of the Tait graph of a link diagram $\widetilde{L}$ is $c(\widetilde{L})$.

In this paper, we propose fast algorithms for computing Jones polynomials of 2–bridge links and closed 3–braid links. The 2–bridge links and the closed 3–braid links are basic classes of links and have been well studied. It is known that both 2–bridge links and closed 3–braid links can be represented in integer sequences. For designing our fast algorithms, we show that Jones polynomials of 2–bridge links and closed 3–braid links can be computed with $\mathcal{O}(c(\widetilde{L}))$ operations of polynomials of degree $\mathcal{O}(c(\widetilde{L}))$ from their sequences by means of simple recurrence formulas. We also show that the sequences of 2–bridge links and closed 3–braid links are able to be constructed in $\mathcal{O}(c(\widetilde{L}))$ time from their Tait graphs.

[†]Partially supported by Research and Study Program of Tokai University Educational System General Research Organization.
[‡]Partially supported by Chuo University Grant for Special Research 2004-2005.
[§]Partially supported by "Academic Frontier" Project for Private Universities: matching fund subsidy from MEXT (Ministry of Education, Culture, Sports, Science and Technology), 2003-2007.

The paper is organized in the following way. Section 2 contains some basic notations and definitions of knot theory. In Section 3, we provide algorithms for 2–bridge links. Section 4 deals with algorithms for closed 3–braid links.

2 Preliminaries

A *link* of n components is n simple closed curves in $\mathbb{R}^3$ that are mutually disjoint. A link of one component is a *knot*. An image of a link by an orthogonal projection from $\mathbb{R}^3$ to a plane is *regular* if it contains only finitely many multiple points, all multiple points are double points and these are traverse points. A regular image of a link is called a *link diagram* if the overcrossing line is marked at every double point in the image. Furthermore, the double points are called *crossings*. For any link diagram $\widetilde{L}$, we denote the number of the crossings of $\widetilde{L}$ by $c(\widetilde{L})$. A link is *oriented* if each of its components is given an orientation.

Definition 1 The *Kauffman bracket polynomial* is a function from link diagrams to the Laurent polynomial ring $\mathbb{Z}[A^{\pm 1}]$ with integer coefficients in an indeterminate A. It maps a link diagram $\widetilde{L}$ to $\langle \widetilde{L} \rangle \in \mathbb{Z}[A^{\pm 1}]$ and is characterized by

$$\text{(i) } \langle \bigcirc \rangle = 1, \quad \text{(ii) } \langle \widetilde{L} \sqcup \bigcirc \rangle = (-A^{-2} - A^2)\langle \widetilde{L} \rangle \quad \text{and} \quad \text{(iii) } \langle \times \rangle = A\langle \;) (\; \rangle + A^{-1} \langle \asymp \rangle.$$

Here, $\bigcirc$ is the knot diagram without a crossing and $\widetilde{L} \sqcup \bigcirc$ is disjoint sum of $\widetilde{L}$ and $\bigcirc$. In (iii), the formula refers to three link diagrams that are exactly the same except near a point where they differ in the way indicated.

The *writhe* $w(\widetilde{L})$ of an oriented link diagram $\widetilde{L}$ is the sum of the signs of the crossings of $\widetilde{L}$, where each crossing has sign $+1$ or -1 as defined (by convention) in Figure 1. The *Jones polynomial* $V(L)$ of an oriented link L is defined by $V(L) = (-A)^{-3w(\widetilde{L})} \langle \widetilde{L} \rangle \Big|_{t^{1/2}=A^{-2}}$, where $\widetilde{L}$ is an oriented link diagram of L. It is known that $V(L)$ is well-defined and $V(L) \in \mathbb{Z}[t^{\pm 1/2}]$.

Given any link diagram $\widetilde{L}$, we can color the faces black and white in such a way that no two faces with a common edge are the same color. We color the unique unbounded face white. Such a coloring is called the *Tait coloring* of $\widetilde{L}$. As in shown Figure 2, we can get an edge-labeled planar graph G of $\widetilde{L}$, its vertices are the black faces of the Tait coloring and two vertices are joined by a labeled edge if they share a crossing. The label of the edge is $+1$ or -1 according to the (conventional) rule shown in Figure 3. We may call the label the sign. We call G the *Tait graph* of $\widetilde{L}$. Note that the number of the edges of G is $c(\widetilde{L})$. A Tait graph G is *isomorphic* to a Tait graph G' if there exists a bijection f from the vertex set of G to the vertex set of G' satisfies the followings:

1. For any pair of vertices u and v of G, the number of the edges in G that joins u and v and are labeled "$+1$" is equal to the number of the edges in G' that joins $f(u)$ and $f(v)$ and are labeled "$+1$".
2. For any pair of vertices u and v of G, the number of the edges in G that joins u and v and are labeled "-1" is equal to the number of the edges in G' that joins $f(u)$ and $f(v)$ and are labeled "-1".

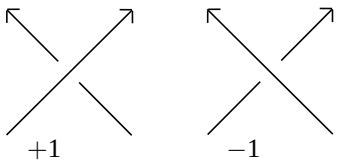


Figure 1: Signs of crossings

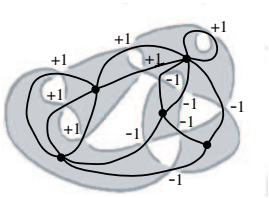


Figure 2: A Tait graph

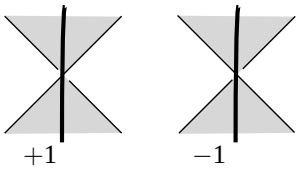


Figure 3: Signs of edges

A *tangle* is a portion of a link diagram from which there emerge just 4 arcs pointing in the compass directions NW, NE, SW, SE. The tangle consisting of two vertical strings without a crossing is called 0*–tangle*. The 0–tangle twisted k times is called k*–tangle* and is denoted by I_k. They are called *integer tangles* (see Figure 4). For a link diagram $\widetilde{L}$, a link diagram $\widetilde{L}\#(k)$ is shorthand for a link diagram twisted it k times as in shown Figure 5. For convenience, $\widetilde{L}\#(0)$ denotes $\widetilde{L}$ itself.

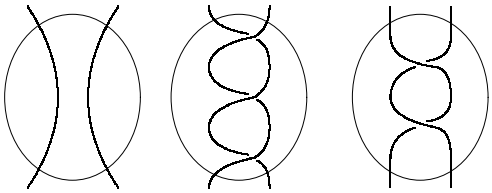


Figure 4: Integer tangles

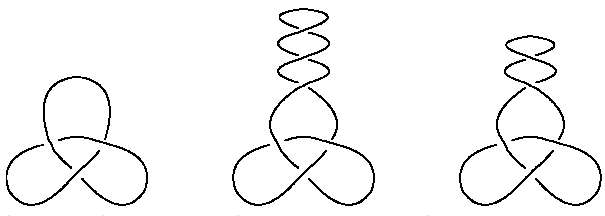


Figure 5: $\widetilde{L}\#(k)$

Proposition 2 *For any link diagram $\widetilde{L}$ and any integer k, $\langle \widetilde{L}\#(k)\rangle = (-A^{-3})^k \langle \widetilde{L}\rangle$ holds.*

Let $G=(V,E,l)$ be a Tait graph, where V is the vertex set of G, E is the edge set of G and l is the edge-labeling function from E to $\{-1,1\}$. For any vertex $v \in V$, $\deg_G(v)$ denotes the degree of v in G, $N_G(v)$ denotes the set of the neighbors of v in G and $G-v$ denotes the subgraph of G induced by $V-\{v\}$. We define a function edge_sign$_G$ from $V\times V$ to $\mathbb{Z}$ such that for any pair of vertices $u,v \in V$, edge_sign$_G(u,v)$ is the sum of the signs of the edges of G that join u and v. For a set S, $|S|$ denotes the size of S. For an integer n, we set

$$Q_n(X) = \frac{1-X^n}{1-X} = \begin{cases} 1+X+\cdots+X^{n-1} & \text{if } n>0, \\ 0 & \text{if } n=0, \\ -X^{-1}-X^{-2}-\cdots-X^n & \text{if } n<0. \end{cases}$$

Note that

$$XQ_n(X)+1 = Q_{n+1}(X) \text{ and } X^{-1}Q_{n+1}(X) - X^{-1} = Q_n(X). \tag{1}$$

Lemma 3 *Let $\widetilde{L}$ be a link diagram, k an integer and $\{b_n\}_{n\in\mathbb{Z}}$ a sequence of polynomials in $\mathbb{Z}[A^{\pm 1}]$. Suppose that for any integer n,*

$$b_n = Ab_{n-1} + A^{-1}\langle \widetilde{L}\#(n+k)\rangle.$$

Then,

$$b_n = A^n b_0 - (-A)^{-3(n+k)-1}\langle \widetilde{L}\rangle Q_n(-A^4).$$

PROOF: We prove by induction on $|n|$. By Proposition 2, we have

$$b_n = Ab_{n-1} + A^{-1}\langle \widetilde{L}\#(n+k)\rangle = Ab_{n-1} + A^{-1}(-A^{-3})^{n+k}\langle \widetilde{L}\rangle = Ab_{n-1} - (-A)^{-3(n+k)-1}\langle \widetilde{L}\rangle.$$

If $n=0$, then $b_0 = A^0 b_0 - (-A)^{-3k-1}\langle \widetilde{L}\rangle Q_0(-A^4)$ since $Q_0(-A^4)=0$.
If $n>0$, then, by induction hypothesis and the equations (1), we have

$$\begin{aligned} b_n &= Ab_{n-1} - (-A)^{-3(n+k)-1}\langle \widetilde{L}\rangle \\ &= A\{A^{n-1}b_0 - (-A)^{-3(n+k-1)-1}\langle \widetilde{L}\rangle Q_{n-1}(-A^4)\} - (-A)^{-3(n+k)-1}\langle \widetilde{L}\rangle \\ &= A^n b_0 - (-A)^{-3(n+k)-1}\langle \widetilde{L}\rangle\{(-A^4)Q_{n-1}(-A^4)+1\} \\ &= A^n b_0 - (-A)^{-3(n+k)-1}\langle \widetilde{L}\rangle Q_n(-A^4). \end{aligned}$$

If $n<0$, then, by $b_{n+1} = Ab_n - (-A)^{-3(n+k+1)-1}\langle \widetilde{L}\rangle$, induction hypothesis and the equations (1), we have

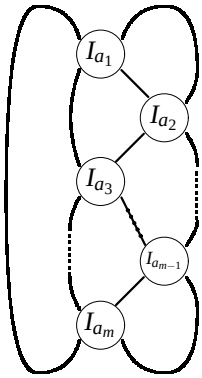
$$\begin{aligned} b_n &= A^{-1}b_{n+1} + A^{-1}(-A)^{-3(n+k+1)-1}\langle \widetilde{L}\rangle \\ &= A^{-1}\{A^{n+1}b_0 - (-A)^{-3(n+k+1)-1}\langle \widetilde{L}\rangle Q_{n+1}(-A^4)\} - A^{-4}(-A)^{-3(n+k)-1}\langle \widetilde{L}\rangle \\ &= A^n b_0 - (-A)^{-3(n+k)-1}\langle \widetilde{L}\rangle\{(-A^{-4})Q_{n+1}(-A^4) - A^{-4}\} \\ &= A^n b_0 - (-A)^{-3(n+k)-1}\langle \widetilde{L}\rangle Q_n(-A^4). \end{aligned}$$

□

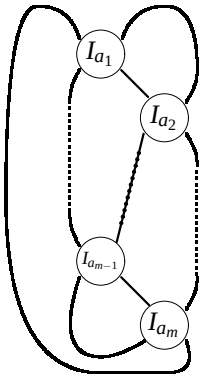
3 Algorithms for 2–bridge links

We denote the link diagram consisting of integer tangles I_{a_k} as in shown Figure 6 by $\widetilde{R}(a_1,\ldots,a_m)$ where $a_1,\ldots,a_m$ are integers. The Tait graph of $\widetilde{R}(a_1,\ldots,a_m)$ is denoted by $G_R(a_1,\ldots,a_m)$. Note that any such Tait graph $G=(V,E,l)$ has a vertex $v\in V$ such that $G-v$ is a path (see Figure 7). We call a link diagram $\widetilde{L}$ a *2–bridge diagram* if the Tait graph $G=(V,E,l)$ of $\widetilde{L}$ satisfies the followings:

1. There exists a vertex $v\in V$ such that
   (a) $G-v$ is a path,
   (b) both of endvertices of the path $G-v$ are adjacent to v in G and
   (c) for any vertex $u\in N_G(v)$, all of the edges that join u and v have the same sign.
2. For any vertex $u\in V$, if $\deg_G(u)=2$, then the two edges incident to u have the same sign.
3. G has no loop edge.

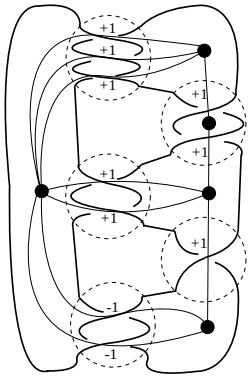


m is an odd number



m is an even number

Figure 6: $\widetilde{R}(a_1,\dots,a_m)$



$\widetilde{R}(-3,2,-2,1,2)$

Figure 7: An example of a 2–bridge diagram and its Tait graph

It is clear that for any 2–bridge diagram $\widetilde{L}$, there exists a sequence $(a_1,\dots,a_m)$ such that $\widetilde{R}(a_1,\dots,a_m)=\widetilde{L}$, namely $G_R(a_1,\dots,a_m)$ is the Tait graph of $\widetilde{L}$. We remark that the expression $G_R(a_1,\dots,a_m)$ is not unique. A link L is a *2–bridge link*[*] if L has a 2–bridge diagram.

Given the Tait graph G of a 2–bridge diagram, Procedure seq_2–bridge constructs a sequence $(a_1,\dots,a_m)$ such that $G_R(a_1,\dots,a_m)$ and G are isomorphic.

Procedure seq_2–bridge
Input: The Tait graph $G=(V,E,l)$ of a 2–bridge diagram.
Output: A sequence $(a_1,\dots,a_m)$ such that $G_R(a_1,\dots,a_m)$ and G are isomorphic.
Index a vertex $u_0\in V$ in the way which $\deg_G(u_0)$ is the maximum degree of G;
Compute $N_{G-u_0}(v)$ for all vertices $v\in V-\{u_0\}$;
Index a vertex $u_1\in V-\{u_0\}$ in the way which u_1 is an endvertex of the path $G-u_0$;
`for` i := 2 `to` $|V|-1$ `do` index a vertex $u_i\in N_{G-u_0}(u_{i-1})$ in the way which u_i is not u_{i-2};
Compute edge_sign$_G(u_0,u_i)$ for $i=1,\dots,|V|-1$;
Compute edge_sign$_G(u_j,u_{j+1})$ for $j=1,\dots,|V|-2$;
Initialize i as "1" and k as "1";
`while` $i<|V|-1$ `do begin`
  { k is an odd number }
  a_k := $-$edge_sign$_G(u_0,u_i)$; increment k;
  { k is an even number }
  initialize a_k as "0";
  `repeat` a_k := a_k+edge_sign$_G(u_i,u_{i+1})$; increment i;
  `until` $\deg_G(u_i)\neq 2$ `or` $i=|V|-1$;
  increment k;
`end`;
$a_k:=-$edge_sign$_G(u_0,u_{|V|-1})$;

Lemma 4 Procedure seq_2–bridge *constructs a sequence* $(a_1,\dots,a_m)$ *such that* $G_R(a_1,\dots,a_m)$ *and* G *are isomorphic in* $\mathcal{O}(|E|)$ *time.*

Lemma 5 *For any integer sequence* $(a_1,\dots,a_m)$*, the following recurrence formula holds.*

$$\langle\widetilde{R}(a_1,\dots,a_m)\rangle=\begin{cases}A^{a_1}(-A^{-2}-A^2)-(-A)^{-3a_1+2}Q_{a_1}(-A^4) & \text{if } m=1,\\ A^{a_2}(-A^{-3})^{a_1}-(-A)^{-3a_2+2}\langle\widetilde{R}(a_1)\rangle Q_{a_2}(-A^4) & \text{if } m=2,\\ A^{a_m}(-A^{-3})^{a_{m-1}}\langle\widetilde{R}(a_1,\dots,a_{m-2})\rangle & \\ \quad -(-A)^{-3a_m+2}\langle\widetilde{R}(a_1,\dots,a_{m-1})\rangle Q_{a_m}(-A^4) & \text{if } m\geq 3.\end{cases}$$

PROOF: We consider the case $m=1$. We have

$$\langle\widetilde{R}(a_1)\rangle=A\langle\widetilde{R}(a_1-1)\rangle+A^{-1}\langle\bigcirc\#(a_1-1)\rangle \tag{2}$$

[*]Schubert [7] defi ned a numerical link invariant called bridge number. For any link diagram, an *overpass* is a subarc of the link diagram that goes over at least one crossing but never goes under a crossing. A *maximal overpass* is an overpass that could not be made any longer. The *bridge number* of a link diagram is the number of maximal overpasses in the link diagram. The *bridge number* of a link is the least bridge number of all of the link diagrams of the link. A *2–bridge link* is a link whose bridge number is 2.

by applying Definition 1 (iii) to a crossing of I_{a_1} of $\widetilde{R}(a_1)$. We also have

$$\langle \widetilde{R}(0) \rangle = -A^{-2} - A^2 \tag{3}$$

by Definition 1 (i) and (ii) because $\widetilde{R}(0)$ is $\bigcirc \sqcup \bigcirc$. Hence, the equations (2) and (3) imply the case where $m = 1$ by Lemma 3.

We consider the case $m = 2$. We have

$$\langle \widetilde{R}(a_1, a_2) \rangle = A \langle \widetilde{R}(a_1, a_2 - 1) \rangle + A^{-1} \langle \widetilde{R}(a_1) \# (a_2 - 1) \rangle \tag{4}$$

by applying Definition 1 (iii) to a crossing of I_{a_2} of $\widetilde{R}(a_1, a_2)$. We also have

$$\langle \widetilde{R}(a_1, 0) \rangle = (-A^{-3})^{a_1} \langle \bigcirc \rangle \tag{5}$$

by Proposition 2 because $\widetilde{R}(a_1, 0)$ is $\bigcirc \# (a_1)$. Hence, the equations (4) and (5) imply the case where $m = 2$ by Lemma 3.

We consider the case $m \geq 3$. We have

$$\langle \widetilde{R}(a_1, \ldots, a_m) \rangle = A \langle \widetilde{R}(a_1, \ldots, a_{m-1}, a_m - 1) \rangle + A^{-1} \langle \widetilde{R}(a_1, \ldots, a_{m-1}) \# (a_m - 1) \rangle \tag{6}$$

by applying Definition 1 (iii) to a crossing of I_{a_m} of $\widetilde{R}(a_1, \ldots, a_m)$. We also have

$$\langle \widetilde{R}(a_1, \ldots, a_{m-1}, 0) \rangle = (-A^{-3})^{a_{m-1}} \langle \widetilde{R}(a_1, \ldots, a_{m-2}) \rangle \tag{7}$$

by Proposition 2 because $\widetilde{R}(a_1, \ldots, a_{m-1}, 0)$ is $\widetilde{R}(a_1, \ldots, a_{m-2}) \# (a_{m-1})$. Hence, the equations (6) and (7) imply the case where $m \geq 3$ by Lemma 3. □

Given a sequence $(a_1, \ldots, a_m)$, Procedure bra_2–bridge computes the Kauffman bracket polynomial $\langle \widetilde{R}(a_1, \ldots, a_m) \rangle$ by using the recurrence formula in Lemma 5. While the procedure is running, every Kauffman bracket polynomial is computed once at most.

Procedure bra_2–bridge
  Input: An integer sequence $(a_1, \ldots, a_m)$.
  Output: The Kauffman bracket polynomial $\langle \widetilde{R}(a_1, \ldots, a_m) \rangle$.
  Compute $\langle \widetilde{R}(a_1) \rangle$ and $\langle \widetilde{R}(a_1, a_2) \rangle$;
  `for` i := 3 `to` m `do begin`
    $T := Q_{a_i}(-A^4)$;
    Compute $\langle \widetilde{R}(a_1, \ldots, a_i) \rangle$ from $\langle \widetilde{R}(a_1, \ldots, a_{i-2}) \rangle$, $\langle \widetilde{R}(a_1, \ldots, a_{i-1}) \rangle$ and T;
  `end;`

Lemma 6 Procedure bra_2–bridge *computes the Kauffman bracket polynomial* $\langle \widetilde{R}(a_1, \ldots, a_m) \rangle$ *with* $\mathcal{O}(c(\widetilde{R}(a_1, \ldots, a_m)))$ *operations of polynomials of degree* $\mathcal{O}(c(\widetilde{R}(a_1, \ldots, a_m)))$.

PROOF: It is clear that the procedure computes the Kauffman bracket polynomial $\langle \widetilde{R}(a_1, \ldots, a_m) \rangle$ from a sequence $(a_1, \ldots, a_m)$. We estimate the running time of the procedure. For $i = 1, \ldots, m$, we can compute $Q_{a_i}(-A^4)$ in $\mathcal{O}(|a_i|)$ time. We can compute $\langle \widetilde{R}(a_1) \rangle$ with $\mathcal{O}(1)$ operations of polynomials of degree $\mathcal{O}(c(\widetilde{R}(a_1, \ldots, a_m)))$ from $Q_{a_1}(-A^4)$ by Lemma 5. We can compute $\langle \widetilde{R}(a_1, a_2) \rangle$ with $\mathcal{O}(1)$ operations of polynomials of degree $\mathcal{O}(c(\widetilde{R}(a_1, \ldots, a_m)))$ from $\langle \widetilde{R}(a_1) \rangle$ and $Q_{a_2}(-A^4)$ by Lemma 5. For $i = 3, \ldots, m$, we can compute $\langle \widetilde{R}(a_1, \ldots, a_i) \rangle$ with $\mathcal{O}(1)$ operations of polynomials of degree $\mathcal{O}(c(\widetilde{R}(a_1, \ldots, a_m)))$ from $\langle \widetilde{R}(a_1, \ldots, a_{i-2}) \rangle$, $\langle \widetilde{R}(a_1, \ldots, a_{i-1}) \rangle$ and $Q_{a_i}(-A^4)$ by Lemma 5. Therefore, the procedure finishes with $\mathcal{O}(c(\widetilde{R}(a_1, \ldots, a_m)))$ operations of polynomials of degree $\mathcal{O}(c(\widetilde{R}(a_1, \ldots, a_m)))$ since $c(\widetilde{R}(a_1, \ldots, a_m)) = |a_1| + \cdots + |a_m|$. □

Theorem 7 *The Jones polynomial of a 2–bridge link is computed from the Tait graph of a 2–bridge diagram* $\widetilde{L}$ *with* $\mathcal{O}(c(\widetilde{L}))$ *operations of polynomials of degree* $\mathcal{O}(c(\widetilde{L}))$.

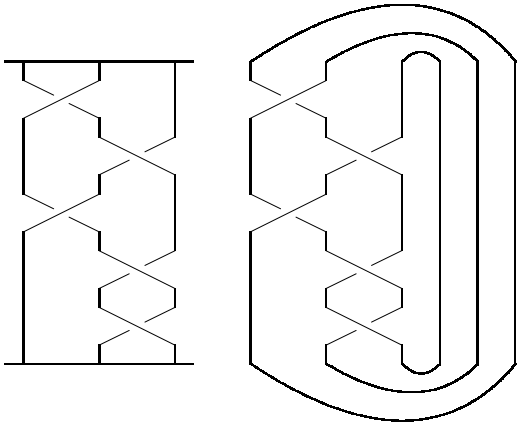
4 Algorithms for closed 3–braid links

A *3–braid* is a mutually disjoint 3 strings, all of which are attached to a horizontal bar at the top and at the bottom and each string intersects any horizontal plane between the two bars exactly once. Given any 3–braid, its ends on the bottom edge may be joined to those on the top edge to produce a *closed 3–braid link* (see Figure 8).

We denote the link diagram consisting of integer tangles I_{a_k} as in shown Figure 9 by $\widetilde{B}(a_1,\dots,a_m)$ where and $a_1,\dots,a_m$ are integers. The Tait graph of $\widetilde{B}(a_1,\dots,a_m)$ is denoted by $G_B(a_1,\dots,a_m)$. Note that any such Tait graph $G=(V,E,l)$ has a vertex $v\in V$ such that $G-v$ is a cycle or a graph consisting of one vertex with no loop edge. We call a link diagram $\widetilde{L}$ a *closed 3–braid diagram* if the Tait graph $G=(V,E,l)$ of $\widetilde{L}$ has no loop edge and there exists a vertex $v\in V$ satisfying the followings:

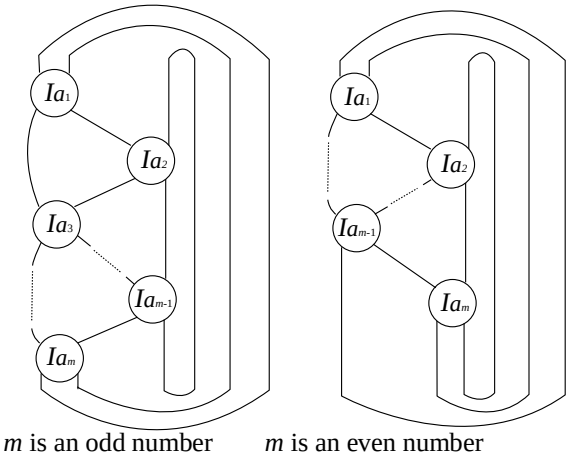
1. $G-v$ is a cycle or a graph consisting of one vertex.
2. For any vertex $u\in N_G(v)$, all of the edges that join u and v have the same sign.
3. For any vertex $u\in V-\{v\}$, if $\deg_G(u)=2$, then the two edges incident to u have the same sign.

It is clear that for any closed 3–bridge diagram $\widetilde{L}$, there exists a sequence $(a_1,\dots,a_m)$ such that $\widetilde{B}(a_1,\dots,a_m)=\widetilde{L}$, namely $G_B(a_1,\dots,a_m)$ is the Tait graph of $\widetilde{L}$. We remark that the expression $G_B(a_1,\dots,a_m)$ is not unique. Any closed 3–braid link has a closed 3–braid diagram.



a 3–braid a closed 3–braid link

Figure 8: A 3–braid and a closed 3–braid link



m is an odd number m is an even number

Figure 9: $\widetilde{B}(a_1,\dots,a_m)$

Given the Tait graph G of a closed 3–braid diagram, Procedure seq_3–braid constructs a sequence $(a_1,\dots,a_m)$ such that $G_B(a_1,\dots,a_m)$ and G are isomorphic.

Procedure seq_3–braid
Input: The Tait graph $G=(V,E,l)$ of a closed 3–braid diagram.
Output: A sequence $(a_1,\dots,a_m)$ such that $G_B(a_1,\dots,a_m)$ and G are isomorphic.
`if` $|V|>3$ and there exists a vertex $v\in V$ such that $|N_G(v)|\geq 4$ or $|N_G(v)|\leq 1$ `then`
  index v in the way which u_0 is v
`else` index a vertex $u_0\in V$ such that $G-u_0$ is a cycle or a graph consisting of one vertex
  and for any vertex $u\in N_G(u_0)$, all of the edges that join u_0 and u have the same sign;
Compute $N_{G-u_0}(v)$ for all vertices $v\in V-\{u_0\}$;
`if` $|N_G(u_0)|>0$ `then` index a vertex $v\in N_G(u_0)$ in the way which u_1 is v
`else` index a vertex $v\in V-\{u_0\}$ in the way which u_1 is v;
`for` i := 2 `to` $|V|-1$ `do` index a vertex $u_i\in N_{G-u_0}(u_{i-1})$ in the way which u_i is not u_{i-2};
Compute edge_sign$_G(u_0,u_i)$ for $i=1,\dots,|V|-1$;
Compute edge_sign$_G(u_{|V|-1},u_0)$ and edge_sign$_G(u_j,u_{j+1})$ for $j=1,\dots,|V|-2$;
Initialize i as "1" and k as "1";
`repeat`
  { k is an odd number }
  initialize a_k as "0";
  `repeat`
    `if` $i<|V|-1$ `then` a_k := a_k+edge_sign$_G(u_i,u_{i+1})$;
    `else` a_k := a_k+edge_sign$_G(u_{|V|-1},u_0)$;
    increment i;
  `until` $i=|V|$ `or` $\deg_G(u_i)\neq 2$;
  increment k;
  { k is an even number }

```
  if i < |V| then a_k := −edge_sign_G(u_0, u_i) else a_k := −edge_sign_G(u_0, u_1);
  increment k;
until i = |V|;
if for any v ∈ V, |N_G(v)| = 2 then begin
  if edge_sign_G(u_1, u_2) = 2 then begin a_1 := 1; a_3 := 1; end;
  if edge_sign_G(u_1, u_2) = 0 then begin a_1 := 1; a_3 := −1; end;
  if edge_sign_G(u_1, u_2) = −2 then begin a_1 := −1; a_3 := −1; end;
end;
```

Lemma 8 Procedure seq_3–braid *constructs a sequence* $(a_1,\dots,a_m)$ *such that* $G_B(a_1,\dots,a_m)$ *and* G *are isomorphic in* $\mathcal{O}(|E|)$ *time.*

Lemma 9 *For any integer sequence* $(a_1,\dots,a_m)$*, the following recurrence formula holds.*

$$\langle\widetilde{B}(a_1,\dots,a_m)\rangle = \begin{cases} (-A^{-2}-A^2)\langle\widetilde{R}(a_1)\rangle & \text{if } m=1,\\ A^{a_m}\langle\widetilde{B}(a_1,\dots,a_{m-1})\rangle - (-A)^{-3a_m+2} & \text{if } m\geq 2 \text{ and}\\ \quad\times\langle\widetilde{R}(a_1,\dots,a_{m-1})\rangle Q_{a_m}(-A^4) & \quad m \text{ is an even number,}\\ A^{a_m}\langle\widetilde{B}(a_1,\dots,a_{m-1})\rangle - (-A)^{-3(a_m+a_1)+2} & \text{if } m\geq 3 \text{ and}\\ \quad\times\langle\widetilde{R}(a_2,\dots,a_{m-1})\rangle Q_{a_m}(-A^4) & \quad m \text{ is an odd number.} \end{cases}$$

PROOF: It implies the case where $m=1$ by Definition 1 (ii) that $\widetilde{B}(a_1)$ is $\widetilde{R}(a_1)\sqcup\bigcirc$.

We consider the case where $m\geq 2$ and m is an even number. We have

$$\langle\widetilde{B}(a_1,\dots,a_m)\rangle = A\langle\widetilde{B}(a_1,\dots,a_m-1)\rangle + A^{-1}\langle\widetilde{R}(a_1,\dots,a_{m-1})\#(a_m-1)\rangle \tag{8}$$

by applying Definition 1 (iii) to a crossing of I_{a_m} of $\widetilde{B}(a_1,\dots,a_m)$. We also have

$$\langle\widetilde{B}(a_1,\dots,a_{m-1},0)\rangle = \langle\widetilde{B}(a_1,\dots,a_{m-1})\rangle\rangle \tag{9}$$

because $\widetilde{B}(a_1,\dots,a_{m-1},0)$ is $\widetilde{B}(a_1,\dots,a_{m-1})$. Hence, the equations (8) and (9) imply the case where $m\geq 2$ and m is an even number by Lemma 3.

We consider the case where $m\geq 3$ and m is an odd number, we get

$$\langle\widetilde{B}(a_1,\dots,a_m)\rangle = A\langle\widetilde{B}(a_1,\dots,a_m-1)\rangle + A^{-1}\langle\widetilde{R}(a_2,\dots,a_{m-1})\#(a_m+a_1-1)\rangle\rangle \tag{10}$$

by applying Definition 1 (iii) to a crossing of I_{a_m} of $\widetilde{B}(a_1,\dots,a_m)$. We also have

$$\langle\widetilde{B}(a_1,\dots,a_{m-1},0)\rangle = \langle\widetilde{B}(a_1,\dots,a_{m-1})\rangle\rangle \tag{11}$$

because $\widetilde{B}(a_1,\dots,a_{m-1},0)$ is $\widetilde{B}(a_1,\dots,a_{m-1})$. Hence, the equations (8) and (9) imply the case where $m\geq 3$ and m is an odd number by Lemma 3. □

Given a sequence $(a_1,\dots,a_m)$, Procedure bra_3–braid computes the Kauffman bracket polynomial $\langle\widetilde{B}(a_1,\dots,a_m)\rangle$ by using the recurrence formulas in Lemma 5 and Lemma 9. While the procedure is running, every Kauffman bracket polynomial is computed once at most.

Procedure bra_3–braid
  Input: An integer sequence $(a_1,\dots,a_m)$.
  Output: The Kauffman bracket polynomial $\langle\widetilde{B}(a_1,\dots,a_m)\rangle$.
  Compute $\langle\widetilde{R}(a_1)\rangle$, $\langle\widetilde{B}(a_1)\rangle$, $\langle\widetilde{B}(a_1,a_2)\rangle$, $\langle\widetilde{R}(a_1,a_2)\rangle$, $\langle\widetilde{R}(a_2)\rangle$, $\langle\widetilde{B}(a_1,a_2,a_3)\rangle$, $\langle\widetilde{R}(a_1,a_2,a_3)\rangle$ and $\langle\widetilde{R}(a_2,a_3)\rangle$;

```
  for i := 4 to m do begin
    T := Q_{a_i}(−A^4);
    if i is an even number then
      Compute ⟨B̃(a_1,...,a_i)⟩ from ⟨B̃(a_1,...,a_{i−1})⟩, ⟨R̃(a_1,...,a_{i−1})⟩ and T
    else Compute ⟨B̃(a_1,...,a_i)⟩ from ⟨B̃(a_1,...,a_{i−1})⟩, ⟨R̃(a_2,...,a_{i−1})⟩ and T;
    Compute ⟨R̃(a_1,...,a_i)⟩ from ⟨R̃(a_1,...,a_{i−2})⟩, ⟨R̃(a_1,...,a_{i−1})⟩ and T;
    Compute ⟨R̃(a_2,...,a_i)⟩ form ⟨R̃(a_2,...,a_{i−2})⟩, ⟨R̃(a_2,...,a_{i−1})⟩ and T;
  end;
```

Lemma 10 Procedure bra_3–braid *computes the Kauffman bracket polynomial* $\langle\widetilde{B}(a_1,\dots,a_m)\rangle$ *with* $\mathcal{O}(c(\widetilde{B}(a_1,\dots,a_m)))$ *operations of polynomials of degree* $\mathcal{O}(c(\widetilde{B}(a_1,\dots,a_m)))$.

Theorem 11 *The Jones polynomial of a closed 3–braid link is computed from the Tait graph of a closed 3–braid diagram with* $\mathcal{O}(c(\widetilde{L}))$ *operations of polynomials of degree* $\mathcal{O}(c(\widetilde{L}))$.

References

[1] M. HARA, S. TANI, M. YAMAMOTO, A polynomial-time algorithm for computing the Jones polynomials of arborescent links (in Japanese), *Information Technology Letters* (2002) **1**

[2] F. JAEGER, D. L. VERTIGAN, AND D. J. A. WELSH, On the computational complexity of the Jones and Tutte polynomials, *Math. Proc. Camb. Phil. Soc.* (1990) **108**

[3] V. F. R. JONES, A polynomial invariant for knots via Von Neumann algebras, *Bull. Amer. Math. Soc.* (1985) **12**

[4] L. H. KAUFFMAN, State models and the Jones polynomial, *Topology* (1987) **26**

[5] J. A. MAKOWSKY, Colored Tutte polynomials and Kauffman brackets for graphs of bounded tree width, *in Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, Philadelphia* (2001)

[6] J. MIGHTON, Knot Theory on Bipartite Graphs, *PhD thesis, Dept. of Math., University of Toronto, Canada* (1999)

[7] H. SCHUBERT, Über eine Numerische Knoteninvariante, *Math. Z.* (1954) **61**

[8] K. SEKINE, H. IMAI, AND K. IMAI, Computation of Jones Polynomial (in Japanese), *Transaction of the Japan Society for Industrial and Applied Mathematics* (1998) **8** No. 3

[9] T. UTSUMI, K. IMAI, Computation of the Jones Polynomials for Pretzel Links (in Japanese), *IPSJ SIG Technical Reports* (2002) No.085

[10] D. J. A. WELSH, *Complexity: Knots, Colorings and Counting*, *Cambridge Univ. Press, Cambridge*, (1993)

M-Convex Functions on Jump Systems: Generalization of Minsquare Factor Problem

KAZUO MUROTA*

Department of Mathematical Informatics
University of Tokyo
Tokyo 113-8656, Japan
murota@mist.i.u-tokyo.ac.jp

Abstract: The concept of M-convex functions is generalized for functions defined on constant-parity jump systems. Such function arises from minimum weight perfect *b*-matchings and from a separable convex function (sum of univariate convex functions) on the degree sequences of an undirected graph. As a generalization of a recent result of Apollonio and Sebő for the minsquare factor problem, a local optimality criterion is given for minimization of an M-convex function subject to a component sum constraint. The proposed framework leads to a polynomial-time algorithm for an edge-weighted extension of the minsquare factor problem.

Keywords: jump system, degree sequence, graph factor, discrete convex function, local optimality

1 Introduction

A recent paper of Apollonio and Sebő [2] has shown that the minsquare factor problem on a graph can be solved in polynomial time. The problem is, given an undirected graph possibly containing loops and parallel edges, to find a subgraph with a specified number of edges that minimizes the sum of squares of the degrees (= numbers of incident edges) of vertices. The key observation in [2] is that global optimality is guaranteed by local optimality in the neighborhood of ℓ_1-distance four in the space of degree sequences. It has also been observed in [2] that this local optimality criterion remains valid when the objective function is generalized to a separable convex function (= sum of univariate convex functions) of the degree sequence.

The objective of this paper is to put the above results in a more general context of discrete convex analysis [13] by introducing the concept of M-convex functions on constant-parity jump systems. A separable convex function of the degree sequences of a graph is an M-convex function in this sense. Furthermore, the proposed framework leads to a polynomial-time algorithm for an edge-weighted extension of the minsquare factor problem.

A jump system [3] is a set of integer points with an exchange property (described in Section 2); see also [9], [10]. Minimization of a separable convex function over a jump system has been studied in [1], where a local criterion for optimality as well as a greedy algorithm is given.

Study of nonseparable nonlinear functions on matroidal structures was started with valuated matroids [4], [5], which have come to be accepted as discrete concave functions; see [12]. This concept has been generalized to M-convex functions on base polyhedra, [11], which play a central role in discrete convex analysis [13]. Valuated delta-matroids [6] afford another generalization of valuated matroids; see also [15]. In all these generalizations global optimality is equivalent to local optimality defined in an appropriate manner. In addition, discrete duality such as discrete separation and min-max formula holds for valuated matroids and M-convex functions on base polyhedra, whereas it fails for valuated delta-matroids. M-convex functions on constant-parity jump systems, to be introduced in this paper, are a common generalization of valuated delta-matroids and M-convex functions on base polyhedra.

In this paper, we investigate into minimization of an M-convex function on a constant-parity jump system. It is shown, in particular, that (i) global optimality for unconstrained minimization is equivalent to local optimality in the neighborhood of ℓ_1-distance two (Theorem 11), and (ii) global optimality for constrained minimization on a hyperplane of a constant component sum is equivalent to local optimality in the neighborhood of ℓ_1-distance four (Theorem 14). The former generalizes the optimality criterion in [1] for separable convex function minimization over a jump system, and the latter the optimality criterion in [2] for the minsquare factor problem. Theorem 18 reveals convexity of the optimal values with respect to the component sum, on the basis of which algorithms are constructed for the constrained minimization in Section 5.

*This work is supported by PRESTO JST, by the 21st Century COE Program on Information Science and Technology Strategic Core, and by a Grant-in-Aid for Scientifi c Research from the Ministry of Education, Culture, Sports, Science and Technology of Japan.

2 Exchange Axioms

Let V be a finite set. For $u \in V$ we denote by χ_u the characteristic vector of u, with $\chi_u(u) = 1$ and $\chi_u(v) = 0$ for $v \neq u$. For $x = (x(v)), y = (y(v)) \in \mathbf{Z}^V$ define

$$x(V) = \sum_{v \in V} x(v), \quad ||x||_1 = \sum_{v \in V} |x(v)|, \quad \mathrm{supp}\,(x) = \{v \in V \mid x(v) \neq 0\},$$
$$\mathrm{supp}^+(x) = \{v \in V \mid x(v) > 0\}, \quad \mathrm{supp}^-(x) = \{v \in V \mid x(v) < 0\},$$
$$[x,y] = \{z \in \mathbf{Z}^V \mid \min(x(v),y(v)) \leq z(v) \leq \max(x(v),y(v)), \forall v \in V\}.$$

A vector $s \in \mathbf{Z}^V$ is called an (x,y)-*increment* if $s = \chi_u$ or $s = -\chi_u$ for some $u \in V$ and $x + s \in [x,y]$. An (x,y)-*increment pair* will mean a pair of vectors (s,t) such that s is an (x,y)-increment and t is an $(x+s,y)$-increment.

A nonempty set $J \subseteq \mathbf{Z}^V$ is said to be a *jump system* if satisfies an exchange axiom, called the *2-step axiom*: for any $x,y \in J$ and for any (x,y)-increment s with $x + s \notin J$, there exists an $(x+s,y)$-increment t such that $x + s + t \in J$. A set $J \subseteq \mathbf{Z}^V$ is a *constant-sum system* if $x(V) = y(V)$ for any $x,y \in J$, and a *constant-parity system* if $x(V) - y(V)$ is even for any $x,y \in J$.

We introduce a stronger exchange axiom:

(J-EXC) For any $x,y \in J$ and for any (x,y)-increment s, there exists an $(x+s,y)$-increment t such that $x + s + t \in J$ and $y - s - t \in J$.

This property characterizes a constant-parity jump system, a fact communicated to the author by J. Geelen (see [14] for a proof).

Lemma 1 (Geelen [8]) *A nonempty set J is a constant-parity jump system if and only if it satisfies* (J-EXC).

It turns out that (J-EXC) can be replaced by a weaker axiom:

(J-EXC$_{\mathrm{w}}$) For any distinct $x,y \in J$ there exists an (x,y)-increment pair (s,t) such that $x + s + t \in J$ and $y - s - t \in J$.

Lemma 2 (see [14]) *A set J satisfies* (J-EXC) *if and only if it satisfies* (J-EXC$_{\mathrm{w}}$).

We call $f : J \to \mathbf{R}$ an *M-convex function* if it satisfies the following exchange axiom:

(M-EXC) For any $x,y \in J$ and for any (x,y)-increment s, there exists an $(x+s,y)$-increment t such that $x + s + t \in J$, $y - s - t \in J$, and $f(x) + f(y) \geq f(x+s+t) + f(y-s-t)$.

We adopt the convention that $f(x) = +\infty$ for $x \notin J$.

It turns out that the exchange axiom (M-EXC) is equivalent to a local exchange axiom:

(M-EXC$_{\mathrm{loc}}$) For any $x,y \in J$ with $||x-y||_1 = 4$ there exists an (x,y)-increment pair (s,t) such that $x + s + t \in J$, $y - s - t \in J$, and $f(x) + f(y) \geq f(x+s+t) + f(y-s-t)$.

Theorem 3 (see [14]) *A function $f : J \to \mathbf{R}$ defined on a constant-parity jump system J satisfies* (M-EXC) *if and only if it satisfies* (M-EXC$_{\mathrm{loc}}$).

This implies that (M-EXC) can be replaced by a weaker axiom:

(M-EXC$_{\mathrm{w}}$) For any distinct $x,y \in J$ there exists an (x,y)-increment pair (s,t) such that $x + s + t \in J$, $y - s - t \in J$, and $f(x) + f(y) \geq f(x+s+t) + f(y-s-t)$.

Theorem 4 *A function $f : J \to \mathbf{R}$ satisfies* (M-EXC) *if and only if it satisfies* (M-EXC$_{\mathrm{w}}$).

PROOF: It suffices to prove the "if" part. (M-EXC$_{\mathrm{w}}$) implies (J-EXC$_{\mathrm{w}}$) for J, and hence J is a constant-parity jump by Lemma 2. Then the claim follows from Theorem 3. □

Note that addition of a linear function preserves M-convexity. That is, for an M-convex function f and a vector $p = (p(v)) \in \mathbf{R}^V$, the function $f[-p]$ defined by $f[-p](x) = f(x) - \langle p,x \rangle$ with $\langle p,x \rangle = \sum_{v \in V} p(v)x(v)$ is M-convex.

Remark 5 Our definition of an M-convex function is consistent with the previously considered special cases where (i) J is a constant-sum jump system, and (ii) J is a constant-parity jump system contained in $\{0,1\}^V$. Case (i) is equivalent to J being the set of integer points in the base polyhedron of an integral submodular system [7], and then our M-convex function is the same as the M-convex function investigated in [11], [13]. Case (ii) is equivalent to J being an even delta-matroid [15], and then f is M-convex in our sense if and only if $-f$ is a valuated delta-matroid in the sense of [6]. ■

Example 6 A *separable convex function* on a constant-parity jump system J, i.e., a function $f : J \to \mathbf{R}$ of the form $f(x) = \sum_{v\in V} \varphi_v(x(v))$ with univariate (one-dimensional) convex functions φ_v, is M-convex. In particular, the sum of squares $f(x) = \sum_{v\in V}(x(v))^2$ is M-convex. Such functions have been investigated in [1], [2]. ■

Example 7 Minimum weight factors in a graph yield an M-convex function. Let $G = (V,E)$ be an undirected graph that may contain loops and parallel edges. For a subgraph $H = (V,F)$, denote its *degree sequence* by $\deg_H = \sum\{\chi_u + \chi_v \mid (u,v) \in F\} \in \mathbf{Z}^V$. It is known [3], [10] that

$$J = \{\deg_H \mid H \text{ is a subgraph of } G\}$$

forms a constant-parity jump system, called the *degree system* of G. Given edge weighting $w : E \to \mathbf{R}$, define a function $f : J \to \mathbf{R}$ by

$$f(x) = \min\{w(F) \mid H = (V,F) \text{ is a subgraph of } G \text{ with } \deg_H = x\}$$

with notation $w(F) = \sum_{e\in F} w(e)$, where $f(x)$ represents the minimum weight of a subgraph with degree sequence x.

This f is an M-convex function. In fact, (M-EXC) can be verified by the alternating path argument as follows. For distinct $x,y \in J$ let F_x and F_y be subsets of edges such that $f(x) = w(F_x)$ and $f(y) = w(F_y)$ with $x = \sum\{\chi_u + \chi_v \mid (u,v) \in F_x\}$ and $y = \sum\{\chi_u + \chi_v \mid (u,v) \in F_y\}$. Let s be an (x,y)-increment, and put $u_* = \mathrm{supp}\,(s)$. We may assume, without loss of generality, that $s = \chi_{u_*}$. Starting with an edge in $F_y \setminus F_x$ incident to u_* we construct an alternating path P by adding an edge in $F_x \setminus F_y$ and an edge in $F_y \setminus F_x$ alternately. The path P is not necessarily simple so that it may contain the same vertex more than once, whereas it consists of distinct edges. We assume that P is maximal in the sense that it cannot be extended further beyond the end vertex, say, v_*. Then there exists an $(x+\chi_{u_*},y)$-increment t with $\mathrm{supp}\,(t) = v_*$; more specifically, $t = \chi_{v_*}$ or $-\chi_{v_*}$ according to whether P consists of odd or even number of edges. Denote by $F_x\Delta P$ the symmetric difference of F_x and P, and by $F_y\Delta P$ that of F_y and P. Since $x+s+t = \sum\{\chi_u + \chi_v \mid (u,v) \in F_x\Delta P\}$ and $y-s-t = \sum\{\chi_u + \chi_v \mid (u,v) \in F_y\Delta P\}$, we have $f(x+s+t) \le w(F_x\Delta P)$ and $f(y-s-t) \le w(F_y\Delta P)$, whereas $w(F_x\Delta P) + w(F_y\Delta P) = w(F_x) + w(F_y) = f(x) + f(y)$. Hence (M-EXC). Note that the alternating path argument above serves also as a proof of (J-EXC) for J. ■

Example 8 Combination of Examples 6 and 7 shows that

$$f(x) = \min\{w(F) \mid H = (V,F) \text{ is a subgraph of } G \text{ with } \deg_H = x\} + \sum_{v\in V} \varphi_v(x(v))$$

is an M-convex function on the degree system of G. By our present results, therefore, we can efficiently solve an edge-weighted extension of Apollonio–Sebő's problem: Given undirected graph $G = (V,E)$, edge weighting $w : E \to \mathbf{R}$, and univariate convex functions φ_v indexed by $v \in V$, find a subgraph $H = (V,F)$ that minimizes $w(F) + \sum_{v\in V} \varphi_v(\deg_H(v))$. Note that we can evaluate $f(x)$ in polynomial time. ■

3 Unconstrained Minimization

We consider minimization of an M-convex function $f : J \to \mathbf{R}$ defined on a constant-parity jump system $J \subseteq \mathbf{Z}^V$. First we note a property of an M-convex function that indicates its discrete convexity. Given $f : J \to \mathbf{R}$ and $x,y \in J$, a sequence of points in J, say, $x_0,x_1,\ldots,x_m$, is called a *steepest-descent chain* connecting x to y if $x_0 = x$, $x_m = y$, and for $i = 1,\ldots,m$ we have $x_i = x_{i-1} + s_i + t_i$ for some (x_{i-1},y)-increment pair (s_i,t_i) such that $f(x_{i-1}+s_i+t_i) \le f(x_{i-1}+s+t)$ for every (x_{i-1},y)-increment pair (s,t); we have $m = ||x-y||_1/2$. An M-convex function turns out to be convex along a steepest-descent chain, as follows.

Proposition 9 *Let $f : J \to \mathbf{R}$ be an M-convex function, and $x_0,x_1,\ldots,x_m$ be a steepest-descent chain connecting $x \in J$ to $y \in J$. Then*

$$f(x_{i-1}) + f(x_{i+1}) \ge 2f(x_i) \qquad (i = 1,\ldots,m-1). \tag{1}$$

PROOF: Put $x_i = x_{i-1} + s + t$ and $x_{i+1} = x_i + s' + t'$. By (M-EXC) we have

$$\begin{aligned} f(x_{i-1}) + f(x_{i+1}) \ \ge\ \min[&f(x_{i-1}+s+t) + f(x_{i-1}+s'+t'),\\ &f(x_{i-1}+s+t') + f(x_{i-1}+s'+t),\\ &f(x_{i-1}+s+s') + f(x_{i-1}+t+t')] \ \ge 2f(x_i). \end{aligned}$$

□

As an immediate corollary we see that a nonoptimal point can be improved with a suitable increment pair.

Proposition 10 (1) *If $x,y \in J$ and $f(x) > f(y)$, there exists an (x,y)-increment pair (s,t) such that $f(x) > f(x+s+t)$.*
(2) *If $x,y \in J$ and $f(x) \ge f(y)$, there exists an (x,y)-increment pair (s,t) such that $f(x) \ge f(x+s+t)$.*

This implies, in turn, that global optimality (minimality) of an M-convex function is guaranteed by local optimality in the neighborhood of ℓ_1-distance two.

Theorem 11 *Let $f : J \to \mathbf{R}$ be an M-convex function on a constant-parity jump system J, and let $x \in J$. Then $f(x) \le f(y)$ for all $y \in J$ if and only if $f(x) \le f(y)$ for all $y \in J$ with $||x-y||_1 \le 2$.*

PROOF: The "only if" part is obvious, and the "if" part follows from Proposition 10. □

The minimizers of an M-convex function form a constant-parity jump system, as follows. We denote by $\arg\min f[-p]$ the set of minimizers of function $f[-p]$.

Proposition 12 *For any $p \in \mathbf{R}^V$, $\arg\min f[-p]$ is a constant-parity jump system, if it is nonempty.*

PROOF: Let β denote the minimum value of $f[-p]$, and let $x, y \in \arg\min f[-p]$. Then, in (M-EXC) we have $2\beta = f[-p](x) + f[-p](y) \ge f[-p](x+s+t) + f[-p](y-s-t) \ge 2\beta$, which implies $x+s+t, y-s-t \in \arg\min f[-p]$. □

Remark 13 The local optimality criterion for M-convex functions on jump systems in Theorem 11 contains a number of previous results as special cases. In the case of constant-sum jump systems, case (i) in Remark 5, the present theorem reduces to the optimality criterion for M-convex functions on base polyhedra established in [11] (see Theorem 6.26 of [13]), and, moreover, Proposition 10 (1) above coincides with Proposition 6.23 of [13]. In the case of constant-parity jump systems contained in $\{0,1\}^V$, case (ii) in Remark 5, Theorem 11 reduces to the optimality criterion for valuated delta-matroids established in [6]. Both of these are generalizations, in different directions, of the optimality criterion for valuated matroids given in [4], [5]. It is noted that the optimality criterion for valuated matroids given in [4], [5] is the origin of this type of optimality criteria for nonseparable nonlinear objective functions, and the above two special cases are generalizations in different directions thereof. Separable convex functions on jump systems have been considered in [1]. ■

4 Minimization under Sum Constraint

In this section we investigate into minimization of an M-convex function $f(x)$ when the sum of the components of x is specified. Recalling the notation $x(V)$ for the sum of components of a vector x, we introduce some other notations concerning the feasible regions of our problem:

$$\begin{aligned} k_{\min} &= \min\{x(V) \mid x \in J\}, \quad k_{\max} = \max\{x(V) \mid x \in J\}, \\ \Lambda &= \{k \mid k_{\min} \le k \le k_{\max},\ k \equiv k_{\min} (\mathrm{mod}\ 2)\}, \\ J_k &= \{x \in J \mid x(V) = k\} \qquad (k \in \Lambda), \end{aligned}$$

where $J_k \neq \emptyset$ for each $k \in \Lambda$ by (J-EXC) and it may be that $k_{\min} = -\infty$ and/or $k_{\max} = +\infty$.

Our problem is to minimize $f(x)$ subject to $x \in J_k$, where $k \in \Lambda$ is a parameter. Denote by f_k and M_k the minimum value and the set of minimizers, respectively, i.e.,

$$f_k = \min\{f(x) \mid x \in J_k\}, \quad M_k = \{x \in J_k \mid f(x) = f_k\} \qquad (k \in \Lambda),$$

where we assume that, for each $k \in \Lambda$, f_k is finite and M_k is nonempty. By convention we put $f_k = +\infty$ for $k \notin \Lambda$.

Global optimality (minimality) on J_k is guaranteed by local optimality in the neighborhood of ℓ_1-distance four. Compare this with the unconstrained optimization treated in Theorem 11, which refers to the neighborhood of ℓ_1-distance two. It is emphasized that J_k is not necessarily a jump system, and accordingly, Theorem 11 does not apply to minimization of f over J_k.

Theorem 14 *Let $f : J \to \mathbf{R}$ be an M-convex function on a constant-parity jump system J, and let $x \in J_k$ with $k \in \Lambda$. Then $f(x) \le f(y)$ for all $y \in J_k$ if and only if $f(x) \le f(y)$ for all $y \in J_k$ with $||x-y||_1 \le 4$.*

PROOF: The "only if" part is obvious. To prove the "if" part by contradiction, assume that $f(x) > f(y)$ for some $y \in J_k$ and take such y with minimum $||y-x||_1$. Since $x(V) = y(V)$ and $x \neq y$, both $\mathrm{supp}^+(y-x)$ and $\mathrm{supp}^-(y-x)$ are nonempty.

Claim 1: If $u \in \mathrm{supp}^+(y-x)$ and $v \in \mathrm{supp}^-(y-x)$, then

$$f(x) + f(y) < f(x+\chi_u - \chi_v) + f(y - \chi_u + \chi_v).$$

Proof of Claim 1: We have $f(x) \le f(x+\chi_u-\chi_v)$ by the assumed local optimality, and $f(x) \le f(y-\chi_u+\chi_v)$ since $y-\chi_u+\chi_v$ is closer to x than y. Adding these two and $f(y) < f(x)$ yields the desired inequality.

By (M-EXC) for (x,y), together with Claim 1, there exist $u_1 \in \mathrm{supp}^+(y-x)$, $u_2 \in \mathrm{supp}^+(y-x-\chi_{u_1})$, $v_1 \in \mathrm{supp}^-(y-x)$, and $v_2 \in \mathrm{supp}^-(y-x-\chi_{v_1})$ such that

$$f(x)+f(y) \geq f(x+\chi_{u_1}+\chi_{u_2})+f(y-\chi_{u_1}-\chi_{u_2}), \tag{2}$$
$$f(x)+f(y) \geq f(x-\chi_{v_1}-\chi_{v_2})+f(y+\chi_{v_1}+\chi_{v_2}). \tag{3}$$

By (M-EXC) for $(x+\chi_{u_1}+\chi_{u_2}, x-\chi_{v_1}-\chi_{v_2})$ and the local optimality, we obtain

$$\begin{aligned} &f(x+\chi_{u_1}+\chi_{u_2})+f(x-\chi_{v_1}-\chi_{v_2}) \\ &\geq \min[f(x+\chi_{u_1}-\chi_{v_1})+f(x+\chi_{u_2}-\chi_{v_2}), \\ &\qquad f(x+\chi_{u_1}-\chi_{v_2})+f(x+\chi_{u_2}-\chi_{v_1}), \\ &\qquad f(x)+f(x+\chi_{u_1}+\chi_{u_2}-\chi_{v_1}-\chi_{v_2})] \\ &\geq 2f(x). \end{aligned} \tag{4}$$

Similarly, by (M-EXC) for $(y-\chi_{u_1}-\chi_{u_2}, y+\chi_{v_1}+\chi_{v_2})$, we obtain

$$\begin{aligned} &f(y-\chi_{u_1}-\chi_{u_2})+f(y+\chi_{v_1}+\chi_{v_2}) \\ &\geq \min[f(y-\chi_{u_1}+\chi_{v_1})+f(y-\chi_{u_2}+\chi_{v_2}), \\ &\qquad f(y-\chi_{u_1}+\chi_{v_2})+f(y-\chi_{u_2}+\chi_{v_1}), \\ &\qquad f(y)+f(y-\chi_{u_1}-\chi_{u_2}+\chi_{v_1}+\chi_{v_2})] \\ &\geq f(x)+f(y), \end{aligned} \tag{5}$$

since $f(y-\chi_{u_i}+\chi_{v_j}) \geq f(x)$ and $f(y-\chi_{u_1}-\chi_{u_2}+\chi_{v_1}+\chi_{v_2}) \geq f(x)$ by the choice of y. Adding (2), (3), (4) and (5) yields a contradiction. □

The ℓ_1-distance of four in Theorem 14 cannot be replaced by ℓ_1-distance of two, as we see in the following example.

Example 15 ([2]) Let $J \subseteq \mathbf{Z}^6$ be the degree system (see Example 7) of an undirected graph consisting of vertex-disjoint two triangles, and $f : J \to \mathbf{R}$ be an M-convex function representing the sum of squares of the components (see Example 6). Let $k = 8$ and $x = (2,2,2,1,1,0)$, for which $f(x) = 14$. For any point $y \in J_8$ with $||y-x||_1 = 2$ we have $f(y) = 14$, whereas for $x^* = (2,1,1,2,1,1)$ we have $f(x^*) = 12$ and $||x^*-x||_1 = 4$. ■

Remark 16 Theorem 14 above is a generalization of Theorem 1 of [2], since the degree system of a graph is a constant-parity jump system (Example 7) and a separable convex function on a constant-parity jump system is an M-convex function (Example 6). In fact, the result of [2] was the primary motivation behind Theorem 14. ■

The following theorem reveals a kind of monotonicity of the minimizers of f on J_k.

Theorem 17 *For any $x_k \in M_k$ with $k \in \Lambda$ there exists $(x_l \in M_l \mid l \in \Lambda \setminus \{k\})$ such that $x_{k_{\min}} \leq \cdots \leq x_{k-2} \leq x_k \leq x_{k+2} \leq \cdots \leq x_{k_{\max}}$.*

PROOF: We show the existence of such x_{k-2}. Then x_{k+2} can be shown to exist in a similar manner, and the other x_l (with $l \leq k-4$ or $l \geq k+4$) are by induction.

Take $y \in M_{k-2}$ with minimum $||y-x_k||_1$. If $y \leq x_k$, we are done with $x_{k-2} = y$. Otherwise, take $u \in \mathrm{supp}^-(y-x_k)$ and apply (M-EXC) to obtain either

$$\exists v \in \mathrm{supp}^-(y-x_k):\ f_{k-2}+f_k \geq f(y+\chi_u+\chi_v)+f(x_k-\chi_u-\chi_v) \quad \text{or}$$
$$\exists v \in \mathrm{supp}^+(y-x_k):\ f_{k-2}+f_k \geq f(y+\chi_u-\chi_v)+f(x_k-\chi_u+\chi_v).$$

In the first case the right-hand side is lower bounded by f_k+f_{k-2} and hence $y+\chi_u+\chi_v \in M_k$ and $x_k-\chi_u-\chi_v \in M_{k-2}$; then we can take $x_{k-2} = x_k-\chi_u-\chi_v$. The second case cannot occur, since the right-hand side is lower bounded by $f_{k-2}+f_k$, from which follows $y+\chi_u-\chi_v \in M_{k-2}$, whereas $||(y+\chi_u-\chi_v)-x_k||_1 = ||y-x_k||_1-2$; a contradiction to the choice of y. □

Theorem 18 *Minimum values f_k form a convex sequence:*

$$f_{k-2}+f_{k+2} \geq 2f_k \qquad (k \in \Lambda \setminus \{k_{\min}, k_{\max}\}). \tag{6}$$

PROOF: By Theorem 17 we can take $x_{k-2} \in M_{k-2}$ and $x_{k+2} \in M_{k+2}$ with $x_{k-2} \le x_{k+2}$, and also $u \in \mathrm{supp}^+(x_{k+2} - x_{k-2})$. By (M-EXC) there exists $v \in \mathrm{supp}^+(x_{k+2} - x_{k-2})$ such that $f_{k-2} + f_{k+2} \ge f(x_{k-2} + \chi_u + \chi_v) + f(x_{k+2} - \chi_u - \chi_v) \ge 2f_k$. □

Convexity of the minimum values motivates us to consider the subgradient. For $\alpha \in \mathbf{R}$ define $f^\alpha : J \to \mathbf{R}$ by

$$f^\alpha(x) \;=\; f(x) - \alpha x(V). \tag{7}$$

Then we have $\min_{x\in J} f^\alpha(x) = \min_{l\in\Lambda} \min_{x\in J_l} f^\alpha(x) = \min_{l\in\Lambda}(f_l - \alpha l)$. By Theorem 18, the minimum of $f_l - \alpha l$ over $l \in \Lambda$ is attained by $l = k$ if

$$(f_k - f_{k-2})/2 \le \alpha \le (f_{k+2} - f_k)/2. \tag{8}$$

Hence

$$f_k = k\alpha + \min\{ f^\alpha(x) \mid x \in J\} \tag{9}$$

for α in the range of (8). This shows that the optimal value f_k can be computed by solving an unconstrained minimization problem for another M-convex function f^α.

It is noted, however, that not every minimizer of f^α belongs to J_k. A point $x \in J$ minimizes $f^\alpha(x)$ if and only if $x \in M_k$ for some k with $k_-(\alpha) \le k \le k_+(\alpha)$, where

$$k_-(\alpha) \;=\; \min\{k \mid \min_l (f_l - \alpha l) = f_k - \alpha k\}, \tag{10}$$

$$k_+(\alpha) \;=\; \max\{k \mid \min_l (f_l - \alpha l) = f_k - \alpha k\}. \tag{11}$$

5 Algorithms

The local optimality criteria in Theorems 11 and 14 for unconstrained and constrained minimization, respectively, naturally suggest descent-type algorithms. At each feasible nonoptimal point, an improved point can be found with $O(|V|^2)$ function evaluations in unconstrained minimization and $O(|V|^4)$ function evaluations in constrained minimization. Although we do not enter into further technical details, the number of updates of the solution point may be bounded by the ℓ_1-distance from the initial point to the optimal point, or by the difference of the objective function values at the initial point and at the optimal point if the objective function is integer-valued.

Two other algorithms can be constructed for constrained minimization, to minimize $f(x)$ subject to $x \in J_k$, on the basis of Theorems 17 and 18. It is assumed that an algorithm is available for unconstrained minimization. For the convenience of descriptions it is also assumed that $k_{\min}$ and $k_{\max}$ are finite.

An increasing sequence of optimal solutions, the existence of which is guaranteed by Theorem 17, can be generated by the following algorithm. Once a global minimizer x^* is found, the algorithm computes the whole set of f_k ($k \in \Lambda$) with $\mathrm{O}((k_{\max} - k_{\min})|V|^2)$ evaluations of f. Note that the algorithm works even if $k_{\min}$ and/or $k_{\max}$ are not known in advance.

Algorithm I
Compute $x^* \in J$ that minimizes f;
Set $k^* := x^*(V)$, $x_{k^*} := x^*$, $f_{k^*} := f(x_{k^*})$;
for $k := k^*+2, k^*+4, \cdots, k_{\max}$ **do**
  Find $\{u,v\} \subseteq V$ that minimizes $f(x_{k-2} + \chi_u + \chi_v)$
  and put $x_k := x_{k-2} + \chi_u + \chi_v$ and $f_k := f(x_k)$;
for $k := k^*-2, k^*-4, \cdots, k_{\min}$ **do**
  Find $\{u,v\} \subseteq V$ that minimizes $f(x_{k+2} - \chi_u - \chi_v)$
  and put $x_k := x_{k+2} - \chi_u - \chi_v$ and $f_k := f(x_k)$.

Convexity of the sequence f_k makes it possible to convert the constrained minimization to an unconstrained minimization of f^α with an appropriate value of α; see (9). Here f^α is M-convex and, by our assumption, the minimum of $f^\alpha(x)$ over $x \in J$ can be computed efficiently. We assume that we can find $k_+(\alpha)$ and $k_-(\alpha)$ of (11) and (10) by maximizing (resp. minimizing) $x(V)$ among the minimizers of $f^\alpha(x)$ by means of some variant of an unconstrained minimization algorithm.

The following algorithm computes $k_{\min}$, $k_{\max}$ and f_k ($k \in \Lambda$) by searching for appropriate values of α. It requires $\mathrm{O}((k_{\max} - k_{\min})|V|^2)$ evaluations of f.

Algorithm II
Let α be sufficiently large;
Minimize f^α to find $k_{\min} = k_+(\alpha) = k_-(\alpha)$ and $f_{k_{\min}}$;
Let α be sufficiently small
  (α is a negative number with a large absolute value);
Minimize f^α to find $k_{\max} = k_+(\alpha) = k_-(\alpha)$ and $f_{k_{\max}}$;
if $k_{\max} - k_{\min} \ge 4$ **then** search$(k_{\min}, k_{\max})$.

Here the procedure "search(k_1,k_2)" is defined when $k_1+4\le k_2$ as follows.

procedure search(k_1,k_2)
  $\alpha := (f_{k_2}-f_{k_1})/(k_2-k_1)$;
  Minimize f^α to find $k_+=k_+(\alpha)$, $k_-=k_-(\alpha)$, $f_+=f_{k_+}$ and $f_-=f_{k_-}$;
  for $k := k_-+2, k_-+4, \cdots, k_+-2$ **do**
    $f_k := ((k-k_-)f_+ + (k_+-k)f_-)/(k_+-k_-)$;
  if $k_1+4\le k_-$ **then** search(k_1,k_-);
  if $k_++4\le k_2$ **then** search(k_+,k_2).

The second algorithm, as it stands, computes the values of f_k, and not the optimal solutions x_k. If x_k's are wanted, they can be computed easily in procedure "search" by generating a sequence of points $x_k\in J_k\cap\arg\min f^\alpha$ by applying (J-EXC) to the pair of the optimal solutions x_{k_-} and x_{k_+}.

Acknowledgments

The author thanks Jim Geelen and Satoru Iwata for discussion when we were at RIMS, Kyoto University, in April and May, 1996. He is also thankful to András Sebő for a stimulating comment that led to Proposition 9, and to Akihisa Tamura and Ken'ichiro Tanaka for checking the proofs.

References

[1] K. Ando, S. Fujishige, and T. Naitoh: A greedy algorithm for minimizing a separable convex function over a finite jump system, *Journal of Operations Research Society of Japan*, **38** (1995), 362–375.

[2] N. Apollonio and A. Sebő: Minsquare factors and maxfix covers of graphs, in: D. Bienstock and G. Nemhauser, eds., *Integer Programming and Combinatorial Optimization*, Lecture Notes in Computer Science, **3064**, Springer-Verlag, 2004, 388–400.

[3] A. Bouchet and W. H. Cunningham: Delta-matroids, jump systems, and bisubmodular polyhedra, *SIAM Journal on Discrete Mathematics*, **8** (1995), 17–32.

[4] A. W. M. Dress and W. Wenzel: Valuated matroid: A new look at the greedy algorithm, *Applied Mathematics Letters*, **3** (1990), 33–35.

[5] A. W. M. Dress and W. Wenzel: Valuated matroids, *Advances in Mathematics*, **93** (1992), 214–250.

[6] A. W. M. Dress and W. Wenzel: A greedy-algorithm characterization of valuated Δ-matroids, *Applied Mathematics Letters*, **4** (1991), 55–58.

[7] S. Fujishige: *Submodular Functions and Optimization*, Annals of Discrete Mathematics, **47**, North-Holland, Amsterdam, 1991.

[8] J. F. Geelen: Private communication, April 1996.

[9] S. N. Kabadi and R. Sridhar: Δ-matroid and jump system, *Journal of Applied Mathematics and Decision Sciences*, to appear.

[10] L. Lovász: The membership problem in jump systems, *Journal of Combinatorial Theory (B)*, **70** (1997), 45–66.

[11] K. Murota: Convexity and Steinitz's exchange property, *Advances in Mathematics*, **124** (1996), 272–311.

[12] K. Murota: *Matrices and Matroids for Systems Analysis*, Springer-Verlag, Berlin, 2000.

[13] K. Murota: *Discrete Convex Analysis*, Society for Industrial and Applied Mathematics, Philadelphia, 2003.

[14] K. Murota: M-convex functions on jump systems: A general framework for minsquare graph factor problem, Technical Report, METR 2004-43, Department of Mathematical Informatics, University of Tokyo, November 2004.

[15] W. Wenzel: Pfaffian forms and Δ-matroids, *Discrete Mathematics*, **115** (1993), 253–266.

A Deterministic Algorithm for Finding All Minimum k-Way Cuts

YOKO KAMIDOI

Faculty of Information Sciences,
Hiroshima City University
3-4-1, Ozuka-Higashi, Asaminami-ku, Hiroshima
731-3194, Japan
yoko@ce.hiroshima-cu.ac.jp

NORIYOSHI YOSHIDA

Faculty of Information Sciences,
Hiroshima City University
3-4-1, Ozuka-Higashi, Asaminami-ku, Hiroshima
731-3194, Japan

HIROSHI NAGAMOCHI*

Dept. of Applied Mathematics and Physics,
Kyoto University
Sakyo, Kyoto 606-8501, Japan
nag@amp.i.kyoto-u.ac.jp

Abstract: Let $G=(V,E)$ be an edge-weighted undirected graph with n vertices and m edges. We present a deterministic algorithm to compute a minimum k-way cut of G for a given k. Our algorithm is a divide-and-conquer method based on a procedure that reduces an instance of the minimum k-way cut problem to $O(n^{2k-5})$ instances of the minimum $(\lfloor (k+\sqrt{k})/2 \rfloor + 1)$-way cut problem, and can be implemented to run in $O(n^{4k/(1-1.71/\sqrt{k})-31})$ time. With a slight modification, the algorithm can find all minimum k-way cuts in $O(n^{4k/(1-1.71/\sqrt{k})-16})$ time.

Keywords: minimum cut, multiway cut, divide-and-conquer

1 Introduction

For an edge-weighted graph $G=(V,E)$, a subset F of edges is called a *k-way cut* if removal of F from G results in at least k connected components. The *minimum k-way cut problem* asks to find a minimum weight k-way cut in G. For given k vertices (called *terminals*), a k-way cut F is called a *k-terminal cut* if no two terminal are connected after removal of F, the problem of finding a minimum weight k-terminal cut is called the *minimum k-terminal cut problem*. These problem have several important applications such as VLSI design [1, 7, 26], task allocation in distributed computing systems [25, 33], graph strength [5, 10, 31] and network reliability [4, 34].

For $k=2$, the minimum 2-terminal cut problem in a graph can be solved by applying a maximum flow algorithm. Let $F(n,m)$ denote the time complexity of a maximum flow algorithm in an edge-weighted graph with n vertices and m edges, where $F(n,m)$ is $O(n^3)$ [24] and $O(nm\log(n^2/m))$ [8]. Dahlhaus et al. [6] proved that the minimum k-terminal cut problem is NP-hard for any fixed $k \geq 3$. Several approximation algorithms have been proposed [2, 6, 21], among which a 1.3438-approximation algorithm is obtained by Karger et al. [21]. An extension of this problem to a general setting defined by submodular set functions can be found in the articles by Zhao et al. [38, 39]. For planar graphs, the minimum k-terminal cut problem admits a polynomial time algorithm [6], and currently $O((k-\frac{3}{2})^{k-1}(n-k)^{2k-4}[nk-\frac{3}{2}k^2+\frac{1}{2}k]\log(n-k))$ time algorithm is known [36].

On the other hand, Goldschmidt and Hochbaum [9] proved that the minimum k-way cut problem is NP-hard if k is an input parameter, but admits a polynomial time algorithm if k is regarded as a constant. The minimum 2-way cut problem (i.e., the problem of computing edge-connectivity) can be solved by $O(nm+n^2\log n)$ and $O(nm\log(n^2/m))$ time deterministic algorithms [13, 28] and by $O(n^2(\log n)^3)$ and $O(m(\log n)^3)$ time randomized algorithms [20, 22, 23]. Approximation algorithms for a minimum k-way cut problem of G have been proposed in [16, 32, 37]. Saran and Vazirani [32] first proposed a $2(1-1/k)$-approximation algorithm for the minimum k-way cut problem, which runs in $O(nF(n,m))$ time. Kapoor [16] also gave a $2(1-1/k)$-approximation algorithm for the minimum k-way cut problem of G, which requires $O(k(nm+n^2\log n))$ time. Zhao *et al.* [37] also presented an approximation algorithm by using a set of minimum 3-way cuts. Their algorithm has the performance ratio $2-3/k$ for an odd k and $2-(3k-4)/(k^2-k)$ for an even k, and the runs in $O(kmn^3\log(n^2/m))$ time. Approximation algorithms for a multiway cut problem defined by submodular set functions are discussed in the articles by Zhao et al. [38, 39].

*Research is supported by the Scientifi c Grant-in-Aid from Ministry of Education, Science, Sports and Culture of Japan.

Goldschmidt and Hochbaum [9] presented an $O(n^{k^2/2-3k/2+4}F(n,m))$ time algorithm for solving the minimum k-way cut problem. This running time is polynomial for any fixed k. The algorithm is based on a divide-and-conquer approach. Suppose that we can choose a family $\mathscr{X}$ of subsets of V such that at least one subset $X \in \mathscr{X}$ has a property that a minimum $(k-1)$-way cut $\{V_1,\ldots,V_{k-1}\}$ in the subgraph $G[V-X]$ induced by $V-X$ gives rise to a minimum k-way cut $\{X,V_1,\ldots,V_{k-1}\}$ in the original graph G. Then we can find a minimum k-way cut in G by solving the $(k-1)$-way cut problem instances $G[V-X]$ for all $X \in \mathscr{X}$. Goldschmidt and Hochbaum [9] proved that such a family $\mathscr{X}$ of $O(n^{2k-3})$ subsets of V can be found in polynomial time, implying an $O(n^{O(k^2)})$ time algorithm for the minimum k-way cut problem.

For small $k \leq 6$ or planar graphs, faster algorithms have been obtained [16, 17, 18, 29, 30]. For $k \leq 6$, the above family $\mathscr{X}$ can be constructed in polynomial time by collecting $O(n)$ subsets of V, and an $O(mn^k \log(n^2/m))$ time algorithm is known [29, 30]. For planar graphs, Hartvigsen [14] gave an $O(n^{2k-1})$ time algorithm, and Nagamochi et al. [29, 30] showed that the problem can be solved in $O(n^k)$ time if $k \leq 6$. The case of unweighted planar graphs with $k = 3$ can be solved in $O(n \log n)$ time [15].

Randomized algorithms have been developed for the k-way cut problem. Karger and Stein [23] proposed a Monte Carlo algorithm for the minimum k-way cut problem which runs in $O(n^{2(k-1)}\log^3 n)$ time. Afterwards, Levine [27] gave a Monte Carlo algorithm for $k \leq 6$ that runs in $O(mn^{k-2}\log^3 n)$ time. However, for a general k and a general graph G, no faster deterministic algorithm has been discovered since Goldschmidt and Hochbaum [9] found an $O(n^{O(k^2)})$ time algorithm.

In this paper, we present the first $O(n^{O(k)})$ time deterministic algorithm to compute a minimum k-way cut of G. Our algorithm is based on a divide-and-conquer method which consists of a procedure that reduces an instance of the minimum k-way cut problem to $O(n^{2k-5})$ instances of the minimum $(\lfloor (k+\sqrt{k})/2 \rfloor + 1)$-way cut problem, and can be implemented to run in $O(n^{4k/(1-1.71/\sqrt{k})-31})$ time. With a slight modification, we can also find all minimum k-way cuts in $O(n^{4k/(1-1.71/\sqrt{k})-16})$ time.

The paper is organized as follows. Section 2 introduces notations and reviews basic properties of 2-way cuts. Section 3 presents our divide-and-conquer algorithm, assuming an efficient procedure for computing a family $\mathscr{X}$ of subsets required to reduce a given problem instance, which is discussed in section 5, after proving a key property on crossing 2-way cuts in section 4. Section 6 analyzes the runtime of our algorithm. Section 7 shows how to modify the algorithm so that all minimum k-way cuts can be computed, and section 8 makes some concluding remarks.

2 Preliminaries

Let $G = (V,E)$ stand for an edge-weighted undirected graph consisting of a vertex set V and an edge set E with an edge weight function $cost : E \to R^+$, where R^+ is the set of nonnegative real numbers. Let $n = |V|$ be the number of vertices and $m = |E|$ be the number of edges. We may simply call G a graph. Let $comp(G)$ denote the number of connected components in G. An edge $e \in E$ with end vertices u and v may be denoted by $e = (u,v)$, and its weight is denoted by $cost(e)$. For a nonempty subset $F \subseteq E$, we denote by $cost(F)$ $\sum_{e\in F} cost(e)$. Let $X_1, X_2, \ldots, X_p$ be mutually disjoint subsets of V. We denote the set of edges $e = (u,v)$ with $u \in X_i$ and $v \in X_j$ for some $i \neq j$ by $(X_1;X_2;\ldots;X_p)$, and the sum of the weights of these edges by $cost(X_1;X_2;\ldots;X_p)$, which is defined to be 0 if $(X_1;X_2;\ldots;X_p) = \emptyset$. For a subset X of V, we may denote $f(X) = cost(X;V-X)$, where f is called a *cut function* of G and satisfies the following identities:

$$f(X) + f(Y) = f(X \cap Y) + f(X \cup Y) + 2cost(X-Y, Y-X) \text{ for all } X,Y \subseteq V, \tag{1}$$

$$f(X) + f(Y) = f(X-Y) + f(Y-X) + 2cost(X \cap Y, V-(X \cup Y)) \text{ for all } X,Y \subseteq V. \tag{2}$$

Let F be a subset of E in G. We denote by $G-F$ the graph obtained from G by deleting edges in F. We call F a *k-way cut* if $comp(G-F) \geq k$. A k-way cut F is *minimum* if it has the minimum $cost(F)$ over all k-way cuts. Given a graph G and an integer $k(\geq 2)$, the *minimum k-way cut problem* asks to find a minimum k-way cut in G. We denote the cost of a minimum k-way cut in G by $opt(G,k)$. Note that $opt(G,k) = 0$ if and only if the set of edges with positive weights induces from G at least k connected components. Any inclusionwise minimal k-way cut F is given by $F = (X_1;X_2;\ldots;X_k)$ for some partition $\{X_1,X_2,\ldots,X_k\}$ of V. Conversely, for any partition $\{V_1,V_2,\ldots,V_k\}$ of V, $F' = (V_1;V_2;\ldots;V_k)$ is a k-way cut, where possibly $comp(G-F') > k$. For a set $\mathscr{C}$ of subsets F of E, we denote the union $\cup_{F\in\mathscr{C}} F$ by $E(\mathscr{C})$.

Given a nonempty vertex subset X, let $G[X] = (X,E_X)$ be the subgraph of G induced by X, where $G[X]$ has the edge weight function $cost_X : E_X \to R^+$ is naturally defined such that $cost_X(e) = cost(e)$ for every edge $e \in E_X$. For a subset Y of vertices of V, we denote $V-Y$ by $\overline{Y}$ if V is clear from the context.

Given p mutually disjoint nonempty subsets $T_1, T_2, \ldots, T_p$ of V, called *terminal sets*, a subset set $F \subseteq E$ is called a *$(T_1,T_2,\ldots,T_p)$-terminal cut* of G if the removal of F from G disconnects each terminal set from the others. A $(T_1,T_2,\ldots,T_p)$-terminal cut is called minimum if it has the minimum $cost(F)$ among all $(T_1,T_2,\ldots,T_p)$-terminal cuts.

3 Divide-and-Conquer Algorithm

Any of previously known deterministic algorithms reduces a minimum k-way cut problem instance into a set of minimum $(k-1)$-way cut problem instances, where the target k on the number of components is reduced only by 1. In this paper, we reduce a minimum k-way cut problem instance into a set of minimum k'-way cut problem instances with k' nearly equal to $k/2$. For this, we first observe the following property.

Lemma 1 *Let $(V_1;V_2;\ldots;V_k)$ be a minimum k-way cut in a graph $G=(V,E)$, where $k\in[2,n]$. Then for any integer $p\in[1,k]$, there is a union X of p subsets in $\{V_1,V_2,\ldots,V_k\}$ such that*

$$f(X)\leq\frac{2(kp-p^2)}{(k^2-k)}opt(G,k).$$

PROOF: Let $\mathscr{X}$ be the family of all such unions X. Then $|\mathscr{X}|=\binom{k}{p}$. For each edge $e=(u,v)\in E$, there are $\binom{k-2}{p-1}$ unions $X\in\mathscr{X}$ such that $u\in X$ and $v\in\overline{X}$. Therefore it holds $\sum_{X\in\mathscr{X}}f(X)=2\binom{k-2}{p-1}opt(G,k)$, and the average of $f(X)$ over all $X\in\mathscr{X}$ is $2\binom{k-2}{p-1}opt(G,k)/\binom{k}{p}=2(kp-p^2)/(k^2-k)opt(G,k)$. This implies the lemma. □

Let $p=\lceil(k-\sqrt{k})/2\rceil-1$, which satisfies $2(kp-p^2)/(k^2-k)<1/2$ for $k\geq 5$. Hence the set of all subsets Y of V such that

$$f(Y)<opt(G,k)/2$$

contains a subset X which is a union of p subsets in $\{V_1,V_2,\ldots,V_k\}$ for a minimum k-way cut $(V_1;V_2;\ldots;V_k)$ in G. For such a subset X, we can reduce the current instance (G,k) into two instances $(G[X],p)$ and $G([V-X],k-p)$, where a minimum k-way cut F for (G,k) is obtained from a minimum p-way cut F' for $(G[X],p)$ and a minimum $(k-p)$-way cut F'' for $(G[V-X],k-p)$ by constructing a k-way cut $F=F'\cup F''$. Note that the size k is reduced to at most $k-\lceil(k-\sqrt{k})/2\rceil+1=\lfloor(k+\sqrt{k})/2\rfloor+1$, which is a nearly half of k for a large k.

Section 5 shows that the number of such subsets $X\in V$ with $f(X)<opt(G,k)/2$ is at most n^{2k-5} and a family $\mathscr{X}$ of n^{2k-5} subsets including these subsets X (possibly together with some other subsets) can be obtained in $O(n^{2k-5}F(n,m))$ time. With this property, our divide-and-conquer algorithm can be described as follows.

Algorithm MULTIWAY(G,k)
Input: A graph $G=(V,E)$ and an integer $k\in[1,|V|]$.
Output: A minimum k-way cut F in G.
1 **if** $opt(G,k)=0$ **then** Return $F:=\emptyset$
2 **else** /* $opt(G,k)>0$ */
3 **if** $k\leq 6$ **then** Return a minimum k-way cut F of G by $O(|V|^{k-1})$
maximum flow computations
4 **else** /* $k\geq 7$ */
5 Compute a set $\mathscr{X}$ of at most $|V|^{2k-5}$ subsets of V such that any 2-way cut with cost
less than $opt(G,k)/2$ is given by $(X;V-X)$ for some $X\in\mathscr{X}$;
6 $p:=\lceil(k-\sqrt{k})/2\rceil-1$;
7 **for** each $X\in\mathscr{X}$ with $|X|\geq p$ and $|V-X|\geq k-p$ **do**
8 $F_X:=(X;V-X)\cup$MULTIWAY$(G[X],p)\cup$MULTIWAY$(G[V-X],k-p)$;
9 **end** /* for */
10 Choose a k-way cut F_X with the minimum cost over all X, and return $F:=F_X$
11 **end** /* if */
12 **end**. /* if */

For the correctness of algorithm MULTIWAY, we only have to give a procedure in line 5, which will be discussed in section 5. The runtime of MULTIWAY will be analyzed in section 6.

4 A Crossing Property

This section provides a property on crossing 2-way cuts, based on which a procedure for collecting all subsets $X\in V$ with $f(X)<opt(G,k)/2$ is designed in section 5.

Lemma 2 *For a graph $G=(V,E)$, let $\{Y_1,Y_2,\ldots,Y_q,W,Z\}$ be a partition of V, and let X be a subset of V such that each subset in $\{Y_1-X,Y_2-X,\ldots,Y_q-X,W\cap X,X\cap Z,Z-X\}$ is nonempty. Then partition $\{Y'_i=Y_i-X\ (i=1,2,\ldots,q),\ Y'_{q+1}=X\cap Z,\ W'=(W\cup X)-Z,\ Z'=Z-X\}$ of V satisfies*

$$\begin{aligned}&2cost(Y_1;Y_2;\ldots;Y_q;W;Z)-f(Y_{1,q})+f(X)\\ &\geq 2cost(Y'_1;Y'_2;\ldots;Y'_q;Y'_{q+1};W';Z')-f(Y'_{1,q+1})+f(W\cap X),\end{aligned}$$

where we denote $Y_{1,i}=Y_1\cup Y_2\cup\cdots\cup Y_i$ and $Y'_{1,j}=Y'_1\cup Y'_2\cup\cdots\cup Y'_j$.

PROOF: We obtain

$$\begin{aligned}&f(Y_1)+f(Y_2)+\cdots+f(Y_q)+f(Y'_{1,q+1})\\ &\geq\ f(Y'_1)+f(Y'_2)+\cdots+f(Y'_q)+f(Y_{1,q}\cup(X\cap Z))+2cost(Y_{1,q}\cap X;X\cap Z),\end{aligned}\tag{3}$$

by summing up the following q inequalities implied by (1):

$$\begin{aligned}f(Y_1)+f((Y_{1,q}-X)\cup(X\cap Z))&\geq f(Y_1-X)+f(Y_1\cup(Y_{1,q}-X)\cup(X\cap Z))+2cost(Y_1\cap X;X\cap Z)\\ f(Y_2)+f(Y_1\cup(Y_{1,q}-X)\cup(X\cap Z))&\geq f(Y_2-X)+f(Y_{1,2}\cup(Y_{1,q}-X)\cup(X\cap Z))+2cost(Y_2\cap X;X\cap Z)\\ &\cdots\\ f(Y_q)+f(Y_{1,q-1}\cup(Y_{1,q}-X)\cup(X\cap Z))&\geq f(Y_q-X)+f(Y_{1,q}\cup(Y_{1,q}-X)\cup(X\cap Z))+2cost(Y_q\cap X;X\cap Z).\end{aligned}$$

On the other hand, (1) and (2) mean

$$\begin{aligned}&f(Z)+f(W)+f(X)\geq f(Z)+f(W\cap X)+f(W\cup X)\\ &\geq\ f((W\cup X)-Z)+f(Z-X)+2cost(Y_{1,q}-X;X\cap Z)+f(W\cap X).\end{aligned}\tag{4}$$

From (4) and (3), we have

$$\begin{aligned}&f(Y_1)+f(Y_2)+\cdots+f(Y_q)+f(W)+f(Z)+f(Y'_{1,q+1})+f(X)\\ &\geq\ f(Y'_1)+f(Y'_2)+\cdots+f(Y'_q)+f((W\cup X)-Z)+f(Z-X)\\ &\quad+f(Y_{1,q}\cup(X\cap Z))+2cost(Y_{1,q}\cap X;X\cap Z)+2cost(Y_{1,p}-X;X\cap Z)+f(W\cap X)\\ &=\ f(Y'_1)+f(Y'_2)+\cdots+f(Y'_q)+f(W')+f(Z')+f(X\cap Z)+f(Y_{1,q})+f(W\cap X),\end{aligned}$$

implying the lemma. □

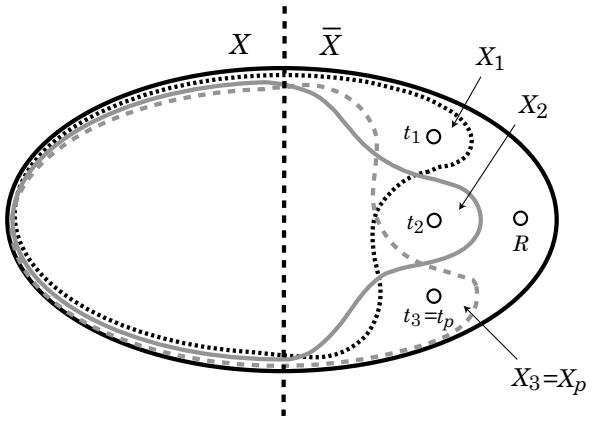


Figure 1: Illustration for a 2-way cut $(X;\overline{X})$ and a minimum $(X;T\cup R-\{t_i\})$-terminal cut $(X_i;\overline{X_i})$, $i=1,2,3(=p)$.

Lemma 3 *For a graph $G=(V,E)$ and an integer $k\in[5,n-1]$, let $(X;\overline{X})$ be a 2-way cut of G, R be a nonempty subset of $\overline{X}$, and $T=\{t_1,t_2,\ldots,t_p\}$ be a set of $p\geq 2$ vertices in $\overline{X}-R$. Assume that, for each t_i, there exists a minimum $(X,T\cup R-\{t_i\})$-terminal cut $(X_i;\overline{X_i})$ which satisfies $X\cup\{t_i\}\subseteq X_i$ (see Fig. 1). Let $\mathscr{C}=\{(X_i;\overline{X_i})\mid 1\leq i\leq p\}$. Then $E(\mathscr{C})$ is a $(p+2)$-way cut which partitions V into $p+2$ subsets*

$$Z=\cap_{1\leq i\leq p}\overline{X_i},\quad W=\cup_{1\leq i<j\leq p}(X_i\cap X_j),\ \text{and}\ \ Y_i=X_i-W\ (i=1,2,\ldots,p).$$

Furthermore $(p+2)$-way cut $E(\mathscr{C})=(Y_1;Y_2;\ldots;Y_p;Z;W)$ satisfies

$$cost(E(\mathscr{C}))+cost(Z;W)+cost(Y_1;Y_2;\ldots;Y_p)\leq f(X_1)+f(X_2).\tag{5}$$

PROOF: Since $X \subseteq W$, $t_i \in Y_i$ $(1 \leq i \leq p)$ and $R \subseteq Z$ hold, we see that $E(\mathscr{C})$ is a $(p+2)$-way cut. We prove (5) by an induction on p.

(Basis case) For $p = 2$, $Z = V - (X_1 \cup X_2)$, $W = X_1 \cap X_2$, $Y_1 = X_1 - X_2$, and $Y_2 = X_2 - X_1$. Then it holds $cost(Y_1; Y_2; Z; W) + cost(Z; W) + cost(Y_1; Y_2) = cost(X_1 - X_2; X_2 - X_1; V - (X_1 \cup X_2); X_1 \cap X_2) + cost(V - (X_1 \cup X_2); X_1 \cap X_2) + cost(X_1 - X_2; X_2 - X_1) = f(X_1) + f(X_2)$, as required.

(Inductive case) Let $p \geq 3$, and assume that (5) holds for $p-1$. Let R, $T = \{t_1, t_2, \ldots, t_p\}$, $\mathscr{C}$ be subsets of $\overline{X}$ and a set of p 2-way cuts satisfying the condition of the lemma for p. We here consider $R' = R \cup \{t_p\}$, $T' = \{t_1, t_2, \ldots, t_{p-1}\}$ and $\mathscr{C}' = \mathscr{C} - \{(X_p; \overline{X_p})\}$, which satisfy the condition of the lemma for $p-1$ (see Fig. 2). Hence, by the induction hypothesis, we have

$$\begin{aligned} & f(X_1) + f(X_2) \\ & \geq cost(E(\mathscr{C}')) + cost(Z'; W') + cost(Y'_1; Y'_2; \ldots; Y'_{p-1}) = 2cost(E(\mathscr{C}')) - f(Y'_{1,p-1}), \end{aligned} \tag{6}$$

where $Z' = \cap_{1 \leq i \leq p-1} \overline{X_i}$, $W' = \cup_{1 \leq i < j \leq p-1} (X_i \cap X_j)$ and $Y'_i = X_i - W'$ $(i = 1, 2, \ldots, p-1)$. By Lemma 2, we obtain

$$(6) \geq 2cost(E(\mathscr{C})) - f(Y_{1,p}) + f(W' \cap X_p) - f(X_p) \geq 2cost(E(\mathscr{C})) - f(Y_{1,p}),$$

where $f(W' \cap X_p) \geq f(X_p)$ holds since $(W' \cap X_p; \overline{W' \cap X_p})$ is an $(X, T \cup R - \{t_p\})$-terminal cut in G. This implies that (5) holds for p. □

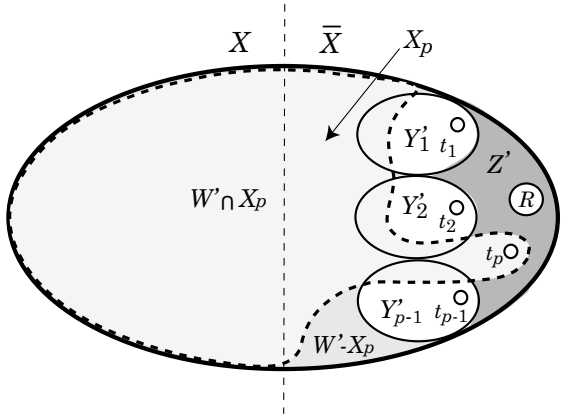


Figure 2: Illustration for a minimum $(X, T \cup R - \{t_p\})$-terminal cut $(X_p; \overline{X_p})$ and subsets $Y'_1, Y'_2, \ldots, Y'_{p-1}$.

5 Computing Small Cuts

With Lemma 3, we are ready to present an $O(n^{2k-5}F(n,m))$ time procedure for collecting all subsets $X \in V$ with $f(X) < opt(G,k)/2$.

Theorem 4 *For a graph $G = (V,E)$ and an integer $k \in [5, n-1]$, let $(X; \overline{X})$ be a 2-way cut with $f(X) < opt(G,k)/2$. Then, for any vertices $s^* \in X$ and $t^* \in \overline{X}$, there are subsets $S \subseteq X$ and $T \subseteq \overline{X}$ with $|S| \leq k-3$ and $|T| \leq k-3$ such that $(X; \overline{X})$ is a unique minimum $(S \cup \{s^*\}, T \cup \{t^*\})$-terminal cut in G.*

PROOF: Let $(X; \overline{X})$ be a 2-way cut with $f(X) \leq opt(G,k)/2$. We first prove the next claim.

Claim 5 *A set T of at most $k-3$ vertices in $\overline{X}$ can be chosen so that $(X; \overline{X})$ becomes a unique minimum $(X, T \cup \{t^*\})$-terminal cut.*

PROOF: Let $\mathscr{Y}$ be the family of all subsets Y with $X \subset Y \subseteq V - \{t^*\}$ such that

$$f(Y) \leq f(X).$$

We choose a subset T of $\overline{X} - \{t^*\}$ so that T becomes a minimal transversal of $\mathscr{Y}$ (i.e., T is an inclusion-wise minimal subset of $\overline{X} - \{t^*\}$ such that $Y \cap T \neq \emptyset$ for all $Y \in \mathscr{Y}$). Since T is a transversal of $\mathscr{Y}$, no other $(X, T \cup \{t^*\})$-terminal cut than $(X; \overline{X})$ has cost less than or equal to $f(X)$. Hence, to prove the claim, it suffices to show that $|T| \leq k-3$.

For each $t \in T$, let $(X_t;\overline{X}_t)$ denote a minimum $(X,(T-\{t\})\cup\{t^*\})$-terminal cut. We show that

$$f(X_t) \le f(X), \quad t \in X_t.$$

By the minimality of T, each vertex $t \in T$ has a subset $Y' \in \mathscr{Y}$ such that $t \in Y'$ and $Y' \cap (T-\{t\})\cup\{t^*\}) = \emptyset$. Hence $f(X_t) \le f(Y') \le f(X)$. This also implies that $X_t \in \mathscr{Y}$ and hence t must belong to X_t (since otherwise $X_t \cap T = \emptyset$ would hold). See Fig. 1, where $R = \{t^*\}$.

The above sets $R = \{t^*\}$, T, $\mathscr{C} = \{(X_t;\overline{X}_t) \mid t \in T\}$ satisfy the condition of Lemma 3. By Lemma 3 and assumption on $f(X)$, $E(\mathscr{C})$ is a $(|T|+2)$-way cut with $cost(E(\mathscr{C})) \le 2\max\{f(X_t) \mid t \in T\} \le 2f(X) < opt(G,k)$. Therefore $|T| \le k-3$ holds, since otherwise $comp(G-E(\mathscr{C})) \ge |T|+2 \ge k$ would hold, contradicting the definition of $opt(G,k)$. This proves the claim. □

By applying the above claim to X, we see that a set S of at most $k-3$ vertices in X can be chosen so that $(X;\overline{X})$ becomes a unique minimum $(S\cup\{s^*\},\overline{X})$-terminal cut.

Finally we show that $(X;\overline{X})$ is a unique minimum $(S\cup\{s^*\},T\cup\{t^*\})$-terminal cut in G. Assume indirectly that G has another minimum $(S\cup\{s^*\},T\cup\{t^*\})$-terminal cut $(Z;\overline{Z})$. By the property of S and T, neither $Z \subseteq X$ nor $Z \supseteq X$; The remaining case is $X-Z \ne \emptyset \ne Z-X$. In this case, by the submodularity of cost function,

$$f(X)+f(Z) \ge f(X\cap Z)+f(Z\cup X)$$

holds, and we see that at least one of $(X\cap Z;\overline{X\cap Z})$ and $(Z\cup X;\overline{Z\cup X})$ is a minimum $(S\cup\{s^*\},T\cup\{t^*\})$-terminal cut. This, however, contradicts the above property of S and T. This completes the proof of the theorem. □

Based on this theorem, we can find all 2-way cuts $(X;\overline{X})$ with $f(X) < opt(G,k)/2$ by $O(n^{2k-5})$ maximum flow computations. For this, choose a vertex $s^* \in V$, and execute the following procedure for each vertex $t^* \in V-\{s^*\}$: Choose disjoint sets $S,T \subseteq V-\{s^*,t^*\}$ with $2 \le |S| \le k-3$ and $2 \le |T| \le k-3$, and compute a minimum $(S\cup\{s^*\},T\cup\{t^*\})$-terminal cut $(X;\overline{X})$ in G. Then the set of these 2-way cuts $(X;\overline{X})$ for all $t^* \in V-\{s^*\}$ include those with cost less than $opt(G,k)/2$. For fixed s^* and t^*, there are at most n^{2k-6} such pairs of S and T. Hence, we need $O(n^{2k-6}\cdot n) = O(n^{2k-5})$ maximum flow computations. We also have the following corollary.

Corollary 6 *Let $G=(V,E)$ be a graph, $k \in [5,n]$ be an integer and $opt(G,k) > 0$. Then the number of subsets $X \subset V$ such that $f(X) < (1/2)opt(G,k)$ is $O(n^{2k-5})$.*

6 Runtime of MULTIWAY

In this section, we analyze the runtime of algorithm MULTIWAY.

Theorem 7 *For a graph $G=(V,E)$ with n vertices m edges and an integer $k \in [1,n]$, MULTIWAY(G,k) runs in $O(n^{k-1}F(n,m))$ time for $k \le 6$ and in $O(n^{4k/(1-1.71/\sqrt{k})-34}F(n,m))$ time for $k \ge 7$, where $F(n,m)$ denotes the time complexity for computing a maximum flow in a graph with n vertices and m edges.*

PROOF: Let $N(k,n)$ denote an upper bound on the number of maximum flow computations to execute MULTIWAY(G,k) for a graph G with n vertices, where we assume that $N(k,n)$ is a nondecreasing function with respect to k and n. For $k \le 6$, it is known that a minimum k-way cut can be obtained by at most n (resp., $2n^2$, $4n^3$, $60n^4$ and $900n^5$) maximum flow computations for $k=2$ (resp., $k=3,4,5,6$ [29, 30]). Then we set $N(2,n)=n$, $N(3,n)=2n^2$, $N(4,n)=4n^3$, $N(5,n)=60n^4$ and $N(6,n)=900n^5$. For $k \ge 5$, $N(k,n)$ satisfies

$$\begin{aligned} N(k,n) &= n^{2k-5} + n^{2k-5}N(k-\lceil(k-\sqrt{k})/2\rceil+1, n-\lceil(k-\sqrt{k})/2\rceil+1) \\ &\le n^{2k-5}N(\lfloor(k+\sqrt{k})/2\rfloor+1, n). \end{aligned}$$

Then we define $M(k)$ by a recursive formula $M(k) = 2k-5+M(\lfloor(k+\sqrt{k})/2\rfloor+1)$ for $k \ge 7$ and $M(2)=1$, $M(3)=2$, $M(4)=3$ and $M(5)=4$. We see that $900n^{M(k)}$ gives an upper bound on the number of maximum flow computations needed to execute MULTIWAY(G,k). We see that $M(k) \le 4k/(1-1.71/\sqrt{k})-34$ holds for $k \le 300000$ by generating all those $M(k)$ with a computer program. We prove that $M(k) \le 4k/(1-1.71/\sqrt{k})-34$ holds for $k > 300000$ with the recursive formula. Let $a=1.71$, and $k' = \lfloor(k+\sqrt{k})/2\rfloor+1 \le (k+\sqrt{k}+2)/2$. Then by the induction hypothesis we have

$$M(k) = 2k-5+M(k') \le 2k-5+4k'\sqrt{k'}/(\sqrt{k'}-a)-34.$$

Then it suffices to show that

$$4k\sqrt{k}/(\sqrt{k}-a)-34-(2k-5)-2(k+\sqrt{k}+2)\sqrt{k+\sqrt{k}+2}/(\sqrt{k+\sqrt{k}+2}-\sqrt{2}a)+34$$

is nonnegative for $k > 300000$. For this, we prove the next is nonnegative.

$$\begin{aligned}
&4k\sqrt{k}(\sqrt{k+\sqrt{k}+2}-\sqrt{2}a)-(\sqrt{k}-a)(2k-5)(\sqrt{k+\sqrt{k}+2}-\sqrt{2}a)\\
&\quad-(\sqrt{k}-a)2(k+\sqrt{k}+2)\sqrt{k+\sqrt{k}+2}\\
&=\left((4a-2)k+(2a+1)\sqrt{k}-a)\right)\sqrt{k+\sqrt{k}+2}+\sqrt{2}a(-2k\sqrt{k}-2ak-5\sqrt{k}+5a)
\end{aligned}$$

Since $(4a-2)k+(2a+1)\sqrt{k}-a>0$ for $k>300000$, (7) is at least

$$\begin{aligned}
&((4a-2)k+(2a+1)\sqrt{k}-a)\sqrt{k}+\sqrt{2}a(-2k\sqrt{k}-2ak-5\sqrt{k})\\
&=\sqrt{k}\Big[((4-2\sqrt{2})a-2)k+(2a(1-\sqrt{2}a)+1)\sqrt{k}-(1+5\sqrt{2})a)\Big],
\end{aligned}$$

which is nonnegative, since $4a-2\sqrt{2}a-2>0$ holds for $k=300000$ and in this case it holds

$$((4-2\sqrt{2})1.71-2)300000+(2\cdot 1.71(1-\sqrt{2}\cdot 1.71)+1)\sqrt{300000}-(1+5\sqrt{2})1.71)>0.$$

This proves that $M(k)\le 4k/(1-1.71/\sqrt{k})-34$ holds for $k>300000$. □

7 Enumerating All Minimum k-Way Cuts

In this section, we show that algorithm MULTIWAY can be modified so that all minimum k-way cuts in G can be enumerated in the nearly same time complexity. Given a graph $G=(V,E)$ and integer k, we can construct all minimum k-way cuts F by combining a minimum p-way cut F' in $G[X]$ and a minimum $(k-p)$-way cut F'' in $G[V-X]$ for all possible subsets $X\subset V$ such that X is a union of p subsets in $\{V_1,\ldots,V_k\}$ for a minimum k-way cut in G.

However, we cannot apply Corollary 6 to instances (G,k) with $k\le 4$. From Lemma 1 with $p=1$, we see that for any minimum k-way cut $(V_1;V_2;\ldots;V_k)$ in a graph $G=(V,E)$, there is a subset $X\in\{V_1,V_2,\ldots,V_k\}$ such that

$$f(X)\le(2/k)opt(G,k).$$

For $k\in\{2,3,4\}$, we derive an upper bound on the number of subsets $X\subset V$ with $f(X)\le(2/k)opt(G,k)$.

Lemma 8 *Let $G=(V,E)$ be a graph, $k\in[2,4]$ be an integer and $opt(G,k)>0$. Then the number of subsets $X\subset V$ such that $f(X)\le(2/k)opt(G,k)$ for $k=2$ (resp., $k=3,4$) is $O(n^2)$ (resp., $O(n^4)$ and $O(n^4)$).*

PROOF: Let $\mathscr{X}_k$ be the family of subsets X with $f(X)\le(2/k)opt(G,k)$. Let $\lambda(G)$ denote the edge-connectivity of G (i.e., $\lambda(G)=opt(G,2)$). It is [19] known that, for $\lambda(G)>0$, there are $O(n^{2\alpha})$ subsets X with $f(X)\le\alpha\lambda(G)$.

(i) Let $k=2$. Then $\lambda(G)>0$ holds by assumption $opt(G,2)>0$. This proves that $|\mathscr{X}_2|=O(n^{2\alpha})=O(n^2)$ holds for $\alpha=1$.

For $k\in\{3,4\}$, we assume that $\mathscr{X}_k$ contains two subsets X_1 and X_2 such that each of $Y_1=X_1\cap X_2$, $Y_2=X_1-X_2$, $Y_3=V-(X_1\cup X_2)$ and $Y_4=X_2-X_1$ is nonempty (otherwise $|\mathscr{X}_k|\le 2n$ holds).

(ii) Let $k=3$. If $\lambda(G)=0$, then the edges with positive weights induce from G exactly two connected components G_1 and G_2, where we see that a minimum 3-way cut in G is a minimum 2-way cut in G_1 or G_2. By the result of (i) we have $|\mathscr{X}_k|=O(n_1^2+n_2^2)=O(n^2)$, where n_i is the number of vertices in G_i. We next assume $\lambda(G)>0$. Then for the above two crossing subsets $X_1,X_2\in\mathscr{X}_3$, we have

$$\begin{aligned}
4opt(G,3)&\le cost(Y_1;Y_2;V-(Y_1\cup Y_2))+cost(Y_2;Y_3;V-(Y_2\cup Y_3))\\
&\quad+cost(Y_3;Y_4;V-(Y_3\cup Y_4))+cost(Y_4;Y_1;V-(Y_4\cup Y_1))\\
&=3(f(X_1)+f(X_2))-2cost(Y_1;Y_3)-2cost(Y_2;Y_4)\\
&\le 3((2/3)opt(G,3)+(2/3)opt(G,3))-2cost(Y_1;Y_3)-2cost(Y_2;Y_4)\\
&=4opt(G,3)-2cost(Y_1;Y_3)-2cost(Y_2;Y_4),
\end{aligned}$$

implying that $f(X_1)=f(X_2)=(2/3)opt(G,3)$ and $cost(Y_1;Y_3)=cost(Y_2;Y_4)=0$. Hence no two subsets $X,X'\in\mathscr{X}_3$ with $f(X)<(2/3)opt(G,3)$ and $cost(X';V-X')<(2/3)opt(G,3)$ cross each other, indicating that the number of subsets X with $f(X)<(2/3)opt(G,3)$ is $O(n)$. If $\lambda(G)<(1/3)opt(G,3)$, then there is a subset $Z\subset V$ with $cost(Z;V-Z)=\lambda(G)<(1/3)opt(G,3)(<cost(X_1;V-X_1))$, where $Z\ne X_1\ne V-Z$ holds, and $F=(Z;V-Z)\cup(X_1;V-X_1)$ is a 3-way cut with $cost(F)<(1/3)opt(G,3)+(2/3)opt(G,3)=opt(G,3)$, a contradiction. Therefore $\lambda(G)\ge(1/3)opt(G,3)$

holds, and $f(X) = (2/3)opt(G,3) \leq 2\lambda(G)$ holds for every subset X with $f(X) = (2/3)opt(G,3)$, and this implies that $|\mathscr{X}_3| = O(n^{2\alpha}) = O(n^4)$ holds for $\alpha = 2$.

(iii) Let $k = 4$. If $\lambda(G) = 0$, then it is not difficult to see that $|\mathscr{X}_3| = O(n^4)$ holds by using the results in (i)-(ii). Assume $\lambda(G) > 0$. For the above two crossing subsets $X_1, X_2 \in \mathscr{X}_4$, we have

$$\begin{aligned} opt(G,4) &\leq cost(Y_1;Y_2;Y_3;Y_4) \\ &= f(X_1) + f(X_2) - cost(Y_1;Y_3) - cost(Y_2;Y_4) \\ &\leq (1/2)opt(G,4) + (1/2)opt(G,4) - cost(Y_1;Y_3) - cost(Y_2;Y_4) \\ &= opt(G,4) - cost(Y_1;Y_3) - cost(Y_2;Y_4), \end{aligned}$$

implying that $f(X_1) = f(X_2) = (1/2)opt(G,4)$ and $cost(Y_1;Y_3) = cost(Y_2;Y_4) = 0$. From this, the number of subsets X with $f(X) < (1/2)opt(G,4)$ is $O(n)$. If $\lambda(G) \geq (1/2)opt(G,4)$, then $f(X) = (1/2)opt(G,4) \leq 2\lambda(G)$ holds for every subset X with $f(X) = (1/2)opt(G,4)$, implying that $|\mathscr{X}_4| = O(n^{2\alpha}) = O(n^4)$ holds for $\alpha = 2$. Assume $\lambda(G) < (1/2)opt(G,4)$. Let Z be a subset of V with $f(Z) = \lambda(G)$. Consider two crossing subsets $X_1, X_2 \in \mathscr{X}_4$ with $Z \cap X_1 \neq \emptyset \neq Z \cap X_2$. As observed in the above, we have $f(X_1) = f(X_2) = (1/2)opt(G,4)$ and $cost(Y_1;Y_3) = cost(Y_2;Y_4) = 0$ hold, indicating that one of 3-way cuts $F_1 = (Y_1;Y_2;Y_3 \cup Y_4)$, $F_2 = (Y_1;Y_2 \cup Y_3;Y_4)$, $F_3 = (Y_1 \cup Y_2;Y_3;Y_4)$ and $F_4 = (Y_1 \cup Y_4;Y_2;Y_3)$ has cost at most $(3/4)opt(G,4)$. Such a 3-way cut F_i and $(Z;V-X)$ have $cost(F_i) + cost(Z;V-X) \leq (3/4)opt(G,4) + \lambda(G) < (3/4)opt(G,4) + (1/2)opt(G,4) = opt(G,4)$, and this means that $F_i \cup (Z;V-X)$ remains to be a 3-way cut, i.e., $X_1 \cap X_2 = Z$ (otherwise $F_i \cup (Z;V-X)$ would be a 4-way cut with cost less than $opt(G,4)$). Hence we have a property that, for two crossing subsets $X, X' \in \mathscr{X}_4$ with $Z \cap X \neq \emptyset \neq Z \cap X'$, $X - Z$ and $X' - Z$ are disjoint, and we see that there are $O(n)$ subsets $X \in \mathscr{X}_4$ with $Z \cap X \neq \emptyset$. Since there are $O(n^2)$ subsets Z with $f(Z) = \lambda(G)$, we have $|\mathscr{X}_4| = O(n^2 \cdot n) = O(n^3)$. This completes the proof of the lemma. □

It is known that the h minimum 2-way cuts can be enumerated in $O(hnF(n,m))$ time [11, 12, 35]. Thus, for $k = 2,3,4$ (resp., $k \geq 5$), all subsets X with $f(X) \leq (2/k)opt(G,k)$ (resp., $f(X) < (1/2)opt(G,k)$) can be obtained in $O(n^3F(n,m))$ time for $k = 2$, $O(n^5F(n,m))$ time for $k = 3,4$ (by Lemma 8), and in $O(n^{2k-5}F(n,m))$ time for $k \geq 5$ (by Theorem 4).

We are ready to describe our algorithm for enumerating all minimum k-way cuts.

Algorithm ALL_CUTS(G,k)
Input: A graph $G = (V,E)$ and an integer $k \in [1,|V|]$.
Output: The set $\mathscr{F}$ of all minimum k-way cuts in G.

```
1  if opt(G,k) = 0 then Return F := ∅
2  else /* opt(G,k) > 0 */
3     if k = 2 then Return {(X;V−X) | f(X) = λ(G)}
4     else if k ∈ {3,4} then
         Compute the set 𝒳_k of all subsets X ⊂ V such that
         f(X) ≤ (2/k)opt(G,k) and |V−X| ≥ k−1;
5        𝓕 := {(X;V−X) ∪ F_2 | X ∈ 𝒳_k, F_2 ∈ ALL_CUTS(G[V−X], k−1)};
6        Return 𝓕 := 𝓕 − {F ∈ 𝓕 | cost(F) > opt(G,k)} /* min_{F∈𝓕} cost(F) = opt(G,k) */
7        else /* k ≥ 5 */
8           Compute the set 𝒳_k of all subsets X ⊂ V such that f(X) < (1/2)opt(G,k);
9           p := ⌈(k − √k)/2⌉ − 1;
10          𝓕 := ⋃_{X∈𝒳_k : |X|≥p, |V−X|≥k−p} {F_1 ∪ F_2 |
                  F_1 ∈ ALL_CUTS(G[X], p), F_2 ∈ ALL_CUTS(G[V−X], k−p)};
11          Return 𝓕 := 𝓕 − {F ∈ 𝓕 | cost(F) > opt(G,k)} /* min_{F∈𝓕} cost(F) = opt(G,k) */
12       end /* if */
13    end /* if */
14 end. /* if */
```

We have the same recursive formula on the number of instances generated during the execution of ALL_CUT, where the difference from the analysis for the runtime of MULTIWAY is some of initial terms in $M(k)$. By checking a solution to the formula up to 300000 by a computer program, we see that the total number of instances in $O(n^{4k/(1-1.71/\sqrt{k})-20})$. Since each subset $X \in \mathscr{X}_k$ is generated in $O(nF(n,m)) = O(n^4)$ time. Therefore we establish the next result.

Theorem 9 *For a graph $G = (V,E)$ with n vertices m edges and an integer $k \in [3,n]$, ALL_CUTS(G,k) finds all minimum k-way cuts in $O(n^{4k/(1-1.71/\sqrt{k})-19}F(n,m))$ time, where $F(n,m)$ denotes the time complexity for computing a maximum flow in a graph with n vertices and m edges.*

8 Concluding Remarks

In this paper, we have shown that, for general k, all minimum k-way cuts can be computed in $O(n^{4k/(1-1.71/\sqrt{k})-16})$ time. This is the first deterministic algorithm with time complexity whose exponent is $O(k)$ for a general graph. The new time bound improves the previous time bound $O(n^{k^2/2-3k/2+4}F(n,m))$ for deterministic algorithms, but is still higher than the bound $O(n^{2(k-1)}\log^3 n)$ of a Monte Carlo algorithm due to Karger and Stein [23]. So, it is left open to derive a better upper bound on the number of subsets X with $f(X) < opt(G,k)/2$. The key property for our algorithm is Theorem 4. A similar property is found in the article by Goldschmidt and Hochbaum [9], but we have a simpler proof for this property under a less constrained setting, which allows us to apply Lemma 1 to generate instances with nearly halved k. However, based on Theorem 4, we can find *all* minimum k-way cuts, requiring a high time complexity. Another future work is to find a more restricted characterization for a family $\mathscr{X}$ of subsets X based on which we can construct *at least one* minimum k-way cut.

References

[1] C. J. ALPERT AND A. B. KAHNG, Recent directions in netlist partitioning: a survey, *Integration, the VLSI journal* (1995) **19** 1-81.

[2] D. BERTSIMAS, C. P .TEO, AND R. VOHRA, Analysis of LP relaxations for multiway and multicut problems, *Networks* (1999) **34** 102-114.

[3] M. BURLET AND O. GOLDSCHMIDT, A new and improved algorithm for the 3-cut problem, *Operations Research Letters* (1997) **21** 225-227.

[4] C. J. COLBOURN, *The Combinatorics of Network Reliability*. Oxford Univ. Press (1987).

[5] W. H. CUNNINGHAM, Optimal attack and reinforcement of a network, *J. ACM* (1985) **32** 549-561.

[6] E. DAHLHAUS, D. S. JOHNSON, C. H. PAPADIMITRIOU, P. D. SEYMOUR AND M. YANNAKAKIS, The complexity of multiterminal cuts, *SIAM J. Comput.* (1994) **23** 864-894.

[7] W. E. DONATH, Logic Partitioning, in B. T. Preas and M. J. Lorenzetti (eds.), *Physical Design Automation of VLSI Systems*, Benjamin Cummings, Menlo Park, CA (1988) 65-86.

[8] A. V. GOLDBERG AND R. E. TARJAN, A new approach to the maximum-flow problem, *J. ACM* (1998) **35** 921-940.

[9] O. GOLDSCHMIDT AND D. S. HOCHBAUM, Polynomial algorithm for the k-cut problem for fixed k, *Mathematics of Operation Research* (1994) **19** 24-37.

[10] D. GUSFIELD, Connectivity and edge-disjoint spanning trees, *Inf. Process. Lett.* (1983) **16** 87-89.

[11] H. W. HAMACHER, J.-C. PICARD AND M. QUEYRANNE, Ranking the cuts and cut-sets of a network, *Annals of Discrete Applied Mathematics* **19** (1984) 183-200.

[12] H. W. HAMACHER, J.-C. PICARD AND M. QUEYRANNE, On finding the K best cuts in a network, *Operations Research Letters* **2** (1984) 303-305.

[13] J. HAO AND J. ORLIN, A faster algorithm for finding the minimum cut in a directed graph, *J. Algorithms* (1994) **17** 424-446.

[14] D. HARTVIGSEN, Minimum path basis, *J. Algorithms* (1993) **15** 125-142.

[15] X. HE, An improved algorithm for the planar 3-cut problem, *J. Algorithms* (1991) **12** 23-37.

[16] S. KAPOOR, On minimum 3-cuts and approximating k-cuts using cut trees, *Lecture Notes in Computer Science* Springer (1996) **1084** 132-146,.

[17] Y. KAMIDOI, S. WAKABAYASHI AND N. YOSHIDA, Faster algorithms for finding a minimum k-way cut in a weighted graph, *Proc. IEEE International Symposium on Circuits and Systems* (1997) 1009-1012.

[18] Y. KAMIDOI, S. WAKABAYASHI AND N. YOSHIDA, A divide-and-conquer approach to the minimum k-way cut problem, *Algorithmica* (2002) **32** 262-276 .

[19] D. R. KARGER, Global min-cuts in RNC, and other ramifications of a simple min-cut algorithm, *Proc. the 4th ACM-SIAM Symposium on Discrete Algorithms* (1993) 21-30.

[20] D. R. KARGER, Minimum cuts in near-linear time, *Proc. 28th ACM Symposium on Theory of Computing* (1996) 56-63.

[21] D. R. KARGER, P. KLEIN, C. STEIN, M. THORUP AND N. YOUNG, Rounding algorithms for a geometric embedding of minimum multiway cut, *Proc. ACM Symp. on Theory of Computing* (1999) 668-678.

[22] D. R. KARGER AND C. STEIN, An $\widetilde{O}(n^2)$ algorithm for minimum cuts, *Proc. 25th ACM Symposium on Theory of Computing*, (1993) 757-765.

[23] D. R. KARGER AND C. STEIN, A new approach to the minimum cut problems, *J. ACM* (1996) **43** 601-640.

[24] A. V. Karzanov, Determining the maximal flow in a network by the method of preflows, *Soviet Math. Dokl.* (1974) **15** 434-437.

[25] C. H. LEE, M. KIM AND C. I. PARK, An efficient *k*-way graph partitioning algorithm for task allocation in parallel computing systems, *Proc. IEEE Int. Conf. on Computer-Aided Design* (1990) 748-751.

[26] T. LENGAUR, *Combinatorial Algorithms for Integrated Circuit Layout*, Wiley (1990).

[27] M. S. LEVINE, Faster randomized algorithms for computing minimum $\{3,4,5,6\}$-way cuts, *Proc. of ACM-SIAM Symposium on Discrete Algorithms* (2000) 735-742.

[28] H. NAGAMOCHI AND T. IBARAKI, Computing the edge-connectivity of multigraphs and capacitated graphs, *SIAM J. Disc. Math.* (1992) **5** 54-66.

[29] H. NAGAMOCHI AND T. IBARAKI, A fast algorithms for computing minimum 3-way and 4-way cuts, *Mathematical Programming* (2000) **88** 507-520.

[30] H. NAGAMOCHI, S. KATAYAMA AND T. IBARAKI, Faster algorithm for computing minimum 5-way and 6-way cuts in graphs, *J. Combinatorial Optimization* (2000) **4** 35-78.

[31] J. NAOR AND Y. RABANI, Tree packing and approximating *k*-cuts, *Proc. ACM-SIAM Symp. on Discrete Algorithms* (2001) 26-27.

[32] H. SARAN AND V. V. VAZIRANI, Finding *k* cuts within twice the optimal, *SIAM J. Comput.* (1995) **24** 101-108.

[33] H. S. STONE, Multiprocessor scheduling with the aid of network flow algorithms, *IEEE Trans. on Software Engg.* (1977) **SE-3** 85-93.

[34] P. TITTMANN, Partitions and network reliability, *Discrete Applied Mathematics* (1999) **95** 445-453.

[35] V. VAZIRANI AND M. YANNAKAKIS, Suboptimal cuts: Their enumeration, weight, and number, *Lecture Notes in Computer Sciences* Springer-Verlag (1992) **632** 366-377.

[36] W. C. YEH, A simple slgorithm for the planar pultiway put problem, *J. of Algorithms* (2001) **39** 68-77.

[37] L. ZHAO, H. NAGAMOCHI AND T. IBARAKI, Approximating the minimum *k*-way cut in a graph via minimum 3-way cuts, *J. of Combinatorial Optimization* (2001) **5** 397-410.

[38] L. ZHAO, H. NAGAMOCHI AND T. IBARAKI, On generalized greedy splitting algorithms for multiway partition problems, *Discrete Applied Mathematics* (2004) **143** 130-143.

[39] L. ZHAO, H. NAGAMOCHI AND T. IBARAKI, A greedy splitting algorithm for approximating multiway partition problems, *Mathematical Programming* (2005) **102** 167-183.

A Strongly Polynomial Algorithm for Line Search in Submodular Polyhedra

KIYOHITO NAGANO

Department of Mathematical Informatics
University of Tokyo
Tokyo 113-8656, Japan
kiyohito_nagano@mist.i.u-tokyo.ac.jp

Abstract: A submodular polyhedron is a polyhedron associated with a submodular function. This paper presents a strongly polynomial time algorithm for line search in submodular polyhedra with the aid of a fully combinatorial algorithm for submodular function minimization as a subroutine. The algorithm is based on the parametric search method proposed by Megiddo.

Keywords: submodular functions, network flows, exchange capacities

1 Introduction

Let U be a finite nonempty set. A function ρ defined on 2^U is *submodular* if

$$\rho(X)+\rho(Y) \geq \rho(X \cup Y)+\rho(X \cap Y), \quad \forall X, Y \subseteq U. \tag{1}$$

Iwata, Fleischer and Fujishige [8] and Schrijver [13] independently presented combinatorial, strongly polynomial time algorithms for submodular function minimization. Iwata [6] presented a fully combinatorial strongly polynomial time algorithm, which uses only additions, subtractions, comparisons, and the oracle calls for function values.

For a vector $x \in \mathbf{R}^U$ and $u \in U$, we denote by $x(u)$ the component of x on u. For a submodular function $\rho : 2^U \to \mathbf{R}$ with $\rho(\emptyset) = 0$, the *submodular polyhedron* $\mathbf{P}(\rho)$ and the *base polyhedron* $\mathbf{B}(\rho)$ are defined by

$$\mathbf{P}(\rho) = \{ x \in \mathbf{R}^U \,|\, x(X) \leq \rho(X) \ (\forall X \subseteq U) \}, \tag{2}$$

$$\mathbf{B}(\rho) = \{ x \in \mathbf{R}^U \,|\, x \in \mathbf{P}(\rho), x(U) = \rho(U) \}, \tag{3}$$

where $x(X) = \sum_{u \in X} x(u)$.

Let V be a finite nonempty set with $|V| = n$ and let $f : 2^V \to \mathbf{R}$ be a submodular function with $f(\emptyset) = 0$. In this paper we consider the following problem:

Problem Line Search in Submodular Polyhedra (LSSP)
Instance: A submodular function $f : 2^V \to \mathbf{R}$ with $f(\emptyset) = 0$, a starting point $x_0 \in \mathbf{P}(f)$ and a direction vector $a \in \mathbf{R}^V$.
Task: Find $t^* = \max\{ t \in \mathbf{R} \,|\, x_0 + ta \in \mathbf{P}(f) \}$.

Problem LSSP is a basic problem, but it is previously unknown if Problem LSSP can be solved in strongly polynomial time. If $a \leq \mathbf{0}$, Problem LSSP does not have an optimal solution. Hence throughout we assume that $a \not\leq \mathbf{0}$. We can assume $f(X) \geq 0$, $\forall X \subseteq V$, and $x_0 = 0$, by resetting $f(X) := f(X) - x_0(X)$ for all $X \subseteq V$. So throughout we assume that f is nonnegative, $f(\emptyset) = 0$ and $x_0 = 0$. An example of Problem LSSP is illustrated in Fig. 1.

The *lexicographically optimal base* problem, which was introduced by Fujishige [4], is a generalization of the lexicographically optimal flow problem. Fujishige [4] showed that the lexicographically optimal base can be obtained by repeatedly solving Problem LSSP with $a \geq \mathbf{0}$.

Let us consider the following problem, which is a special case of Problem LSSP:

Problem Line Search in Base Polyhedra (LSBP)
Instance: A submodular function $f : 2^V \to \mathbf{R}$ with $f(\emptyset) = 0$, a starting point $x_0 \in \mathbf{B}(f)$ and a direction vector $a \in \mathbf{R}^V$ with $a(V) = 0$.
Task: Find $t^* = \max\{ t \in \mathbf{R} \,|\, x_0 + ta \in \mathbf{B}(f) \}$ $(= \max\{ t \in \mathbf{R} \,|\, x_0 + ta \in \mathbf{P}(f) \})$.

As is the case with Problem LSSP, it is previously unknown if Problem LSBP can be solved in strongly polynomial time.

For each $v \in V$, let χ_v be a vector that has value 1 on v and 0 elsewhere. As an instance of Problem LSBP, if $a = \chi_v - \chi_{v'}$ for $v, v' \in V$, $v \neq v'$, $t^* = \max\{ t \in \mathbf{R} \,|\, x_0 + t(\chi_v - \chi_{v'}) \in \mathbf{B}(f) \}$ is said to be an *exchange capacity* (see, for example, Fujishige [5]). Problem LSBP can be interpreted as a problem of computing a *generalized exchange capacity*.

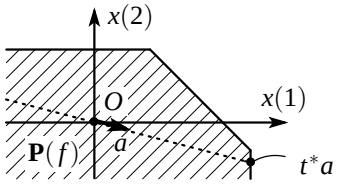


Figure 1: Problem LSSP ($n = 2$)

The problems of finding maximum flow values in capacitated networks can be reduced to Problem LSBP. Consider a network $\mathcal{N}$ with a directed graph $G = (V, E)$, where V is a vertex set and E is an arc set, and with a nonnegative capacity vector $c \in \mathbf{R}^E$. For $X \subseteq V$, we denote by $\delta(X)$ the set $\{e = (v, v') \in E \,|\, v \in X,\ v' \in V \setminus X\}$. We define a function $\kappa_c : 2^V \to \mathbf{R}$ as $\kappa_c(X) = c(\delta(X))$ $(X \subseteq V)$. This function κ_c is called a *cut function* and κ_c is a nonnegative submodular function with $\kappa_c(\emptyset) = \kappa_c(V) = 0$. A vector $x \in \mathbf{R}^V$ is said to be *feasible* for $\mathcal{N}$ if there exists a vector $y \in \mathbf{R}^E$ such that

$$0 \le y \le c, \ \text{ and } \ \sum\{y(e) \,|\, e \in \delta(\{v\})\} - \sum\{y(e) \,|\, e \in \delta(V \setminus \{v\})\} = x(v),\ \forall v \in V. \tag{4}$$

Maximum r-s flows Let $r, s \in V$, $r \neq s$ and we consider finding a maximum flow value from r to s. The maximum r-s flow value is $t^* = \max\{t \in \mathbf{R} \,|\, t(\chi_r - \chi_s)$ is feasible for $\mathcal{N}\}$. It is known that the set $\{x \in \mathbf{R}^V \,|\, x$ is feasible for $\mathcal{N}\}$ is equal to the base polyhedron $\mathbf{B}(\kappa_c)$. Therefore we can find the maximum r-s flow value by solving Problem LSBP.

Maximum w-proportional flows Let $S^+, S^- \subseteq V$, $S^+ \cap S^- = \emptyset$ and let $w \in \mathbf{R}^V$ be a vector which satisfies

$$\sum\{w(v) \,|\, v \in S^+\} = 1,\ \sum\{w(v) \,|\, v \in S^-\} = -1 \ \text{ and } \ \begin{cases} w(v) > 0 \ \ (v \in S^+), \\ w(v) < 0 \ \ (v \in S^-), \\ w(v) = 0 \ \ (v \in V \setminus (S^+ \cup S^-)). \end{cases}$$

For $t \in \mathbf{R}$, we say that $y \in \mathbf{R}^E$ is a w-proportional flow with flow value t if y satisfies the condition (4) w.r.t. $x = tw$. The maximum w-propotional flow value is $t^* = \max\{t \in \mathbf{R} \,|\, tw \in \mathbf{B}(\kappa_c)\}$ and this is the optimal value of Problem LSBP.

The Newton method (Section 4) is a simple approach to Problem LSSP. If $a \ge \mathbf{0}$, it is known that Problem LSSP can be solved in strongly polynomial time. (See, e.g., Fujishige [5, Section 7.2].) If $a \in \mathbf{R}^V$ and $a \not\le \mathbf{0}$, however, it is left open to verify if the Newton method for Problem LSSP runs in strongly polynomial time.

In Section 5, we propose an algorithm for Problem LSSP, which is quite different from the Newton method. The algorithm uses a fully combinatorial algorithm for submodular function minimization [6, 7] as a subroutine, within the framework of the parametric search method proposed by Megiddo [9]. It solves Problem LSSP in strongly polynomial time.

From the definition of a submodular polyhedron (2), it is easy to see that the optimal value t^* of Problem LSSP is equal to $\min\{f(X)/a(X) \,|\, X \subseteq V, a(X) > 0\}$. So Problem LSSP can be regarded as a minimum-ratio problem. We also show that a minimum-ratio problem which is a generalization of Problem LSSP can be solved in strongly polynomial time in Section 6.

2 Preliminaries

Definitions and basic properties

Let U be a finite nonempty set. A family $\mathscr{D} \subseteq 2^U$ is said to be a *ring family* if it satisfies $X, Y \in \mathscr{D} \Rightarrow X \cup Y, X \cap Y \in \mathscr{D}$. Let $\mathscr{D} \subseteq 2^U$ be a ring family and ρ be a function defined on $\mathscr{D}$. A function ρ is said to be *submodular* if $\rho(X) + \rho(Y) \ge \rho(X \cup Y) + \rho(X \cap Y)$ $(X, Y \in \mathscr{D})$. We say that ρ is *supermodular* if $-\rho$ is submodular and that ρ is *modular* if ρ is submodular and supermodular.

Let $\mathscr{D} \subseteq 2^U$ be a ring family and let $\rho : \mathscr{D} \to \mathbf{R}$ be a submodular function. Let $\arg\min\rho \subseteq \mathscr{D}$ denote a family of all the minimizers of ρ. It is not difficult to see that $\arg\min\rho$ forms a ring family by the submodularity of ρ. As $\arg\min\rho$ is closed under union and intersection, there exists the *minimal minimizer* $X_{\min} = \bigcap \arg\min\rho \in \arg\min\rho$ and exists the *maximal minimizer* $X_{\max} = \bigcup \arg\min\rho \in \arg\min\rho$.

Let $\rho : 2^U \to \mathbf{R}$ be a submodular function with $\rho(\emptyset) = 0$ and let $x \in \mathbf{P}(\rho)$. A subset $X \subseteq U$ is said to be a *x-tight* w.r.t. ρ if $x(X) = \rho(X)$. We denote the family of x-tight sets w.r.t. ρ by $\mathscr{D}_\rho(x)$. Namely, $\mathscr{D}_\rho(x) = \{X \subseteq U \,|\, x(X) = \rho(X)\}$. For any $y \in \mathbf{R}^U$, a function $\rho_y : 2^U \to \mathbf{R}$ defined by $\rho_y(X) = \rho(X) - y(X)$ $(X \subseteq V)$ is obviously a submodular function and $\rho_y(\emptyset) = 0$. As $x \in \mathbf{P}(\rho)$, $\rho_x(\emptyset) = 0$ and $\rho_x(X) \ge 0$, $\forall X \subseteq U$. This implies $X \in \mathscr{D}_\rho(x) \iff X \in \arg\min\rho_x$. So $\mathscr{D}_\rho(x) = \arg\min\rho_x$, therefore $\mathscr{D}_\rho(x)$ forms a ring family. Note that $\emptyset \in \mathscr{D}_\rho(x)$.

Let $\mathscr{D} \subseteq 2^U$ be a ring family. Now we assume $\{\emptyset, U\} \subseteq \mathscr{D}$. Although the cardinality of $\mathscr{D}$ may be exponentially large in general, $\mathscr{D}$ can always be represented by a directed graph with at most $|U|$ vertices. For a directed graph $G = (U, A)$, we call

$X \subseteq U$ a *closure* of G if $(u,u') \in A$ and $u \in X$ imply $u' \in X$. It is known that there exists a directed graph $G_{\mathscr{D}} = (U, A_{\mathscr{D}})$ such that $\mathscr{D}$ is the family of closures of G. For example, if $G_{\mathscr{D}} = (U, A_{\mathscr{D}})$ is a directed graph with

$$A_{\mathscr{D}} = \big\{(u,u') \,\big|\, u,u' \in U,\ u \neq u',\ u' \in \bigcap\{X \,|\, u \in X \in \mathscr{D}\}\big\},$$

then $\mathscr{D} = \{X \,|\, X$ is a closure of $G_{\mathscr{D}}\}$ (see, e.g., [5, Section 3.2]). We say that $G_{\mathscr{D}}$ is a *directed graph representation* of $\mathscr{D}$. We decompose $G_{\mathscr{D}}$ into strongly connected components $\{\Gamma_{\mathscr{D}}(s) \,|\, s \in S\}$. Let $\mathscr{G}_{\mathscr{D}} = (S, F_{\mathscr{D}})$ be a directed acyclic graph determined by $G_{\mathscr{D}}$ in a natural way. For each $T \subseteq S$, we define $\Gamma_{\mathscr{D}}(T) = \bigcup_{s\in T} \Gamma_{\mathscr{D}}(s)$. Now $\mathscr{D} = \big\{\Gamma_{\mathscr{D}}(T) \,\big|\, T \subseteq S$ is a closure of $\mathscr{G}_{\mathscr{D}}\big\}$. We say that $(\Gamma_{\mathscr{D}}, \mathscr{G}_{\mathscr{D}})$ is a *contracted directed graph representation* of $\mathscr{D}$. A ring family $\mathscr{D}$ is called *simple* if $G_{\mathscr{D}}$ is acyclic. For a simple ring family $\mathscr{D}$, we can identify $(\Gamma_{\mathscr{D}}, \mathscr{G}_{\mathscr{D}})$ with $G_{\mathscr{D}}$.

For a ring family $\mathscr{D}$, we consider the case when $\emptyset \notin \mathscr{D}$ and/or $U \notin \mathscr{D}$. Let $U_{\mathrm{res}} = \bigcup \mathscr{D} \setminus \bigcap \mathscr{D}$ and let $\mathscr{D}_{\mathrm{res}} = \{X \setminus \bigcap \mathscr{D} \,|\, X \in \mathscr{D}\}$. Note that $\mathscr{D}_{\mathrm{res}}$ is a ring family with $\{\emptyset, U_{\mathrm{res}}\} \subseteq \mathscr{D}_{\mathrm{res}} \subseteq 2^{U_{\mathrm{res}}}$. We define $G_{\mathscr{D}} := G_{\mathscr{D}_{\mathrm{res}}}$, $\mathscr{G}_{\mathscr{D}} := \mathscr{G}_{\mathscr{D}_{\mathrm{res}}}$, and $\Gamma_{\mathscr{D}} := \Gamma_{\mathscr{D}_{\mathrm{res}}}$. We say that $(\bigcap \mathscr{D}, \bigcup \mathscr{D}, G_{\mathscr{D}})$ $((\bigcap \mathscr{D}, \bigcup \mathscr{D}, (\Gamma_{\mathscr{D}}, \mathscr{G}_{\mathscr{D}})))$ is a (contracted) directed graph representation of $\mathscr{D}$.

Optimality conditions for Problem LSSP

As an instance of Problem LSSP, w.l.o.g., we assume that f is nonnegative, $f(\emptyset) = 0$, $x_0 = 0$, and $a \not\leq \mathbf{0}$. We explain that the optimal value t^* of Problem LSSP is nonnegative and finite. The optimal value is by definition $t^* = \max\{t \,|\, ta \in \mathbf{P}(f)\}$. Since $0 \in \mathbf{P}(f)$, t^* is nonnegative. Let $A \subseteq V$ be a subset which satisfies $a(A) > 0$. If $t > f(A)/a(A)$, then $ta(A) > f(A)$ and hence $ta \notin \mathbf{P}(f)$. So $t^* \leq f(A)/a(A)$ and t^* is finite.

For any $t \in \mathbf{R}$ we consider deciding whether $ta \in \mathbf{P}(f)$ or $ta \notin \mathbf{P}(f)$. Since, for any $x \in \mathbf{R}^V$, $f(\emptyset) - x(\emptyset) = 0$, we have $x \in \mathbf{P}(f) \iff \min\{f_x(X) \,|\, X \subseteq V\} = 0$, and if x can be represented as ta, using $ta(\emptyset) = 0$,

$$\begin{aligned} ta \in \mathbf{P}(f) &\iff \min\{f_{ta}(X) \,|\, X \subseteq V\} = 0, \\ ta \notin \mathbf{P}(f) &\iff \min\{f_{ta}(X) \,|\, X \subseteq V\} < 0. \end{aligned} \tag{5}$$

So we can decide whether $ta \in \mathbf{P}(f)$ or $ta \notin \mathbf{P}(f)$ by minimizing f_{ta}.

Now, let us consider the optimality condition for Problem LSSP. For $t \geq 0$, we consider the conditions of "$t < t^*$", "$t = t^*$" and "$t > t^*$". See Figure 2 to understand each condition intuitively. Note that $t = t^*$ and ta in the boundary of $\mathbf{P}(f)$ are not equivalent (see Ex. 1. 2 and Ex. 1. 3 in Figure 2).

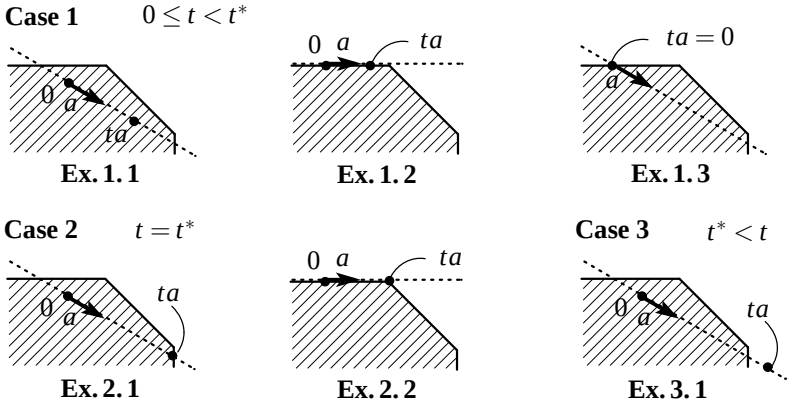


Figure 2: Relation between t and t^*

By the definition of t^*, it is easy to see that

$$t = t^* \iff \max\{a(X) \,|\, X \in \mathscr{D}(ta)\} > 0. \tag{6}$$

For $ta \in \mathbf{P}(f)$, $\mathscr{D}(ta)$ always includes $\emptyset$, so $\max\{a(X) \,|\, X \in \mathscr{D}(ta)\} \geq 0$. Thus using (5) and (6), we obtain the following conditions for t^* and any $t \geq 0$:

$$\begin{aligned} t \lhd t^* &\iff \begin{cases} \min\{f_{ta}(X) \,|\, X \subseteq V\} = 0, \\ \max\{a(X) \,|\, X \in \mathscr{D}(ta)\} = 0, \end{cases} \\ t = t^* &\iff \begin{cases} \min\{f_{ta}(X) \,|\, X \subseteq V\} = 0, \\ \max\{a(X) \,|\, X \in \mathscr{D}(ta)\} > 0, \end{cases} \\ t \rhd t^* &\iff \min\{f_{ta}(X) \,|\, X \subseteq V\} < 0. \end{aligned} \tag{7}$$

3 Submodular function minimization

Finding a minimizer of a submodular function

Let U be a finite nonempty set and let $\rho : 2^U \to \mathbf{R}$ be any submodular function. We assume that for any given $X \subseteq U$ a function value $\rho(X)$ can be acquired by an *oracle call*. Let $\gamma(\rho)$ denote the upper bound on the time to compute the function value of ρ.

An algorithm for submodular function minimization is said to be a strongly polynomial time algorithm if the total number of oracle calls for function evaluation and *arithmetic operations*, that is, additions, subtractions, multiplications, divisions and comparisons, is bounded by some polynomial in $|U|$. Combinatorial strongly polynomial time algorithms for submodular function minimization are given independently by Iwata, Fleischer and Fujishige (IFF) [8] and Schrijver [13]. Iwata [7] described an improved variant of the IFF algorithm and this algorithm achieves the best known bound on the running time, $O((|U|^6 \log|U|)\cdot\gamma(\rho)+|U|^7\log|U|)$. Let Algorithm SFM be an algorithm which finds a minimizer of a submodular function $\rho : 2^U \to \mathbf{R}$ with $O(\mathscr{T}^{\mathsf{O}}(|U|))$ oracle calls for function evaluation and $O(\mathscr{T}^{\mathsf{A}}(|U|))$ arithmetic operations where $\mathscr{T}^{\mathsf{O}}(|U|)$ and $\mathscr{T}^{\mathsf{A}}(|U|)$ are some polynomials in $|U|$. Let $\mathscr{T}(|U|,\gamma(\rho)) = \mathscr{T}^{\mathsf{O}}(|U|)\cdot\gamma(\rho)+\mathscr{T}^{\mathsf{A}}(|U|)$.

An algorithm for submodular function minimization is said to be a *fully combinatorial* strongly polynomial time algorithm if the total number of oracle calls for function evaluation of ρ and *fully combinatorial operations*, that is, additions, subtractions and comparisons, is bounded by some polynomial in $|U|$. Iwata [6] presented a fully combinatorial strongly polynomial time algorithm for submodular function minimization as a variant of the IFF algorithm [8], and later, Iwata [7] described an improved algorithm, which runs in $O(|U|^8\log^2|U|\cdot\gamma(\rho))$ time. Let Algorithm FC-SFM be some algorithm which finds a minimizer of a submodular function $\rho : 2^U \to \mathbf{R}$ with $O(\mathscr{T}^{\mathsf{O}}_{\mathsf{FC}}(|U|))$ oracle calls for function evaluation of ρ and $O(\mathscr{T}^{\mathsf{FC}}_{\mathsf{FC}}(|U|))$ fully combinatorial operations, where $\mathscr{T}^{\mathsf{O}}_{\mathsf{FC}}(|U|)$ and $\mathscr{T}^{\mathsf{FC}}_{\mathsf{FC}}(|U|)$ are some polynomials in $|U|$. Let $\mathscr{T}_{\mathsf{FC}}(|U|,\gamma(\rho)) = \mathscr{T}^{\mathsf{O}}_{\mathsf{FC}}(|U|)\cdot\gamma(\rho)+\mathscr{T}^{\mathsf{FC}}_{\mathsf{FC}}(|U|)$.

Constructing all the minimizers of a submodular function

Let $\rho : 2^U \to \mathbf{R}$ be a submodular function and $X_{\min}$, $X_{\max}$ be the minimal, maximal minimizer of ρ respectively. We are interested in constructing $\arg\min\rho$, that is, finding $X_{\min}$, $X_{\max}$ and a (contracted) directed graph representation of $\arg\min\rho$.

It is not difficult to show that a (fully combinatorial) algorithm which computes $X_{\min}$, $X_{\max}$ and $G_{\arg\min\rho}$ can be designed by using any (fully combinatorial) submodular function minimization algorithm $O(|U|^2)$ times, or by using any (fully combinatorial) algorthm which finds the minimal minimizer of a submodular function $|U|$ times. As for presently known combinatorial, strongly polynomial algorithms for submodular function minimization [13, 6, 7, 8], we can do much better than that: using each one of them, we can construct $\arg\min\rho$ in the same asymptotic running time as a single computation of the original algorithm. Now we explain how to achieve this.

For $x \in \mathbf{R}^U$, we define $x^- \in \mathbf{R}^U$ by $x^-(u) = \min\{0, x(u)\}$ for $u \in U$, and define $\mathrm{supp}^-(x), \mathrm{supp}^+(x) \subseteq U$ by $\{u \in U \mid x(u) < 0\}$, $\{u \in U \mid x(u) > 0\}$ respectively. For any $x \in \mathbf{B}(\rho)$ and any $X \subseteq U$, we have $x^-(U) \le x(X) \le \rho(X)$, and the vector reduction theorem on polymatroids due to Edmonds [3] immediately implies

$$\max\{x^-(U) \mid x \in \mathbf{B}(\rho)\} = \min\{\rho(X) \mid X \subseteq U\}. \tag{8}$$

So, for a maximizer x' of the left-hand side of (8), we have

$$\arg\min\rho = \{X \mid X \in \mathscr{D}(x'), \mathrm{supp}^-(x') \subseteq X \subseteq U \setminus \mathrm{supp}^+(x')\}. \tag{9}$$

An extreme point of a base polyhedron is said to be an *extreme base*. The following theorem plays an important role for the construction of $\arg\min\rho$.

Theorem 1 (Bixby, Cunningham and Topkis [2]) *Let $\rho : 2^U \to \mathbf{R}$ be a submodular function, and let b be an extreme base of $\mathbf{B}(\rho)$.*

(i) $\mathscr{D}_\rho(b)$ includes $\{\emptyset, U\}$ and is a simple ring family.

(ii) A directed graph representation of $\mathscr{D}_\rho(b)$ can be constructed in $O(|U|^2\cdot\gamma(\rho))$ time. □

If a maximizer x' of the left-hand side of (8) is given as a convex combination of k extreme bases, then using (9) and Theorem 1 we can construct $\arg\min\rho$ in $O(k|U|^2\cdot\gamma(\rho))$ time. (See Note 10.11 in [10] for details.) Schrijver's algorithm finds a maximizer of the left-hand side of (8) which is represented as a convex combination of (we may assume) at most $|U|$ extreme points of $\mathbf{B}(\rho)$. Schrijver's algorithm runs in $O(|U|^7\cdot\gamma(\rho)+|U|^8)$ time. So $\arg\min\rho$ can also be constructed in $O(|U|^7\cdot\gamma(\rho)+|U|^8)$ time.

Unfortunately, the IFF algorithm and its variants do not find a maximizer of the left-hand side of (8). But we can overcome this difficulty. They use the same framework and we can construct $\arg\min\rho$ in the same way. In the algorithms, we maintain $Z, H \subseteq U$, a partition $\{\Gamma(s) \mid s \in S\}$ of $U \setminus (Z \cup H)$ and a directed acyclic graph $\mathscr{G} = (S, F)$ such that

⟨IFF-1⟩ $Z \subseteq X_{\min}$, $H \subseteq (U \setminus X_{\max})$,

⟨IFF-2⟩ for each $s \in S$ and $X \in \operatorname{argmin}\rho$, $\Gamma(s) \subseteq X$ or $\Gamma(s) \cap X = \emptyset$,

⟨IFF-3⟩ for each arc $(s, s') \in F$ and $X \in \operatorname{argmin}\rho$, $\Gamma(s) \subseteq X$ implies $\Gamma(s') \subseteq X$.

Intuitively, we update $(Z, U \setminus H, (\Gamma, \mathscr{G}))$ toward $(X_{\min}, X_{\max}, (\Gamma_{\operatorname{argmin}\rho}, \mathscr{G}_{\operatorname{argmin}\rho}))$ in the algorithms. We define $\Gamma(T) = \bigcup_{s\in T}\Gamma(s)$ $(T \subseteq S)$, and define a function $\widehat{\rho} : 2^S \to \mathbf{R}$ by $\widehat{\rho}(T) = \rho(\Gamma(T) \cup Z) - \rho(Z)$ $(T \subseteq S)$. It is obvious that $\widehat{\rho}$ is submodular. By ⟨IFF-1⟩ and ⟨IFF-2⟩, it is easy to see that for each minimizer $T \subseteq S$ of $\widehat{\rho}$ $\Gamma(T) \cup Z$ is minimizer of ρ, and for each minimizer X of ρ there exists a minimizer T of $\widehat{\rho}$ such that $X = \Gamma(T) \cup Z$. For $s \in S$, let $R(s) \subseteq S$ denote the set of vertices reachable from s in $\mathscr{G}$. The algorithms finally obtain Z, H, Γ, $\mathscr{G} = (S, F)$ which satisfy

$$\eta := \max\{\widehat{\rho}(R(s)) - \widehat{\rho}(R(s) \setminus \{s\}) \,|\, s \in S\} \leq 0.$$

We assume $\eta \leq 0$. As $\mathscr{G} = (S, F)$ is acyclic, there exists a linear order $\widehat{L}$ on S in which for each $(s, s') \in F$, s' is a predecessor of s. Let $\widehat{L} = (s_1, \ldots, s_{|S|})$ be such a linear order. We define $\widehat{L}(s_j) = \{s_1, \ldots, s_j\}$ $(j = 1, \ldots, |S|)$. By definition, for each $s \in S$, $R(s) \subseteq \widehat{L}(s)$. Let $\widehat{b} \in \mathbf{R}^S$ be a vector defined by $\widehat{b}(s) = \widehat{\rho}(\widehat{L}(s)) - \widehat{\rho}(\widehat{L}(s) \setminus \{s\})$ $(s \in S)$. Now $\widehat{b}$ is an extreme base of $\mathbf{B}(\widehat{\rho})$ (see Edmonds [3]). For each $s \in S$ we have, by the submodularity of $\widehat{\rho}$, $\widehat{b}(s) \leq \widehat{f}(R(s)) - \widehat{f}(R(s) \setminus \{s\}) \leq \eta \leq 0$, that is, $\widehat{b} \leq \mathbf{0}$. Since $\widehat{b} \in \mathbf{B}(\widehat{\rho})$ and $\widehat{b} \leq \mathbf{0}$, we have

$$\widehat{b}^-(S) = \widehat{b}(S) = \widehat{\rho}(S) \leq \widehat{b}(T) \leq \widehat{\rho}(T) \;\;, \forall T \subseteq S. \tag{10}$$

By (10), S is a minimizer of $\widehat{\rho}$ and, of course, the maximal minimizer of $\widehat{\rho}$. So $\Gamma(S) \cup Z$ is the maximal minimizer of ρ. The original algorithms output $U \setminus H = \Gamma(S) \cup Z$ as a minimizer of ρ and stops. We can construct $\operatorname{argmin}\widehat{\rho}$ using $\widehat{b} \in \mathbf{B}(\widehat{\rho})$ as follows. By (10) we have

$$\operatorname{argmin}\widehat{\rho} = \{T \,|\, T \in \mathscr{D}_{\widehat{\rho}}(\widehat{b}),\ \operatorname{supp}^-(\widehat{b}) \subseteq T \subseteq S\}. \tag{11}$$

As $\widehat{b}$ is an extreme base of $\mathbf{B}(\widehat{\rho})$, we can easily obtain $G_{\mathscr{D}_{\widehat{\rho}}(\widehat{b})} = (S, A_{\mathscr{D}_{\widehat{\rho}}(\widehat{b})})$ by Theorem 1. It follows from (11) that $\bigcap \operatorname{argmin}\widehat{\rho}$ is the set of vertices reachable from $\operatorname{supp}^-(\widehat{b})$ in $G_{\mathscr{D}_{\widehat{\rho}}(\widehat{b})}$. Let G' be the subgraph of $G_{\mathscr{D}_{\widehat{\rho}}(\widehat{b})}$ induced by $S \setminus \bigcap \operatorname{argmin}\widehat{\rho}$. It is easy to see that $(\bigcap \operatorname{argmin}\widehat{\rho}, S, G')$ is a directed graph representation of $\operatorname{argmin}\widehat{\rho}$. From the correspondence between $\operatorname{argmin}\widehat{\rho}$ and $\operatorname{argmin}\rho$, we can construct $\operatorname{argmin}\rho$ straightforward. Let $\Gamma' : S \setminus \bigcap \operatorname{argmin}\widehat{\rho} \to 2^U$ be a function defined by $\Gamma'(s) = \Gamma(s)$ $(s \in S \setminus \bigcap \operatorname{argmin}\widehat{\rho})$. Then $(X_{\mathsf{min}}, X_{\mathsf{max}}, (\Gamma', G'))$ is a contracted directed graph representation of $\operatorname{argmin}\rho$ where $X_{\mathsf{min}} = \Gamma(\bigcap \operatorname{argmin}\widehat{\rho}) \cup Z$ and $X_{\mathsf{max}} = \Gamma(S) \cup Z$. As a result, we can design a combinatorial algorithm which constructs $\operatorname{argmin}\rho$ in $\mathrm{O}((|U|^6 \log|U|) \cdot \gamma(\rho) + |U|^7 \log|U|)$ time. Moreover, we can design a fully combinatorial algorithm which constructs $\operatorname{argmin}\rho$ in $\mathrm{O}(|U|^8 \log^2|U| \cdot \gamma(\rho))$ time.

Let Algorithm $\mathsf{SFM}_{\mathsf{am}}$ be some combinatorial strongly polynomial time algorithm which constructs all the minimizers of a submodular function $\rho : 2^U \to \mathbf{R}$, and let Algorithm $\mathsf{FC\text{-}SFM}_{\mathsf{am}}$ be some fully combinatorial strongly polynomial time algorithm which constructs all the minimizers of a submodular function $\rho : 2^U \to \mathbf{R}$. For simplicity we assume that the running time of $\mathsf{SFM}_{\mathsf{am}}$ is $\mathrm{O}(\mathscr{T}(|U|, \gamma(\rho)))$ and that of $\mathsf{FC\text{-}SFM}_{\mathsf{am}}$ is $\mathrm{O}(\mathscr{T}_{\mathsf{FC}}(|U|, \gamma(\rho)))$.

4 The Newton method for Problem LSSP

In this section we describe the Newton method for Problem LSSP. It is left open to verify if the Newton method for Problem LSSP runs in strongly polynomial time. The Newton method for Problem LSSP uses an algorithm for submodular function minimization as a subroutine.

We define a function $h : \mathbf{R} \to \mathbf{R}$ as

$$h(t) = \min_{X\subseteq V}\{f_{ta}(X)\} = \min_{X\subseteq V}\{f(X) - ta(X)\}. \tag{12}$$

It is obvious that h is concave. As $0 \in \mathbf{P}(f)$ and $f(\emptyset) = 0$, $h(0) = 0$. Since $f_{ta}(\emptyset) = 0$ for any $t \in \mathbf{R}$, $h(t) \leq 0$ for any $t \in \mathbf{R}$. Using the definition of t^*, (5) and (12), we have $t^* = \max\{t \in \mathbf{R} \,|\, h(t) = 0\}$. The graph of h is illustraited in Figure 3 by a thick curve. For any $t \in \mathbf{R}$ we can obtain the value $h(t)$ by running $\mathsf{SFM}(f - ta)$. For simplicity, we assume $n = \mathrm{O}(\gamma(f))$ in the rest of this paper (remember that we reset $f(X) := f(X) - x_0(X)$ for all $X \subseteq V$ in Section 1). Using this assumption, for any $x \in \mathbf{R}^V$, the function value of f_x can be acquired in $\mathrm{O}(\gamma(f))$ time. So $f - ta$ can be minimized in $\mathrm{O}(\mathscr{T}(n, \gamma(f)))$ time. The Newton method is described below. Figure 3 illustrates the process of the algorithm.

The Newton method for Problem LSSP

Step 0: For $X_0 \subseteq V$ such that $a(X_0) > 0$, set $t_1 := f(X_0)/a(X_0)$ $(\geq t^*)$. Set $i := 1$.

Step 1: Obtain $X_i \subseteq V$ such that $h(t_i) = f(X_i) - t_i a(X_i)$ by running $\mathsf{SFM}(f - t_i a)$.

Step 2: If $h(t_i) = 0$, return $t^* := t_i$ and stop. If $h(t_i) < 0$ then set $t_{i+1} := f(X_i)/a(X_i)$ and $i := i + 1$. Go to Step 1.

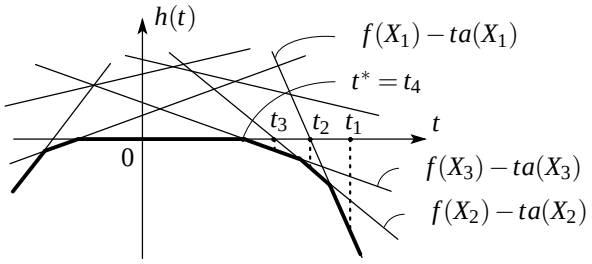


Figure 3: The Newton method

As $h(t)$ has at most 2^n linear segments, the Newton method terminates in a finite number of iterations. If $a \geq \mathbf{0}$, it is known that the number of iterations of the Newton method for Problem LSSP is at most $n+1$. (See, e.g., [5, Section 7.2] for details.) It is left open to verify if the Newton method for Problem LSSP runs in a strongly polynomial number of iterations. An analysis based on Radzik [12] gives a weakly polynomial bound on the number of iterations.

Theorem 2 *Let f be an integer-valued nonnegative submodular function with $f(\emptyset)$, and let a be an integer vector which satisfies $a \not\leq \mathbf{0}$. If $\max_{X\subseteq V}|f(X)| \leq U_1$, $\max_{X\subseteq V}|a(X)| \leq U_2$, the Newton method for Problem* LSSP *runs in $O(\log U_1 + \log U_2)$ iterations.* □

5 A strongly polynomial algorithm for Problem LSSP

In this section we present a combinatorial strongly polynomial time algorithm for Problem LSSP. We use a fully combinatorial strongly polynomial algorithm for submodular function minimization [6, 7] as a subroutine and the parametric search method proposed by Megiddo [9].

Framework

Later we will describe two procedures for Comparison with the Optimal Value; Procedure COV and Procedure L-COV. For any given nonnegative value $t \geq 0$, we can tell whether "$t < t^*$", "$t = t^*$" or "$t > t^*$" by running $\mathsf{COV}(t)$ in $O(\gamma(f)\cdot\mathscr{T}^{\mathsf{O}}_{\mathsf{COV}}(n) + \mathscr{T}^{\mathsf{A}}_{\mathsf{COV}}(n))$ time, where $\mathscr{T}^{\mathsf{O}}_{\mathsf{COV}}(n)$ and $\mathscr{T}^{\mathsf{A}}_{\mathsf{COV}}(n)$ are some polynomials in n. Procedure L-COV is a similar procedure. For any given $t \geq 0$, once $ta(v)$ is computed for each $v \in V$, it compares t to t^* with $O(\mathscr{T}^{\mathsf{O}}_{\mathsf{L\text{-}COV}}(n))$ oracle calls for function evaluation of f, and $O(\mathscr{T}^{\mathsf{FC}}_{\mathsf{L\text{-}COV}}(n))$ *fully combinatorial operations*, that is, additions, subtractions and comparisons, where $\mathscr{T}^{\mathsf{O}}_{\mathsf{L\text{-}COV}}(n)$ and $\mathscr{T}^{\mathsf{FC}}_{\mathsf{L\text{-}COV}}(n)$ are some polynomials in n. Moreover, if $t = t^*$, Procedure L-COV returns a subset $X \subseteq V$ such that $f(X) = t^*a(X)$ and $a(X) > 0$.

By running $\mathsf{COV}(0)$ we can tell whether $t^* = 0$ or $t^* > 0$. So we can assume that $t^* > 0$. If we knew the value of t^* and run $\mathsf{L\text{-}COV}(t^*)$, then it would return "$t^* = t^*$" and a subset $X \subseteq V$ s. t. $f(X) = t^*a(X)$ and $a(X) > 0$, that is, $t^* = f(X)/a(X)$. We try to run $\mathsf{L\text{-}COV}(t^*)$ *without knowing* the value of t^*. If we can run $\mathsf{L\text{-}COV}(t^*)$ successfully without knowing the value of t^*, we can obtain t^* by $f(X)/a(X)$ using $X \subseteq V$ s. t. $f(X) = t^*a(X)$ and $a(X) > 0$. The point is *how* to run $\mathsf{L\text{-}COV}(t^*)$ successfully without knowing the value of t^*. To achieve this goal, we use Megiddo's parametric search method [9].

Megiddo's parametric search

We give a strongly polynomial time algorithm for Problem LSSP using the parametric search technique of Megiddo [9]. We explain this technique in the following paragraphs.

Operations used in running $\mathsf{L\text{-}COV}(t^*)$ are additions, subtractions, comparisons, oracle calls for function evaluation of f, and only n multiplications to obtain $t^*a(v)$ for each $v \in V$. So each value which appears in running $\mathsf{L\text{-}COV}(t^*)$ can be represented as the form $p - qt^*$ where values p, q are known values and not functions of t^*. We consider trying to run $\mathsf{L\text{-}COV}(t^*)$ without knowing the value of t^* with all the values represented as linear functions of t^*. When values are represented as linear functions of t^*, each operation is done as follows:

Operation

An addition : $(p_1 - t^*q_1) + (p_2 - t^*q_2) := (p_1 + p_2) - t^*(q_1 + q_2)$.

A subtraction : $(p_1 - t^*q_1) - (p_2 - t^*q_2) := (p_1 - p_2) - t^*(q_1 - q_2)$.

A comparison : $(p_1 - t^*q_1) \overset{?}{\gtrless} (p_2 - t^*q_2) :=$
$(p_1 - t^*q_1) \gtrless (p_2 - t^*q_2)$
or
$(p_1 - t^*q_1) \gtrless (p_2 - t^*q_2)$
or
$(p_1 - t^*q_1) \lessgtr (p_2 - t^*q_2)$.

An addition of two linear functions of t^* needs 2 scalar additions. A subtraction of two linear functions of t^* needs 2 scalar subtractions. So, even though t^* is not known additions and subtractions do not change the asymptotic running time of the procedure. A comparison of two linear functions of t^*, however, is not so easy. We now consider comparing two linear functions of t^*. Let p_1, p_2, q_1, q_2 be known values. Let us consider the comparison of $p_1 - t^*q_1$ and $p_2 - t^*q_2$. Setting $p = p_1 - p_2$, $q = q_1 - q_2$, we want to decide whether $p - t^*q > 0$, $p - t^*q = 0$ or $p - t^*q < 0$. Now we assume $t^* > 0$. A comparison of $p - t^*q$ can be resolved either immediately, if $pq \leq 0$, or by running Procedure COV with parameter p/q to compare p/q with t^*. We describe below Algorithm LSSP, which solves Problem LSSP within Megiddo's parametric search method.

Algorithm LSSP

Step 1: Decide whether "$t^* = 0$" or "$t^* > 0$" by running COV(0). If $t^* = 0$, then stop.

Step 2: Run L-COV(t^*) without knowing the value of t^* with all the values represented as linear functions of t^*. Each comparison of two linear functions of t^* encounterd during the computation can be evaluated (if necessary) by running Procedure COV. We can obtain $X \subseteq V$ s. t. $f(X) = t^*a(X)$ and $a(X) > 0$.

Step 3: Return $t^* := f(X)/a(X)$.

We will show that Algorithm LSSP solves Problem LSSP in strongly polynomial time after describing two procedures; Procedure COV and Procedure L-COV.

Comparison of t with t^*

Now let us consider describing Procedure COV and Procedure L-COV using (7). As a preparation for describing them, we introduce Algorithm MFM.

Let U be a finite nonempty set. For a vector $b \in \mathbf{R}^U$ and a ring family $\mathscr{D} \subseteq 2^U$, let us consider minimizing a modular function $b_{\mathscr{D}} : \mathscr{D} \to \mathbf{R}$ defined by

$$b_{\mathscr{D}}(X) = b(X) \ (X \in \mathscr{D}).$$

We assume that a directed graph representation of $\mathscr{D}$ is known. Using a result of Picard [11] the modular function minimization problem can be reduced to the minimum cut problem of a network with $O(|U|)$ vertices in $O(|U|^2)$ time. For the minimum cut problem, many combinatorial strongly polynomial time algorithms are known [1], and most of them are fully combinatorial. So we can design a fully combinatorial strongly polynomial time algorithm for modular function minimization over ring families. For example, $b_{\mathscr{D}}$ can be minimized with $O(|U|^3)$ fully combinatorial operations. For a vector $b \in \mathbf{R}^U$ and a ring family $\mathscr{D} \subseteq 2^U$ (we know a directed graph representation of $\mathscr{D}$), let Algorithm MFM be an algorithm which finds a minimizer of a modular function $b_{\mathscr{D}}$ with $\mathscr{T}_{\mathsf{MFM}}(|U|)$ fully combinatorial operations, where $\mathscr{T}_{\mathsf{MFM}}(|U|)$ is some polynomial in $|U|$. We assume $\mathscr{T}_{\mathsf{MFM}}(|U|) = O(\mathscr{T}^{\mathsf{A}}(|U|))$ and $\mathscr{T}_{\mathsf{MFM}}(|U|) = O(\mathscr{T}^{\mathsf{FC}}_{\mathsf{FC}}(|U|))$.

We describe below Procedure COV, which decide, for any given nonnegative value $t \geq 0$, whether "$t < t^*$", "$t = t^*$" or "$t > t^*$" using conditions (7) directly. In Step 1, we examine whether $ta \in \mathbf{B}(f)$ or not. In Step 2, we maximize $a_{\mathscr{D}(ta)}$ and examine whether $t = t^*$ or $t < t^*$.

Procedure COV (Comparison with the Optimal Value)

Input: A nonnegative value $t \geq 0$.

Output: A decision whether "$t < t^*$", "$t = t^*$" or "$t > t^*$".

Operation: Oracle calls for function evaluation, arithmetic operations.

Step 1: Minimize f_{ta} on 2^V by running $\mathsf{SFM}_{\mathsf{am}}(f_{ta})$.
If $\min\{f_{ta}(X) \mid X \subseteq V\} < 0$ then stop ($t > t^*$).
(If $\min\{f_{ta}(X) \mid X \subseteq V\} = 0$ we obtain $G_{\mathscr{D}_f(ta)}$.)

Step 2: Maximize $a_{\mathscr{D}(ta)} : \mathscr{D}(ta) \to \mathbf{R}$ by running $\mathsf{MFM}(-a, \mathscr{D}(ta))$.
If $\max\{a(X) \mid X \in \mathscr{D}(ta)\} = 0$ then stop ($t < t^*$).
If $\max\{a(X) \mid X \in \mathscr{D}(ta)\} > 0$ then return the maximizer of $a_{\mathscr{D}(ta)}$ and stop ($t = t^*$).

As we assumed $n = O(\gamma(f))$ (see Section 4), for any $x \in \mathbf{R}^V$, the function value of f_x can be acquired in $O(\gamma(f))$ time. So the running time of Step 1 is $O(\mathscr{T}(n, \gamma(f)))$. Thus, the total running time of Procedure COV is $O(\mathscr{T}(n, \gamma(f)) + \mathscr{T}_{\mathsf{MFM}}(n))$ = $O(\mathscr{T}(n, \gamma(f))$. Let $\mathscr{T}^{\mathsf{O}}_{\mathsf{COV}}(n) = \mathscr{T}^{\mathsf{O}}(n)$, $\mathscr{T}^{\mathsf{A}}_{\mathsf{COV}}(n) = \mathscr{T}^{\mathsf{A}}(n)$, and let $\mathscr{T}_{\mathsf{COV}}(n, \gamma(f)) = \mathscr{T}(n, \gamma(f))$.

Let Procedure L-COV be a procedure which is obtained by replacing Algorithm $\mathsf{SFM}_{\mathsf{am}}$ by Algorithm $\mathsf{FC\text{-}SFM}_{\mathsf{am}}$ in Procedure COV. For any given $t \geq 0$, once $ta(v)$ is computed for each $v \in V$, Procedure L-COV compares t to t^* with $O(\mathscr{T}^{\mathsf{O}}_{\mathsf{L\text{-}COV}}(n))$ oracle calls for function evaluation of f and $O(\mathscr{T}^{\mathsf{FC}}_{\mathsf{L\text{-}COV}}(n))$ fully combinatorial operations where $\mathscr{T}^{\mathsf{O}}_{\mathsf{L\text{-}COV}}(n) = \mathscr{T}^{\mathsf{O}}_{\mathsf{FC}}(n)$ and $\mathscr{T}^{\mathsf{FC}}_{\mathsf{L\text{-}COV}}(n) = \mathscr{T}^{\mathsf{FC}}_{\mathsf{FC}}(n)$. Let $\mathscr{T}_{\mathsf{L\text{-}COV}}(n, \gamma(f)) = \mathscr{T}_{\mathsf{FC}}(n, \gamma(f))$. And moreover if $t = t^*$, Procedure L-COV returns a subset $X \subseteq V$ such that $f(X) = t^* a(X)$ and $a(X) > 0$.

Complexity of Algorithm LSSP

The following theorem is the main result in the paper.

Theorem 3 *Algorithm* LSSP *solves Problem* LSSP *in strongly polynomial time.*

PROOF The running time of Step 1 is $O(\mathscr{T}_{\mathsf{COV}}(n, \gamma(f)))$. In Step 2, $O(\mathscr{T}^{\mathsf{FC}}_{\mathsf{L\text{-}COV}}(n))$ comparisons of linear functions of t^* are evaluated and the running time of the other part is $O(\mathscr{T}_{\mathsf{L\text{-}COV}}(n, \gamma(f)))$. So Algorithm LSSP solves Problem LSSP in strongly polynomial time. □

6 The minimum-ratio problem

Let $f : 2^V \to \mathbf{R}$ be a submodular function with $f(\emptyset) = 0$ and $f' : 2^V \to \mathbf{R}$ be a supermodular function with $f'(\emptyset) = 0$. We assume that $f(X) \geq 0$ for all $X \subseteq V$ and $f'(X) > 0$ for some $X \subseteq V$. Let us consider the following minimum ratio problem.

Problem MR : Minimize $\dfrac{f(X)}{f'(X)}$ subject to $X \subseteq V,\ f'(X) > 0$.

We give a strongly polynomial time algorithm for Problem MR. We can easily see that Problem MR is equivalent to the following maximization problem of a parameter t .

Problem P-MR : Find $t^* = \max\{t \in \mathbf{R} \mid f(X) - tf'(X) \geq 0,\ \forall X \subseteq V\}$, and find $X \subseteq V$ such that $f(X) - tf'(X) = 0$ and $f'(X) > 0$.

Problem P-MR is a generalization of Problem LSSP. The optimal value t^* of Problem P-MR is nonnegative. In the same way as the discussion in Section 2, we have the following conditions for t^* and any $t \geq 0$:

$$
\begin{aligned}
t \lhd t^* &\iff \begin{cases} \min\{f(X) - tf'(X) \mid X \subseteq V\} = 0, \\ \max\{f'(X) \mid X \in \arg\min(f - tf')\} = 0, \end{cases} \\
t = t^* &\iff \begin{cases} \min\{f(X) - tf'(X) \mid X \subseteq V\} = 0, \\ \max\{f'(X) \mid X \in \arg\min(f - tf')\} > 0, \end{cases} \\
t \rhd t^* &\iff \min\{f(X) - tf'(X) \mid X \subseteq V\} < 0.
\end{aligned}
\tag{13}
$$

For any $t \geq 0$, $f - tf'$ is a submodular function defined on 2^V. Using (13) and the same technique as the algorithm for Problem LSSP (Section 5), we can develop a strongly polynomial time algorithm for Problem P-MR and simultaneously for Problem MR. Note that a supermodular function maximization problem on a ring family $\mathscr{D}$, or a submodular function minimization problem on $\mathscr{D}$, can be reduced to a normal submodular function minimization problem if we know a directed graph representation of $\mathscr{D}$. (See Schrijver [13].)

Acknowledgments

I am grateful to Satoru Iwata for a number of useful suggestions, including the efficient method of constructing all the minimizers of a submodular function via the IFF algorithm.

References

[1] R. K. AHUJA, T. L. MAGNANTI, AND J. B. ORLIN, *Network Flows—Theory, Algorithms, and Applications*, Prentice-Hall, Englewood, NJ, 1993.

[2] R. E. BIXBY, W. H. CUNNINGHAM AND D. M. TOPKIS, The partial order of a polymatroid extreme point, *Mathematics of Operations Research*, (1985) **10**, pp. 367–378.

[3] J. EDMONDS, Submodular functions, matroids, and certain polyhedra, in R. Guy, H. Hanai, N. Sauer, and J. Schönheim, editors, *Combinatorial Structures and Their Applications*, Gordon and Breach, New York, 1970, pp. 69–87.

[4] S. FUJISHIGE, Lexicographically optimal base of a polymatroid with respect to a weight vector, *Mathematics of Operations Research* (1980) **5**, pp. 186–196.

[5] S. FUJISHIGE, *Submodular Functions and Optimization*, North-Holland, Amsterdam, 1991.

[6] S. IWATA, A fully combinatorial algorithm for submodular function minimization, *Journal of Combinatorial Theory (B)* (2002) **84**, pp. 203–212.

[7] S. IWATA, A faster scaling algorithm for minimizing submodular functions, *SIAM Journal on Computing* (2003) **32**, pp. 833–840.

[8] S. IWATA, L. FLEISCHER, AND S. FUJISHIGE, A combinatorial strongly polynomial algorithm for minimizing submodular functions, *Journal of the ACM* (2001) **48**, pp. 761–777.

[9] N. MEGIDDO, Combinatorial optimization with rational objective functions, *Mathematics of Operations Research* (1979) **4**, pp. 414–424.

[10] K. MUROTA, *Discrete Convex Analysis*, Society for Industrial and Applied Mathematics, Philadelphia, 2003.

[11] J. C. PICARD, Maximal closure of a graph and applications to combinatorial problems, *Management Science* (1976) **22**, pp. 1268–1272.

[12] T. RADZIK, Parametric flows, weighted means of cuts, and fractional combinatorial optimization, in P. M. Pardalos, ed., *Complexity in Numerical Optimization*, pp. 351–386, World Scientific, Singapore, 1993.

[13] A. SCHRIJVER, A combinatorial algorithm minimizing submodular functions in strongly polynomial time, *Journal of Combinatorial Theory (B)* (2000) **80** , pp. 346–355.

Comparing the strengths of the non-realizability certificates for oriented matroids

HIROKI NAKAYAMA

Graduate School of Information Science and Technology,
The University of Tokyo, Japan
nak-den@is.s.u-tokyo.ac.jp

SONOKO MORIYAMA

Graduate School of Information Science and Technology,
The University of Tokyo, Japan
moriso@is.s.u-tokyo.ac.jp

KOMEI FUKUDA

Swiss Federal Institue of Technology
Zurich and Lausanne, Switzerland
fukuda@ifor.math.ethz.ch

YOSHIO OKAMOTO

Department of Information and Computer Sciences,
Toyohashi University of Technology, Japan
okamotoy@ics.tut.ac.jp

Abstract: An oriented matroid is a combinatorial abstraction of a vector configuration in a real vector space. An oriented matroid is *realizable* if it is obtained from a vector configuration. Deciding realizability of an oriented matroid is known to be NP-hard. There are several classes of non-realizable oriented matroids which admit succinct certificates derived from the non-Euclidean, the non-Shannon, the non-HK, the non-HK* methods, and the method of biquadratic final polynomials. In the present paper we focus on $\mathrm{OM}(4,8)$, the class of rank-4 oriented matroids on an 8-element ground set. We report the complete classification of $\mathrm{OM}(4,8)$ with respect to the five non-realizability certificates. In particular, the result shows the superiority of the certificate using biquadratic final polynomials to the other certificates, both in uniform and non-uniform cases.

Keywords: biquadratic final polynomial, oriented matroid, realizability

1 Introduction

1.1 Background

An oriented matroid is a combinatorial abstraction of a vector configuration in a real vector space. Formally speaking, a *rank-r oriented matroid* is defined as a pair of a finite set E, called the *ground set*, and a mapping, called a *chirotope*, from the set of all ordered r-tuples of E to $\{0,+1,-1\}$ which fulfills certain conditions. An oriented matroid is *uniform* if no r-tuple is mapped to zero. A rank-r oriented matroid is *realizable* if its chirotope is obtained from some vector configuration in $\mathbb{R}^r$ in the following way. "Given a vector configuration, for each ordered r-tuple of vectors in the configuration we introduce a square matrix by arranging these vectors in columns and associate with the matrix the sign of its determinant." Otherwise, an oriented matroid is *non-realizable*.

A problem to decide realizability of a given oriented matroid is known to be NP-hard even if the rank is three [19]. Hence various conditions to give succinct certificates of realizability and non-realizability have been proposed and tried. As a sufficient condition for realizability, the existence of a solvability sequence of an oriented matroid [4] is known. On the contrary, as a sufficient condition for non-realizability, the non-Euclideanness [8], a condition from Shannon's theorem [18], a biquadratic final polynomial [16], and the non-HK and the non-HK* properties [9] using polytopes combinatorially equivalent to bounded cells of a pseudo-hyperplane arrangement, have been proposed.

The numbers of rank-r oriented matroids on an n-element ground set for small r and n are given in Table 1 [6]. (In this paper, the number of oriented matroids are considered to be the number of *non-isomorphic* oriented matroids unless stated otherwise.)

Particularly, the numbers of *uniform* oriented matroids are given in Table 2.

The following results are known about non-realizable oriented matroids.

Proposition 1 (see Björner et al. [1]) *All rank-r oriented matroids on an n-element ground set are realizable if and only if*

1. $r \leq 2$,
2. $r = 3$ *and* $n \leq 8$,
3. $r = 4$ *and* $n \leq 7$,

Table 1: The number of rank-r oriented matroids on an n-element ground set.

$n=$	2	3	4	5	6	7	8	9	10
$r=2$	1	1	1	1	1	1	1	1	1
$r=3$		1	2	4	17	143	4890	461053	95052532
$r=4$			1	3	12	206	181472	$\cdots$	$\cdots$
$r=5$				1	4	25	6029	$\cdots$	$\cdots$
$r=6$					1	5	50	508321	$\cdots$
$r=7$						1	6	91	$\cdots$
$r=8$							1	7	164

Table 2: The number of rank-r uniform oriented matroids on an n-element ground set.

$n=$	2	3	4	5	6	7	8	9	10
$r=2$	1	1	1	1	1	1	1	1	1
$r=3$		1	1	1	4	11	135	4382	312356
$r=4$			1	1	1	11	2628	$\cdots$	$\cdots$
$r=5$				1	1	1	135	$\cdots$	$\cdots$
$r=6$					1	1	1	4382	$\cdots$
$r=7$						1	1	1	$\cdots$
$r=8$							1	1	1

4. $r=5$ *and* $n\leq 8$, *or*
5. $r\geq 6$ *and* $n\leq r+2$.

In all other cases, there exist non-realizable uniform oriented matroids.

Theorem 2 (Richter [13], Bokowski & Richter-Gebert [3]) *1. There are precisely* 4382 *uniform rank-*3 *oriented matroids on a* 9*-element ground set. All these oriented matroids except* `Rin(9)` *are realizable.*

2. There are precisely 2628 *uniform rank-*4 *oriented matroids on an* 8*-element ground set. Including* `RS(8)` *and* `EFM(8)`*, precisely* 24 *of these oriented matroids are non-realizable.*

In this paper, we focus on $\mathrm{OM}(4,8)$, the class of rank-4 oriented matroids on an 8-element ground set. Although the paper including the latter of Theorem 2 is still unpublished, it is assumed that the non-realizability proof was done via biquadratic final polynomials with a floating-point arithmetic. Hence, we need to reconfirm the proof via biquadratic final polynomials with the rational arithmetic. On the other hand, other methods like the non-Euclidean method, the non-Shannon method and the non-HK* method, found only 18 out of 24 non-realizable uniform oriented matroids. For the non-uniform case, although some non-realizable oriented matroids were found by the non-Enclidean method and the non-HK* method, the exact number of non-realizable oriented matroids is not known. In fact, Richter-Gebert [15] showed that all non-realizable uniform oriented matroids found by the non-Euclidean method have biquadratic final polynomials, but for the non-uniform case, such a property fails to hold in general. Indeed, Richter-Gebert [16] constructed a rank-3 non-realizable oriented matroid on a 14-element ground set which has no biquadratic final polynomial.

In this study, we try to enumerate both uniform and non-uniform non-realizable oriented matroids in $\mathrm{OM}(4,8)$, using biquadratic final polynomials with the rational arithmetic libraries cdd [5] and lrs [11].

1.2 Organization of this paper

In Section 2, we introduce oriented matroids as chirotopes and define realizability. In Section 3, we introduce final polynomials and biquadratic final polynomials. Then we present a method to show non-realizability of oriented matroids in Section 4. The computational results of are presented in Section 5. Finally in Section 6, we conclude this paper.

2 Oriented Matroids

In this paper, we denote the number of elements of a ground set by n and the rank by r. For more details of oriented matroids, see Björner et al. [1]. In the sequel, we use a notation OM as an abbreviation of oriented matroid.

A *rank-r oriented matroid* (OM shortly)is defined as a pair (E,χ) of a finite ground set $E=\{1,\ldots,n\}$ and a mapping $\chi:E^r\rightarrow\{0,+1,-1\}$ called a chirotope, which is defined as follows.

Definition 3 *A* rank-r chirotope *on E is a mapping $\chi:E^r\rightarrow\{0,+1,-1\}$ which satisfies the following three properties.*

1. *χ is not identically zero.*
2. *χ is alternating, that is, $\chi(x_{\sigma_1},\ldots,x_{\sigma_r}) = \mathrm{sgn}(\sigma)\cdot\chi(x_1,\ldots,x_r)$ for every $(x_1,\ldots,x_r)\in E^r$ and every permutation σ on $\{1,\ldots,r\}$.*
3. *If $x_1,\ldots,x_r,y_1,\ldots,y_r \in E$ satisfy $\chi(y_i,x_2,\ldots,x_r)\cdot\chi(y_1,\ldots,y_{i-1},x_1,y_{i+1},\ldots,y_r)\geq 0$ for all $i=1,\ldots,r$, then it holds that $\chi(x_1,\ldots,x_r)\cdot\chi(y_1,\ldots,y_r)\geq 0$.*

A class of oriented matroids are obtained from vector configurations. Let $X=(x_1,\ldots,x_n)\in\mathbb{R}^{r\times n}$ be a configuration of n vectors in $\mathbb{R}^r$, which is also regarded as a $r\times n$ real matrix. For any ordered r-tuple $(i_1,\ldots,i_r)\in E^r$ we determine its sign $\chi_X(i_1,\ldots,i_r)$ as follows:

$$\chi_X(i_1,\ldots,i_r) = \mathrm{sgn}\det(x_{i_1},\ldots,x_{i_r}).$$

We can observe that the mapping $\chi_X : E^r\to\{0,+1,-1\}$ defined above is a rank-r chirotope. We call an OM $\mathscr{M}=(E,\chi)$ *realizable* if there exists a vector configuration X such that $\chi_X=\chi$; Otherwise *non-realizable*. Moreover, we call $\mathscr{M}$ *uniform* if 0 does not belong to the image of χ; Otherwise call *non-uniform*. These definitions of realizability and uniformity are equivalent to the definitions in terms of hyperplane arrangements.

The next theorem characterize chirotopes in terms of the so-called 3-term Grassmann-Plücker relations.

Theorem 4 (see Björner et al. [1]) *A mapping $\chi : E^r\to\{0,+1,-1\}$ is a chirotope if and only if it satisfies the following two properties:*

1. *χ is alternating, and the set of r-subsets $\{x_1,\ldots,x_r\}$ of E such that $\chi(x_1,\ldots,x_r)\neq 0$ is the set of bases of a matroid of rank r on E,*
2. *for any $x_1,\ldots,x_r,y_1,y_2\in E$,*
   if $\chi(y_1,x_2,\ldots,x_r)\cdot\chi(x_1,y_2,x_3,\ldots,x_r)\geq 0$ and $\chi(y_2,x_2,\ldots,x_r)\cdot\chi(y_1,x_1,x_3,\ldots,x_r)\geq 0$,
   then $\chi(x_1,\ldots,x_r)\cdot\chi(y_1,\ldots,y_r)\geq 0$.

The second property of Theorem 4 is a special case of the third one of Definition 3 where $x_i=y_i$ for $i=3,\ldots,r$.

Because of the alternating property of chirotopes, it is sufficient to set signs to the r-tuples $(i_1,i_2,\ldots i_r)$ where $i_1<i_2\ldots<i_r$.

As the most primitive algorithm for deciding realizability of an OM (E,χ), first we let a vector configuration X be $(I_r\mid x_{r+1},\ldots,x_n)$ without loss of generality, and solve a system of $\binom{n}{r}-1$ inequalities and equalities with $r\times(n-r)$ variables. The OM is realizable if this system has a solution, otherwise non-realizable. However, solving such a system is a nasty task. Therefore, several methods to derive a certificate of non-realizability have been proposed. A biquadratic final polynomial, which we adopt in this paper, is one of them.

3 Final polynomials

3.1 Definition of final polynomials

The systematic method of final polynomials was introduced by Bokowski, Richter & Sturmfels [2] which implements the naive idea above in the algebraic setting so that we can obtain a certificate of non-realizability. In the following, we denote the set $\{(i_1,\ldots,i_r)\mid 1\leq i_1<\cdots<i_r\leq n\}$ by $\Lambda(n,r)$. Given an $r\times n$ matrix $X=(x_1,\ldots,x_n)$ of $r\times n$ indeterminates, for every $(i_1,\ldots,i_r)\in\Lambda(n,r)$ we denote by a bracket $[i_1,\ldots,i_r]$ the determinant of the $r\times r$ matrix $(x_{i_1},\ldots,x_{i_r})$. A polynomial of such brackets is called a *bracket polynomial*.

Let K be an ordered field. Consider the polynomial algebra $K[\Lambda(n,r)]$ freely generated over K by the brackets $[i_1,\ldots,i_r]$, where $(i_1,\ldots,i_r)\in\Lambda(n,r)$. Given a rank-$r$ OM $\mathscr{M}=(E,r)$ on an n-element ground set, we assign to $\mathscr{M}$ the three sets $I^K_{\mathscr{M}}$, $P^K_{\mathscr{M}}$, and $N^K_{\mathscr{M}}$ of bracket polynomials as follows.

- Let I^K_χ denote the ideal in $K[\Lambda(n,r)]$ generated by $\{[i_1,\ldots,i_r]\in\Lambda(n,r)\mid\chi(i_1,\ldots,i_r)=0\}$. (If χ is uniform, then $I^K_\chi=\{0\}$.)
- Let P^K_χ denote the multiplicative semigroup with unit generated by the positive brackets $\{[i_1,\ldots,i_r]\mid\chi(i_1,\ldots,i_r)=+1\}$, the negated negative brackets $\{-[i_1,\ldots,i_r]\mid\chi(i_1,\ldots,i_r)=-1\}$, and the positive elements in K.
- Let N^K_χ denote the quadratic semiring in $K[\Lambda(n,r)]$ which is generated by P^K_χ and the set $K[\Lambda(n,r)]^2$ of all squares in $K[\Lambda(n,r)]$.

Definition 5 *A bracket polynomial $f\in K[\Lambda(n,r)]$ is called a* final polynomial *for χ if f is identically zero and $f\in I^K_\chi+P^K_\chi+N^K_\chi$.*

Theorem 6 *A chirotope χ is non-realizable if and only if there exists a final polynomial for χ with integer coefficients.*

3.2 Biquadratic final polynomials

As a consequence of Theorem 6, if there exists a final polynomial for a given chirotope, then the chirotope is guaranteed non-realizable. However, the problem to find a final polynomial is not so easy. Thus we introduce a *biquadratic final polynomial*.

A biquadratic final polynomial is a special form of a final polynomial. We can obtain a biquadratic final polynomial by solving a linear program. Hence it is found more easily than a general final polynomial. Given a rank-r oriented matroid $\mathscr{M}$ on an n-element set, we denote its realization space by $\mathscr{R}(\mathscr{M}) \subset \bigwedge_r \mathbb{R}^n$. With $\mathscr{M}$ we associate a convex polyhedron $\mathscr{P}(\mathscr{M}) \subset \bigwedge_r \mathbb{R}^n$, defined by a system of linear inequalities in $\binom{n}{r}$ variables such that

1. $\mathscr{P}(\mathscr{M}) = \emptyset$ implies $\mathscr{R}(\mathscr{M}) = \emptyset$,
2. every dual solution of linear program proving $\mathscr{P}(\mathscr{M}) = \emptyset$ can be tarnsformed into a biquadratic final polynomial for $\mathscr{M}$.

With properties of biquadratic final polynomials and Theorem 6, the following corollary is derived.

Corollary 7 *A chirotope χ is non-realizable if there exists a biquadratic final polynomial for χ with integer coefficients.*

4 Proof of non-realizability

From now on, we fix the rank of oriented matroids and the size of their ground set with 4 and 8, respectively. Then, the 3-term Grassmann-Plücker relation can be written in the form

$$[\tau_1\tau_2\lambda_1\lambda_2][\tau_1\tau_2\lambda_3\lambda_4] + [\tau_1\tau_2\lambda_1\lambda_4][\tau_1\tau_2\lambda_2\lambda_3] = [\tau_1\tau_2\lambda_1\lambda_3][\tau_1\tau_2\lambda_2\lambda_4]. \tag{1}$$

For any realization of an oriented matroid, the above equality holds. Furthermore, for any τ_1, τ_2 and $\lambda_1, \lambda_2, \lambda_3, \lambda_4$, there is always a suitable permutation $\pi \in S_4$ of the elements $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ such that all three biquadratic terms in the equality (1) are nonnegative, which we call *normalized*. From a normalized 3-term Grassmann-Plücker relation, the following two inequalities are derived:

$$\begin{aligned} [\tau_1\tau_2\lambda_1\lambda_2][\tau_1\tau_2\lambda_3\lambda_4] &< [\tau_1\tau_2\lambda_1\lambda_3][\tau_1\tau_2\lambda_2\lambda_4] \quad \text{and} \\ [\tau_1\tau_2\lambda_1\lambda_4][\tau_1\tau_2\lambda_2\lambda_3] &< [\tau_1\tau_2\lambda_1\lambda_3][\tau_1\tau_2\lambda_2\lambda_4]. \end{aligned} \tag{2}$$

Given an oriented matroid $\mathscr{M}$, let $\mathscr{P}'(\mathscr{M})$ be the solution set of the biquadratic inequality system (2), where τ_1, τ_2 and $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ range over all index sets. The realization space $\mathscr{R}(\mathscr{M})$ is a subset of $\mathscr{P}'(\mathscr{M})$. Thus "$\mathscr{P}'(\mathscr{M}) = \emptyset$" is a sufficient condition for the non-realizability of $\mathscr{M}$. In order to test the solvability of the inequality system (2), we now apply the bijective transformation $\log : [i_1, \ldots, i_r] \mapsto \log[i_1, \ldots, i_r]$ and define $\mathscr{P}(\mathscr{M}) := \log(\mathscr{P}'(\mathscr{M}))$. The convex polyhedron $\mathscr{P}(\mathscr{M})$ is the solution set of the following linear system of inequalities:

$$\begin{aligned} [\tau_1\tau_2\lambda_1\lambda_2] + [\tau_1\tau_2\lambda_3\lambda_4] &< [\tau_1\tau_2\lambda_1\lambda_3] + [\tau_1\tau_2\lambda_2\lambda_4] \quad \text{and} \\ [\tau_1\tau_2\lambda_1\lambda_4] + [\tau_1\tau_2\lambda_2\lambda_3] &< [\tau_1\tau_2\lambda_1\lambda_3] + [\tau_1\tau_2\lambda_2\lambda_4]. \end{aligned} \tag{3}$$

If $\mathscr{P}(\mathscr{M})$ is empty, then we obtain a dual solution of this linear system, i.e. a positive integer linear combination of the left hand sides of (3) which equals the same linear combination of the right hand sides, resulting in the contradiction $0 < 0$.

By exponentiation we get a product of the left hand sides in (2) equal to the same product of the right hand sides. Using the syzygies (1), we transform this equality to an explicit representation, which is a biquadratic final polynomial for $\mathscr{M}$.

In the case of non-uniform OMs, for some τ_1, τ_2 and $\lambda_1, \lambda_2, \lambda_3, \lambda_4$, the relation (1) has a biquadratic term whose value is 0. By an appropriate permutation of $\lambda_1, \lambda_2, \lambda_3, \lambda_4$, either of two left-hand side terms of the relation (1) becomes zero and a right-hand side term becomes positive, unless the values of all three terms become zero.

Now we assume that the bracket $[\tau_1\tau_2\lambda_1\lambda_4]$ is zero in (1). Then the following equality

$$[\tau_1\tau_2\lambda_1\lambda_2][\tau_1\tau_2\lambda_3\lambda_4] = [\tau_1\tau_2\lambda_1\lambda_3][\tau_1\tau_2\lambda_2\lambda_4] \tag{4}$$

is derived. By taking the logarithms of (4), we obtain the equality

$$[\tau_1\tau_2\lambda_1\lambda_2] + [\tau_1\tau_2\lambda_3\lambda_4] = [\tau_1\tau_2\lambda_1\lambda_3] + [\tau_1\tau_2\lambda_2\lambda_4] \tag{5}$$

as a constraint of $\mathscr{P}(\mathscr{M})$.

We denote the set of inequalities (3) and equalities (5) by $\mathscr{A}_\chi$ and $\mathscr{B}_\chi$, respectively. Then $\mathscr{P}(\mathscr{M})$ is the feasible space of the system whose constraints are all relations in $\mathscr{A}_\chi$ and $\mathscr{B}_\chi$. Note that if the oriented matroid is uniform, then $\mathscr{B}_\chi = \emptyset$.

Table 3: Chirotope of `RS(8)`.

1234+	1235+	1236+	1237+	1238+	1245+	1246+	1247−	1248+	1256+
1257−	1258−	1267−	1268−	1278+	1345−	1346+	1347−	1348−	1356+
1357+	1358+	1367−	1368−	1378+	1456+	1457−	1458−	1467+	1468−
1478+	1567+	1568+	1578−	1678+	2345−	2346−	2347−	2348−	2356+
2357+	2358+	2367−	2368+	2378+	2456+	2457−	2458+	2467−	2468+
2478+	2567+	2568+	2578−	2678−	3456+	3457−	3458−	3467−	3468−
3478+	3567+	3568+	3578−	3678−	4567−	4568+	4578−	4678−	5678+

Example 8 *We show the non-relizability of the uniform rank-*4 *oriented matroid* `RS(8)`*.* `RS(8)` *is defined by the chirotope in Table 3. In this table,* $i_1 \cdots i_r +\ (-)$ *means* $\chi(i_1,\ldots,i_r) = +1\ (-1)$*, respectively.*

The following six equalities are obtained from this chirotope. Their indices are sorted, where underlined indices mean τ_1, τ_2*, otherwise* $\lambda_1, \lambda_2, \lambda_3, \lambda_4$*.*

$$\begin{aligned}
[\underline{12}34][\underline{12}56] + [\underline{12}45][\underline{12}36] &= [\underline{12}35][\underline{12}46],\\
[\underline{123}4][\underline{13}57] + [\underline{13}74][\underline{12}35] &= [\underline{13}54][\underline{12}37],\\
[\underline{12}34][\underline{14}85] + [\underline{13}54][\underline{12}48] &= [\underline{1}384][\underline{12}45],\\
[\underline{12}34][\underline{23}76] + [\underline{23}64][\underline{12}37] &= [\underline{23}74][\underline{12}36],\\
[\underline{12}34][\underline{24}68] + [\underline{23}84][\underline{12}46] &= [\underline{2}364][\underline{12}48],\\
[12\underline{34}][\underline{34}78] + [2\underline{37}4][\underline{13}84] &= [2\underline{38}4][\underline{13}74].
\end{aligned} \tag{6}$$

This implies the six inequalities for the set $\mathscr{P}'(\texttt{RS(8)})$*. Taking logarithms on both sides, we obtain the following system of linear inequalities:*

$$\begin{aligned}
[1245] + [1236] &< [1235] + [1246],\\
[1374] + [1235] &< [1354] + [1237],\\
[1354] + [1248] &< [1384] + [1245],\\
[2364] + [1237] &< [2374] + [1236],\\
[2384] + [1246] &< [2364] + [1248],\\
[2374] + [1384] &< [2384] + [1374].
\end{aligned} \tag{7}$$

In this system, the left hand sides equal to the right hand sides. Thus it is inconsistent. This means $\mathscr{P}'(\texttt{RS(8)}) = \emptyset$*, which implies the non-realizability of* `RS(8)`*.*

From the equalities (6), we obtain the following biquadratic final polynomial:

$$\begin{aligned}
&([1234][1256] + [1245][1236]) \cdot ([1234][1357] + [1374][1235]) \cdot ([1234][1485] + [1354][1248]) \cdot\\
&([1234][2376] + [2364][1237]) \cdot ([1234][2468] + [2384][1246]) \cdot ([1234][3478] + [2374][1384])\\
&\quad - [1235][1246][1354][1237][1384][1245][2374][1236][2364][1248][2384][1374]
\end{aligned} \tag{8}$$

which is zero in $K[\Lambda(8,4)]$*. Expanding the product of (8), a sum of* 63 *terms contained in* $P^{\mathbb{R}}_{\texttt{RS(8)}}$ *is left. By the definition of a final polynomial, the bracket polynomial (8) is a final polynomial.*

5 Results

In our computation, we use the $\mathrm{OM}(4,8)$ catalog, constructed by Finschi [6]. The software packages cdd [5] and lrs [11] are used as LP-solvers with exact rational arithmetic for testing the feasibilities of the linear programs defined by $\mathscr{A}_\chi$ and $\mathscr{B}_\chi$.

5.1 Case of uniform oriented matroids

Bokowski & Richter-Gebert [3] showed that there exist 2628 uniform OMs and 24 of them are non-realizable. By our computation, we found all of the 24 non-realizable OMs, whose indices in the catalog are #2–#5,#7–#21 and #114–#118, where #*i*–#*j* means all indices from *i* to *j*. This result reconfirms the correctness of the theorem by Bokowski and Richter-Gebert.

5.2 Case of non-uniform oriented matroids

There exist 178844 non-uniform OMs. Although the exact number of non-realizable OMs has not been known, some of them were obtained by the non-Euclidean method and the non-HK* method [9]. The following results were shown in that paper.

- By the non-Euclidean method, 3444 non-uniform OMs were found non-realizable.
- By the non-HK* method, 1364 non-uniform OMs were found non-realizable, all of which were indeed found by non-Euclidean method.
- By the non-Shannon method, no non-uniform OMs was found to be non-realizable.

By our computation with biquadratic final polynomials, 3944 non-realizable OMs were obtained, which include all of the 3444 non-realizable OMs found by the non-Euclidean method. The relation among the certificates using a biquadratic final polynomial, the non-Shannon method, the non-Euclidean method and the non-HK* method is given in Figure 1.

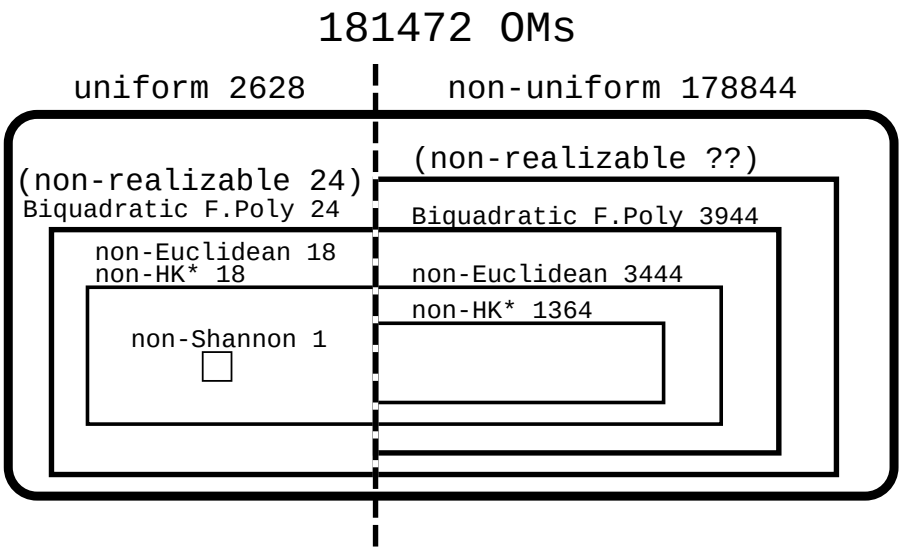


Figure 1: Relations of the number of non-realizable OMs.

6 Concluding Remarks

Using biquadratic final polynomials, for both of the uniform and the non-uniform cases, we found all non-realizable OMs that were already found by the non-Euclidean method. Additionally, we obtained new 500 non-uniform non-realizable OMs, which have never been found non-realizable. This suggests the superiority of biquadratic final polynomials well. Furthermore, recently we have shown that every non-Euclidean OM admits a biquadratic final polynomial [10]. This extends the same result by Richter-Gebert [15] for the uniform case.

For $\mathrm{OM}(4,8)$, all non-realizable OMs found by the non-HK* method are also found by the non-Euclidean method. The problem whether the same property also holds for general $\mathrm{OM}(r,n)$ is still open.

Next, in the non-uniform case, non-realizable OMs which cannot be decided with biquadratic final polynomials may exist. Therefore we need to decide non-realizability of OMs by certificates which give a sufficient condition of realizability, i.e. a solvability sequence.

Finally, for the class of rank-3 OMs on an 9-element set, the number of non-uniform non-realizable OMs is not known. Because the non-Euclidean and the non-HK* properties cannot be applied to rank-3 OMs, enumerating the non-realizable OMs by biquadratic final polynomials is meaningful.

References

[1] A. BJÖRNER, M. LAS VERGNAS, B. STURMFELS, N. WHITE AND G. M. ZIEGLER, Oriented Matroids, *Encyclopedia of Mathematics, Cambridge University Press* (1993) **46**

[2] J. BOKOWSKI, J. RICHTER AND B. STURMFELS, Nonrealizability proofs in computational geometry, *Discrete and Comput. Geometry* (1990) **5**, 333–350.

[3] J. BOKOWSKI AND J. RICHTER-GEBERT, On the classification of non-realizable oriented matroids, Part I: Generation, *Preprint 1283, TH Darmstadt* (1990), 17 pages.

[4] J. BOKOWSKI AND B. STURMFELS, On the coordinatization of oriented matroids, *Discrete Comput. Geometry* (1986) **1**, 293–306.

[5] cdd, *http://www.cs.mcgill.ca/~fukuda/soft/cdd_home/cdd.html*

[6] L. FINSCHI, A graph theoretical approach for reconstruction and generation of oriented matroids, *Ph.D. Thesis, Swiss Federal Institute of Technology Zürich* (2001), 179 pages.

[7] J. FOLKMAN AND J. LAWRENCE, Oriented matroids, *J. Combinatorial Theory, Ser. B* (1978) **25**, 199–236.

[8] K. FUKUDA, Oriented matroid programming, *Ph.D. Thesis, University of Waterloo* (1982), 223 pages.

[9] K. FUKUDA, S. MORIYAMA AND Y. OKAMOTO, Non-LP orientations, nonlinear shellings and non-representable oriented matroids, *Technical Group on Computation (COMP)* (2004)

[10] K. FUKUDA, H. NAKAYAMA AND S. MORIYAMA, Every non-Euclidean oriented matroid admits a biquadratic final polynomial, *in preparation*

[11] lrs, *http://cgm.cs.mcgill.ca/~avis/C/lrs.html*

[12] A. MANDEL, Topology of oriented matroids, *Ph.D. Thesis, University of Waterloo* (1982), 333 pages.

[13] J. RICHTER, Kombinatorische Realisierbarkeitskriterien für orientierte Matroide, *Mitteilungen Math. Seminar Gießen* (1989) **194**, 112 pages.

[14] J. RICHTER-GEBERT, New construction methods for oriented matroids, *Dissertation, KTH Stockholm* (1992), 102 pages.

[15] J. RICHTER-GEBERT, Euclideaness and final polynomials in oriented matroid theory, *Combinatorica* (1993) **13**, 259–268.

[16] J. RICHTER-GEBERT, Two interesting oriented matroids, *Documenta Mathematica* (1996) **1**, 137–148.

[17] J.-P. ROUDNEFF AND B. STURMFELS, Simplicial cells in arrangements and mutations of oriented matroids, *Geometriae Dedicata* (1988) **27**, 153–170.

[18] R. W. SHANNON, Simplicial cells in arrangements of hyperplanes, *Geometriae Dedicata*, (1979) **8**, 179–187.

[19] P. SHOR, Stretchability of pseudolines is NP-hard, *DIMACS Series in Discrete Math. and Theoretical Computer Science* (1991) **4**, 531–554.

An $O(n^3)$ Time Algorithm for Obtaining the Minimum Vertex Ranking Spanning Tree on Permutation Graphs

SHIN-ICHI NAKAYAMA

Department of Mathematical Sciences
The University of Tokushima
Tokushima 770-8502, JAPAN
shin@ias.tokushima-u.ac.jp

SHIGERU MASUYAMA

Department of Knowledge-Based Information Engineering
and
Intelligent Sensing System Research Center
Toyohashi University of Technology
Toyohashi 441-8580, JAPAN
masuyama@tutkie.tut.ac.jp

Abstract: The minimum vertex ranking spanning tree problem is to find a spanning tree of G whose vertex ranking is minimum. This paper proposes an $O(n^3)$ time algorithm for solving the minimum vertex ranking spanning tree problem on a permutation graph.

Keywords: algorithm, graph theory, vertex ranking, spanning tree, permutation graph

1 Introduction

Consider a simple connected undirected graph $G = (V,E)$. A vertex ranking of G is labeling r from the vertices of G to the positive integers such that for each path between any two vertices u and v, $u \neq v$, with $r(u) = r(v)$, there exists at least one vertex w on the path with $r(w) > r(u) = r(v)$. The value $r(v)$ of a vertex v is called the rank of vertex v. A vertex ranking r of G is minimum if the largest rank k assigned by r is the smallest among all rankings of G. Such rank k is called the vertex ranking number of G, denoted by $\chi(G)$. The vertex ranking problem is to find a minimum ranking of given graph G. The vertex ranking problem has interesting applications to e.g., communication network design, planning efficient assembly of products in manufacturing systems and VLSI layout design.

As for the complexity, this problem is NP-complete even when restricted to cobipartite graphs [15] and bipartite graphs [2], and a number of polynomial time algorithms for this problem have been developed on several subclasses of graphs. Much work has been done in finding the minimum vertex ranking of a tree; a linear time algorithm for trees is proposed in [19]. The problem is trivial on split graphs and is solvable in linear time on cographs [20]. As regards interval graphs, Deogun et al has given an $O(n^3)$ time algorithm recently[5], which outperforms the previously known $O(n^4)$ time algorithm [1] where n is the number of vertices. They also presented $O(n^6)$ time algorithms on permutation graphs and on trapezoid graphs, respectively, and showed that a polynomial time algorithm on d-trapezoid graphs exists [5]. Moreover, a polynomial time algorithm on graphs with treewidth at most k was developed [3].

The problem described above is the ranking to vertices, while a ranking to edges is similarly defined as follows. An edge ranking of G is labeling r_e from the edges of G to the positive integers such that for each path between any two edges e_u and e_v, $e_u \neq e_v$, with $r(e_u) = r(e_v)$, there exists at least one edge e_w on the path with $r(e_w) > r(e_u) = r(e_v)$. The value $r(e_v)$ of an edge e_v is called the rank of edge e_v. An edge ranking of G is minimum if the largest rank k assigned is the smallest among all rankings of G. Such rank k is called the edge ranking number of G, denoted by $\chi_e(G)$. The edge ranking problem is to find a minimum edge ranking of given graph G. Before the proof of this problem to be NP-complete was given, an $O(n^3)$ time algorithm for trees was known [21]. By now, a linear time algorithm for trees is shown in [11]. Recently, it has finally been shown that this problem on general graphs is NP-complete [10].

Makino et al. introduced a minimum edge ranking spanning tree problem related to the minimum edge ranking problem but is essentially different [12]. The minimum edge ranking spanning tree problem is to find a spanning tree of G whose edge ranking is minimum. They proved that this problem is NP-complete and presented an approximation algorithm for this problem. This problem has interesting applications, e.g., scheduling the parallel assembly of a multipart product from its components and the relational database [12].

In this paper, we consider the vertex version of this problem, i.e., the minimum vertex ranking spanning tree problem. The minimum vertex ranking spanning tree problem is to find a spanning tree of G whose vertex ranking is minimum. We recently proved that this problem is NP-complete [13] and developed an $O(n^3)$ time algorithm when an input graph is an interval graph [14]. We show that, in this paper, an $O(n^3)$ time algorithm for the minimum vertex ranking spanning tree exists when an input graph is a permutation graph. It is interesting that, for permutation graphs, the minimum vertex ranking spanning tree problem is solved in $O(n^3)$ time, although the time complexity of known algorithm for the minimum vertex ranking problem is $O(n^6)$.

2 Permutation graph

Let $V = \{v_1, v_2, \cdots, v_n\}$ and $\pi = [\,\pi[1],\pi[2],\cdots,\pi[n]]$ be a permutation on V. We construct an undirected graph $G(\pi) = (V,E)$ such that $\{v_i, v_j\} \in E$ iff $(i-j)(\pi^{-1}[i] - \pi^{-1}[j]) < 0$, where $\pi^{-1}[i]$ denotes the position of vertex v_i in π. An undirected graph G is a *permutation graph* if there exists a π such that G is isomorphic to $G(\pi)$ [6]. Pnueli et al.[16] describe an $O(n^3)$ algorithm for testing if a given undirected graph is a permutation graph. This result was improved to $O(n^2)$ by Spinrad [18], whose algorithm produces the corresponding permutation if the graph is a permutation graph.

A permutation graph can also be visualized by its corresponding *permutation diagram*. The permutation diagram consists of two horizontal parallel channels, named the top channel and the bottom channel, respectively. Put the index $1, 2, \cdots, n$ of vertices on the top channel, in the order from left to right, and put the index of vertex in $\pi[1], \pi[2], \cdots, \pi[n]$ on the bottom channel in the same way. Finally, for each i, draw a straight line joining the two i's, one on the top channel and the other on the bottom channel, respectively [6]. The index number i of vertex v_i is same as that of the corresponding line l_i. Note that line l_i intersects line l_j in the diagram iff l_i and l_j appear in the reversed order in π. That is, lines l_i and l_j intersect iff vertices v_i and v_j of the corresponding permutation graph are adjacent. The reader is encouraged to draw the permutation diagram for given π's since they are sometimes quite useful in visualizing the properties of the original permutation graphs.

Permutation graphs are a useful discrete mathematical structure for modeling practical problems [6]. Moreover, permutation graphs construct an important class of perfect graphs and many problems that are NP-complete on arbitrary graphs are shown to admit polynomial time algorithms on this class [6][7][9][17].

3 The basic idea of the algorithm

The basic idea of our algorithm is as follows: First find a shortest path P^* of G between a certain pair of vertices, then construct a spanning tree with the minimum vertex ranking by joining each vertex $v \in V - V(P^*)$ to a vertex of P^* using an edge of G, based on the fact, to be proven in this paper, that, for permutation graphs, $v \in V - V(P^*)$ not included in P^* is adjacent to some vertex on P^*. For preparation, we introduce a known result on the vertex ranking of paths.

Lemma 1 (17) *The ranking $\chi(P)$ of a path $P = x_1, x_2, \cdots, x_n$ is $\lfloor \log n \rfloor^* + 1$.* □

In the following, we explain what kind of shortest path P^* is selected and how each vertex in $V - V(P^*)$ should be joined to some vertex on P^* (with an edge) in order to construct a minimum vertex ranking spanning tree.

A shortest path to be selected in our algorithm is one between a vertex corresponding to the rightmost line and that corresponding to the leftmost line on the permutation diagram. Namely, denoting the vertex corresponding to a line whose position is 1 and n on the top (resp. bottom) channel by v_1^t (resp. v_1^b) and v_n^t (resp. v_n^b), respectively, we select a path whose length is shortest among four shortest paths from v_1^t to v_n^t, from v_1^t to v_n^b, from v_1^b to v_n^t and from v_1^b to v_n^b. Note here that the length of each edge is 1. Let P^* be the selected shortest path. On a spanning tree T of permutation graph G, as the length of a diameter of T is equal to or greater than that of P^*, for the minimum ranking $\chi(P^*)$ of P^* on G, $\chi(P^*) \leq \chi(T)$.

Our algorithm first finds the shortest path P^* described above and then constructs a spanning tree by joining each vertex in $V - V(P^*)$ to a vertex on P^* using an edge of G. Now, we show that, for permutation graph G, each vertex in $V - V(P^*)$ is adjacent to some vertices on P^*.

Lemma 2 *Let a shortest path selected by the above process be $P^* = v_1, v_2, \ldots, v_l$. For permutation graphs $G = (V,E)$, each vertex in $V - V(P^*)$ is adjacent to some vertex on P^* in G.*

The proof is omitted due to the space limit.

We now consider how each vertex in $V - V(P^*)$ should be joined to a vertex on P^* in order to construct a minimum vertex ranking spanning tree. Let a vertex set $V - V(P^*)$ be V'. By lemma 2, each vertex $v' \in V'$ is adjacent to a vertex on P^*. Then, our algorithm finds a path P^* of G and joins each vertex in V' to a vertex on P^* using an edge of G.

By lemma 2, the relation of connections by edges between $v' \in V'$ and vertices on P^* are classified into the following three cases.

(1) $v' \in V'$ is adjacent to only one vertex on P^*.

(2) $v' \in V'$ is adjacent to two consecutive vertices v_j, v_{j+1} on P^* or three consecutive vertices v_j, v_{j+1}, v_{j+2} on P^*.

(3) $v' \in V'$ is not adjacent to consecutive vertices on P^* but adjacent to two vertices v_j, v_{j+2} having one skip on P^*.

Note: As P^* is the shortest path, $v' \in V'$ is adjacent to neither more than three consecutive vertices on P^* in the case (2) nor two vertices that have more than one skip on P^* in the case (3).

Let V_1' denote a subset of V' that contains vertices in V' each of which is adjacent to only one vertex on P^*, let V_2' denote a subset of V' that contains vertices in V' each of which is adjacent to two or three consecutive vertices on P^* and let V_3' denote a subset of V' that contains vertices in V' each of which is adjacent to two vertices v_j, v_{j+2} having one skip on P^*.

*Throughout this paper, log denotes $\log_2$.

We first consider $v'' \in V_2'$ adjacent to two or three consecutive vertices on P^*. As for $v'' \in V_2'$ adjacent to at least two vertices on P^*, we can select a vertex on P^* to be joined to v'' in order to construct a spanning tree. Then, let us consider to which vertex of P^* $v'' \in V_2'$ should be joined. After finding the minimum vertex ranking of P^*, for consecutive vertices v_i, v_{i+1} on P^*, either $r(v_i) > r(v_{i+1})$ or $r(v_i) < r(v_{i+1})$ holds by the definition of the vertex ranking. As $v'' \in V_2'$ is adjacent to at least two consecutive vertices on P^*, v'' is adjacent to a vertex v on P^* whose rank is at least 2. Then, joining v'' to v and assigning rank 1 to v'', we can construct a spanning tree T with $\chi(T) = \chi(P^*)$, without changing the rank of vertices on P^*.

Next, we consider $v' \in V_1'$ adjacent to only one vertex on P^* and $v' \in V_3'$ adjacent to two vertices having one skip on P^*. In this case, depending on the result of vertex ranking of P^*, v' may be adjacent to a vertex v on P^* with rank 1. Then, when selecting the edge (v', v) in order to construct a spanning tree, we must modify the rank of v for satisfying the vertex ranking. Moreover, G may not have a spanning tree T such that $\chi(T) = \chi(P^*)$. Fortunately, for permutation graphs, the upper bound on $\chi(T)$ is determined as shown in the following lemma.

Lemma 3 *For permutation graph G, the ranking $\chi(T)$ of a spanning tree T satisfies the following inequality: $\chi(T) \leq \chi(P^*) + 1$.*

The proof is omitted due to the space limit.

By lemma 3, the ranking of spanning tree $\chi(T)$ is either $\chi(P^*)$ or $\chi(P^*)+1$. Therefore, our algorithm tries to construct a spanning tree T with rank $\chi(P^*)$. As a result, if we can not construct a spanning tree T with rank $\chi(P^*)$, we construct a spanning tree T with rank $\chi(P^*)+1$.

After assigning ranks to vertices on P^* with a minimum ranking, if the rank of a vertex v_j on P^* adjacent to $v' \in V_1'$ is 1, a spanning tree satisfying the ranking condition can not be constructed by joining v' to v_j by this assignment. Similarly, if each rank of vertices v_j, v_{j+2} on P^* adjacent to $v' \in V_3'$ is 1, a spanning tree satisfying the ranking condition can not be constructed by joining v' to v_j or v_{j+2}. In these cases, we may get a spanning tree satisfying the ranking condition either by changing the rank of v_j (or v_{j+2}) to become greater than 1 or by joining v' to a vertex in V'. Then, our algorithm classifies each vertex $v' \in V_1' \cup V_3'$ according to the connection between v' and vertices on P^* and selects an edge to join v'.

For illustration, we now consider the minimum vertex ranking of trees. A tree is divided into more than one components $T_1, T_2, \cdots, T_l$ by removing a vertex v other than a leaf. A path from a vertex of T_i to a vertex of T_j $(i \neq j)$ obviously go through v. Then, by assigning the largest rank $\max\{\chi(T_1), \chi(T_2), \cdots, \chi(T_l)\} + 1$ to v, the condition of vertex ranking of the tree is satisfied. However, the resulting vertex ranking is not necessarily the minimum one. Based on this observation, we develop an algorithm as sketched below. We assign the largest rank $\chi(P^*)(= \lfloor \log |P^*| \rfloor + 1)$ to a vertex v_i on $P^*(= v_1, \cdots, v_l)$. (Here $|P^*|$ denotes the number of vertices on P^*.) Then, we pay attention to two subgraphs $G^1_{v_i}$, $G^2_{v_i}$ of G such that $G^1_{v_i}$ is induced by path $v_1, v_2, \cdots, v_{i-1}$ and vertices in V' $(= V - V(P^*))$ adjacent to $v_1, v_2, \cdots, v_{i-1}$ and $G^2_{v_i}$ is induced by path $v_{i+1}, v_{i+2}, \cdots, v_l$ and vertices in V' adjacent to $v_{i+1}, v_{i+2}, \cdots, v_l$, respectively. As will be described in detail later, the case when $G^1_{v_i}$ and $G^2_{v_i}$ share a common vertex v^* of V' needs to be treated separately. Then, we find a minimum vertex ranking spanning tree T_1 in $G^1_{v_i}$ and T_2 in $G^2_{v_i}$, respectively. If both of minimum vertex rankings of T_1 and T_2 are not greater than $\lfloor \log |P^*| \rfloor$, a spanning tree with ranking $\chi(P^*)(= \lfloor \log |P^*| \rfloor + 1)$ can be constructed by joining T_1, T_2 via v_i. Even when a spanning tree with ranking $\lfloor \log |P^*| \rfloor + 1$ can not be constructed, by using some other vertex on P^* instead of v_i, a spanning tree with ranking $\lfloor \log |P^*| \rfloor + 1$ may be constructed. Hence, we check whether each of $G^1_{v_i}$ and $G^2_{v_i}$ has a spanning tree with ranking at most $\lfloor \log |P^*| \rfloor$ for each v_i, $i = 2, \cdots, l-1$, with the largest rank. For this purpose, we use the dynamic programming. We check whether a subgraph induced by k consecutive vertices $v_j, \cdots, v_{j+k}$ on P^* , $(j = 1, \cdots, l,\ k = 0, \cdots, l-j)$, and vertices in V' adjacent to $v_j, \cdots, v_{j+k}$ has a spanning tree with ranking $\lfloor \log |P^*_{v_j v_{j+k}}| \rfloor + 1$. (Note that $P^*_{v_i v_j}$ denotes a subpath $v_i, \cdots, v_j$ on P^*.) Therefore, we now consider a spanning tree on a subgraph induced by consecutive vertices $v_j, \cdots, v_{j+k}$ on P^* and vertices in V' adjacent to $v_j, \cdots, v_{j+k}$.

Let define some terms needed to explain the algorithm in the following. As for consecutive vertices $v_j, \cdots, v_k$ on P^*, a subgraph of G induced by $v_j, \cdots, v_k$ and vertices in V' adjacent to $v_j, \cdots, v_k$ is called a *subgraph regarding* $v_j, \cdots, v_k$ and denoted by $G[v_j, v_k]$. For $G[v_j, v_k]$, if we can construct a spanning tree such that each rank of vertices in $G[v_j, v_k]$ is at most $\lfloor \log |P^*_{v_j v_k}| \rfloor + 1 (= \chi(P^*_{v_j v_k}))$, we say that $G[v_j, v_k]$ is *minimum-rankable*.

Note: For a subgraph $G[v_i, v_i]$ regarding one consecutive sequence of vertices, as we can always construct spanning tree T with ranking at most 2 by assigning rank 2 to v_i and rank 1 to vertices adjacent to v_i. Then, we say that each subgraph $G[v_i, v_i]$ regarding one consecutive vertices is minimum-rankable.

Using these terms, what we are going to do in the dynamic programming is as follows: Let subpaths of P^* selected in the first step be $P^*_1 = v_i, \cdots, v_{j-1}$ and $P^*_2 = v_{j+1}, \cdots, v_k$, respectively. We check whether $G[v_i, v_{j-1}]$, $G[v_{j+1}, v_k]$ are minimum-rankable or not. If each of $G[v_i, v_{j-1}]$, $G[v_{j+1}, v_k]$ is minimum-rankable, the subgraph $G[v_i, v_k]$ regarding $v_i, \cdots, v_k$ is minimum-rankable by assigning $\lfloor \log |P^*_{v_j v_k}| \rfloor + 1$ to v_j. However, when $G[v_i, v_{j-1}]$ and $G[v_{j+1}, v_k]$ share a common vertex, even if these are not minimum-rankable, we need to check some conditions, to be described later, because $G[v_i, v_k]$ may

be minimum-rankable. If either of $G[v_i, v_{j-1}]$ or $G[v_{j+1}, v_k]$ is not minimum-rankable and do not share a common vertex, $G[v_i, v_k]$ is not minimum-rankable.

As mentioned above, for constructing a minimum vertex ranking spanning tree, our algorithm first check whether subgraphs $G[v_i, v_{i+1}]$, for $i = 1, \cdots, l-1$, regarding two consecutive vertices on P^* is minimum-rankable, and then check whether subgraphs $G[v_i, v_{i+2}]$, for $i = 1, \cdots, l-2$, regarding three consecutive vertices on P^* is minimum-rankable. Concerning subgraphs $G[v_i, v_{i+k}]$, $k \leq 3$, regarding more than three consecutive vertices on P^*, using known information about subgraphs, we check whether $G[v_i, v_{i+k}]$ is minimum-rankable by using the dynamic programming.

We then consider the way to check whether a subgraph regarding consecutive vertices is minimum-rankable. We classify each vertex $v' \in V_1' \cup V_3'$ according to the connection between v' and vertices on P^* and investigate whether each case is minimum-rankable or not.

3.1 Subgraph regarding two consecutive vertices

We consider whether a subgraph $G[v_j, v_{j+1}]$ regarding two consecutive vertices v_j, v_{j+1} on P^* is minimum-rankable or not. That is, we examine whether we can construct a spanning tree such that each rank of vertices in $G[v_j, v_{j+1}]$ is at most $\lfloor \log |P^*_{v_j v_{j+1}}| \rfloor + 1$ $(= \chi(P^*_{v_j v_{j+1}}) = 2)$. We classify the cases by connection between $v' \in V_1' \cup V_3'$ and a vertex of P^*. However, we omit detailed discussions due to limitations of space.

Case 1: $v' \in V_1'$ is adjacent to only one vertex on P^*.

Case 1-1: If each of v_j and v_{j+1} is adjacent to a vertex $v \in V_1'$ whose degree is 1, $G[v_j, v_{j+1}]$ is not minimum-rankable. However, if either of v_j and v_{j+1} is adjacent to a vertex $v \in V_1'$ whose degree is 1, $G[v_j, v_{j+1}]$ is minimum-rankable.

Case 1-2: v_j is adjacent to $v_j' \in V_1'$ whose degree is at least 2, or v_{j+1} is adjacent to $v_{j+1}' \in V_1'$ whose degree is at least 2.

Case 1-2-1: If v_j, v_{j+1} are adjacent to $v_j', v_{j+1}' \in V_1'$, respectively, and v_j' and v_{j+1}' are only adjacent to each other, $G[v_j, v_{j+1}]$ is not minimum-rankable.

Case 1-2-2: If v_j, v_{j+1} are adjacent to $v_j', v_{j+1}' \in V_1'$, respectively, and v_j' and v_{j+1}' are adjacent to a vertex $v'' \in V_2'$, $G[v_j, v_{j+1}]$ is minimum-rankable.

Case 1-2-3: If v_j, v_{j+1} are adjacent to $v_j', v_{j+1}' \in V_1'$, respectively, and v_{j+1}' is adjacent to a vertex $v^* \in V_2'$ adjacent to v_{j+2}', then $G[v_j, v_{j+1}]$ is minimum-rankable. (By symmetry, the case where v_j' is adjacent to a vertex $v^* \in V_2'$ adjacent to v_{j-1}', can be discussed in a similar way.)

Case 2: $v''' \in V_3'$ is adjacent to not consecutive vertices on P^* but two vertices v_j, v_{j+2} having one skip on P^*.

Case 2-1: If $v''' \in V_3'$ is adjacent to only two vertices v_j and v_{j+2}, then $G[v_j, v_{j+1}]$ is minimum-rankable. (By symmetry, the case where $v''' \in V_3'$ is only adjacent to v_{j-1} and v_{j+1}, can be discussed in a similar way.)

Case 2-2: If $v''' \in V_3'$ is adjacent to only two vertices v_{j+1} and v_{j+3}, then $G[v_j, v_{j+1}]$ is minimum-rankable.

Case 3: Vertices in V_1' and in V_3' both exist in $G[v_j, v_{j+1}]$.

Case 3-1: A vertex in V_1' and a vertex in V_3' share a common vertex on P^*.

Case 3-1-1: $v''' \in V_3'$ is adjacent to v_j and v_{j+2} on P^* and either v_j or v_{j+1} is adjacent to vertices in V_1'.

Lemma 4 *For permutation graph G, when $v''' \in V_3'$ is adjacent to v_j and v_{j+2} on P^*, if $v_{j+1}' \in V_1'$ is adjacent to v_{j+1}, v_{j+1}' is also adjacent to v'''.*

The proof is omitted due to the space limit.

By the above lemma, we need to consider two cases where v_{j+1} is not adjacent to a vertex in V_1' and v_{j+1} is adjacent to both a vertex $v_{j+1}' \in V_1'$ and v'''.

Case 3-1-2: If $v''' \in V_3'$ is adjacent to v_{j+1} and v_{j+3} on P^* and v_{j+1} is adjacent to a vertex in V_1', $G[v_j, v_{j+1}]$ is minimum-rankable.

Case 3-2: Vertices in V_1' and those in V_3' do not share a common vertex: $v''' \in V_3'$ is adjacent to v_{j+1}, v_{j+3} on P^*, the degree of v''' is 2 and a vertex in V_1' with degree 1 is adjacent to v_j.

In the following, we call a vertex like v''' a *suspension vertex* and if $G[v_j, v_{j+1}]$ has a suspension vertex, we say that $G[v_j, v_{j+1}]$ is not minimum-rankable by a suspension vertex.

3.2 Subgraph regarding three consecutive vertices

We consider whether a subgraph $G[v_j, v_{j+2}]$ regarding three consecutive vertices v_j, v_{j+1}, v_{j+2} on P^* is minimum-rankable or not. We classify the cases with respect to connection between $v' \in V_1' \cup V_3'$ and a vertex of P^*. However, we omit detailed discussions due to limitations of space.

Case 4: $v' \in V_1'$ is adjacent to only one vertex on P^*.

Case 4-1: If v_j is an articulation (1-cut) vertex in G and $v_j' \in V_1'$ adjacent to v_j is not adjacent to a vertex adjacent to v_{j-1}, then $G[v_j, v_{j+2}]$ is not minimum-rankable. Note that an *articulation vertex* is a vertex of a connected graph whose deletion

disconnects the graph. (By symmetry, the case where v_{j+2} is an articulation vertex and $v'_{j+2} \in V_1'$ adjacent to v_{j+2} is not adjacent to a vertex adjacent to v_{j+3}, can be discussed in a similar way.)

Case 4-2: If $v'_j \in V_1'$ adjacent to v_j is adjacent to v'_{j-1} adjacent to v_{j-1}, $G[v_j, v_{j+2}]$ is minimum-rankable. (By symmetry, the case where $v'_{j+2} \in V_1'$ adjacent to v_{j+2} is adjacent to v'_{j+3} adjacent to v_{j+3}, can be discussed in a similar way.)

Case 4-3: v_j and v_{j+2} are not articulation vertices: Whereas $v'_j \in V_1'$ is adjacent to v_j, if $v^*_j \in V_2' \cup V_3'$ that is adjacent to v_{j-1} and v_{j+1} exists, $G[v_j, v_{j+2}]$ is minimum-rankable. (As for v_{j+2}, we can discuss in a similar way.)

Case 5: $v''' \in V_3'$ is adjacent to not consecutive vertices on P^* but adjacent to two vertices v_j, v_{j+2} having one skip on P^*.

Case 5-1: If $v''' \in V_3'$ is adjacent to only two vertices v_j and v_{j+2} on P^* and v_j and v_{j+2} are articulation vertices, then $G[v_j, v_{j+2}]$ is not minimum-rankable.

Case 5-2: If $v''' \in V_3'$ is adjacent to two vertices v_j, v_{j+2} and $v'_{j+3} \in V'$ is adjacent to v_{j+3}, then $G[v_j, v_{j+2}]$ is minimum-rankable.

Case 5-3: If $v''' \in V_3'$ is adjacent to both v_j and v_{j+2} on P^* and $v^* \in V_2' \cup V_3'$ that is adjacent to both v_{j+1} and v_{j+3} exits, then $G[v_j, v_{j+2}]$ is minimum-rankable.

Case 5-4: If $v''' \in V_3'$ is adjacent to two vertices v_{j+1}, v_{j+3} on P^*, then $G[v_j, v_{j+2}]$ is minimum-rankable. (By symmetry, the case where $v''' \in V_3'$ is adjacent to two vertices v_{j-1}, v_{j+1} on P^*, can be discussed in a similar way.)

Case 5-5: If $v''' \in V_3'$ is adjacent to only two vertices v_{j+2} and v_{j+4} on P^* and v_{j+2} and v_{j+4} are articulation vertices, then $G[v_j, v_{j+2}]$ is not minimum-rankable by a suspension vertex. (By symmetry, the case where $v''' \in V_3'$ is adjacent to only two vertices v_j and v_{j-2} on P^*, and the fact that v_j and v_{j-2} are articulation vertices can be discussed in a similar way.)

Case 5-6: If $v''' \in V_3'$ is adjacent to v_{j+2}, v_{j+4} on P^* and is adjacent to $v'_{j+3} \in V'$ adjacent to v_{j+3}, then $G[v_j, v_{j+2}]$ is minimum-rankable.

Case 5-7: If $v''' \in V_3'$ is adjacent to v_{j+2}, v_{j+4} on P^* and $v^* \in V_2' \cup V_3'$ that is adjacent to v_{j+1} and v_{j+3} exits, then $G[v_j, v_{j+2}]$ is minimum-rankable.

Case 6: Both vertices in V_1' and in V_3' exists in $G[v_j, v_{j+2}]$.

Case 6-1: If a vertex in V_3' is adjacent to two vertices v_j, v_{j+2}, $v'_j \in V_1'$ (resp. $v'_{j+2} \in V_1'$) is adjacent to v_j (resp. v_{j+2}) and v_j (resp. v_{j+2}) is articulation vertices, then $G[v_j, v_{j+2}]$ is not minimum-rankable.

Case 6-2: A vertex $v''' \in V_3'$ is adjacent to two vertices v_j, v_{j+2}, $v'_{j+2} \in V_1'$ (resp. $v'_j \in V_1'$) is adjacent to v_{j+2} (resp. v_j) and a vertex $v'_{j+3} \in V'$ adjacent to v_{j+3} is adjacent to v''' or v'_{j+2}. (By symmetry, the case where a vertex $v'_{j-1} \in V'$ adjacent to v_{j-1} is adjacent to v''' or v'_j, can be treated in a similar way.)

Case 6-2-1: If $v'_{j+3} \in V'$ is adjacent to $v''' \in V_3'$, then $G[v_j, v_{j+2}]$ is minimum-rankable.

Case 6-2-2: If $v'_{j+3} \in V'$ is adjacent to $v'_{j+2} \in V_1'$ but not adjacent to v''', then $G[v_j, v_{j+2}]$ is not minimum-rankable.

Case 6-3: A vertex $v''' \in V_3'$ is adjacent to two vertices v_j, v_{j+2}, $v'_{j+2} \in V_1'$ (resp. $v'_j \in V_1'$) is adjacent to v_{j+2} (resp. v_j) and a vertex $v^* \in V_2' \cup V_3'$ is adjacent to v_{j+1} and v_{j+3}. In this case, $G[v_j, v_{j+2}]$ is minimum-rankable.

Case 6-4: If a vertex $v''' \in V_3'$ is adjacent to two vertices v_{j+2}, v_{j+4} and $v'_{j+2} \in V_1'$ adjacent to v_{j+2} is adjacent to v''', then $G[v_j, v_{j+2}]$ is not minimum-rankable.

Case 6-5: If a vertex $v''' \in V_3'$ is adjacent to two vertices v_{j+2}, v_{j+4}, $v'_{j+2} \in V_1'$ adjacent to v_{j+2} is adjacent v''' and v''' is adjacent to $v'_{j+3} \in V'$ adjacent to v_{j+3}, then $G[v_j, v_{j+2}]$ is minimum-rankable.

Case 6-6: If a vertex $v''' \in V_3'$ is adjacent to two vertices v_{j+2}, v_{j+4}, $v'_{j+2} \in V_1'$ adjacent to v_{j+2} is adjacent v''' and a vertex $v^* \in V_2' \cup V_3'$ is adjacent to v_{j+1} and v_{j+3}. In this case, $G[v_j, v_{j+2}]$ is minimum-rankable.

4 An algorithm for solving the minimum vertex ranking spanning tree problem

Following the above explanations given in sections 3.1 and 3.2, we can check whether spanning trees with rank 2 can be constructed in subgraphs regarding two consecutive vertices and subgraphs regarding three consecutive vertices.

Our algorithm is described as follows. In the algorithm, we use an array $R[v_i, v_j]$, for i, $j = 1, \cdots, l$. If $G[v_i, v_j]$ is minimum-rankable, 'OK' is assigned to $R[v_i, v_j]$.

Procedure Find_Minimum_Ranking_Spanning_Tree
begin

Step 1. Find a path $P^*(= v_1, v_2, \ldots, v_l)$ whose length is shortest among four shortest paths from v_1^t to v_n^t, from v_1^t to v_n^b, from v_1^b to v_n^t and from v_1^b to v_n^b.
Step 2. For $V - V(P^*)$, find vertex sets $V_1^{'}$, $V_2^{'}$ and $V_3^{'}$.
Step 3. If every vertex in $V - V(P^*)$ is in $V_2^{'}$, a spanning tree with $\chi(T) = \lfloor \log |P^*| \rfloor + 1$ can be constructed. Stop.
Step 4. For $i, j = 1$ to l, $R[v_i, v_j] \leftarrow$ 'null'
For $k = 1$ to l, $R[v_k, v_k] \leftarrow$ 'OK'.
Step 5. For subgraph $G[v_j, v_{j+1}]$ regarding two consecutive vertices v_j, v_{j+1}, $j = 1, \cdots, l-1$, on P^*, check whether $G[v_j, v_{j+1}]$ is minimum-rankable. If $G[v_j, v_{j+1}]$ is minimum-rankable, $R[v_j, v_{j+1}] \leftarrow$ 'OK'.
Step 6. For subgraph $G[v_j, v_{j+2}]$ regarding three consecutive vertices v_j, v_{j+1}, v_{j+2}, $j = 1, \cdots, l-2$, on P^*, check whether $G[v_j, v_{j+2}]$ is minimum-rankable. If $G[v_j, v_{j+2}]$ is minimum-rankable, $R[v_j, v_{j+2}] \leftarrow$ 'OK'.
Step 7. For the pairs of vertices on P^* whose distance is greater than 3, sort $R[v_i, v_k]$'s in increasing order according to the value of the distance between v_i and v_k.
Step 8. Compute $R[v_i, v_k]$'s in the order of step 7 as follows :
for each j such that $i < j < k$ *do*
begin
If $G[v_i, v_{j-1}]$ is not minimum-rankable by a suspension vertex $v^{'''}$, we check whether the rank of v_{j+1} adjacent to $v^{'''}$ in $G[v_{j+1}, v_k]$ is 1. If the rank of v_{j+1} is not 1, as a suspension vertex $v^{'''}$ can be joined to v_{j+1} in $G[v_{j+1}, v_k]$ for $G[v_i, v_{j-1}]$ to be minimum-rankable, then $R[v_i, v_{j-1}] \leftarrow$ 'OK'.
If $G[v_{j+1}, v_k]$ is not minimum-rankable by a suspension vertex $v^{'''}$, we check whether the rank of v_{j-1} adjacent to $v^{'''}$ in $G[v_i, v_{j-1}]$ is 1. If the rank of v_{j-1} is not 1, as a suspension vertex $v^{'''}$ can be joined to v_{j-1} in $G[v_i, v_{j-1}]$ for $G[v_{j+1}, v_k]$ to be minimum-rankable, then $R[v_{j+1}, v_k] \leftarrow$ 'OK'.
If the value of $R[v_i, v_{j-1}]$ is 'OK', that of $R[v_{j+1}, v_k]$ is 'OK' and $\max\{\lfloor \log |P^*_{v_i v_{j-1}}| \rfloor + 1, \lfloor \log |P^*_{v_{j+1} v_k}| \rfloor + 1\} \leq \lfloor \log |P^*_{v_i v_k}| \rfloor$ then, $R[v_i, v_k] \leftarrow$ 'OK'.
end
Step 9. If the value of $R[1, l]$ is 'OK', a spanning tree with $\chi(T) = \lfloor \log |P^*| \rfloor + 1$ can be constructed. Otherwise, a spanning tree with $\chi(T) = \lfloor \log |P^*| \rfloor + 1 + 1 (= \chi(P^*) + 1)$ can be constructed.
end.

Theorem 5 *Procedure Find_Minimum_Ranking_Spanning_Tree solves the minimum vertex ranking spanning tree problem in $O(n^3)$ time.*

The proof is omitted due to the space limit.

5 Conclusion

In this paper, we proposed an $O(n^3)$ time algorithm for solving the minimum vertex ranking spanning tree problem, when an input graph is a permutation graph. It is interesting that, for permutation graphs, the minimum vertex ranking spanning tree problem is solved in $O(n^3)$ time, although the time complexity of known algorithm for the minimum vertex ranking problem is $O(n^6)$.

Acknowledgement. This work was partially supported by the Grant-in-Aid "Scientic Research on Priority Areas: New Horizons in Computing" (B)(2)16092213 and by The 21st Century COE Program "Intelligent Human Sensing" both from the Ministry of Education, Culture, Sports and Science of Japan.

References

[1] B. Aspvall, P. Heggernes : "Finding minimum height elimination tree for interval graphs in polynomial time", *BIT*, **34**, pp. 484-509, 1994.

[2] H. L. Bodlaender, J. S. Deogun, K. Jansen, T. Kloks, D. Kratsch, H. Müller, Z. Tuza, "Rankings of graphs", *Lecture Notes in Computer Science*, vol. 903, Springer, Berlin, pp.292-304, 1996.

[3] H. Bodlaender, J. R. Gilbert, H. Hafsteinsson, T. Kloks, D. Kratsch, H. Müller, Z. Tuza, "Rankings of Graphs", *SIAM J. Discrete Math*, **11** pp.168-181, 1998.

[4] J.A.Bondy and U.S.R.Murty : "Graph Theory with Applications", North-Holland, 1976.

[5] J. S. Deogun, T. Kloks, D. Kratsch, H. Müller, "On the vertex ranking problem for trapezoid, circular-arc and other graphs", *Discrete Appl. Math.*, **98**, pp.39-63, 1999.

[6] M. C. Golumbic, "Algorithmic graph theory and perfect graphs", *Academic Press*, New York, 1980.

[7] O. H. Ibarra, Q. Zheng, “Some Efficient Algorithms for Permutation Graphs”, *J. Algorithms*, **16**, pp.453-469, 1994.

[8] D.E.Knuth, “The Art of Computer Programming, Vol. III: Sorting and Searching”, *Addison-Wesley*, Reading Mass., 1973.

[9] H. Kim, “Finding a Maximum Independent set in a permutation graph”, *Information Process. Lett.*, **36**, pp.19-23, 1990.

[10] T. W. Lam, F. L. Yue, “Edge ranking of graphs is hard”, *Discrete Appl. Math.*. **85**, pp.71-86, 1998.

[11] T. W. Lam, F. L. Yue, “Optimal edge ranking of trees in linear time”, Proceeding of the ninth Annual ACM-SIAM Symposium on Discrete Algorithms, pp.436-445, 1998.

[12] K. Makino, Y. Uno, T, Ibaraki, “On minimum edge ranking spanning trees”, *J. Algorithms*, **38**, pp.411-437, 2001.

[13] K. Miyata, S. Masuyama and S. Nakayama, “Computational complexity of the minimum vertex ranking spanning tree problem” (in Japanese), *Technical Report of IEICE*, COMP2003-55, pp. 9-4, 2003.

[14] S. Nakayama, S. Masuyama, “An algorithm for solving the minimum vertex ranking spanning tree problem on interval graphs”, *IEICE Trans, Fundamentals*, Vol.E86-A, No.5, pp.1019-1026, 2003.

[15] A. Pothen, “The complexity of optimal elimination trees”, Technical Report CS-88-13, Pennsylvania State University, USA, 1988.

[16] A. Pnueli, A. Lempel, S. Even, ”Transitive orientation of graphs and identification of permutation graphs”, *Can. J. Math.*, **23**, pp.160-175, 1971.

[17] C. Rhee, Y. D. Liang, ”Finding a maximum matching in a permutation graph”, *ACTA Informatica*, **32**, pp.779-792, 1995.

[18] J. Spinrad, ”On comparability and permutation graphs”, *SIAM J. Computing*, **14**, pp.658-670, 1985.

[19] A. A. Schäffer, “Optimal node ranking of trees in linear time”, *Information Process. Lett.*, **33**, pp.91-96, 1989/1990.

[20] P. Scheffler, “Node ranking and searching on graphs (Abstract)”, in: U. Faigle, C.Hoede (Eds.), Third Twente Workshop on Graphs and Combinatorial Optimization, Memorandum No. 1132, Faculty of Applied Mathematics, University of Twente, The Netherlands, 1993.

[21] P. de la Torre, R. Greenlaw, A. A. Schäffer, “Optimal edge ranking of trees in polynomial time, *Algorithmica*, **13**, pp.592-618, 1995.

Counting the Independent Sets of a Chordal Graph in Linear Time

YOSHIO OKAMOTO*

Department of Information and Computer Sciences,
Toyohashi University of Technology,
1-1 Hibarigaoka, Tempaku-cho, Toyohashi,
Aichi, 441-8580, Japan
okamotoy@ics.tut.ac.jp

TAKEAKI UNO

National Institute of Informatics,
2-1-2 Hitotsubashi, Chiyoda-ku,
Tokyo, 101-8430, Japan
uno@nii.jp

RYUHEI UEHARA

Department of Information Processing,
School of Information Science, JAIST,
1-1 Asahidai, Nomi, Ishikawa,
923-1292, Japan
uehara@jaist.ac.jp

Abstract: We study some counting problems for chordal graphs, especially concerning independent sets (or stable sets). We first provide the following efficient algorithms for a chordal graph: (1) a linear-time algorithm for counting the number of independent sets; (2) a linear-time algorithm for counting the number of maximum independent sets; (3) a polynomial-time algorithm for counting the number of independent sets of a fixed size. On the other hand, we prove that the following problems for a chordal graph are #P-complete: (1) counting the number of maximal independent sets; (2) counting the number of minimum maximal independent sets.

Keywords: chordal graph, counting, graph algorithm, independent set

1 Introduction

How can we cope with computationally hard graph problems? There are several possible answers, and one of them is to utilize the special graph structures arising from a particular context. This has been motivating the study of special graph classes in algorithmic graph theory [1, 7].

Recently, counting and enumeration of some specified sets in a graph have been widely investigated, e.g., in the data mining area. In general, however, from the graph-theoretic point of view, those problems are hard even if input graphs are quite restricted. For example, counting the number of independent sets in a planar bipartite graph of maximum degree 4 is #P-complete [11]. Therefore, we wonder what kind of graph structures makes counting and enumeration problems tractable.

In this paper, we consider chordal graphs. A *chordal graph* is a graph in which every cycle of length at least four has a chord. It is known that many graph optimization problems can be solved in polynomial time for chordal graphs [6, 4]. However, relatively fewer problems have been studied for enumeration and counting in chordal graphs [5, 8, 2, 3].

This paper investigates problems concerning independent sets in a chordal graph. We first give the following efficient algorithms for a chordal graph; (1) a linear-time algorithm to count the number of independent sets, (2) a linear-time algorithm to count the number of maximum independent sets, and (3) a polynomial-time algorithm to count the number of independent sets of a given size. The running time of the third algorithm is linear when the size is constant. Let us note that the time complexity here refers to the arithmetic operations, not to the bit operations.

The basic idea of these efficient algorithms is to invoke a clique tree associated with a chordal graph and perform the dynamic programming on the clique tree. A clique tree is based on the characterization of a chordal graph as an intersection graph of subtrees of a tree. Since a clique tree can be constructed in linear time and the structure of a clique tree is simple, this approach leads to simple and efficient algorithms for the problems above.

Along the same idea, we can also enumerate all independent sets, all maximum independent sets, and all independent sets of constant size in a chordal graph in $O(1)$ amortized time per output.

On the other hand, we show that the following counting problems are #P-complete: (1) counting the number of maximal independent sets in a chordal graph, and (2) counting the number of minimum maximal independent sets in a chordal graph.

*The work was done while he was at Institute of Theoretical Computer Science, ETH Zurich, under the support of the Berlin/Zurich Graduate Program "Combinatorics, Geometry, and Computation," financed by ETH Zurich and the German Science Foundation

Due to the space constraint, this version omits proofs of some statements and the whole part on enumeration algorithms. The complete version should appear elsewhere.

2 Preliminaries

In this article, we assume that the reader has a moderate familiarity with graph theory. This section aims at fixing the notation and introducing a chordal graph and concepts around that. Let $G = (V,E)$ be a graph, which we always assume to be simple and finite. The *neighborhood* of a vertex v in a graph $G = (V,E)$ is the set $N_G(v) = \{u \in V \mid \{u,v\} \in E\}$. For a vertex subset U of V, we denote by $N_G(U)$ the set $\{v \in V \mid v \in N(u) \text{ for some } u \in U\}$. If no confusion can arise we will omit the subscript G. We denote the closed neighborhood $N(v) \cup \{v\}$ by $N[v]$. A vertex set I is an *independent set* of G if any pair of vertices in I is not an edge of G, and a vertex set C is a *clique* if every pair of vertices in C is an edge of G. An edge which joins two vertices of a cycle but is not itself an edge of the cycle is a *chord* of the cycle. A graph is *chordal* if each cycle of length at least four has a chord.

To a chordal graph $G = (V,E)$, we associate a tree T, called a *clique tree* of G, satisfying the following three properties. (A) The nodes of T are the maximal cliques of G. (B) For every vertex v of G, the subgraph T_v of T induced by the maximal cliques containing v is a tree. (In the literature, the condition (A) is sometimes weakened as each node is a vertex subset of G.) It is well known that a graph is chordal if and only if it has a clique tree, and in such a case a clique tree can be constructed in linear time. Some details are explained in books [1, 10]. The following property is important in the running time analysis of our algorithms.

Lemma 1 Let $G = (V,E)$ be a chordal graph, and denote by $\mathscr{K}$ the family of maximal cliques of G. Then, it holds that $\sum_{K \in \mathscr{K}} |K| = O(|V| + |E|)$.

3 Linear-Time Algorithm to Count the Independent Sets

In this section, we describe an algorithm for counting the number of independent sets in a chordal graph. The basic idea of our algorithm is to divide the input graph into subgraphs induced by subtrees of the clique tree. Any two of these subtrees share a vertex of a clique if they are disjoint in the clique tree. This property is very powerful for counting the number of independent sets since any independent set can include at most one vertex of a clique. We compute the number of independent sets including each vertex of the clique, or no vertex of the clique by using the recursions.

First, we introduce some notations and state some lemmas. Given a chordal graph $G = (V,E)$, we construct a clique tree T of G. We now pick any node in T, regard the node as the root of T, and denote it by K_r. This is what we call a *rooted clique tree*. For a maximal clique K in a chordal graph G and a rooted clique tree T of G, a maximal clique K' in G is a *descendant* of K (with respect to T) if K' is a descendant of K in T. For convenience, we consider K itself a descendant of K as well, and when no confusion arises we omit saying "with respect to T." Let $\mathrm{PRT}(K)$ be the parent of K in T. For convenience, we define $\mathrm{PRT}(K_r)$ as the empty set. We denote by $T(K)$ the subtree of T rooted at K. Let $G(K)$ denote the subgraph of G induced by the vertices included in at least one node of $T(K)$. Observe that $G(K)$ is a chordal graph of which $T(K)$ is a clique tree.

For a graph G, let $\mathscr{IS}(G)$ be the family of independent sets in G. For a vertex v of G, let $\mathscr{IS}(G,v)$ be the family of independent sets in G including v. For a vertex set U, let $\overline{\mathscr{IS}}(G,U)$ be the family of independent sets in G including no vertex of U.

Lemma 2 Let G be a chordal graph and T be a rooted clique tree of G. Choose a maximal clique K of G, and let $K_1,\ldots,K_\ell$ be the children of K in T. (If K is a leaf of the clique tree, we set $\ell := 0$.) Furthermore let $v \in K$ and $S \subseteq V(G(K))$. Then, $S \in \mathscr{IS}(G(K),v)$ if and only if S is represented by the union of $\{v\}$ and $S_1,\ldots,S_\ell$ such that $S_i \in \mathscr{IS}(G(K_i),v)$ if v belongs to K_i, and $S_i \in \overline{\mathscr{IS}}(G(K_i),K \cap K_i)$ otherwise. Furthermore, such a representation is unique.

PROOF: Assume that $S \in \mathscr{IS}(G(K),v)$. Let $S_i := S \cap G(K_i)$ for every $i \in \{1,\ldots,\ell\}$. Then, S includes the union of $\{v\}$ and $S_1,\ldots,S_\ell$. Let us show the converse inclusion. Choose an arbitrary vertex $x \in S$. If $x = v$, then x is certainly included in the union of $\{v\}$ and $S_1,\ldots,S_\ell$. Otherwise, we have $x \in V(G(K)) \setminus K$. Since $V(G(K)) = K \cup \bigcup_{i=1}^{\ell} V(G(K_i))$, the vertex x belongs to S_i for some $i \in \{1,\ldots,\ell\}$. Therefore, S is included in the union of $\{v\}$ and $S_1,\ldots,S_\ell$. Now, we need to show that for every $i \in \{1,\ldots,\ell\}$ the set S_i satisfies the property required in the lemma. Fix $i \in \{1,\ldots,\ell\}$. If v belongs to K_i, then S_i belongs to $\mathscr{IS}(G(K_i),v)$ since v also belongs to S. If $v \notin K_i$, then S_i belongs to $\overline{\mathscr{IS}}(G(K_i),K \cap K_i)$ since v is adjacent to all vertices of $K \cap K_i$. Thus the required property is satisfied. This completes the proof of the only-if part.

Next, we prove the if part. Assume that S is the union of $\{v\}$ and $S_1,\ldots,S_\ell$ satisfying that $S_i \in \mathscr{IS}(G(K_i),v)$ if $v \in K_i$, and $S_i \in \overline{\mathscr{IS}}(G(K_i),K \cap K_i)$ otherwise. When $v \in K_i$, since v is adjacent to all vertices of $K \setminus \{v\}$, every vertex in $S_i \setminus \{v\}$ belongs to $V(G(K_i)) \setminus K$. When $v \notin K_i$, by the definition of $\overline{\mathscr{IS}}(G(K_i),K \cap K_i)$, every vertex in $S_i \setminus \{v\}$ belongs to $V(G(K_i)) \setminus K$. Therefore, for each $i \in \{1,\ldots,\ell\}$ it holds that $S_i \setminus \{v\} \subseteq V(G(K_i)) \setminus K$. This implies that $S \setminus \{v\} \subseteq V(G(K)) \setminus K$. Now, we show that for every $i,j \in \{1,\ldots,\ell\}$, $i \neq j$, $(S_i \setminus \{v\}) \cup (S_j \setminus \{v\})$ is independent. To show that, suppose not. Since S_i and S_j

are independent, there must be an edge $\{x,y\} \in E$ such that $x \in S_i \setminus \{v\}$ and $y \in S_j \setminus \{v\}$. Since $\{x,y\}$ is an edge of G, it is included in some maximal clique of G. Then, the definition of a clique tree implies that x or y belongs to K. Without loss of generality, assume that x belongs to K. (Remember that $x \in S_i \setminus \{v\}$.) If $S_i \in \mathscr{I}\mathscr{S}(G(K_i),v)$, then $S_i \cap K \supseteq \{v,x\}$. This is a contradiction to S_i being independent. If $S_i \in \overline{\mathscr{I}\mathscr{S}}(G(K_i),K \cap K_i)$, then S_i cannot contain any vertex of K, particularly x. This is also a contradiction. Thus the claim is verified, and it implies that $S \setminus \{v\}$ is an independent set of $G(K)$. Together with the observation that no vertex of $G(K_i) \setminus K$ is adjacent to v if $v \notin K_i$, this further implies that S is an independent set of $G(K)$. Since $v \in S$, this shows that $S \in \mathscr{I}\mathscr{S}(G(K),v)$.

To show the uniqueness, suppose that S is the union of $\{v\},S_1,\ldots,S_\ell$ and also the union of $\{v\},S'_1,\ldots,S'_\ell$ such that there exists i with $S_i \neq S'_i$. Without loss of generality assume that $S_i \setminus S'_i \neq \emptyset$. Choose a vertex $u \in S_i \setminus S'_i$, where $u \neq v$. Then, there must exist $j \neq i$ with $u \in S'_j$. Hence, there exists a node $L \in T(K_i)$ such that $u \in L$ and a node $L' \in T(K_j)$ such that $u \in L'$. (Note that L and L' are maximal cliques of G.) Then, by Property (B) in the definition of a clique tree, the nodes on the path connecting L and L' in T contain u. In particular we have $u \in K$. Therefore, u and v belong to the clique K and at the same time they belong to the independent set S. This is a contradiction. □

By a close inspection of the proof above, we can observe that for every $i,j \in \{1,\ldots,\ell\}$, $i \neq j$, it holds that $V(G(K_i)) \setminus K$ is disjoint from $V(G(K_j)) \setminus K$. This property gives a nice decomposition of the problem into several independent parts, and enables us to perform the dynamic programming on a clique tree.

By similar discussion as above, we obtain the following lemma.

Lemma 3 Let G be a chordal graph and T be a rooted clique tree of G. Choose a maximal clique K of G, and let $K_1,\ldots,K_\ell$ be the children of K in T. (If K is a leaf of the clique tree, we set $\ell := 0$.) 1. We have $S \in \overline{\mathscr{I}\mathscr{S}}(G(K),K)$ if and only if S is the union of $S_1,\ldots,S_l$ such that $S_i \in \overline{\mathscr{I}\mathscr{S}}(G(K_i),K \cap K_i)$. Furthermore, such a representation is unique. 2. For each $i = 1,\ldots,\ell$, we have $S_i \in \overline{\mathscr{I}\mathscr{S}}(G(K_i),K \cap K_i)$ if and only if S_i belongs either to $\mathscr{I}\mathscr{S}(G(K_i),v)$ for some $v \in K_i \setminus K$ or to $\overline{\mathscr{I}\mathscr{S}}(G(K_i),K_i)$. Furthermore, S_i belongs to exactly one of them.

From these lemmas, we have the following recursive equations for $\mathscr{I}\mathscr{S}$.

Equations 1 Let G be a chordal graph and T be a rooted clique tree of G. For a maximal clique K of G which is not a leaf of the clique tree, let $K_1,\ldots,K_\ell$ be the children of K in T. Furthermore, let $v \in K$. Then, the following identities hold. (We remind that $\dot\cup$ means "disjoint union.")

$$\mathscr{I}\mathscr{S}(G(K)) = \overline{\mathscr{I}\mathscr{S}}(G(K),K) \dot\cup \dot{\bigcup_{v \in K}} \mathscr{I}\mathscr{S}(G(K),v);$$

$$\mathscr{I}\mathscr{S}(G(K),v) = \{S \cup \{v\} \mid S = \bigcup_{i=1}^{\ell} S_i, S_i \in \left\{ \begin{array}{ll} \mathscr{I}\mathscr{S}(G(K_i),v) & \text{if } v \in K_i \\ \overline{\mathscr{I}\mathscr{S}}(G(K_i),K \cap K_i) & \text{otherwise} \end{array} \right\};$$

$$\overline{\mathscr{I}\mathscr{S}}(G(K),K) = \{S \mid S = \bigcup_{i=1}^{\ell} S_i, S_i \in \overline{\mathscr{I}\mathscr{S}}(G(K_i),K \cap K_i)\};$$

$$\overline{\mathscr{I}\mathscr{S}}(G(K_i),K \cap K_i) = \overline{\mathscr{I}\mathscr{S}}(G(K_i),K_i) \dot\cup \dot{\bigcup_{u \in K_i \setminus K}} \mathscr{I}\mathscr{S}(G(K_i),u) \quad \text{for each } i \in \{1,\ldots,\ell\}.$$

These equations lead us to the following algorithm to count the number of independent sets in a chordal graph. For a maximal clique K of a chordal graph G, we denote the set of children of K in a rooted clique tree of G by $\mathrm{CHD}(K)$.

Algorithm 2: #IndSets

Input : A chordal graph $G = (V,E)$;
Output: The number of independent sets in G;
1 construct a rooted clique tree T of G with root K_r;
2 call #IndSetsIter(K_r);
3 **return** $\left|\overline{\mathscr{I}\mathscr{S}}(G,K_r)\right| + \sum_{v \in K_r} |\mathscr{I}\mathscr{S}(G(K_r),v)|$.

Procedure #IndSetsIter(K)

Input : A maximal clique K of the chordal graph G;
if *K is a leaf of T* **then**
  set $\left|\overline{\mathscr{I}\mathscr{S}}(G(K),K)\right| := 0$ and $|\mathscr{I}\mathscr{S}(K,v)| := 1$ for each $v \in K$;
else
  foreach *child K' of K* **do** call #IndSetsIter(K');
  foreach *child K' of K* **do** compute $\left|\overline{\mathscr{I}\mathscr{S}}(G(K'),K\cap K')\right|$ by $\left|\overline{\mathscr{I}\mathscr{S}}(G(K'),K')\right| + \sum_{u\in K'\setminus K} |\mathscr{I}\mathscr{S}(G(K'),u)|$;
  compute $\left|\overline{\mathscr{I}\mathscr{S}}(G(K),K)\right|$ by $\prod_{K'\in \mathrm{CHD}(K)} \left|\overline{\mathscr{I}\mathscr{S}}(G(K'),K\cap K')\right|$;
  foreach $v \in K$ **do** compute $|\mathscr{I}\mathscr{S}(G(K),v)|$ by
  $\prod_{K'\in \mathrm{CHD}(K),v\in K'} |\mathscr{I}\mathscr{S}(G(K'),v)| \times \prod_{K'\in \mathrm{CHD}(K),v\notin K'} \left|\overline{\mathscr{I}\mathscr{S}}(G(K'),K\cap K')\right|$.

Theorem 4 The algorithm #IndSets outputs the number of independent sets in a chordal graph $G=(V,E)$ in $O(|V|+|E|)$ time.

PROOF: From Equations 1, the algorithm correctly computes the number of independent sets in G. Let us consider the computation time $t(K)$ taken by a call to #IndSetsIter(K). The overall running time of #IndSets is $t(K_r)+O(|K_r|)$. Steps 7 and 8 take $O(t(K'))$ and $O(|K'|)$ time for each $K' \in \mathrm{CHD}(K)$ respectively. Step 9 can be done in $O(|\mathrm{CHD}(K)|)$. Next, we analyze the computation time for Step 10. Since $\left|\overline{\mathscr{I}\mathscr{S}}(G(K),K)\right| = \prod_{K'\in \mathrm{CHD}(K)} \left|\overline{\mathscr{I}\mathscr{S}}(G(K'),K\cap K')\right|$, we have that $\prod_{K'\in \mathrm{CHD}(K),v\in K'} |\mathscr{I}\mathscr{S}(G(K'),v)| \times \prod_{K'\in \mathrm{CHD}(K),v\notin K'} \left|\overline{\mathscr{I}\mathscr{S}}(G(K'),K\cap K')\right| = \left|\overline{\mathscr{I}\mathscr{S}}(G(K),K)\right| \times \frac{\prod_{K'\in \mathrm{CHD}(K),v\in K'} |\mathscr{I}\mathscr{S}(G(K'),v)|}{\prod_{K'\in \mathrm{CHD}(K),v\in K'} |\overline{\mathscr{I}\mathscr{S}}(G(K'),K\cap K')|}$, and we use this equation in Step 10. Then $|\mathscr{I}\mathscr{S}(G(K),v)|$ can be computed in $O(|\{K'\in \mathrm{CHD}(K) \mid v\in K'\}|)$ time, thus Step 10 can be done in $O(\sum_{v\in K}|\{K'\in \mathrm{CHD}(K)\mid v\in K'\}|)$ time. Therefore, the accumulated time taken by a call to #IndSetsIter(K_r) is $\sum_{K'\in \mathrm{CHD}(K_r)}(O(t(K'))+O(|K'|))+O(|\mathrm{CHD}(K_r)|)+O(\sum_{v\in K_r}|\{K'\in \mathrm{CHD}(K_r)\mid v\in K'\}|)$. By expanding $t(K')$ inside the sum, we can see that this is at most $O(\sum_{K\in\mathscr{K}}(|K|+\sum_{v\in K}|\{K'\in \mathrm{CHD}(K)\mid v\in K'\}|))$, where $\mathscr{K}$ denotes the set of nodes in the clique tree, i.e., the family of maximal cliques of G. By Lemma 1, we have $\sum_{K\in\mathscr{K}}|K| = O(|V|+|E|)$. Furthermore, it follows that $\sum_{K\in\mathscr{K}}\sum_{v\in K}|\{K'\in \mathrm{CHD}(K)\mid v\in K'\}| = \sum_{v\in V}|\{K'\in\mathscr{K}\mid v\in K'\}| = \sum_{K\in\mathscr{K}}|K| = O(|V|+|E|)$ again by Lemma 1. Hence, the overall running time is $O(|V|+|E|)$. □

4 Linear-Time Algorithm to Count the Maximum Independent Sets

In this section, we modify Algorithm #IndSets to count the number of maximum independent sets in a chordal graph. For a set family $\mathscr{S}$, we denote by $\max(\mathscr{S})$ the size of a largest set in $\mathscr{S}$, and $\mathrm{argmax}(\mathscr{S})$ denotes the family of largest sets in $\mathscr{S}$. For a graph G, let $\mathscr{M}\mathscr{I}\mathscr{S}(G)$ be the family of maximum independent sets in G. For a vertex v, let $\mathscr{M}\mathscr{I}\mathscr{S}(G,v)$ be the family of maximum independent sets in G including v. For a vertex set U, let $\overline{\mathscr{M}\mathscr{I}\mathscr{S}}(G,U)$ be the family of maximum independent sets in G including no vertex of U.

From lemmas stated in the previous section and Equations 1, we immediately have the following equations.

Equations 2 With the same set-up as Equations 1, the following identities hold.

$$\mathscr{M}\mathscr{I}\mathscr{S}(G(K)) = \mathrm{argmax}(\overline{\mathscr{M}\mathscr{I}\mathscr{S}}(G(K),K) \dot\cup \dot\bigcup_{v\in K} \mathscr{M}\mathscr{I}\mathscr{S}(G(K),v));$$

$$\mathscr{M}\mathscr{I}\mathscr{S}(G(K),v) = \mathrm{argmax}(\{S \mid S=\bigcup_{i=1}^{\ell} S_i, S_i \in \left\{ \begin{array}{ll} \mathscr{M}\mathscr{I}\mathscr{S}(G(K_i),v) & \text{if } v\in K_i \\ \overline{\mathscr{M}\mathscr{I}\mathscr{S}}(G(K_i),K\cap K_i) & \text{otherwise} \end{array} \right\});$$

$$\overline{\mathscr{M}\mathscr{I}\mathscr{S}}(G(K),K) = \mathrm{argmax}(\{S \mid S=\bigcup_{i=1}^{\ell} S_i, S_i \in \mathscr{M}\mathscr{I}\mathscr{S}(G(K_i),K\cap K_i)\});$$

$$\overline{\mathscr{M}\mathscr{I}\mathscr{S}}(G(K_i),K\cap K_i) = \mathrm{argmax}(\overline{\mathscr{M}\mathscr{I}\mathscr{S}}(G(K_i),K_i) \dot\cup \dot\bigcup_{u\in K_i\setminus K} \mathscr{M}\mathscr{I}\mathscr{S}(G(K_i),u)).$$

Since the sets of each family on the left hand side have the same size in each equation, the cardinality of the set can be computed in the same order as Algorithm #IndSets. For example, $\mathscr{M}\mathscr{I}\mathscr{S}(G(K))$ can be computed as follows.

1. Set $N := 0$ and $M := \max(\overline{\mathscr{M}\mathscr{I}\mathscr{S}}(G(K),K) \cup \bigcup_{v\in K} \mathscr{M}\mathscr{I}\mathscr{S}(G(K),v))$;
2. if the size of a member of $\overline{\mathscr{M}\mathscr{I}\mathscr{S}}(G(K),K)$ is equal to M, then $N := N + \left|\overline{\mathscr{M}\mathscr{I}\mathscr{S}}(G(K),K)\right|$;
3. for each $v\in K$, if the size of a member of $\mathscr{M}\mathscr{I}\mathscr{S}(G(K),v))$ is equal to M, then $N := N + |\mathscr{M}\mathscr{I}\mathscr{S}(G(K),v))|$;
4. output N.

In this way we have the following theorem.

Theorem 5 The number of maximum independent sets in a chordal graph $G=(V,E)$ can be computed in $O(|V|+|E|)$ time.

5 Efficient Algorithm to Count the Independent Sets of Size k

In this section, we modify Algorithm #IndSets to count the number of independent sets of size k. For a graph G and a number k, let $\mathscr{IS}(G;k)$ be the family of independent sets in G of size k. For a vertex v, let $\mathscr{IS}(G,v;k)$ be the family of independent sets in G of size k including v. For a vertex set U, let $\overline{\mathscr{IS}}(G,U;k)$ be the family of independent sets in G of size k including no vertex of U. From lemmas stated in Section 3 and Equations 1, we immediately obtain the following equations.

Equations 3

$$\mathscr{IS}(G(K);k)=\overline{\mathscr{IS}}(G(K),K;k)\dot\cup\dot\bigcup_{v\in K}\mathscr{IS}(G(K),v;k);$$

$$\mathscr{IS}(G(K),v;k)=\{S\mid S=\bigcup_{i=1}^{\ell}S_i,|S|=k,S_i\in\left\{\begin{array}{ll}\mathscr{IS}(G(K_i),v) & \text{if } v\in K_i\\ \overline{\mathscr{IS}}(G(K_i),K\cap K_i) & \text{otherwise}\end{array}\right\}\};$$

$$\overline{\mathscr{IS}}(G(K),K;k)=\{S\mid S=\bigcup_{i=1}^{\ell}S_i,|S|=k,S_i\in\mathscr{IS}(G(K_i),K\cap K_i)\};$$

$$\overline{\mathscr{IS}}(G(K_i),K\cap K_i;k)=\overline{\mathscr{IS}}(G(K_i),K_i;k)\dot\cup\dot\bigcup_{u\in K_i\setminus K}\mathscr{IS}(G(K_i),u;k).$$

In contrast to Equations 1, the second and third equations of Equations 3 do not give a straightforward way to compute $|\mathscr{IS}(G(K),v;k)|$ and $\left|\overline{\mathscr{IS}}(G(K),K;k)\right|$, respectively, since we have to count the number of combinations of $S_1,\ldots,S_\ell$ which generate an independent set of size k. To compute them, we need a more detailed algorithm.

Here we only explain a method to compute $|\mathscr{IS}(G(K),v;k)|$ since $\left|\overline{\mathscr{IS}}(G(K),K;k)\right|$ can be computed in a similar way. Fix an arbitrary vertex $v\in K$. Then, according to v, we give indices to the children of K such that $K_1,\ldots,K_p$ include v and $K_{p+1},\ldots,K_\ell$ do not. For $k'\le k$ and $\ell'\le p$, let $\textsc{Num}(\ell';k'):=\{S\mid S=\bigcup_{i=1}^{\ell'}S_i,S_i\in\mathscr{IS}(K_i,v),|S|=k'\}$. For $k'\le k$ and $\ell'\ge p+1$, let $\overline{\textsc{Num}}(\ell';k'):=\{S\mid S=\bigcup_{i=\ell'}^{\ell}S_i,S_i\in\mathscr{IS}(K_i,K_i\setminus K),|S|=k'\}$. Then, it holds that $|\mathscr{IS}(G(K),v;k)|=\sum_{h=0}^{k}(|\textsc{Num}(p;h)|\times\left|\overline{\textsc{Num}}(p+1;k-h)\right|)$.

For each ℓ' and k', $|\textsc{Num}(\ell';k')|$ can be computed in $O(k\times p)$ time based on the following recursive equation:

$$\left|\textsc{Num}(\ell';k')\right|=\left\{\begin{array}{ll}\sum_{h=0}^{k'}|\textsc{Num}(\ell'-1;h)|\times|\mathscr{IS}(G(K_{\ell'}),v;k'-h)| & \text{if } \ell'>1,\\ |\mathscr{IS}(G(K_1),v;k')| & \text{otherwise.}\end{array}\right.$$

Similarly, $\left|\overline{\textsc{Num}}(\ell';k')\right|$ can be computed in $O(k')$ time. The computation of $|\textsc{Num}(\ell';k')|$ and $\left|\overline{\textsc{Num}}(\ell';k')\right|$ for all combinations of ℓ' and k' can be done in $O(k^2|\textsc{chd}(K)|)$ time, thus we can count the number of independent sets of size k in a chordal graph in $O(k^2|V|^2)$ time. In the following, we reduce the computation time by the same technique used in the previous sections. Observe that $\left|\overline{\mathscr{IS}}(G(K),K;k')\right|=\sum_{h=0}^{k'}\left|\overline{\textsc{Num}}(p;h)\right|\times\left|\overline{\textsc{Num}}(p+1;k'-h)\right|$, which gives $\left|\overline{\textsc{Num}}(p+1;k')\right|\times\left|\overline{\textsc{Num}}(p;0)\right|=\left|\overline{\mathscr{IS}}(G(K),K;k')\right|-\sum_{h=1}^{k'}\left|\overline{\textsc{Num}}(p;h)\right|\times\left|\overline{\textsc{Num}}(p+1;k'-h)\right|$. This implies that we can compute $\left|\overline{\textsc{Num}}(k';p+1)\right|$ from $\left|\overline{\mathscr{IS}}(G(K),K;h)\right|$ and $\left|\overline{\textsc{Num}}(p;h)\right|$ in the increasing order of k'. The computation time for this task is $O(k\times p)$.

In summary, we can compute $|\mathscr{IS}(G(K),v;k')|$ for all $v\in K$ and $k'\in\{0,\ldots,k\}$ in $O(k^2\sum_{v\in K}|\{K'\in\textsc{chd}(K)\mid v\in K'\}|)$ time. Therefore, the total computation time over all iterations can be bounded in the same way as the above section, and we obtain the following theorem.

Theorem 6
1. The number of independent sets of size k in a chordal graph $G=(V,E)$ can be computed in $O(k^2(|V|+|E|))$ time.
2. The numbers of independent sets of all sizes from 0 to $|V|$ in a chordal graph $G=(V,E)$ can be simultaneously computed in $O(|V|^2(|V|+|E|))$ time.

6 Hardness of Counting the Maximal Independent Sets

In this section, we show the hardness results for counting the number of maximal independent sets in a chordal graph. Although finding a maximal independent set is easy even in a general graph, we show that the counting version of the problem is actually hard.

Theorem 7 Counting the number of maximal independent sets in a chordal graph is #P-complete.

The proof is based on a reduction from the counting problem of the number of set covers. Let X be a finite set, and $\mathcal{S} \subseteq 2^X$ be a family of subsets of X. A *set cover* of X is a subfamily $\mathcal{F} \subseteq \mathcal{S}$ such that $\bigcup \mathcal{F} = X$. Counting the number of set covers is #P-complete [9].

PROOF: The membership in #P is immediate. To show the #P-hardness, we use a polynomial-time reduction of the problem for counting the number of set covers to our problem.

Let X be a finite set and $\mathcal{S} \subseteq 2^X$ be a family of subsets of X, and consider them as an instance of the set cover problem. Let us put $\mathcal{S} := \{S_1, \ldots, S_t\}$. From X and $\mathcal{S}$, we construct a chordal graph $G = (V, E)$ in the following way.

We set $V := X \cup \mathcal{S} \cup \mathcal{S}'$, where $\mathcal{S}' := \{S'_1, \ldots, S'_t\}$. Namely, $\mathcal{S}'$ is a copy of $\mathcal{S}$. Now, we draw edges. There are three kinds of edges. (1) We connect every pair of vertices in X by an edge. (2) For every $S \in \mathcal{S}$, we connect $x \in X$ and S by an edge if and only if $x \in S$. (3) For every $S \in \mathcal{S}$, we connect S and S' (a copy of S) by an edge. Formally speaking, we define $E := \{\{x,y\} \mid x,y \in X\} \cup \{\{x,S\} \mid x \in X, S \in \mathcal{S}, x \in S\} \cup \{\{S,S'\} \mid S \in \mathcal{S}\}$. This completes our construction. Note that this construction can be done in polynomial time.

First, let us check that the constructed graph G is indeed chordal. Let C be a cycle of length at least four in G. Since the degree of a vertex in $\mathcal{S}'$ is one, they do not take part in any cycle of G. So forget them. Since $\mathcal{S}$ is an independent set of G, vertices in $\mathcal{S}$ cannot appear along C in a consecutive manner. Then, since the length of C is at least four, there have to be at least two vertices of X which appear in C not consecutively. Then, these two vertices give a chord since X is a clique of G. Hence, G is chordal.

Now, we look at the relation between the set covers of X and the maximal independent sets of G. Let U be a maximal independent set of G. We distinguish two cases.

Case 1. Consider the case in which U contains a vertex $x \in X$. Since X is a clique of G, U cannot contain any other vertices of X. Let $G_x := G \setminus N_G[x]$. (Remember that $N_G[x]$ is the closed neighborhood of x, i.e., the set of vertices adjacent to x in G and x itself.) By the construction, we have that $V(G_x) = \{S \in \mathcal{S} \mid x \notin S\} \cup \mathcal{S}'$ and $E(G_x) = \{\{S,S'\} \mid S \in \mathcal{S}, x \notin S\}$. Then, a vertex $S' \in \mathcal{S}'$ such that $x \in S$ is an isolated vertex of G_x. Therefore, this vertex must belong to U by the maximality of U. For each $S \in \mathcal{S}$ such that $x \notin S$, U must contain either S or S', but not both. This means that the number of maximal independent sets containing x is exactly $2^{|\{S \in \mathcal{S} \mid x \notin S\}|}$.

Case 2. Consider the case in which U contains no vertex of X. Then, for each $S \in \mathcal{S}$, due to the maximality, U must contain either S or S'. Furthermore, $U \cap \mathcal{S}$ has to be a set cover of X (otherwise an element of X not covered by $U \cap \mathcal{S}$ could be included in U). Hence, the number of maximal independent sets containing no vertex of X is equal to the number of set covers of X.

To summarize, we obtained that the number of maximal independent sets of G is equal to the number of set covers of X plus $\sum_{x \in X} 2^{|\{S \in \mathcal{S} \mid x \notin S\}|}$. Since the last sum can be computed in polynomial time, this concludes the reduction. □

As a variation, let us consider the problem for counting the minimum maximal independent sets in a chordal graph. Note that a minimum maximal independent set in a chordal graph can be found in polynomial time [4]. In contrast to that, the counting version is hard.

Theorem 8 Counting the minimum maximal independent sets in a chordal graph is #P-complete.

PROOF: We use the same reduction as in the proof of Theorem 7. Look at the case distinction in that proof again. The maximal independent sets arising from Case 1 have $|\mathcal{S}| + 1$ elements, while the maximal independent sets from Case 2 have $|\mathcal{S}|$ elements. Therefore, the minimum maximal independent sets of the graph G constructed in that proof are exactly the maximal independent sets arising from Case 2, which precisely correspond to the set covers of X. □

Modifying the proof of Theorem 7, we may also show that finding a minimum weight maximal independent set in a chordal graph is NP-hard, and even hard to approximate in the sense that there exists no randomized polynomial-time approximation algorithm to find such a set within a factor of $c \ln |V|$, for some constant c, unless $\mathsf{NP} \subseteq \mathsf{ZTIME}(n^{O(\log \log n)})$. The detail can be found in the complete version.

References

[1] A. Brandstädt, V.B. Le, and J.P. Spinrad. *Graph Classes: A Survey.* SIAM, 1999.

[2] L.S. Chandran. A Linear Time Algorithm for Enumerating All the Minimum and Minimal Separators of a Chordal Graph. In *COCOON 2001*, pages 308–317. Lecture Notes in Computer Science Vol. 2108, Springer-Verlag, 2001.

[3] L.S. Chandran, L. Ibarra, F. Ruskey, and J. Sawada. Generating and Characterizing the Perfect Elimination Orderings of a Chordal Graph. *Theoretical Computer Science*, 307:303–317, 2003.

[4] M. Farber. Independent Domination in Chordal Graphs. *Operations Research Letters*, 1(4):134–138, 1982.
[5] D.R. Fulkerson and O.A. Gross. Incidence Matrices and Interval Graphs. *Pacific Journal of Mathematics*, 15:835–855, 1965.
[6] F. Gavril. Algorithms for Minimum Coloring, Maximum Clique, Minimum Covering by Cliques, and Maximum Independent Set of a Chordal Graph. *SIAM Journal on Computing*, 1(2):180–187, 1972.
[7] M.C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Annals of Discrete Mathematics 57. Elsevier, 2nd edition, 2004.
[8] J.Y.-T. Leung. Fast Algorithms for Generating All Maximal Independent Sets of Interval, Circular-Arc and Chordal Graphs. *Journal of Algorithms*, 5:22–35, 1984.
[9] J.S. Provan and M.O. Ball. The Complexity of Counting Cuts and of Computing the Probability that a Graph is Connected. *SIAM Journal on Computing*, 12:777–788, 1983.
[10] J.P. Spinrad. *Efficient Graph Representations*. American Mathematical Society, 2003.
[11] S.P. Vadhan. The Complexity of Counting in Sparse, Regular, and Planar Graphs. *SIAM Journal on Computing*, 31(2):398–427, 2001.

Packing non-returning A-paths

GYULA PAP*

Department of Operations Research
Eötvös University, Budapest
1117 Pázmány Péter sétány 1/C, Budapest, Hungary
gyuszko@cs.elte.hu

Abstract: Chudnovsky et al. gave a min-max formula for the maximum number of node-disjoint non-zero A-paths in group-labeled graphs [1], which is a generalization of Mader's theorem on node-disjoint A-paths [3]. Here we present a further generalization with a shorter proof. The main feature of Theorem 1 is that parity is "hidden" inside ν^*, which is given an oracle for non-bipartite matching.

Keywords: A**-paths, matching**

1 Introduction

This paper is motivated by results in [5] of A. Sebő, L. Szegő on the structure of A-paths and results in [1] of M. Chudnovsky, J. Geelen, B. Gerards, L. Goddyn, M. Lohman and P. Seymour on the packing of non-zero A-paths in group-labeled graphs. The method of proof in this paper is related to the proof of Mader's theorem on $\mathscr{S}$-paths [3] given by A. Schrijver [4]. We give a generalization for Berge's alternating paths' lemma to the setting in the sequel, which can be used efficiently for the inductive method.

The main feature of Theorem 1 is that parity is "hidden" inside ν^* in the dual, and we do not need to deal with parity of components in the proof. This also implies a smart formulation for known special cases, like Mader's theorem and the result of Cudnovsky et al.

Let $G=(V,E)$ be an oriented graph with node-set V, arc-set E and a fixed set $A\subseteq V$ of terminals. Let Ω be an arbitrary *set of "potentials"* and let $\omega : A\to\Omega$ define the *potential of origin* for the terminals. Let $\pi : E\to S(\Omega)$ where $S(\Omega)$ is the set of all permutations of Ω. For an arc $ab=e\in E$ let $\pi(e,a):=\pi(e)$ and $\pi(e,b):=\pi^{-1}(e)$. An *A-path* in G is a sequence of nodes and arcs which correspond to a path in the underlying undirected graph joining two distinct nodes of A, not using any other node in A. For an A-path $P=(v_0,e_0,v_1,e_1,\dots,e_{k-1},v_k)$ let $\pi(P):=\pi(e_0,v_0)\circ\pi(e_1,v_1)\circ\ldots\circ\pi(e_{k-1},v_{k-1})$, let P be called *non-returning* if $\pi(P)(\omega(v_0))\neq\omega(v_k)$. Notice, an A-path is non-returning if and only if its reverse is non-returning. A family of node-disjoint non-returning A-paths will be called a *non-returning family*.

Let $\nu(G,A,\omega,\pi)$ denote the maximum cardinality of a non-returning family. This is a slight generalization of the packing of non-zero A-paths in group-labeled graphs, for which a min-max formula was given by Chudnovsky et al. in [1]. To see this, we need to put the group as the set of potentials, the potential of origin is zero for each terminal, and $\pi(e)$ must be the multiplication by the group-label on arc e.

2 The min-max formula

T. Gallai [2] determined the maximum number $\nu^*(G,A)$ of node-disjoint A-paths by a reduction to non-bipartite matching. This is a relaxation of the above problem by omitting the constraint of non-returning.

Consider a set $F\subseteq E$ of arcs, let $A':=A\cup V(F)$. F is called *A-balanced* if ω can be extended to a function $\omega' : A'\to\Omega$ with the property that each arc $ab=e\in F$ gives a one-arc returning A'-path, i.e. $\pi(ab)(\omega'(a))\neq\omega'(b)$. Notice, one can check A-balance by using a depth-first search. Moreover, an A-path P is returning if and only if $E(P)$ is A-balanced.

Theorem 1 If G,A,ω,π is given as above then the equation

$$\nu(G,A,\omega,\pi)=\min_F \nu^*(G-F,A\cup V(F)) \tag{1}$$

holds, where the minimum is taken over A-balanced arc-sets F.

*Research is supported by OTKA grants T 037547, TS 049788 by European MCRTN ADONET, Contract Grant No. 504438, and by the Egerv´ary Research Group of the Hungarian Academy of Sciences.

First we prove the easy inequality, i.e. for an A-balanced arc-set F we show $\nu(G,A,\omega,\pi) \leq \nu^*(G-F,A\cup V(F))$. Consider a non-returning A-path P. One can easily see that if $E[P] \subseteq F$ then P is returning, hence some section of P must be an $A\cup V(F)$-path in $G-F$. Given any non-returning family, the family of these sections gives a same-size family of node-disjoint $A\cup V(F)$-paths in $G-F$.

To show equality in (1), we need to use the following notion. For a non-returning family $\mathscr{P}$ let $A(\mathscr{P})$ denote the set of terminals covered by the paths in $\mathscr{P}$. A set $Z \subseteq A$ is called *exactly coverable* if there is non-returning family $\mathscr{P}$ with $Z = A(\mathscr{P})$.

Lemma 2 If Z is exactly coverable with $|Z| < 2\nu(G,A,\omega,\pi)$ (i.e. it is not maximal), then there is an exactly coverable set $Z+s+t$ with $s,t \notin Z$.

PROOF: We prove this by induction on $|V|$. Let $\mathscr{P}$ be a non-returning family with $Z = A(\mathscr{P})$, consider a non-returning family $\mathscr{R}$ with $|\mathscr{R}| = |\mathscr{P}|+1$. If $Z \subseteq A(\mathscr{R})$ then we are done, suppose there is a node $r \in Z - A(\mathscr{R})$.

Case I. If r is covered by a one-arc path rr' in $\mathscr{P}$. Then $Z' := Z-r-r'$ is exactly coverable for $G-r-r', A-r-r', \omega, \pi$. It is easy to see that Z' is not maximal, we only need to delete from $\mathscr{R}$ a path incident to r', if any. So, by induction, there is an exactly coverable set $Z'+s+t \subseteq A-r-r'$, then $Z+s+t$ is exactly coverable for G,A,ω,π.

Case II. Suppose r is covered by a path with first arc $rq \in E$, $q \in V-A$. Define $\omega' : A-r+q \to \Omega$ by ω on $A-r$ and by $\omega'(q) := \pi(rq,r)(\omega(r))$. Then $Z' := Z-r+q$ is exactly coverable for $G-r, A-r+q, \omega', \pi$. We claim that Z' is not maximal, which can be seen as follows: if the paths in $\mathscr{R}$ are disjoint from q, then $A(\mathscr{R})$ is exactly coverable for $G-r, A-r+q, \omega', \pi$. Otherwise, if there is a path $R \in \mathscr{R}$ with $q \in V(R)$ joining nodes $q_1, q_2 \in A(\mathscr{R})$, then some $q-q_i$ section of R must be non-returning, thus $A(\mathscr{R}) - q_{3-i} + q$ is exactly coverable for $G-r, A-r+q, \omega', \pi$. So, by induction, there is an exactly coverable set $Z'+s+t \subseteq A-r+q$, then $Z+s+t$ is exactly coverable for G,A,ω,π. □

Let α denote the set of those terminals where there starts a returning A-path. Consider a counterexample with $|V|+|E|$ minimal, and then α minimal.

Claim 3 There is a node-disjoint family of $\nu(G,A,\omega,\pi)+1$ A-paths $\nu(G,A,\omega,\pi)$ of which are non-returning.

PROOF: In case of $\nu(G,A,\omega,\pi) = \nu^*(G,A)$ the formula (1) obviously holds, choose $F = \emptyset$. Otherwise, there is a terminal $t \in T$ where there starts a returning A-path. Let $\Omega' := \Omega + \bullet$ and we define π' by π and $\bullet$ allways mapped onto $\bullet$, we redefine $\omega'(t) := \bullet$. So all paths starting in t will be non-returning, the status of paths disjoint from t does not change. $\alpha' < \alpha$, consider a minimal A-balanced set F with respect to ω', π' with $\nu(G,A,\omega',\pi') = \nu^*(G-F, A\cup V(F))$. By the minimality of F, $t \notin V(F)$, hence F is A-balanced with respect to ω, π. So, by the choice of G,A,ω',π' we must have $\nu(G,A,\omega',\pi') > \nu(G,A,\omega,\pi)$. Consider $\nu(G,A,\omega,\pi)+1$ non-returning A-paths with respect to ω', π', only a path incident with t can be returning with respect to ω, π. □

Consider a family $\mathscr{P}$ of A-paths given by Claim 3, let $P \in \mathscr{P}$ be the returning path. Here $F := E(P)$ is A-balanced, let ω' be the $A\cup V(P) \to \Omega$ function from the definition. Let $G' := G-F$ and $A' := A \cup V(P)$, since the paths in $\mathscr{P}-P$ are non-returning we have

$$\nu(G',A',\omega',\pi') \geq \nu(G,A,\omega,\pi) = \nu. \tag{2}$$

By the choice of G,A,ω,π, there is an A'-balanced arc set F' with respect to ω', π', with $\nu(G',A',\omega',\pi') = \nu^*(G'-F', A'\cup V(F'))$. It is easy to see that $F := F' \cup E(P)$ is A-balanced with respect to ω, π, which gives $\nu^*(G-F, A\cup V(F)) = \nu^*(G'-F', A'\cup V(F'))$. If in (2) we have equality, then we are done.

Otherwise, if there is a non-returning family $\mathscr{P}'$ in G',A',ω',π' with $|\mathscr{P}'| > \nu$, then by Lemma 2 we choose $\mathscr{P}'$ with $A(\mathscr{P}') = A(\mathscr{P}-P)+s+t$. We will get to a contradiction by constructing a $|\mathscr{P}'|$-size non-returning family. To this end, a path ending in a node in $V(P)$ can be extended by a section of P to get a non-returning A-path. Since $\mathscr{P}'$ covers at most two nodes in $V(P)$ we need at most two such extending sections, clearly, two extensions fit into P. This contradiction completes the proof.

References

[1] M. CHUDNOVSKY, J.F. GEELEN, B. GERARDS, L. GODDYN, M. LOHMAN, P. SEYMOUR, Packing non-zero A-paths in group-labeled graphs, submitted

[2] T. GALLAI, Maximum-minimum Sätze und verallgemeinerte Faktoren von Graphen, *Acta Mathematica Academiae Scientarim Hungaricae* (1961) **12**

[3] W. MADER, Über die Maximalzahl kreuzungsfreier H-Wege, *Archiv der Mathematik (Basel)* (1978) **31**

[4] A. SCHRIJVER, A short proof of Mader's $\mathscr{S}$-paths theorem, *Journal of Combinatorial Theory Ser. B* (2001) **85**

[5] A. SEBŐ, L. SZEGŐ, The path-packing structure of graphs, *Proceedings of Integer Programming and Combinatorial Optimization Conference X, Lecture Notes in Computer Science*

Total dual integrality of a description of the stable marriage polyhedron

TAMÁS KIRÁLY*

MTA-ELTE Egerváry Research Group, Department of Operations Research
Eötvös University
Pázmány Péter sétány 1/C, Budapest, Hungary, H-1117.
tkiraly@cs.elte.hu

JÚLIA PAP

Department of Operations Research
Eötvös University
Pázmány Péter sétány 1/C, Budapest, Hungary, H-1117.
papjuli@cs.elte.hu

Abstract: Rothblum showed that the convex hull of the stable matchings of a bipartite preference system can be described by an elegant system of linear inequalities. In this paper we prove that the description given by Rothblum is totally dual integral. We give a constructive proof based on the results of Gusfield and Irving on rotations, which gives rise to a strongly polynomial algorithm for finding an integer optimal dual solution.

Keywords: stable matching, stable marriage, rotation, total dual integrality

1 Introduction

The stable marriage problem was introduced by Gale and Shapley [4], who showed that every bipartite preference system has a stable matching, and gave an algorithm that finds one. Since then, a lot of progress has been made in understanding the problem and its non-bipartite version, the so-called stable roommates problem (see e.g. [3]). Of particular interest are the results of Vande Vate [9] and Rothblum [8], who gave simple systems of linear inequalities that describe the convex hull of stable matchings of a bipartite preference system.

The contribution of the present paper is that the linear system of Rothblum is in fact a totally dual integral (TDI) system (see [1]). This is interesting because, contrary to the case of other known small TDI systems like bipartite matchings and network flows, the matrix of the system is not totally unimodular. We prove the TDI property by showing that integer optimal dual solutions can be derived from the dual solutions of the associated rotation system, which has a totally unimodular matrix. This proof also gives rise to a strongly polynomial algorithm for finding an integer optimal dual solution.

Let $G = (U,V;E)$ be a bipartite graph, and for every $w \in U \cup V$ let $<_w$ be a linear order of the edges incident to w. The set of these linear orders is denoted by $\mathscr{O}$, and the pair $(G,\mathscr{O})$ is called a *bipartite preference system*. The notation $e \leqslant_w f$ is used if $e <_w f$ or $e = f$ (we say that e *dominates* f). An edge e is said to be *better at* w than f if $e <_v f$. We use $e <_U f$ to denote that e and f has a common endnode in U and there e is better than f.

Let $E' \subseteq E$ be a set of edges. An edge $e \in E$ *blocks* E' if e is not dominated by any element of E'. A matching M is called *stable* if it is not blocked by any edge of E, i.e. M dominates every edge. In particular, every stable matching is inclusion-wise maximal. For a node w covered by M, let $p_M(w)$ denote the other endnode of the edge of M covering w.

The rest of this section contains some well-known results on stable matchings that are used in the subsequent proofs. Gale and Shapley proved that every bipartite preference system has a stable matching, and they gave an algorithm for finding one. The structural properties of stable matchings were first described in [7]. It is easy to see that any two stable matchings M,N cover the same node set. For $u \in U$, let $\min_u(M,N)$ be the best edge of $M \cup N$ at u, and let $\max_u(M,N)$ be the worst edge of $M \cup N$ at u (if u is not covered by $M \cup N$, then $\min_u(M,N) = \max_u(M,N) = \emptyset$). Let $M \wedge N := \{\min_u(M,N) \mid u \in U\}$ and $M \vee N := \{\max_u(M,N) \mid u \in U\}$. It is easy to see that $M \wedge N$ and $M \vee N$ are stable matchings. Conway proved that the set $\mathscr{M}$ of stable matchings with the operations $\wedge$ and $\vee$ forms a distributive lattice, which has a unique minimal element (the U-optimal stable matching M_U) and a unique maximal element (the V-optimal stable matching M_V). The algorithm of Gale and Shapley finds M_U or M_V.

2 Rotations

In this section we briefly describe the basic properties of the rotations of a bipartite preference system. These results are taken from the book of Gusfield and Irving [5], but we cite them in a slightly different form. Let $(G,\mathscr{O})$ be a bipartite preference

*Research is supported by the Hungarian National Foundation for Scientifi c Research, OTKA T037547, and by European MCRTN Adonet, Contract Grant No. 504438.

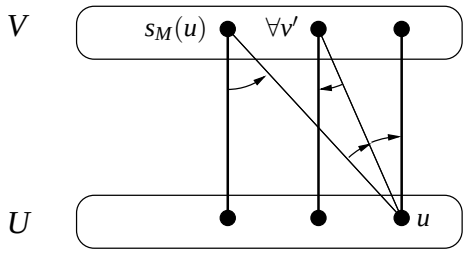


Figure 1: Definition of $s_M(u)$ (the arrows point at the better edge)

system, M a stable matching, and $u \in U$ a node covered by M. Let $s_M(u)$ denote the node $v \in V$ (if it exists) for which the following hold:

- $(u,v) \in E$ and (u,v) is better at v than $(p_M(v),v)$,
- if $(u,v') \in E$ and $(u,v') <_u (u,v)$, then v' is covered by M, and (u,v') is not better at v' than $(p_M(v'),v')$.

It is easy to see that if an edge (u,v) is between $(u,s_M(u))$ and $(u,p_M(u))$ according to the linear order at u, then (u,v) is not in a stable matching.

A cycle $\rho = (v_1,u_1,v_2,u_2,\dots v_k,u_k)$ is called a *rotation* if there is a stable matching M such that $(u_i,v_i) \in M$ and $v_{i+1} = s_M(u_i)$ for every $i = 1,2,\dots k$ (where $v_{k+1} = v_1$). If these properties hold for a rotation ρ and a stable matching M, then we say that ρ can be *eliminated* from M. Let $M/\rho := M \setminus \{(v_i,u_i) : i = 1,2,...k\} \cup \{(u_i,v_{i+1}) : i = 1,2,...k\}$, this is obtained by *eliminating* ρ from M. We say that the edges of type (v_i,u_i) are *discarded edges* in ρ, and the edges of type (u_i,v_{i+1}) are *adopted edges* in ρ. Thus at u_i the discarded edge is better than the adopted edge, and at v_i the adopted edge is better than the discarded edge.

Claim 1 ([5]) *M/ρ is a stable matching, and M/ρ covers M in the lattice $\mathscr{M}$.* □

It turns out that rotations have a very rich structure which completely describes the structure of stable matchings. Let R be the set of rotations of the bipartite preference system. The following are true:

- Let M and N be stable matchings such that $M \wedge N = M$ in the lattice of stable matchings. Then N can be obtained from M by successively eliminating a sequence of rotations. The set of rotations that have to be eliminated is unique (but the sequence is not). In particular, every stable matching can be obtained by eliminating a sequence of rotations from M_U. We say that a rotation ρ is *eliminated* in M if ρ is in the set of rotations that have to be eliminated to obtain M from M_U.
- A partial order can be defined on R: $\rho \preceq \rho'$ if ρ is eliminated in every stable matching where ρ' is eliminated. A set X of rotations can be eliminated from M_U in some order if and only if it is an *ideal* in this partial order ('ideal' means that if $\rho_1 \in X$ and $\rho_2 \preceq \rho_1$, then $\rho_2 \in X$).

It follows from the above facts that there is a one-to-one correspondence between the ideals of the partial order of rotations (including the empty set and the set R) and the stable matchings of the preference system. Let R_M denote the set of rotations corresponding to the stable matching M.

Given a cost function $c : E \to \mathbb{Z}$ on the edges of the graph, we can define a cost function $c' : R \to \mathbb{Z}$ on the rotations the following way: For a rotation $\rho := (v_1,u_1,\dots,v_k,u_k)$ let

$$c'(\rho) := -c(v_1u_1) + c(u_1v_2) - c(v_2u_2) + c(u_2v_3) - \dots + c(u_kv_1). \tag{1}$$

Then for every stable matching M, $c(M) = c(M_U) + c'(R_M)$. This means that a minimum cost stable matching corresponds to a minimum cost ideal in the partial order of rotations.

In the following we define a directed graph $D = (R,A)$ on the set of rotations, with the property that its transitive closure is the partial order $\prec$, i.e. there is a directed path in D from ρ to ρ' if and only if $\rho \preceq \rho'$. The digraph D has two types of edges.

Type 1: $(\rho,\rho') \in A$ if there is an edge $(u,v) \in E$ ($u \in U, v \in V$) contained in both rotations such that in ρ the other edge at u is better than (u,v), and in ρ' the other edge at u is worse than (u,v).

Type 2: $(\rho,\rho') \in A$ if there is an edge $(u,v) \in E$ ($u \in U, v \in V$) which is between the two edges of ρ' incident to u according to $<_u$, and is between the two edges of ρ incident to v according to $<_v$.

It is easy to see that if $(\rho,\rho') \in A$, then $\rho \prec \rho'$.

Theorem 2 ([5]) The transitive closure of the digraph D is the partial order $\preceq$.

Corollary 3 ([5]) There is a one-to-one correspondence between the stable matchings of the preference system $(G,\mathscr{O})$ and the sets of in-degree 0 of D.

The sets of in-degree 0 of D can be described as the integer points of the polyhedron $\{x\in\mathbb{R}^R:\ 0\le x\le 1,\ x(\rho)-x(\rho')\ge 0\ \forall(\rho,\rho')\in A\}$. The matrix of this system is totally unimodular, so the polyhedron has integer vertices. Moreover, integer optimal primal and dual solutions can be obtained using a maximum flow algorithm (this was first observed in [6]). Given a cost function c, finding a minimum cost stable matching corresponds to the problem of finding an integer solution x of the above system for which $c'x$ is minimal, where c' is the cost function defined in (1). The minimum value of $c'x$ is equal to $c(M_{opt})-c(M_U)$, where M_{opt} is a minimum weight stable matching.

If we consider the dual problem, we obtain the following result.

Corollary 4 Let c' be the cost function defined in (1). There exists an integer vector $z\in\mathbb{Z}^{R\cup A}$ such that

$$\begin{aligned} z_\rho &\ge 0 && \text{if } \rho\in R,\\ z_{(\rho,\rho')} &\ge 0 && \text{if } (\rho,\rho')\in A,\\ -z_\rho+z(\Delta^+(\rho))-z(\Delta^-(\rho)) &\le c'(\rho) && \text{if } \rho\in R, \end{aligned} \tag{2}$$

and

$$-\sum_{\rho\in R} z_\rho = c(M_{opt})-c(M_U),$$

where M_{opt} is a minimum weight stable matching, and $\Delta^+(\rho)$ (resp. $\Delta^-(\rho)$) denotes the set of edges of D leaving (resp. entering) ρ.

3 Polyhedral results

This section contains the main results of the paper. First we consider a linear system where variables belong only to the edges that appear in some stable matching. We prove that this system is TDI and it describes the convex hull of stable matchings. This result is then used in the next subsection to prove that the system of Rothblum, which has variables on all edges, is also TDI. Our proofs are constructive, so they give rise to a strongly polynomial algorithm for obtaining an integer optimal dual solution.

3.1 Variables on stable matching edges

Given a bipartite preference system $(G=(U,V;E),\mathscr{O})$, let E_{st} denote the set of edges in E that belong to some stable matching. We first show a TDI system with variables $x\in\mathbb{R}^{E_{st}}$ that describes the convex hull of stable matchings.

Theorem 5 The following system with variables $x\in\mathbb{R}^{E_{st}}$ is TDI:

$$\begin{aligned} \min cx \quad & s.t.\\ x &\ge 0,\\ x(\varphi_{st}(e)) &\ge 1 && \text{if } e\in E\setminus E_{st},\\ x(\varphi_{st}(e)) &= 1 && \text{if } e\in E_{st}, \end{aligned} \tag{3}$$

where $\varphi_{st}(e)$ is the set of edges in E_{st} that dominate e. Furthermore, the system describes the convex hull of stable matchings.

PROOF: It is easy to see that all stable matchings satisfy the inequalities, so they belong to the polyhedron. Let x be an integer element of the polyhedron. If $e=(u,v)\in E_{st}$ is the worst edge at u in E_{st}, then the lattice property implies that e is the best edge at v, so $\varphi_{st}(e)=D_{st}(u)$, where $D_{st}(u)$ is the set of edges in E_{st} incident to u. It follows that $x(D_{st}(u))=1$ if u is covered by a stable matching, so x is a matching that covers the nodes covered by every stable matching. The other inequalities imply that x is a stable matching.

By the above argument, the TDI property implies that the system describes the convex hull of stable matchings. To prove the TDI property, it is enough to show that for every integer cost function $c\in\mathbb{Z}^{E_{st}}$ there exists an integer dual vector $y\in\mathbb{Z}^E$ that satisfies the following:

$$y(e) \geq 0, \qquad \text{if } e \in E \setminus E_{st}, \tag{4}$$

$$y(\psi(e)) \leq c(e), \qquad \text{if } e \in E_{st}, \tag{5}$$

$$\sum_{e \in E} y(e) = c(M_{opt}), \tag{6}$$

where $\psi(e)$ is the set of edges dominated by e, and M_{opt} is a minimum cost stable matching.

We will construct a feasible y using the vector $z \in \mathbb{Z}^{R \cup A}$ which exists according to Corollary 4 (here R is the set of rotations and $D = (R, A)$ is the acyclic digraph referred to in Theorem 2). Let $\rho_1, \ldots, \rho_r$ be a topological order of D, i.e. an order of the rotations in which they can be eliminated. We will denote z_{ρ_i} by z_i and $z_{(\rho_i, \rho_j)}$ by z_{ij}.

The construction of y consists of constructing a sequence of vectors $y_0, y_1, \ldots, y_r$, such that y_t satisfies the inequalities of type (5) on the edges of M_U and on the edges that appear in rotations $\rho_1, \ldots, \rho_t$, and $\sum_{e \in E} y_t(e) = c(M_U) - \sum_{i=1}^{t} z_i$. This would imply that $y := y_r$ satisfies all inequalities of type (5) and $\sum_{e \in E} y(e) = c(M_{opt})$, since $\sum_{i=1}^{r} z_i = c(M_U) - c(M_{opt})$. Thus the constructed y would have the required properties.

In order to make this step-by-step construction possible, some additional technical conditions are required for the vectors $y_0, \ldots, y_r$. For $i = 1, \ldots, r$, let us choose an arbitrary adopted edge e_0^i of the rotation ρ_i. If $(\rho_i, \rho_j) \in A$ and it is an edge of type 2, then we choose an edge $e_{ij} = uv$ that is between the two edges of ρ_i incident to v according to $<_v$, and is between the two edges of ρ_j incident to u according to $<_u$ (such an edge exists because (ρ_i, ρ_j) is an edge of type 2).

For $0 \leq t \leq r$ and $e \in E_{st}$, let

$$c_t(e) = c(e) - \sum \{ z_{li} \mid l \leq t < i,\ e \leqslant_V e_0^l,\ (\rho_l, \rho_i) \in A \}.$$

Note that $c_r = c$. We will define vectors $y_0, y_1, \ldots, y_r$ in $\mathbb{Z}^E$ such that the following conditions hold for every $0 \leq t \leq r$:

(C1) $y_t(e) \geq 0$ if $e \in E \setminus E_{st}$,

(C2) $y_t(\psi(e)) \leq c_t(e)$ if $e \in M_U \cup \rho_1 \cup \rho_2 \cup \cdots \cup \rho_t$,

(C3) $\sum_{e \in E} y_t(e) = c(M_U) - \sum_{i=1}^{t} z_i$,

(C4) $\operatorname{supp} y_t \subseteq M_U \cup \rho_1 \cup \rho_2 \cup \cdots \cup \rho_t \cup \{ e_{ij} :\ (i, j \leq t) \}$.

Condition (C2) means that at some edges inequality (5) should hold with a surplus that depends on t and the digraph D. This surplus can be used in the construction of subsequent y_i vectors. As we have already mentioned, the vector $y := y_r$ is in the dual polyhedron, and $\sum_{e \in E} y_e = c(M_{opt})$, so it is an optimal dual solution.

Let

$$y_0(e) := \begin{cases} c(e) & \text{if } e \in M_U, \\ 0 & \text{otherwise} \end{cases}$$

Then (C2) holds for every edge of M_U, and $\sum_{e \in E} y_0(e) = c(M_U)$, as required. The other conditions are also satisfied.

The vector y_t is obtained from y_{t-1} by changing it only on the edges of $\rho_1 \cup \cdots \cup \rho_t$ and on the edges e_{lt} (for $l \leq t$). Let $w_1^t, w_2^t, \ldots w_{2k}^t$ be the nodes of the rotation ρ_t in reverse order, such that $(w_{2k}^t, w_1^t) = e_0^t$. Thus w_i^t is in U if i is odd and it is in V if i is even.

Let e_i^t denote the edge (w_i^t, w_{i+1}^t). Then $e_i^t <_{w_i^t} e_{i-1}^t$. Every edge e_{2i+1}^t is a discarded edge in ρ_t, hence $y_{t-1}(\psi(e_{2i+1}^t)) \leq c_{t-1}(e_{2i+1}^t)$.

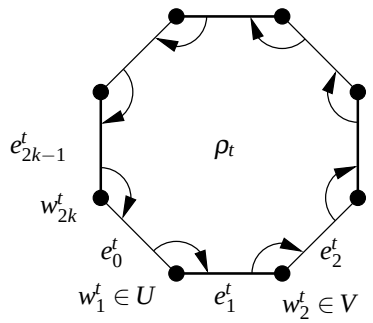


Figure 2: The rotation ρ_t (the thick edges are the discarded edges)

We will use the fact that if en edge e is incident to w^t_{2i+1} and $y_{t-1}(e) \neq 0$, then $e \leqslant_{w^t_{2i+1}} e^t_{2i+1} <_{w^t_{2i+1}} e^t_{2i}$. Analogously, if e is incident to w^t_{2i} and $y_{t-1}(e) \neq 0$, then $e \geq_{w^t_{2i}} e^t_{2i-1} >_{w^t_{2i}} e^t_{2i}$.

Before constructing y_t, we will define a vector y'_t that satisfies conditions (C1), (C3) and (C4) for t, and $y'_t(\psi(e)) \leq c_{t-1}(e)$ for every $e \in M_U \cup \rho_1 \cup \rho_2 \cup \cdots \cup \rho_t$ except for e^t_0. The vector y'_t is obtained from y_{t-1} by changing the values only on the edges of ρ_t. Let

$$y'_t(e^t_{2j+1}) := y_{t-1}(e^t_{2j+1}) + \sum_{i=1}^{2j}(-1)^{i+1}c(e^t_i) \qquad (j = 0,1,\ldots,k-1),$$

$$y'_t(e^t_{2j}) := \sum_{i=1}^{2j}(-1)^i c(e^t_i) \qquad (j = 1,2,\ldots,k-1),$$

$$y'_t(e^t_0) := -z_t.$$

It is easy to check that if $e \in M_U \cup \rho_1 \cup \rho_2 \cup \cdots \cup \rho_{t-1}$, then $y'_t(\psi(e)) \leq y_{t-1}(\psi(e))$, so so $y'_t(\psi(e)) \leq c_{t-1}(e)$.

We also have to check that this inequality holds on the adopted edges of ρ_t except for e^t_0. Observe that $c(e^t_{2j}) - c(e^t_{2j-1}) = c_{t-1}(e^t_{2j}) - c_{t-1}(e^t_{2j-1})$ and $y_{t-1}(\psi(e^t_{2j})) = y_{t-1}(\psi(e^t_{2j-1}))$ for every j. Using these facts,

$$\begin{aligned}
&y'_t(\psi(e^t_{2j})) = y'_t(e^t_{2j}) + y_{t-1}(\psi(e^t_{2j-1})) + y'_t(e^t_{2j-1}) - y_{t-1}(e^t_{2j-1}) = \\
&= \sum_{i=1}^{2j}(-1)^i c(e^t_i) + y_{t-1}(\psi(e^t_{2j-1})) + \sum_{i=1}^{2j-2}(-1)^{i+1}c(e^t_i) = \\
&= y_{t-1}(\psi(e^t_{2j-1})) - c_{t-1}(e^t_{2j-1}) + c_{t-1}(e^t_{2j}) \leq \\
&\leq c_{t-1}(e^t_{2j}),
\end{aligned}$$

so $y'_t(\psi(e)) \leq c_{t-1}(e)$ for every $e \in M_U \cup \rho_1 \cup \rho_2 \cup \cdots \cup \rho_t$ except for e^t_0.

Condition (C3) holds because

$$\sum_{e\in E} y'_t(e) = \sum_{e\in E} y_{t-1}(e) - z_t = c(M_U) - \sum_{i=1}^{t} z_i.$$

To obtain y_t from y'_t, we make the following changes for every l for which $(\rho_l, \rho_t) \in A$.

Suppose that $(\rho_l, \rho_t) \in A$ is an edge of type 1. Then there is a common edge in the two rotations that has even index in ρ_l and odd index in ρ_t, say $e^l_{2i} = e^t_{2j+1}$. For every $0 \leq p \leq i-1$ we increase y'_t by z_{lt} on the edges e^l_{2p}, and decrease by z_{lt} on the edges e^l_{2p+1}. For every $j+1 \leq q \leq k-1$ we increase y'_t by z_{lt} on the edges e^t_{2q}, and we decrease by z_{lt} on the edges e^t_{2q+1}.

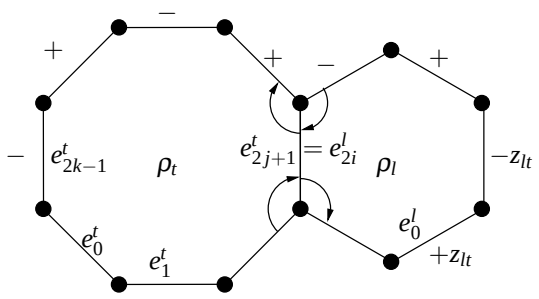


Figure 3: If (ρ_l, ρ_t) is an edge of type 1

If there is an edge of type 2 from ρ_l to ρ_t, then the edge e_{lt} has an endnode of even index in ρ_l, say w^l_{2i}, and it has an endnode of odd index in ρ_t, say w^l_{2j+1}.

For every $0 \leq p \leq i-1$ we increase y'_t by z_{lt} on the edges e^l_{2p}, and decrease by z_{lt} on the edges e^l_{2p+1}. We increase y'_t by z_{lt} on e_{lt} and on the edges e^t_{2q} for every $j+1 \leq q \leq k-1$. For every $j \leq q \leq k-1$ we decrease by z_{lt} on the edges e^t_{2q+1}.

After such a modification, condition (C1) holds because y'_t is increased on e_{lt} by a non-negative value.

The number of edges where we increased by z_{lt} is the same as the number of edges where we decreased, thus

$$\sum_{e\in E} y_t(e) = \sum_{e\in E} y'_t(e) = c(M_U) - \sum_{i=1}^{t} z_i.$$

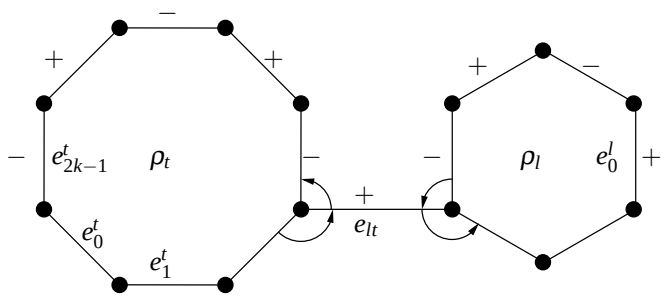


Figure 4: If (ρ_l, ρ_t) is an edge of type 2

For a fixed l, if $e \in M_U \cup \rho_1 \cup \cdots \cup \rho_t \setminus \{e^t_0\}$ and $e \not\preccurlyeq_V e^l_0$ then the number of edges in $\psi(e)$ that were increased by z_{lt} is the same as the number of edges in $\psi(e)$ that were decreased. If $e \in M_U \cup \rho_1 \cup \cdots \cup \rho_t \setminus \{e^t_0\}$ and $e \preccurlyeq_V e^l_0$ then $y'_t(\psi(e))$ increases by z_{lt}, which is exactly the allowed amount, since the term z_{lt} appears in $c_{t-1}(e)$ but not in $c_t(e)$.

Since $\{e \in \rho_t : e \preccurlyeq_V e^t_0\} = \{e^t_0\}$, the only condition we have yet to verify is that $y_t(\psi(e^t_0)) \leq c_t(e^t_0)$. Let us introduce the notation $\alpha := \sum\{z_{lt} \mid l < t,\ e^t_0 \preccurlyeq_V e^l_0,\ (\rho_l, \rho_t) \in A\}$. Then $c_t(e^t_0) = c_{t-1}(e^t_0) - z(\Delta^+(\rho_t)) + \alpha$, so the following holds:

$$\begin{aligned}
y_t(\psi(e^t_0)) &= y'_t(\psi(e^t_0)) - z(\Delta^-(\rho_t)) + \alpha = \\
&= y'_t(e^t_0) + y_{t-1}(\psi(e^t_{2k-1})) + y'_t(e^t_{2k-1}) - y_{t-1}(e^t_{2k-1}) - z(\Delta^-(\rho_t)) + \alpha = \\
&= -z_t + y_{t-1}(\psi(e^t_{2k-1})) + \sum_{i=1}^{2k-2} (-1)^{i+1} c(e^t_i) - z(\Delta^-(\rho_t)) + \alpha \leq \\
&\leq c'(\rho_t) - z(\Delta^+(\rho_t)) + c_{t-1}(e^t_{2k-1}) - c'(\rho_t) - c(e^t_{2k-1}) + c(e^t_0) + \alpha \leq \\
&\leq c'(\rho_t) - z(\Delta^+(\rho_t)) + c_{t-1}(e^t_{2k-1}) - c'(\rho_t) - c_{t-1}(e^t_{2k-1}) + c_{t-1}(e^t_0) + \alpha = \\
&= c_{t-1}(e^t_0) - z(\Delta^+(\rho_t)) + \alpha = c_t(e^t_0),
\end{aligned}$$

where we used the fact that $-z_t - z(\Delta^-(\rho_t)) \leq c'(\rho_t) - z(\Delta^+(\rho_t))$ by (2). Thus y_t satisfies condition (C2) on all the required edges. □

3.2 Variables on all edges

In [8], Rothblum gave a linear system that describes the convex hull of stable matchings of an arbitrary bipartite preference system. We now prove, using Theorem 5, that this system is also TDI.

Theorem 6 The following system, with variables $x \in \mathbb{R}^E$, is totally dual integral:

$$x(e) \geq 0, \tag{7}$$

$$-x(D(w)) \geq -1 \qquad \text{if } w \in U \cup V, \tag{8}$$

$$x(\varphi(e)) \geq 1 \qquad \text{if } e \in E. \tag{9}$$

PROOF: Let $c \in \mathbb{Z}^E$ be an integer cost function. We have to find an integer optimal dual solution for c, i.e. vectors $\pi \in \mathbb{Z}^{U \cup V}$ and $y \in \mathbb{Z}^E$ that satisfy

$$y(e) \geq 0, \tag{10}$$

$$\pi(w) \geq 0, \tag{11}$$

$$-\pi(u) - \pi(v) + y(\psi(e)) \leq c(e) \qquad \text{if } e = (u,v) \in E, \tag{12}$$

and

$$-\sum_{w \in U \cup V} \pi(w) + \sum_{e \in E} y(e) = c(M_{opt}).$$

Let $y_0 \in \mathbb{Z}^E$ be an integral dual optimal solution of the system (3) for the cost function c restricted to E_{st}, which exists by Theorem 5, and let π_0 be the all-zero vector on $U \cup V$. Then (y_0, π_0) satisfies (10) if $e \in E \setminus E_{st}$, it satisfies (12) if $e \in E_{st}$, and $-\sum_{w \in U \cup V} \pi_0(w) + \sum_{e \in E} y_0(e) = c(M_{opt})$.

If we increase y by 1 on the edges of a stable matching M and increase π by 1 on every node in U covered by M, then we get a dual vector for which the objective value is the same, and the left side of (12) does not increase for any edge, since $|\psi(e)\cap M|\leq 1$, and if $|\psi(e)\cap M|=1$ then both endnodes of e are covered by M, otherwise e would block M. Moreover, if e is dominated by 2 edges of M then the left side of (12) decreases for e. Let E' be the set of edges that are dominated by 2 edges of some stable matching. By applying modifications of the above type, a dual vector (y_1,π_1) can be constructed which satisfies (10) for every edge, and satisfies (12) for edges in $E_{st}\cup E'$.

Let $e=(u,v)\notin E_{st}\cup E'$. There is no matching M such that $(u,p_M(u))<_u e<_u (u,s_M(u))$, since then two edges of M would dominate e. It follows that either $e>_u (u,p_{M_V}(u))$ or $e>_v (p_{M_U}(v),v)$. Suppose that the first case holds (the second one can be treated similarly by exchanging U and V). Let $(v=v_1,u_1,v_2,u_2,\dots v_k,u_k)$ be a maximal sequence such that $(u_i,v_i)\in M_V$ for every $i=1,\dots,k$ and $v_{i+1}=s_{M_V}(u_i)$ for every $i=1,\dots,k-1$. Since no rotation can be eliminated from M_V, $s_M(u_k)$ must be undefined. This is only possible in the following two cases.

Case 1: there is no edge (u_k,v') that is better at v' than $(p_{M_V}(v'),v')$. In this case we increase the value of y by 1 on edges in $M_V\setminus\{(u_1,v_1),\dots,(u_k,v_k)\}\cup\{(u_1,v_2),\dots,(u_{k-1},v_k)\}$, and increase π by 1 on every node in $U-u_k$ covered by M_V.

Case 2: there is an edge $(u_k,v_{k+1})\in E$ such that v_{k+1} is not covered by M_V and $(u_k,v_{k+1})<_{u_k}(u_k,v')$ if (u_k,v') is better at v' than $(p_{M_V}(v'),v')$. In this case we increase y by 1 on the edges in $M_V\setminus\{(u_1,v_1),\dots,(u_k,v_k)\}\cup\{(u_1,v_2),\dots,(u_k,v_{k+1})\}$, and increase π by 1 on every node in U covered by M_V.

It is easy to see that in both cases the objective value remains the same, the left side of (12) does not increase for any edge, and it decreases by 1 for e. So by applying such modifications on every edge where (12) does not hold we can obtain an integer optimal dual solution. □

The above proof and the proof of Theorem 5 are constructive, so an integer optimal dual solution can be easily obtained from a solution to system (2). If the rotations, the digraph $D=(R,A)$, and the solution to system (2) are given in a convenient form, then this can be done in $O(mn)$ time, where m is the number of edges and n is the number of nodes of the bipartite preference system.

4 Acknowledgement

We thank Tamás Fleiner for his valuable comments and suggestions.

References

[1] J. EDMONDS, R. GILES, A min-max relation for submodular functions on graphs, *Ann. of Discrete Math.* (1977) **1**, 185–204.

[2] T. FLEINER, A fixed-point approach to stable matchings and some applications, *Math. Oper. Res.* (2003) **28**, 103–126.

[3] T. FLEINER, Some results on stable matchings and fixed points, *EGRES Technical Report* (2002) **08**

[4] D. GALE, L.S. SHAPLEY, College admissions and the stability of marriage, *American Math. Monthly* (1962) **69**, 9–15.

[5] D. GUSFIELD, R.W. IRVING, The stable marriage problem: structure and algorithms, *MIT Press, Cambridge, MA* (1989)

[6] T.B. JOHNSON, Pit mine production scheduling, *Ph. D. Thesis, Univ. of California, Berkeley* (1968)

[7] D.E. KNUTH, Mariages stables, *Les Presses de l'Université de Montréal, Montréal* (1976)

[8] U.G. ROTHBLUM, Characterization of stable matchings as extreme points of a polytope, *Math. Programming Ser. A* (1992) **54**, 57–67.

[9] J.H. VANDE VATE, Linear programming brings marital bliss, *Oper. Res. Letters* (1989) **8**, 147–153.

Line graphs of cubic graphs are normal

ZSOLT PATAKFALVI*

Department of Computer Science and Information Theory
Budapest University of Technology and Economics
1111 Budapest, Műegyetem rkp. 3., Hungary
kopasz@cs.bme.hu

Abstract: A graph is called normal if its vertex set can be covered by cliques $Q_1, Q_2, \ldots, Q_k$ and also by stable sets $S_1, S_2, \ldots, S_l$, such that $S_i \cap Q_j \neq \emptyset$ for every i, j. This notion is due to Körner, who introduced the class of normal graphs as an extension of the class of perfect graphs. Normality has also relevance in information theory. Here we prove, that the line graphs of cubic graphs are normal.

Keywords: snarks, cubic graphs, normal graphs, perfect graphs

1 Introduction

The concept of graph normality was introduced by János Körner in [8], where he proved that all members of the celebrated class of perfect graphs are normal. These two graph classes come up together also in the study of additivity properties of the information theoretic functional graph entropy, cf. [6], [9], [12]. Perfect graphs are important and well-studied for several reasons (cf., e. g., [11] and [1]), one of which is certainly the long-standing conjecture of Claude Berge known as the Strong Perfect Graph Conjecture, which has been proved recently by Chudnovsky, Robertson, Seymour, and Thomas [2]. A conjecture of similar flavour has been formulated by De Simone and Körner [7] to characterize those graphs all induced subgraphs of which are normal. This latter conjecture is still open. (See it below.)

There are several important subclasses of perfect graphs known, and by Körner's theorem in [8] all these are also normal. Very few graph families were, however, identified yet as being normal among those that are not necessarily perfect. The main goal of this paper is to show that the members of a certain family, namely those graphs that are line graphs of cubic graphs are all normal. This class of graphs is interesting, for example, for its connections to the four colour theorem and related conjectures, see, e. g., [10].

Now we give the definition of normality. Consider a graph G. A set $\mathbb{A}$ of subsets of $V(G)$ is a covering, if every vertex of G is contained in an element of $\mathbb{A}$.

Definition 1 A graph G is normal if there exist two coverings, $\mathbb{C}$ and $\mathbb{S}$, of G, where each element of $\mathbb{C}$ induces a clique, each element of $\mathbb{S}$ induces a stable set and $C \cap S \neq \emptyset$ for every $C \in \mathbb{C}$, $S \in \mathbb{S}$.

From the symmetry of Definition 1 it follows that a graph is normal iff its complement is normal. If we require normality for every induced subgraph, we obtain the notion of hereditary normality. Clearly, every perfect graph is hereditarily normal, since every induced subgraph of a perfect graph is perfect and consequently normal.

The simplest graphs which are normal but not perfect are the odd cycles with at least 9 vertices (see [8]). Smaller odd cycles are either perfect, like the triangle, or not even normal, like the cycles with 5 or 7 vertices. Actually, these latter graphs and the complement of the 7-cycle are the only minimal not hereditarily normal graphs known so far. This motivates the following conjecture formulated by De Simone and Körner [7].

Conjecture 2 A graph is hereditarily normal iff neither the graph nor its complement contains a 5-cycle or a 7-cycle as an induced subgraph.

For partial results on Conjecture 2 see also [13]. As De Simone and Körner remarks in [7], from Conjecture 2 it would immediately follow that the class of heredtarily normal graphs can be recognized in polynomial time. The analogous statement for perfect graphs is true, though very far from being trivial, cf. [3], [4], [5].

2 Main result

Recall, that the line graph $L(G)$ of a graph G is a graph whose vertices are the edges of G, and two vertices of $L(G)$ are connected if and only if the edges corresponding to them share a common vertex in G. Our main result is the following.

Theorem 3 The line-graph of every cubic graph is normal

Let G be a cubic graph whose edge-chromatic number is 3. The colour-classes of a good edge-colouring of G form stable sets in $L(G)$. Every 3 edges of G which share a common vertex form a clique in $L(G)$. The set of all these cliques, together with the previously mentioned stable sets fulfill the requirements of normality. So, Theorem 3 is trivial in case of such graphs.

The nontrivial part of the theorem is that the statement also holds for those cubic graphs which have edge-chromatic number 4. Such graphs are often called snarks (cf. [10]), and are widely investigated objects.

References

[1] M. Chudnovsky, N. Robertson, P. D. Seymour, R. Thomas, Progress on perfect graphs, *Math. Program. Ser. B*, **97** (2003), 405–422.

[2] M. Chudnovsky, N. Robertson, P. D. Seymour, R. Thomas, The Strong Perfect Graph Theorem, manuscript, 2002, http://www.math.gatech.edu/~thomas/spgc.html.

[3] M.Chudnovsky, G. Cornuéjols, X. Liu, P. Seymour, K. Vušković, Cleaning For Bergeness, manuscript, 2002, http://www.math.princeton.edu/~mchudnov/paper3_submitted.ps

[4] M. Chudnovsky, P. D. Seymour, Recognizing Berge Graphs, manuscript, 2003, http://www.math.princeton.edu/~mchudnov/paper2_submitted.ps

[5] G. Cornuéjols, X. Liu, K. Vušković, A Polynomial Algorithm for Recognizing Perfect Graphs, manuscript, 2003, http://www.integer.gsia.cmu.edu/webpub/perfectrecogn1.pdf

[6] I. Csiszár, J. Körner, L. Lovász, K. Marton, G. Simonyi, Entropy splitting for antiblocking corners and perfect graphs, *Combinatorica*, **10** (1990), 27–40.

[7] C. De Simone, J. Körner, On the odd cycles of normal graphs, Proceedings of the Third International Conference on Graphs and Optimization, GO-III (Leukerbad, 1998), *Discrete Appl. Math.* **94** (1999), no. 1-3, 161–169.

[8] J. Körner, An extension of the class of perfect graphs, *Studia Sci. Math. Hungar.* **8** (1973), 405–409.

[9] J. Körner and G. Longo, Two-step encoding of finite memoryless sources, *IEEE Trans. Inform. Theory*, **19** (1973), 778-782.

[10] R. Nedela, M. Škoviera, Decompositions and reductions of snarks, *J. Graph Theory*, **22** (1996), no. 3, 253–279.

[11] J. L. Ramírez Alfonsín and B. A. Reed (eds.), *Perfect graphs*, Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley and Sons, Ltd., Chichester, 2001.

[12] G. Simonyi, Perfect graphs and graph entropy. An updated survey, in: *Perfect Graphs* (J. L. Ramírez-Afonsín, B. A. Reed, eds.), John Wiley and Sons, Ltd., Chichester, 2001, 293–328.

[13] A. Wagler, The normal graph conjecture and circulant graphs, manuscript

Dominating and Large Induced Trees in Regular Graphs

DIETER RAUTENBACH

Forschungsinstitut für Diskrete Mathematik
University of Bonn
Lennéstr. 2
D-53113 Bonn
Germany
rauten@or.uni-bonn.de

Abstract: Recently Chen, McRae and Sun (Tree Domination in Graphs, *Ars Combin.* **73** (2004), 193-203) asked for characterizations of the class of graphs and the class of regular graphs that have an induced dominating tree, i.e. for which there exists a dominating set that induces a tree.

We give a somewhat negative answer to their question by proving that the corresponding decision problems are NP-complete. Furthermore, we prove essentially best-possible lower bounds on the maximum order of induced trees in connected cacti of maximum degree 3 and connected cubic graphs.

Finally, we give a forbidden induced subgraph condition for the existence of induced dominating trees.

Keywords: induced dominating tree, regular graph, cactus

1 Introduction

All graph $G = (V_G, E_G)$ will be finite, undirected and simple with vertex set V_G and edge set E_G. The neighbourhood and the degree of a vertex $u \in V_G$ in the graph G are denoted by $N_G(u)$ and $d_G(u) = |N_G(u)|$, respectively. For $U \subseteq V_G$, let $N_G(U) = \left(\bigcup_{u \in U} N_G(u) \right) \setminus U$, $N_G[U] = N_G(U) \cup U$ and let $G[U]$ denote the subgraph of G induced by U.

A set $S \subseteq V_G$ of vertices of some graph G is called dominating if $N_G[S] = V_G$. In [5] Chen, McRae and Sun consider the minimum order of dominating sets that induce trees. They have been motivated by the known notions of connected domination [14, 9] and tree domination [8, 9] and studied bounds on this minimum order and its value for specific classes of graphs.

Since not for every graph G there exists an induced dominating tree, i.e. a dominating set $S \subseteq V_G$ for which $G[S]$ is a tree, they pose the two open problems to characterize the class of graphs (cf. problem **(1)** in [5]) and the class of regular graphs (cf. problem **(2)** in [5]) for which induced dominating trees exist.

As our first result we prove that the corresponding decision problems are NP-complete which means that there are most probably no 'nice' characterizations for these classes.

Since obviously an induced dominating tree of an r-regular graph of order n necessarily has order at least $\frac{n-2}{r-1}$, we are led to consider large induced trees of (regular) graphs. Let the maximum order of an induced tree of a graph G be denoted by $\mathrm{it}(G)$. Whereas there are several publications on the maximum order of induced forests [1, 2, 3, 15, 16] on the one hand and on the maximum order of induced trees in random graphs [6, 7, 11, 12, 13] on the other hand, the maximum order of induced trees in general graphs received much less attention. We prove lower bounds on the maximum order of induced trees in connected cacti of maximum degree at most 3 and in connected cubic graphs. (A cactus is a graph all cycles of which are edge disjoint.)

It was observed by several authors (cf. e.g. [4] and the references therein) that graphs G which do not contain one of the two graphs in Figure 1 as an induced subgraph have a dominating path, i.e. there exists a dominating set $S \subseteq V_G$ such that $G[S]$ is a path. Motivated by this observation we close this paper with a similar forbidden induced subgraph condition for the existence of induced dominating trees.

Figure 1 $K_{1,3}$ and $C_3 \circ K_1$

2 Results

We consider the following decision problem and proceed immediately to our first result.

INDUCED DOMINATING TREE
Instance: A graph $G=(V_G,E_G)$.
Question: Is there a dominating set S of G such that $G[S]$ is a tree?

Theorem 1 The problem INDUCED DOMINATING TREE is NP-complete even when restricted to regular graphs.

PROOF: We describe a polynomial reduction of a 3SAT instance to a regular INDUCED DOMINATING TREE instance. Therefore, let $C_1,C_2,...,C_m$ be the clauses of a 3SAT instance $\mathscr{C}$ over the variables $x_1,x_2,...,x_n$. We describe a construction of a regular graph $G=(V_G,E_G)$ whose order is polynomially bounded in n and m such that G has a dominating set S for which $G[S]$ is a tree if and only if $\mathscr{C}$ is satisfiable.

1. Start with G as the empty graph $(\emptyset,\emptyset)$.
2. For every variable x_i for $1\leq i\leq n$ add a triangle
$$x_it_if_ix_i$$
to G.
3. For every clause C_j for $1\leq j\leq m$ add a vertex c_j to G.
4. If the un-negated variable x_i appears in the clause C_j for some $1\leq i\leq n$ and $1\leq j\leq m$, then add the edge t_ic_j to G. If the negation of the variable x_i appears in the clause C_j for some $1\leq i\leq n$ and $1\leq j\leq m$, then add the edge f_ic_j to G.

For $1\leq i\leq n$ let d_i^t and d_i^f denote the number of clauses of $\mathscr{C}$ in which x_i appears un-negated and negated as a variable. At the end of the construction the graph G will be regular of degree
$$\Delta=2\left\lceil\frac{\max\left\{d_i^t,d_i^f\mid 1\leq i\leq n\right\}+6}{2}\right\rceil.$$

Let the graph H arise from the complete bipartite graph $K_{\Delta-1,\Delta-1}$ by adding two new vertices u and v such that u is adjacent to all vertices of one partite set and v is adjacent to all vertices of the other partite set.

For $1\leq i\leq n$ execute the following Steps 5 to 9.

5. Add to G two disjoint copies H_i and H_i' of the graph H in which we denote the corresponding copies of the vertices u,v by u_i,v_i and u_i',v_i'. Add to G the four edges
$$x_iu_i,x_iv_i,t_iu_i',f_iv_i'.$$
6. Add to G a set A_i of
$$\Delta-\left(\max\left\{d_i^t,d_i^f\right\}+3\right)$$
vertices each of which adjacent to t_i and f_i.
7. Add to G a set B_i^t of
$$\Delta-|A_i|-(d_i^t-3)=\max\left\{d_i^t,d_i^f\right\}-d_i^t$$
vertices each of which adjacent to x_i and t_i.
8. Add to G a set B_i^f of
$$\Delta-|A_i|-(d_i^f-3)=\max\left\{d_i^t,d_i^f\right\}-d_i^f$$
vertices each of which adjacent to x_i and f_i.
9. Add
$$\left\lfloor\frac{1}{2}\left(\Delta-\left(|B_i^t|+|B_i^f|+6\right)\right)\right\rfloor=\left\lfloor\frac{1}{2}\left(\Delta-(|d_i^t-d_i^f|+6)\right)\right\rfloor$$
copies of H and join the vertices u and v in all these copies to x_i.

Note that all vertices in

$$W = \{c_1, c_2, ..., c_m\} \cup A_1 \cup ... \cup A_n \cup B_1^t \cup ... \cup B_n^t \cup B_1^f \cup ... \cup B_n^f$$

have degree $1, 2$ or 3 in the graph constructed so far.

For each vertex $w \in W$ execute the following Steps 10 to 13.

10. Let d denote the degree of w in the graph constructed so far. Add to G a set E_w of $\Delta - d$ vertices each of which adjacent to w.

11. For every $w' \in E_w$ add to G a set $X_{w,w'}$ of $\Delta - 1$ vertices each of which adjacent to w'.

12. For every $w' \in E_w$ add $\Delta - 2$ edges to G such that $X_{w,w'}$ induces a path.

13. For every $w' \in E_w$ and every $w'' \in X_{w,w'}$ add

$$\left\lfloor \frac{\Delta - 3}{2} \right\rfloor$$

copies of H and join the vertices u and v in all these copies to w''.

Note that in the graph constructed so far the set

$$X = \{x_1, x_2, ..., x_n\} \cup \bigcup_{w \in W} \bigcup_{w' \in E_w} X_{w,w'}$$

induces a graph whose components are paths (possibly of length 0).

14. Add edges to G such that X induces a path P.

15. Add a copy of H to G and join each of the corresponding vertices u and v to one of the endvertices of the path P.

By now all vertices of the graph constructed so far that are not in X are of degree Δ and all vertices in X are either of degree Δ or of degree $\Delta - 1$. Since Δ is even, there is an even number of vertices in X of degree $\Delta - 1$ and we partition these vertices into arbirary disjoint pairs $\mathscr{P}$.

16. For every pair $\{x', x''\}$ in $\mathscr{P}$ add a copy of H to G and add the edges $x'u$ and $x''v$ to G.

This completes the construction of the graph G. Clearly, the order of G is polynomially bounded in n and m and G is regular of degree Δ.

We proceed to the proof that G has a dominating set S such that $G[S]$ is a tree if and only if $\mathscr{C}$ is satisfiable.

First, we assume that $\mathscr{C}$ is satisfiable and that

$$\varphi : \{x_1, x_2, ..., x_n\} \to \{\text{true}, \text{false}\}$$

is a satisfying truth assignment of $\mathscr{C}$. It is easy to check that the set S that arises from

$$X \cup \{t_i \mid 1 \leq i \leq n, \varphi(x_i) = \text{true}\} \cup \{f_i \mid 1 \leq i \leq n, \varphi(x_i) = \text{false}\}$$

by adding for every copy of the graph H added in Steps 5, 9, 13, 15 or 16 the two vertices u and v together with possibly two further vertices from the copy of H forming a path of length three together with u and v is a dominating set of G that induces a tree.

Conversely, we assume that G has a dominating set S that induces a tree. Since all vertices in X are cutvertices, $X \subseteq S$. Since $\{t_i, f_i\}$ is a cutset for $1 \leq i \leq n$ and $x_i \in S$, exactly one of the two vertices t_i and f_i is in S. Since for every $w \in W$ and every $w' \in E_w$ the vertices in $X_{w,w'}$ induce a path and belong to S, we have $E_w \cap S = \emptyset$. This implies that for every $1 \leq j \leq m$ the vertex c_j has a neighbour in

$$S \cap \{t_i, f_i \mid 1 \leq i \leq n\}$$

and setting

$$\varphi(x_i) = \begin{cases} \text{true} & , t_i \in S \\ \text{false} & , f_i \in S \end{cases}$$

defines a satisfying truth assignment for $\mathscr{C}$. This completes the proof. □

The graphs constructed in the above proof are regular but the degree of regularity depends on the corresponding 3SAT instance. Whereas the problem to determine an induced forest of maximum order within a given graph is NP-hard in general, there are polynomial time algorithms for cubic graphs [10, 17]. It is therefore conceivable - but unlikely according to our opinion - that also induced trees of maximum order can be determined in polynomial time for cubic graphs.

We proceed to our lower bounds on $\text{it}(G)$ for cacti of maximum degree at most 3 and cubic graphs. The observation that these classes are closely related in this context has already been made and used several times [3, 15, 16]. We begin with a technical lemma.

Lemma 2 Let T be a rooted tree with $c \in \mathbb{N}_0$ non-leaves each of which has exactly two children. Let L_T denote the set of leaves of T and let $w : L_T \to \mathbb{R}_{\geq 0}$. If $\mathrm{depth}(l)$ denotes the depth of a leaf $l \in L_T$ in T, then

$$\max\{w(l) + \mathrm{depth}(l) \mid l \in L_T\} \geq \frac{1}{c+1} \sum_{l \in L_T} l(w) + \log_2(c+1) - 1.$$

PROOF: For some fixed $W \geq 0$ let T and w be chosen as in the statement of the lemma such that

1. $\sum_{l \in L_T} w(l) = W$.
2. Subject to the previous condition, $\max\{w(l) + \mathrm{depth}(l) \mid l \in L_T\}$ is minimum.
3. Subject to the previous conditions, $d := \max\{\mathrm{depth}(l) - \mathrm{depth}(l') \mid l, l' \in L_T\}$ is minimum.
4. Subject to the previous conditions, the number of pairs $(l_1, l_2) \in L_T^2$ for which $d = \mathrm{depth}(l_1) - \mathrm{depth}(l_2)$ is minimum.

Claim $|\mathrm{depth}(l_1) - \mathrm{depth}(l_2)| \leq 1$ for $l_1, l_2 \in L_T$.

PROOF OF THE CLAIM: For contradiction, we assume that $d = \mathrm{depth}(l_1) - \mathrm{depth}(l_2) \geq 2$ where $l_1 \in L_T$ has maximum depth and $l_2 \in L_T$ has minimum depth among all leaves of T.

Let v_i be the parent of l_i for $i \in \{1,2\}$. Let v_1' be the parent of v_1 and let v_1'' be the parent of v_1'. It is easy to see that we may assume without loss of generality that $w(l) + \mathrm{depth}(l)$ has the same value for all leaves $l \in L_T$. Specifically, $w(l_1) = w(l_2) - d$.

Let T' arise from T by deleting the edges in $\{v_1v_1', v_1'v_1'', l_2v_2\}$ and inserting the new edges in $\{v_1v_1'', v_1'v_2, l_2v_1'\}$. Setting $w'(l_1) = w(l_1) + 1$, $w'(l_2) = w(l_2) - 1$ and $w'(l) = w(l)$ for all $l \in L_T \setminus \{l_1, l_2\}$ (note that $L_T = L_{T'}$), we obtain a contradiction to the choice of T (Condition 3 or 4). This completes the proof of the claim. □

Since T has c non-leaves, the maximum depth of a leaf of T is at least $\log_2(c+1)$. Hence, by the claim, the minimum depth of a leaf of T is at least $\log_2(c+1) - 1$. Since T has exactly $c+1$ leaves, we have

$$\max\{w(l) \mid l \in L_T\} \geq \frac{W}{c+1}$$

and the desired result follows. □

Instead of a lower bound on $\mathrm{it}(G)$ for cacti G of maximum degree at most 3 we will prove a slightly stronger result. For a graph G and two sets U_1 and U_2 let

$$\mathrm{it}(G, U_1, U_2)$$

denote the maximum order of an induced tree T in G with $U_1 \subseteq V_T$ and $|U_2 \cap V_T| \geq \min\{1, |U_2|\}$ or 0, if no such tree exists.

Theorem 3 Let G be a connected cactus of order n and maximum degree at most 3 which has $c \geq 1$ cycles. Let U denote the vertex set of a cycle of G.

Then

$$\mathrm{it}(G, \emptyset, U) \geq \max\left\{\frac{2}{3}\left(\frac{n-3c}{c} + 2\log_2(c)\right) - 2, 2\log_2(c) - 2\right\}.$$

PROOF: We assume that G and U are chosen as in the statement of the theorem such that

1. $\mathrm{it}(G, \emptyset, U)$ is minimum.
2. Subject to the previous condition, the number of cycles of length at least 4 is minimum.
3. If O denotes the set of all vertices of G that lie on a cycle, then, subject to the previous conditions, the order of the component of $G[O]$ that contains U is maximum.

Claim 1 All cycles of G have length 3.

PROOF OF CLAIM 1: For contradiction, we assume that $C : u_1u_2u_3 \ldots u_lu_1$ is a cycle with $l \geq 4$. For $1 \leq i \leq l$ with $d_G(u_i) = 3$ let G_i denote the component of $G[V_G \setminus V_C]$ that contains a neighbour v_i of u_i. For $1 \leq i \leq l$ with $d_G(u_i) = 2$ let G_i be the empty graph. Clearly, we may assume that either $U = V_C$ or $U \subseteq V_{G_1}$.

If $d_G(u_l) = 3$, then let G' arise from G by deleting the edges in $\{u_2u_3, u_lv_l\}$ and inserting the new edges in $\{u_2u_l, u_3v_l\}$. Similarly, if $d_G(u_l) = 2$, then let G' arise from G by deleting the edge u_2u_3 and inserting the new edge u_2u_l. We consider two cases.

Case 1 $U = V_C$.

It is easy to see that

$$\mathrm{it}(G,\emptyset,U) \;=\; \max\Big\{(l-1)+\sum_{i=1,i\neq j}^{l}\mathrm{it}(G_i,\{v_i\},\emptyset) \mid 1\leq j\leq l\Big\}.$$

If we set $U' = \{u_1,u_2,u_l\}$, i.e. U' is the vertex set of the cycle $u_1u_2u_lu_1$ in G', then

$$\begin{aligned}\mathrm{it}(G',\emptyset,U') \;=\max\Big\{\;& 2+\mathrm{it}(G_1,\{v_1\},\emptyset)+\mathrm{it}(G_2,\{v_2\},\emptyset),\\ & (l-1)+\sum_{i=1,i\neq 1}^{l}\mathrm{it}(G_i,\{v_i\},\emptyset),\\ & (l-1)+\sum_{i=1,i\neq 2}^{l}\mathrm{it}(G_i,\{v_i\},\emptyset)\Big\}.\end{aligned}$$

Therefore, $\mathrm{it}(G',\emptyset,U') \leq \mathrm{it}(G,\emptyset,U)$ and we obtain a contradiction to the choice of G (Condition 1 or 2).

Case 2 $U \subseteq V_{G_1}$.

As above

$$\begin{aligned}\mathrm{it}(G,\emptyset,U) \;=\max\Big\{\;& \mathrm{it}(G_1,\emptyset,U),\\ & \max\Big\{(l-1)+\mathrm{it}(G_1,\{v_1\},U)+\sum_{i=2,i\neq j}^{l}\mathrm{it}(G_i,\{v_i\},\emptyset) \mid 2\leq j\leq l\Big\}\Big\}.\end{aligned}$$

Clearly, U is the vertex set of a cycle of G' totally contained in V_{G_1} and

$$\begin{aligned}\mathrm{it}(G',\emptyset,U) \;=\max\Big\{\;& \mathrm{it}(G_1,\emptyset,U),\\ & 2+\mathrm{it}(G_1,\{v_1\},U)+\mathrm{it}(G_2,\{v_2\},\emptyset),\\ & (l-1)+\mathrm{it}(G_1,\{v_1\},U)+\sum_{i=3}^{l}\mathrm{it}(G_i,\{v_i\},\emptyset)\Big\}.\end{aligned}$$

Again $\mathrm{it}(G',\emptyset,U) \leq \mathrm{it}(G,\emptyset,U)$ and we obtain a contradiction to the choice of G (Condition 1 or 2). This completes the proof of the claim. □

Claim 2 $G[O]$ is connected.

PROOF OF CLAIM 2: For contradiction, we assume that $G[O]$ is not connected. This implies the existence of vertices $u_1,u_2,\ldots,u_l$ for $l\geq 5$ such that $u_1\in O$, $u_2,\ldots,u_{l-3}\notin O$, $G[\{u_1,u_2,\ldots,u_{l-1}\}]$ is the path $u_1u_2\ldots u_{l-1}$ and $u_{l-2},u_{l-1}\in N_G(u_l)$.

For $2\leq i\leq l$, $i\neq l-2$ with $d_G(u_i)=3$ let G_i denote the component of $G[V_G\setminus\{u_1,u_2,\ldots,u_l\}]$ that contains a neighbour v_i of u_i. For $1\leq i\leq l$, $i\neq l-2$ with $d_G(u_i)=2$ let G_i be the empty graph. Let G_1 denote the component of $G[V_G\setminus\{u_2\}]$ that contains u_1. Clearly, we may assume that U and u_1 are contained in the same component of $G_1[V_{G_1}\cap O]$ and obtain

$$\begin{aligned}\mathrm{it}(G,\emptyset,U) \;=\max\Big\{\;& \mathrm{it}(G_1,\emptyset,U),\\ & (l-2)+\mathrm{it}(G_1,\{u_1\},U)+\sum_{i=2}^{l-3}\mathrm{it}(G_i,\{v_i\},\emptyset)\\ & +\max\{\mathrm{it}(G_{l-1},\{v_{l-1}\},\emptyset),\mathrm{it}(G_l,\{v_l\},\emptyset)\}\Big\}.\end{aligned}$$

If $d_G(u_l)=3$, then let G' arise from G by deleting the edges in $\{u_1u_2,u_lv_l\}$ and inserting the new edges in $\{u_1u_l,u_2v_l\}$. Similarly, if $d_G(u_l)=2$, then let G' arise from G by deleting the edge u_1u_2 and inserting the new edge u_1u_l. Note that U is the vertex set of a cycle of G' and

$$\begin{aligned}\mathrm{it}(G',\emptyset,U) = \max\Big\{ & \mathrm{it}(G_1,\emptyset,U),\\ & 1+\mathrm{it}(G_1,\{u_1\},U)+\mathrm{it}(G_{l-1},\{v_{l-1}\},\emptyset),\\ & (l-2)+\mathrm{it}(G_1,\{u_1\},U)+\sum_{i=2}^{l-3}\mathrm{it}(G_i,\{v_i\},\emptyset)+\mathrm{it}(G_l,\{v_l\},\emptyset)\Big\}.\end{aligned}$$

Thus, $\mathrm{it}(G',\emptyset,U)\leq\mathrm{it}(G,\emptyset,U)$ and we obtain a contradiction to the choice of G (Condition 1 or 3). This completes the proof of the claim. □

Let the graph G' arise from $G[O]$ by inserting for every vertex $u\in O$ with $d_{G[O]}(u)=2$ a new vertex l_u and a new edge ul_u. Let $w(l_u)$ be half the order of the component of $G[V_G\setminus\{u\}]$ that does not intersect O or 0, if no such component exists. Let the tree T' arise from G' by contracting all cycles. Let u denote the vertex of T' that corresponds to the set U and let $N_{T'}(u)=\{u_1,u_2,u_3\}$. For $1\leq i\leq 3$ let T_i be the component of $T'[V_{T'}\setminus\{u\}]$ that contains u_i. If T_i is rooted in u_i, then T_i is a rooted tree with $c_i\geq 0$ non-leaves each of which has exactly two children. Let L_{T_i} denote the set of leaves of T_i. (See Figure 2 for an example of the construction.)

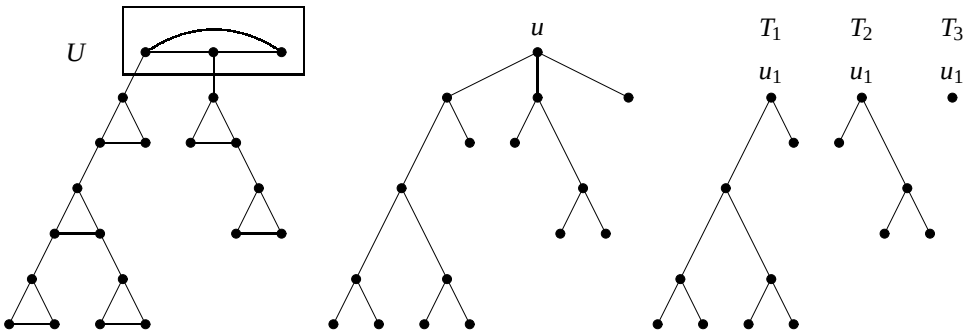


Figure 2 $G[O]$, T' and the trees T_i.

For $1\leq i\leq 3$ let

$$w_i=\sum_{l\in L_{T_i}}w(l).$$

Clearly,

$$c-1=c_1+c_2+c_3$$

and

$$n=3c+2(w_1+w_2+w_3).$$

It is obvious from the construction of the tree T' that the maximal induced subtrees of G that intersect U are in one-to-one correspondence with the maximal paths in T' containing u. Furthermore, if P is a maximal path in T' containing u, then it joins two leaves l and l' of T' that belong to two different subtrees T_i. If $\mathrm{depth}(l)$ and $\mathrm{depth}(l')$ denote the depths of l and l' in the corresponding rooted subtrees T_i, then the maximal induced subtree of G that corresponds to P has order exactly

$$2+2(\mathrm{depth}(l)+w(l))+2(\mathrm{depth}(l')+w(l')).$$

Applying Lemma 2, we obtain

$$\begin{aligned}&\frac{1}{2}\mathrm{it}(G,\emptyset,U)\\ &\geq 1+\sum_{i=1}^{3}\left(\frac{w_i}{c_i+1}+\log_2(c_i+1)-1\right)-\min\left\{\frac{w_i}{c_i+1}+\log_2(c_i+1)-1\mid 1\leq i\leq 3\right\}.\end{aligned}\qquad(1)$$

We assume that $c_1 \geq c_2 \geq c_3$ and estimate the term in (1) in different ways. Clearly, (1) implies

$$\begin{aligned}\frac{1}{2}\mathrm{it}(G,\emptyset,U) &\geq 1+\frac{2}{3}\sum_{i=1}^{3}\left(\frac{w_i}{c_i+1}+\log_2(c_i+1)-1\right)\\ &\geq 1+\frac{2}{3}\left(\frac{w_1+w_2+w_3}{c_1+1}+\log_2(c_1+1)+\log_2(c_2+1)+\log_2(c_3+1)-3\right)\\ &\geq \frac{2}{3}\left(\frac{w_1+w_2+w_3}{c_1+1}+\log_2(c_1+c_2+c_3+1)\right)-1\\ &\geq \frac{2}{3}\left(\frac{n-3c}{2c}+\log_2(c)\right)-1.\end{aligned}$$

Elementary calculus shows that the function $\log_2(x+1)+\log_2(\frac{s-x}{2}+1)$ for $x \in [\frac{s}{3},s]$ is minimum for $x=s$. Therefore, (1) also implies

$$\begin{aligned}\frac{1}{2}\mathrm{it}(G,\emptyset,U) &\geq \log_2(c_1+1)+\log_2(c_2+1)+\log_2(c_3+1)-\min\{\log_2(c_i+1) \mid 1\leq i\leq 3\}-1\\ &\geq \log_2(c_1+1)+\log_2\left(\frac{(c-1)-c_1}{2}+1\right)-1\\ &\geq \log_2(c)-1.\end{aligned}$$

Altogether we obtain

$$\mathrm{it}(G,\emptyset,U) \geq \max\left\{\frac{2}{3}\left(\frac{n-3c}{c}+2\log_2(c)\right)-2, 2\log_2(c)-2\right\}$$

and the proof is complete. □

It is easy to see that Theorem 3 is essentially best-possible in many cases. If for example $c=1$ or $c=\frac{n}{3}$, then one can construct cacti for which the given bound is tight up to some additive constants by reversing the construction illustrated in Figure 2 for appropriate trees T_i.

We proceed to our lower bound for cubic graphs.

Theorem 4 There is some fixed $\alpha \in \mathbb{R}$ such that if G is a connected cubic graph of order n, then

$$\mathrm{it}(G) \geq \frac{4}{3}\log_2(n)+\alpha.$$

PROOF: Iteratively removing vertices $u \in V_G$ from G whose degree (in the remaining graph) is 3 and whose removal does not disconnect the graph yields a cactus G' (cf. [15, 16, 17], the set formed by the removed vertices is usually called a *non-separating independent set*). If i denotes the number of deleted vertices and c denotes the number of cycles of G', then double-counting the edges in G' yields

$$n-i-1=\frac{3}{2}n-3i-c$$

which implies

$$c=\frac{n}{2}-2i+1.$$

Since $0 \leq c \leq \frac{n-i}{3}$, we have $i \in \left[\frac{n+6}{10},\frac{n+2}{4}\right]$. If $c=0$, then G' is a tree and $\mathrm{it}(G) \geq n-i = n-\frac{n+2}{4} = \frac{3n-2}{4}$. Hence we may assume $c \geq 1$.

Since the function

$$f(i) = \frac{(n-i)-3c}{c}+2\log_2(c) = \frac{5i-\frac{n}{2}-3}{\frac{n}{2}-2i+1}+2\log_2\left(\frac{n}{2}-2i+1\right)$$

for $\frac{n}{2}-2i+1 \geq 0$ has a unique minimum in

$$i=\left(\frac{1}{4}-\frac{3\ln(2)}{16}\right)n+\left(\frac{1}{2}+\frac{\ln(2)}{8}\right),$$

we deduce from Theorem 3 that

$$\begin{aligned}\mathrm{it}(G) &\geq \mathrm{it}(G') \geq \frac{2}{3}f(i)-2\\ &\geq \frac{2}{3}\left(\frac{\left(\frac{3}{4}-\frac{15\ln(2)}{16}\right)n+\left(\frac{5\ln(2)}{8}-\frac{1}{2}\right)}{\frac{3\ln(2)}{8}n-\frac{\ln(2)}{4}}+2\log_2\left(\frac{3\ln(2)}{8}n-\frac{\ln(2)}{4}\right)\right)-2\end{aligned}$$

and the proof is complete. □

Whereas the bound given in Theorem 4 is clearly not best-possible with respect to multiplicative and additive constants, it indicates the correct growth rate. Cubic graphs G of arbitrarily large order n for which $\mathrm{it}(G)=O(\log_2(n))$ can easily be constructed generalizing the graph in Figure 3.

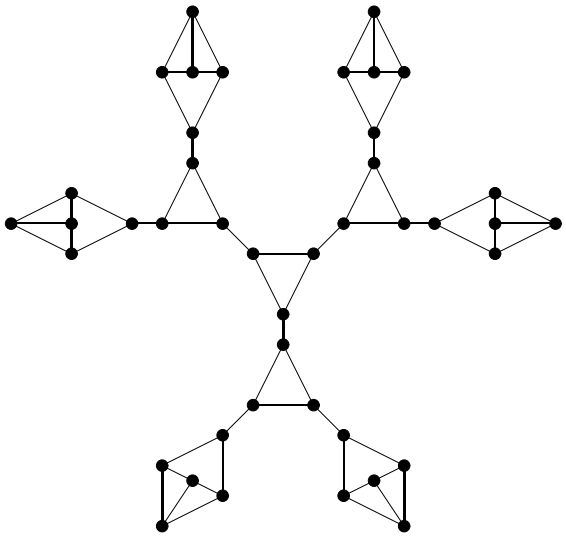


Figure 3

It is possible to extend Theorems 3 and 4 to the case in which the girth of the considered graphs is at least some $g \geq 3$. The essential step of this extension is the proof of a version of Lemma 2 in which one considers rooted trees in which every non-leaf has $g-1$ children and tries to maximize the sum of weights $w(l)+\mathrm{depth}(l)$ over all leaves l in rooted subtrees in which every non-leaf has $g-2$ children. The proofs of Theorems 3 and 4 easily adapt.

Our final result is the following forbidden induced subgraph condition for the existence of induced dominating trees.

Theorem 5 Every induced subgraph of a graph G has an induced dominating tree if and only if G does not contain $C_l \circ K_1$ for any $l \geq 3$ (cf. Figure 1) as an induced subgraph.

PROOF: Since the necessity is trivial, we proceed to the sufficiency. Therefore, let $G=(V_G,E_G)$ be a graph that does not contain $C_l \circ K_1$ for any $l \geq 3$ as an induced subgraph and let T be an induced tree of G with vertex set S such that $|N_G[S]|$ is maximum.

For contradiction, we assume that S is not dominating. This implies that there are vertices $x,y \in V \setminus S$ with $N_G(x) \cap S = \emptyset$ and $y \in N_G(x) \cap N_G(S)$.

Since $G[S \cup \{x,y\}]$ is connected, we can choose a set $S' \subseteq S \cup \{x,y\}$ such that $G[S']$ is connected, $N_G[S'] = N_G[S \cup \{x,y\}]$ and subject to these conditions the number of cycles of $G[S']$ is minimum. We will prove that $G[S']$ is a tree and assume, for contradiction, that $G[S']$ has a cycle C. Clearly, y lies on C and we may assume that

$$C : yt_1t_2\ldots t_l y$$

for some $l \geq 2$ such that

$$N_G(y) \cap \{t_1,t_2,\ldots,t_l\} = \{t_1,t_l\}.$$

If for some $1 \leq i \leq l$

$$N_G(t_i) \subseteq N_G[S' \setminus \{t_i\}],$$

then let $S'' = S' \setminus \{t_i\}$. It is easy to see that $G[S'']$ is connected, $N_G[S''] = N_G[S \cup \{x,y\}]$ and $G[S'']$ has less cycles than $G[S']$ which is a contradiction. Therefore, for $1 \leq i \leq l$ there are vertices $t_i' \subseteq N_G(t_i) \setminus N_G[S' \setminus \{t_i\}]$ and $G[\{t_i, t_i' \mid 1 \leq i \leq l\} \cup \{x,y\}]$ is isomorphic to $C_{l+1} \circ K_1$ which is a contradiction.

Therefore, S' induces a tree in G and $|N_G[S']| > |N_G[S]|$ which is a contradiction to the choice of S. This completes the proof. □

As a concluding remark we want to mention that the third problem posed in [5] "**(3)** *Does there exist a family of* 2*-connected graphs* F*, for which a polynomial time algorithm exists for finding a tree dominating set (of any size) in any graph in* F*, if it exists?*" is phrased a little too vaguely. The trivial answer is 'yes', since the family $\{C_n \mid n \geq 3\}$ of cycles is such a family.

References

[1] S. BAU AND L.W. BEINEKE, The decycling numbers of graphs, *Australas. J. Combin.* **25** (2002), 285-298.

[2] L.W. BEINEKE AND R.C. VANDELL, Decycling graphs, *J. Graph Theory* **25** (1998), 59-77.

[3] J.A. BONDY, G. HOPKINS AND W. STATON, Lower bounds for induced forests in cubic graphs, *Can. Math. Bull.* **30** (1987), 193-199.

[4] A. BRANDSTÄDT AND F.F. DRAGAN, On linear and circular structure of (claw, net)-free graphs, *Discrete Appl. Math.* **129** (2003), 285-303.

[5] X. CHEN, A. MCRAE AND L. SUN, Tree Domination in Graphs, *Ars Combin.* **73** (2004), 193-203.

[6] P. ERDŐS AND Z. PALKA, Trees in random graphs, *Discrete Math.* **46** (1983), 145-150.

[7] W. FERNANDEZ DE LA VEGA, The largest induced tree in a sparse random graph, *Random Struct. Algorithms* **9** (1996), 93-97.

[8] S.M. HEDETNIEMI, S.T. HEDETNIEMI AND D.F. RALL, Acyclic domination, *Discrete Math.* **222** (2000), 151-165.

[9] T.W. HAYNES, S.T. HEDETNIEMI AND P.J. SLATER, Fundamentals of domination in graphs, Marcel Dekker, Inc., New York, 1998.

[10] D. LI AND Y. LIU, A polynomial algorithm for finding the minimum feedback vertex set of a 3-regular simple graph, *Acta Math. Sci.* (English Ed.) **19** (1999), 375-381.

[11] T. LUCZAK AND Z. PALKA, Maximal induced trees in sparse random graphs, *Discrete Math.* **72** (1988), 257-265.

[12] Z. PALKA AND A. RUCINSKI, On the order of the largest induced tree in a random graph, *Discrete Appl. Math.* **15** (1986), 75-83.

[13] M. PROTASI AND M. TALAMO, A parametric analysis of the largest induced tree problem in random graphs, *RAIRO, Inf. Théor. Appl.* **20** (1986), 211-219.

[14] E. SAMPATHKUMAR AND H.B. WALIKAR, The connected domination number of a graph, *J. Math. Phys. Sci.* **13** (1979), 607-613.

[15] E. SPECKENMEYER, On feedback vertex sets and nonseparating independent sets in cubic graps, *J. Graph Theory* **12** (1988), 405-412.

[16] W. STATON, Induced forests in cubic graphs, *Discrete Math.* **49** (1984), 175-178.

[17] S. UENO, Y. KAJITANI AND S. GOTOH, On the nonseparating independent set problem and feedback set problem for graphs with no vertex degree exceeding three, *Discrete Math.* **72** (1988), 355-360.

One-Dimensional Synthesis of Graphs as Tensegrity Frameworks

ANDRÁS RECSKI*

Budapest University of Technology and Economics, Department of Computer Science and Information Theory, and Center for Applied Mathematics and Computational Physics, H-1521 Budapest, Hungary.

OFFER SHAI

Tel-Aviv University, Department of Mechanics Materials and Systems, Faculty of Engineering, Tel-Aviv, Israel.

1 Introduction

The broad definition of tensegrity structures sees them as pin-connected frameworks, where some of the members are cables or struts. Today, tensegrity structures interest researchers in engineering, mathematical and biological communities.

In engineering, tensegrity structures provide efficient solutions in such applications as deployable structures (Tibert, 2002; Guest, 1994), shape-controllable structures, smart sensors (Sultan and Skelton, 2004) and lightweight structures.

Biological community employs tensegrity structures as models underlying the behavior of a number of biological entities, such as the cytoskeleton (Ingber, 1993). Adopting such models enables the biologists to interpret some previously unexplained observed natural phenomena.

The complexity of the behavior on one hand and the special properties on the other are those providing the incentive for mathematical studies of tensegrity structures. The main interest in this respect is concentrated on the issues of checking rigidity (Rescki, 1989; Connelly and Whitley, 1996) and structural analysis of these structures.

A key problem faced when dealing with design of tensegrity structures is the determination of geometrical configurations, at which the given structure becomes rigid. For now, this problem, also referred as the 'form-finding problem' (Tibert and Pellegrino, 2003), does not possess general analytical solution, except for some special relatively simple cases (Connelly and Terrell, 1995).

Current paper addresses a combinatorial approach for treating one-dimensional tensegrity structures, i.e. structures where all members are parallel. The paper establishes a theorem for checking the topological rigidity of these structures, i.e. identifying if for a given graph there exists at least one rigid geometrical embedding. If the latter condition is satisfied, the paper provides a graph-theoretical algorithm for synthesis of the rigid embedding for the given frame topology. This can be regarded as an alternative solution for the 'form-finding problem', although, for now, it is limited for one dimensional structures. Additionally, an algorithm for checking the rigidity of a structure with a given topology is shown to be equivalent to checking whether the corresponding graph is well connected.

It is shown that the methodology can partly be considered as a special case of a more general theorems based on matroid theory (Rescki, 1989), which indicates on the possibility that in the future the method will be expanded for multidimensional cases.

2 Condition for graph embeddability as rigid one-dimensional framework

Let $G=(V,E)$ be a finite graph with vertex set V and edge set E and let χ denote a bipartition $E=E_C\cup E_S$. A function $f:V(G)\to\mathbb{R}$ is called a *one-dimensional embedding* of G if $x\neq y$ implies $f(x)\neq f(y)$.

A function $g:V(G)\to\mathbb{R}$ satisfying

$$|g(x)-g(y)| \quad \begin{cases} \leq |f(x)-f(y)| & \text{if } \{x,y\}\in E_C \\ \geq |f(x)-f(y)| & \text{if } \{x,y\}\in E_S, \end{cases}$$

and

$$sign[g(x)-g(y)]=sign[f(x)-f(y)] \qquad \forall\{x,y\}\in E$$

is called a *motion with respect to the bipartition* χ or shortly a *χ-motion* of the embedded graph G. Such a χ-motion is *trivial* if there exists a constant $c\in\mathbb{R}$ so that $g(x)=f(x)+c$ for every $x\in V(G)$.

A one-dimensional embedding f is called a *one-dimensional rigid embedding* of G with respect to this bipartition, or shortly a *one-dimensional rigid χ-embedding* if every χ-motion of it is trivial.

A circuit C of the graph G is a *mixed circuit* with respect to a bipartition χ, or shortly a *χ-mixed circuit* if neither $C \cap E_C$ nor $C \cap E_S$ is empty.

Theorem 1 A graph has a one-dimensional rigid χ-embedding if and only if the graph is connected and every edge of it is contained by at least one χ-mixed circuit.

Remark 2 The elements of E_C and E_S can be interpreted as cables and struts, respectively, of a tensegrity framework with a given topology G. Since each edge, representing a rod can be replaced by a pair of edges, one representing a cable and one representing a strut, theorem 1 essentially refers to tensegrity frameworks with all three types of elements. Observe that if a framework consists of rods only then the condition of the theorem reduces to the connectivity of the graph, a known condition described in the mathematical literature (Lovász and Yemini (1982)).

PROOF:*I. Necessity.* The connectedness is obvious – if G_0 were a connected component of a disconnected graph G then the function

$$g(x) = \begin{cases} f(x) + c_0 & \text{if } x \in V(G_0) \\ f(x) & \text{otherwise} \end{cases}$$

with $c_0 \neq 0$ would be a nontrivial χ-motion of G. Similarly, if the edge $e = \{a,b\} \in E_S$ (or $\in E_C$, respectively) were a bridge of G and G_0 denotes one of the components of $G - e$ then the same function could be applied using a value of c_0 so that $|g(b) - g(a)|$ must be greater (smaller, respectively) than $|f(b) - f(a)|$.

Hence from now on we may suppose that G is connected and bridgeless. Consider one of its 2-connected components G_0 and suppose indirectly that it has no χ-mixed circuits, that is, all of its edges are in, say, E_C. Let x_0 be a vertex of $V(G_0)$ so that $f(x_0)$ is an internal point of the interval spanned by the values $\{f(v)|v \in V(G_0)\}$. Then $g(x) = f(x_0) + c[f(x) - f(x_0)]$ with some $c < 1$ applied for $x \in V(G_0)$ and then extended by an appropriate constant translation for the remaining elements of $V(G)$ would define a nontrivial χ-motion of G. (If all of the edges of G_0 were in E_S then use the same argument with $c > 1$.)

II. Sufficiency. If every edge of a connected graph G is contained in some circuits then G is clearly bridgeless. Hence it is either 2-connected or has a cactus-decomposition into 2-connected components. It is clearly enough to prove the embeddability for a single 2-connected component.

Lemma 3 A single χ-mixed circuit has a one-dimensional rigid χ-embedding.

PROOF: We may suppose that struts and cables alternate in the circuit (otherwise replace temporarily a maximum path of struts or cables with a single strut (cable, respectively); after embedding this tensegrity framework into the one-dimensional space one can readily finish the original embedding by "subdividing" some struts and cables into smaller ones). Let $[v_0, v_1, v_2, \ldots, v_{k-1}, v_k = v_0]$ be a cyclic description of the vertices of the χ-mixed circuit. Then

- Let $f(v_0)$ be an arbitrary real number and $i = 0$.
- If $i = k - 1$ then stop.
- If $\{v_i, v_{i+1}\} \in E_C$ then "jump to the right", that is, define $f(v_{i+1})$ as an arbitrary value greater than any of the values $f(v_0), f(v_1), \ldots, f(v_i)$.
- If $\{v_i, v_{i+1}\} \in E_S$ then "jump to the left", that is, define $f(v_{i+1})$ as an arbitrary value less than any of the values $f(v_0), f(v_1), \ldots, f(v_i)$.
- Increase the value of i by one and go to the second step.

Figure 1 shows an example of a mixed circuit and its embedding obtained by means of this procedure:

Consider a motion $g(x)$ with respect to the embedding, $f(x)$, obtained by the above process. Without loss of generality let us suppose that $\{v_1, v_2\} \in E_s$, thus by eq(?), the following set of inequalities is satisfied:

$$|g(v_1) - g(v_2)| \geq |f(v_1) - g(f_2)|$$

$$|g(v_2) - g(v_3)| \leq |f(v_2) - g(f_3)|$$

$$\ldots$$

$$|g(v_k) - g(v_1)| \leq |f(v_k) - g(f_1)|$$

By definition of $g(v)$ (Eq?) and the above synthesis procedure for $\{v_i, v_j\} \in E_s$ it follows that $g(v_i) > g(v_j)$ and $f(v_i) > f(v_j)$, while for $\{v_i, v_j\} \in E_c$ it follows that $g(v_i) < g(v_j)$ and $f(v_i) < f(v_j)$. Therefore the above inequalities can now be rewritten without using the absolute values:

$$g(v_1) - g(v_2) \geq f(v_1) - g(f_2)$$

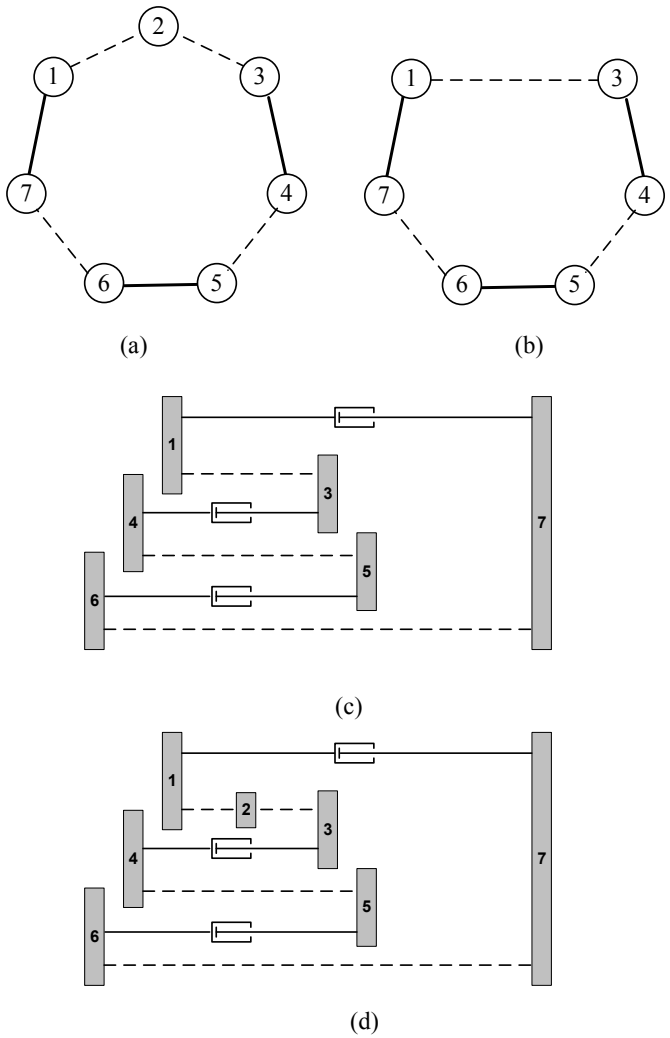


Figure 1: Example of a rigid embedding of a mixed circuit: (a) the mixed circuit; (b) reducing the circuit to an alternating form; (c) rigid embedding of the reduced mixed circuit; (d) the rigid embedding of the original circuit (with the corresponding strut 'subdivided').

$$g(v_2) - g(v_3) \geq f(v_2) - g(f_3)$$
$$\dots$$
$$g(v_k) - g(v_1) \geq f(v_k) - g(f_1)$$

Rearranging terms in above inequalities yields:

$$g(v_1) - f(v_1) \geq g(v_2) - f(v_2) \geq \dots \geq g(v_k) - f(v_k) \geq g(v_1) - f(v_1)$$

Obviously, the latter set of inequalities can be resolved only if $g(x)$ is trivial with respect to $f(x)$, which makes $f(x)$ a rigid embedding. □

Lemma 4 Suppose that a 2-connected proper subgraph G' of a 2-connected graph G has already a one-dimensional rigid χ-embedding and let $[v_0, v_1, \dots, v_k]$ be a path of G so that $\{v_0, v_1, \dots, v_k\} \cap V(G') = \{v_0, v_k\}$. Then this embedding can be extended to a subgraph containing G' and this path. (Here $k \geq 1$, hence we permit that a single edge is added only.)

PROOF: Without loss of generality we may suppose that the edges of the path belong alternatingly to E_C and E_S, see the argument of the first paragraph of the proof of Lemma 3. If $k = 1$ then simply insert the required tensegrity element between the two end points which were already in fixed positions. If $k > 1$ then

- Let $i = 0$.

- If $i = k-1$ then stop.
- If $\{v_i, v_{i+1}\} \in E_C$ then "jump to the right", that is, define $f(v_{i+1})$ as an arbitrary new value greater than any of the values $f(v_0), f(v_1), \ldots, f(v_i)$ and $f(v)$ for any $v \in V(G')$.
- If $\{v_i, v_{i+1}\} \in E_S$ then "jump to the left", that is, define $f(v_{i+1})$ as an arbitrary new value less than any of the values $f(v_0), f(v_1), \ldots, f(v_i)$ and $f(v)$ for any $v \in V(G')$.
- Increase the value of i by one and go to the second step.

The rigidity of the resulting embedding can be proved in a similar fashion, as it was done for Lemma 3. □

Now *the proof of the sufficiency* is obvious by considering the cactus-decomposition of G and realizing the embedding of the individual 2-connected components as follows: Start with a mixed circuit as in Lemma 3 and then extend it gradually, as in Lemma 4, with new paths (including the possibility of single new edges as well). □

Figure 2 shows an example of realizing such embedding of a graph.

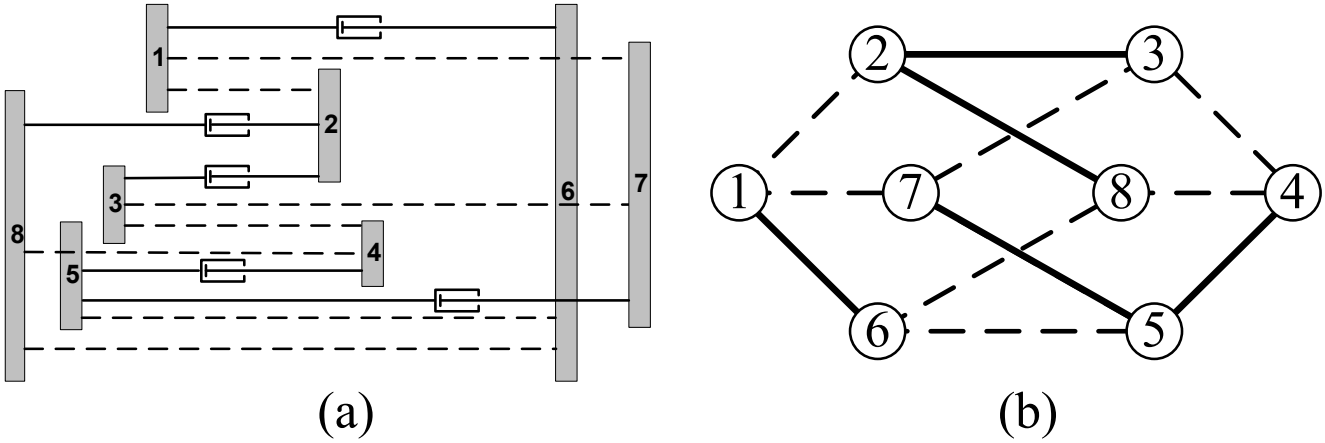


Figure 2: Example of a rigid embedding of a complex graph: (a) the rigid embedding; (b) the corresponding graph

3 Condition for rigidity of a given one-dimensional framework

Consider a one-dimensional embedding, F, of a tensegrity framework. The corresponding directed graph representation, G_F, is constructed through the following steps:

- For each joint i in F add a vertex, v_i, to G_F.
- For each cable/strut, c_{ij}/s_{ij}, in F connected to joint i from the left and joint j from the right, add a directed edge $e =< i, j > / < j, i >$ to G_F.

With respect to G_F the condition for a function, $g(x)$ to be a valid motion function obtains the following form:

$$g(h) - g(t) \leq f(h) - f(t) \quad \forall e =< h,t > \in G_F$$

Theorem 5 A given one-dimensional tensegrity framework, F, is rigid if and only if, the corresponding directed graph, G_F, is strongly connected.

PROOF:*Necessity.* Let us suppose indirectly that G_F possesses a directed cut-set, which separates G_F into two connected subgraphs, G_h and G_t, connected respectively to the head and the tail vertices of the edges belonging to the cut-set. Then the function:

$$g(x) = \begin{cases} f(x) + c_0 & if\ x \in G_h \\ f(x) & if\ x \in G_t \end{cases}$$

with $c_0 \neq 0$ would be a valid nontrivial motion of F.

II.Sufficiency. For any two vertices $x,y \in V(G_F)$ there is a directed path $\{x,v_1,...,v_k,y\}$ from x to y. By, eq..., the motion function g for G_F should satisfy the following inequalities for the vertices belonging to the path:

$$g(x) - g(v_1) \leq f(x) - f(v_1)$$

$$\ldots$$

$$g(v_k) - g(y) \leq f(v_k) - f(y)$$

or $g(x) - f(x) \leq g(y) - f(y)$. Similarly, there is a directed path $\{y,v'_1,\ldots,v'_k,x\}$ from y to x yielding $g(y) - f(y) \leq g(x) - f(x)$. In combination, the two inequalities yield:

$$g(x) - g(y) = f(x) - f(y) \quad \forall x,y \in V(G_F)$$

Obviously, the latter equality means that F can sustain only a trivial motion and is thus rigid. □

It is interesting to note that Theorem 5 can be considered a special case of a more general theorem developed by the first author on the basis of matroid theory. Theorem 18.3.2 in (Recski, 1989), related to tensegrity frameworks of any dimensionality and is formulated as follows:

Theorem 6 Let F be a tensegrity framework and suppose that the underlying system F' is rigid (i.e. dynamically determined). Suppose that the oriented matroid $\mathbf{M}(F)$ is graphic and is described by a directed graph G. Then F is rigid if and only if the tensegrity transformation of G is strongly connected.

$\mathbf{M}(F)$ in Theorem 6 is the oriented matroid coordinated by the row vectors of the rigidity matrix of the tensegrity framework, F; and the tensegrity transformation of G reverses the orientation of the edges corresponding to struts.

In one-dimensional case, the rigidity matrix is actually the transposed incidence of F, where each column is multiplied by the corresponding member length. Thus, in this case, $\mathbf{M}(F)$ would always be a graphic matroid, the description of which is the G_F itself.

References

1. Tibert, G. (2002) Deployable Tensegrity Structures for Space Applications. Doctoral thesis. Royal Institute of Technology, Stockholm
2. Guest, S.D. (1994). Deployable Structures: Concepts and Analysis, PhD Thesis, Cambridge University.
3. Sultan C, Skelton R. (2004). A force and torque tensegrity sensor. Sensors and Actuators A-Physical, vol.A112, no.2-3, 1 May 2004, pp.220-31.
4. Ingber, D.E. (1993), Cellular tensegrity: defining new rules of biological design that govern the cytoskeleton, Journal of Cell Science, 104 , 613-627.
5. Recski (1989)
6. Connelly, R., Whiteley, W. (1996), Second-Order Rigidity and Prestress stability for tensegrity frameworks. SIAM J. Discrete Math., 9:453-491.
7. Tibert, A G. Pellegrino, S. (2003), Review of form-finding methods for tensegrity structures Space Structures, V 18 n 4, p 209-223
8. Connelly, R., Terrell, M., (1995), Globally rigid symmetric tensegrities. Structural Topology, 21:59-78.
9. Lovász and Yemini (1982)
10. Diestel's book

Circuit Switched Broadcastings and Digit Tilings on Torus Networks*

RYOTARO OKAZAKI

Department of Knowledge Engineering and Computer Sciences,
Doshisha University
1-3 Tataramiyakodani, Kyotanabe-shi, Kyoto-fu, 610-0394 Japan,
rokazaki@mail.doshisha.ac.jp

HIROTAKA ONO

Department of Computer Science and Communication Engineering
Kyushu University
6-10-1, Hakozaki, Fukuoka, 812-8581, Japan
ono@csce.kyushu-u.ac.jp

TAIZO SADAHIRO

Department of Administration
Kumamoto Prefectural University
3-1-100 Tsukide Kumamoto, Japan
sadahrio@pu-kumamoto.ac.jp

MASAFUMI YAMASHITA

Department of Computer Science and Communication Engineering
Kyushu University
6-10-1, Hakozaki, Fukuoka, 812-8581, Japan
mak@csce.kyushu-u.ac.jp

Abstract: Peters and Syska[1] showed efficient ciurcuit switched broadcastings on two-dimensional torus networks. We reformulate this algorithm using the numeration systems and tilings on finite tori and extend the algorithm to three-dimensional torus networks.

Keywords: broadcasting, torus network, numeration system, tiling

1 Broadcasting problems on circuit switching networks

Peters and Syska[1] showed efficient ciurcuit switched broadcastings on two-dimensional torus networks. It is interesting that they use certain tilings which seem to be very similar to those so-called digit tilings[2]. This paper shows these tilings are modifications of digit tilings generated by the numeration systems on finite torus. This formulation gives algebraic and clear description of the algorithm and the proof. We also extends the algorithm to higher dimensional cases. Our construction is very similar to those of generalized van der Corput sequence[3].

First we give the mathematical formulation of the *broadcasting* problem we discuss in this paper. We consider the broadcastings of short messages on the circuit switching networks. Let G be an undirected graph with the vertex set $V(G)$ and the edge set $E(G)$. A sequence of distinct vertices $w = v_0 v_1 \cdots v_k$ is called a *path* of G if v_i and v_{i+1} are connected by an edge of G for $0 \leq i \leq k-1$. We call k the *length* of the path w and denote it by $l(w)$. We call v_0 the *starting point* of w and denote it by $s(w)$, and we call v_k the *terminal point* of w and denote it by $t(w)$. For a set W of paths in G we denote the set $\{t(w) \mid w \in W\}$ (resp. $\{s(w) \mid w \in W\}$) by $t(W)$ (resp. $s(W)$). We assume a node (on a vertex) in G can send a message to at most p other nodes at the same time through paths in G where p is a positive integer fixed for G. The transmission time for a message to be sent through the path w is assumed to be $\alpha + l(w)\delta$, where α and δ are some positive constant. Let $o = v_0, v_1, \ldots, v_k = v \in V(G)$ be $k+1$ vertices on a graph G. Consider the situation in which a message is sent from the originator o to the node v_1 through a path w_1, v_1 forwards it to v_2 through a path w_2, and so on finally v_k receive the message:

$$(o =) v_0 \xrightarrow{w_1} v_1 \xrightarrow{w_2} \cdots \xrightarrow{w_{k-1}} v_{k-1} \xrightarrow{w_k} v_k (= v)$$

Then, in our model, it is assumed to take

$$k\alpha + \delta \sum_{i=1}^{k} l(w_i) \tag{1}$$

time for the node v to receive the message. A node which has received a message can forward it to p other nodes which have not received it. By repeating the forwarding, a message which originated from a node is transmitted to all of the nodes. We call this process a *broadcasting*. To be precise, we define the broadcasting on the graph G as follows.

*Partly supported by the Grant-in-Aid for Scientifi c Research on Priority Areas, The Ministry of Education, Culture, Sports, Science and Technology, Japan (No.766)

Definition 1 Let G be an undirected graph with the vertex set $V(G)$ and the edge set $E(G)$. A broadcasting on G with the originator $o \in V(G)$ is a family $\{W_r\}_{r=1}^{R}$ of sets of paths in G which satisfies the following conditions.

1. $s(W_1) = \{o\}$
2. $s(W_r) \subset \bigcup_{i=1}^{r-1} t(W_i) \cup \{o\}$
3. $\sharp\{w \in W_r \mid s(w) = v\} \leq p$ for all $v \in t(W_{r-1})$.
4. $\bigcup_{r=1}^{R} t(W_r) \cup \{o\} = V(G)$, $\sharp\left(\cup_{r=1}^{R} W_r\right) = \sharp V(G) - 1$.
5. Any two paths in W_r do not intersect each other except at their starting points.

We call the index r of W_r the *round* of the broadcasting. For a broadcasting $\{W_r\}_{r=1}^{R}$ we define $I(W,r) = \bigcup_{i=1}^{r} t(W_r) \cup \{o\}$ for $1 \leq r \leq R$ and call it *informed nodes* at the r-th round. For the convenience we define $I(W,0) = \{o\}$. So the condition 2 above can be interpreted as saying that only informed nodes can forward the message, and the condition 4 means that all of the nodes are informed at the R-th round. By the conditions $4,5$ above, for any vertex $v \in V(G)$, there exists a unique sequence of paths $w_1 w_2 \cdots w_k$ where $w_r \in W_r$ for $1 \leq i \leq k$, $t(w_{r-1}) = s(w_r)$ for $2 \leq r \leq k$ and $t(w_k) = v$. We define $l_{W,v} = \sum_{i=1}^{k} l(w_i)$ and call $\max_{v \in V(G)}\{l_{W,v}\}$ the *total path length* of the broadcasting $\{W_r\}$. The problem is to design the efficient broadcasting, the broadcasting with small maximum round R and total path length. We have the following obvious lower bounds for R and total path length :

$$R \geq \log_{p+1} \sharp V(G)$$

and

$$\max_{v \in V(G)} \{l_{W,v}\} \geq \Delta(G)$$

where $\Delta(G)$ is the diameter of the graph.

Here we give the definition of the *torus networks*, on which we construct the efficient broadcastings. Let n, d be positive integers. We define the vector $\mathbf{e}_i$ by

$$\mathbf{e}_i = (\overbrace{0,0,\cdots,0}^{i-1},1,\overbrace{0,\cdots,0}^{d-i})$$

for $i \in \{1,2,\ldots,n\}$ and

$$S = \{\pm \mathbf{e}_k \mid k = 1,2,\cdots,d\}.$$

The *d-dimensional torus network* of size n is the graph G which is defined as follows. The vertex set $V(G)$ of G is $\mathbb{Z}^d/n\mathbb{Z}^d$. Two vertices $\mathbf{x}, \mathbf{y} \in V(G)$ are connected by an edge if and only if $\mathbf{x} - \mathbf{y} \in S$. Hence the torus network G is the Cayley graph of $\mathbb{Z}^d/n\mathbb{Z}^d$ with the generater S. Figure 1 shows a torus network of size 5.

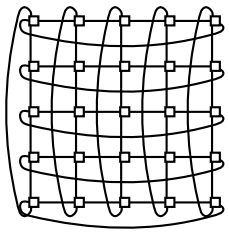


Figure 1: The two dimensinal torus network of size 5

2 Numeration systems and digit tiilngs on fi nite tori

Let $B \in \mathrm{M}_d(\mathbb{Z})$ be a d-dimensional integer matrix which satisfies

$$B^d = bI,$$

where $b = \det B$ and I is the identity matrix. Let $\mathscr{D} \subset \mathbb{Z}^d$ be a set of complete coset representatives of $\mathbb{Z}^d/B\mathbb{Z}^d$, that is, for any point $\mathbf{x} \in \mathbb{Z}^d$, there is a unique element $\mathbf{d}_\mathbf{x}$ such that $\mathbf{x} \equiv \mathbf{d}_\mathbf{x} \pmod{B\mathbb{Z}^d}$. Then we have the following lemma,

Lemma 2 Let m be a positive integer and let $b = \det B$. For any $\mathbf{x} \in \mathbb{Z}^d/b^m\mathbb{Z}^d$ there exist unique sequence $a_k \in \mathscr{D}$, $0 \leq k \leq dm-1$, such that,

$$\mathbf{x} = \mathbf{a}_0 + B\mathbf{a}_1 + \cdots + B^{dm-1}\mathbf{a}_{dm-1} \pmod{b^m\mathbb{Z}^d} \quad (2)$$

This lemma implies that we can express any point of $\mathbb{Z}^d/b^m\mathbb{Z}^d$ can be expressed as a word of length dm over the alphabet $\mathscr{D}$. We define

$$\mathsf{T}^{(k)} = \left\{ \mathbf{a}_0 + B\mathbf{a}_1 + \cdots + B^k\mathbf{a}_k \mid \mathbf{a}_i \in \mathscr{D} \right\},$$

and

$$L^{(k)} = \left\{ B^{k+1}\mathbf{a}_{k+1} + B^{k+2}\mathbf{a}_{k+2} + \cdots + B^{dm-1}\mathbf{a}_{dm-1} \mid \mathbf{a}_i \in \mathscr{D} \right\}.$$

Then the vertices $V(G) = \mathbb{Z}^d/b^m\mathbb{Z}^d$ of the torus network of size b^m is partitioned into disjoint subsets,

$$V(G) = \bigcup_{\mathbf{c}\in L^{(k)}} \left(\mathbf{c} + \mathsf{T}^{(k)} \right) \pmod{b^m\mathbb{Z}^d}. \tag{3}$$

We call $\mathbf{c} + \mathsf{T}^{(k)}$ a *tile of level k* where $\mathbf{c} \in L^{(k)}$. The equation (3) means $\mathsf{T}^{(k)}$ can be covered by the translates of $\mathsf{T}^{(k)}$ without overwrappings. A tile is partitioned into tiles of smaller levels,

$$\mathsf{T}^{(k+1)} = \bigcup_{\mathbf{c}\in B^{k+1}\mathscr{D}} \left(\mathbf{c} + \mathsf{T}^{(k)} \right).$$

We use this property to divide the network recursively.

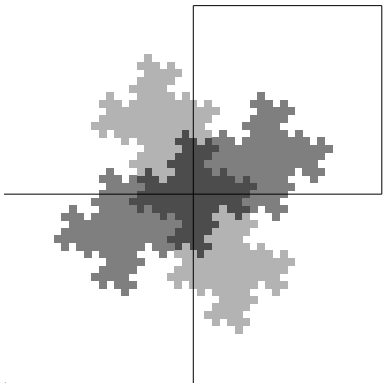


Figure 2: Tilings of the torus

Example 3 Let $B = \begin{pmatrix} 2 & 1 \\ 1 & -2 \end{pmatrix}$ and $\mathscr{D}\{(0,0)^T, \pm(1,0)^T, \pm(0,1)^T\}$. Then $\mathscr{D}$ forms a set of complete coset representatives of $\mathbb{Z}^2/B\mathbb{Z}^2$. To draw the pictures of tilings, we define

$$\widehat{\mathsf{T}}^{(k)} = \mathsf{T}^{(k)} + U$$

where $U = \{(x,y) \mid -0.5 \le x \le 0.5, -0.5 \le y \le 0.5\}$. Figure 2 shows $\widehat{\mathsf{T}}^{(3)}$ and its translates tesselate $\mathbb{R}^2/5^2\mathbb{Z}^2$.

3 The broadcasting algorithm

3.1 The algorithm

In this section we describe the broadcasting algorithm by Peters and Syska on the torus network of size 5^m in our formulation using the numeration systems and the tilings. We use the following base B and the digit sets $\mathscr{D}$,

$$B = \begin{pmatrix} 2 & 1 \\ 1 & -2 \end{pmatrix}, \quad \mathscr{D} = \left\{ (0,0)^T, \pm(1,0)^T, \pm(0,1)^T \right\}.$$

Then we have $B^2 = 5I$ and $\mathscr{D}$ is a set of complete coset representatives of $B\mathbb{Z}^2$. Therefore, for any positive integer m and any point $\mathbf{x} \in \mathbb{Z}^2/5^m\mathbb{Z}^2$, there exists a unique sequence $\{\mathbf{a}_k\}_{k=0}^{2m-1}$ of $\mathscr{D}$ such that

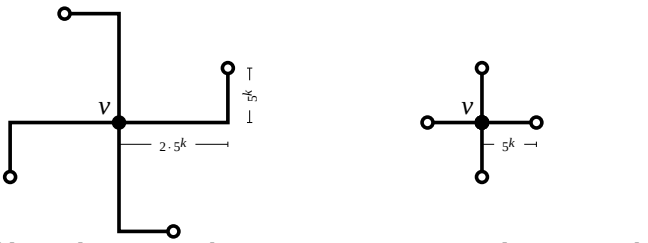
$$\mathbf{x} = \mathbf{a}_0 + B\mathbf{a}_1 + \cdots + B^{2m-1}\mathbf{a}_{2m-1} \pmod{5^m\mathbb{Z}^2}. \tag{4}$$

Every vertex $\mathbf{v}$ is connected to four neighbours,$\mathbf{v}\pm\mathbf{e}_1, \mathbf{v}\pm\mathbf{e}_2$. We denote the edges which connect $\mathbf{v}$ to $\mathbf{v}+\mathbf{e}_1, \mathbf{v}-\mathbf{e}_1, \mathbf{v}+\mathbf{e}_2$ and $\mathbf{v}-\mathbf{e}_2$ by $X, \overline{X}, Y, \overline{Y}$ respectively. Thus we can express a path starting from a vertex by a word over the alphabet $\{X, Y, \overline{X}, \overline{Y}\}$. For example, we denote by $(\mathbf{o}, YXX\overline{Y})$ the path which goes through the vertices,

$$\mathbf{o} = (0,0)^T, (0,1)^T, (1,1)^T, (2,1)^T, (2,0)^T.$$

We denote by α^k the k-times repetition of the letter α. So we can express $YYYY$ by Y^4.

The broadcasting $\{P_r\}_{r=1}^{2m}$ on torus networks of size 5^m consists of $2m$ rounds. A vertex $\mathbf{v}$ which has received the message at the $r-1$ round forwards the message to other four vertices $\mathbf{v}+B^{2m-r}\mathscr{D}^*$ which have not received the message, where $\mathscr{D}^* = (\mathscr{D}\backslash\{\mathbf{0}\})$. We then show the paths in P_r.



odd round $2m-r=2k+1$ even round $2m-r=2k$

Figure 3: paths in P_r starting from v

- $r \equiv 1 \bmod 2$

  There exists an integer $k \in \{0,1,\cdots,m-1\}$ such that $2m-r=2k+1$. The vertex $\mathbf{v}$ sends the message to $\mathbf{v}+B^{2k+1}\mathscr{D}^* = \mathbf{v}+\left\{\pm 5^k(2,1)^T, \pm 5^k(1,-2)^T\right\}$ through the four paths shown in the left side of Figure 3, which can be expressed as

  $$(\mathbf{v}, X^{2u}Y^u),\ (\mathbf{v}, \overline{X}^{2u}\overline{Y}^u)\ (\mathbf{v}, Y^{2u}\overline{X}^u),\ (\mathbf{v}, \overline{Y}^{2u}X^u).$$

  where $u = 5^k$.

- $r \equiv 0 \bmod 2$

  There exists a integer $k \in \{0,1,\cdots,m-1\}$ such that $2m-r=2k$. The vertex $\mathbf{v}$ sends the message to four vertices $\mathbf{v}+B^{2k}\mathscr{D}^* = \mathbf{v}+\left\{\pm 5^k(1,0)^T,\ \pm 5^k(0,1)^T\right\}$ through the paths in the right side of Figure 3. These paths are expressed as

  $$(\mathbf{v}, X^u),\ (\mathbf{v}, \overline{X}^u),\ (\mathbf{v}, Y^u),\ (\mathbf{v}, \overline{Y}^u),$$

  where $u = 5^k$.

Figure 4 shows the four rounds of this broadcasting where $m=2$. The black circles ● show vertices which have already received the message, the white circles ○ show the vertices which has newly received the message at each round.

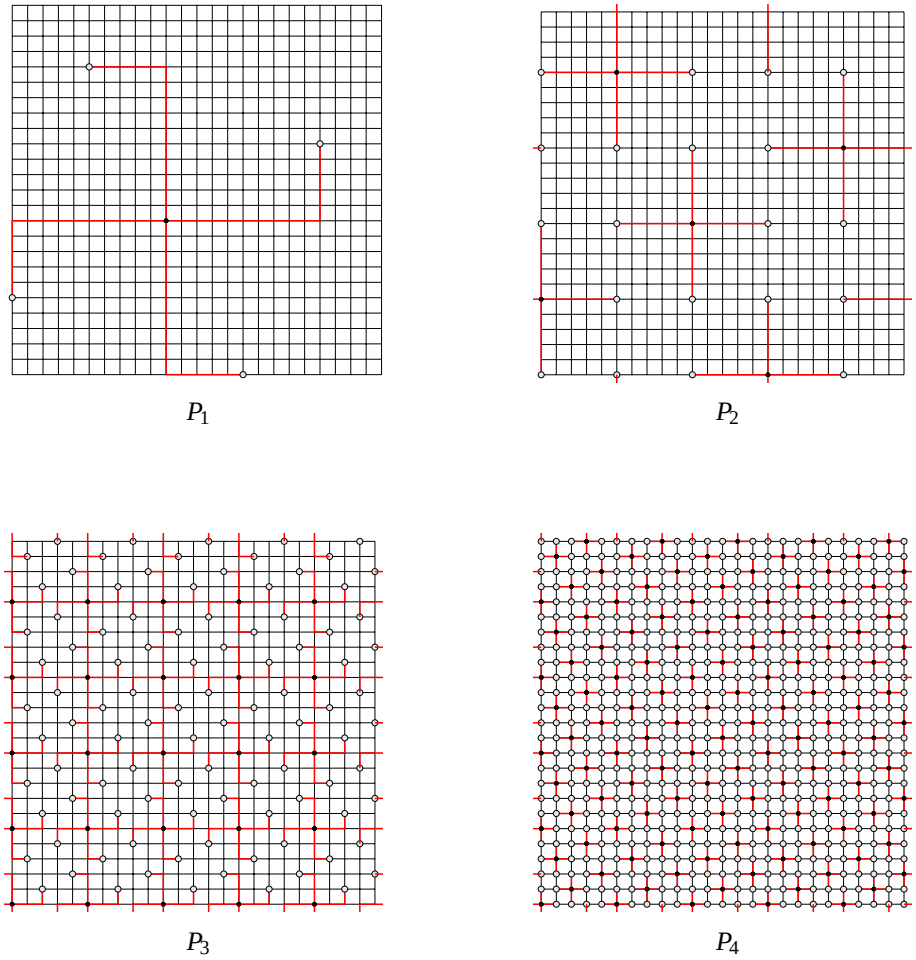


Figure 4: A broadcasting on the torus network $\mathbb{Z}^2/5^2\mathbb{Z}^2$

3.2 The proof of the algorithm

We use the following two lemmas to prove the family $\{P_r\}$ forms a broadcasting on the torus network.

Lemma 4 Let n be a integer with $|n| \leq 5^m$. Then there exists a unique sequence $\{a_i\}_{i=0}^m$ of $\{0, \pm 1, \pm 2\}$ such that

$$n = a_0 + 5a_1 + 5^2 a_2 + \cdots + 5^m a_m, \quad |a_m| \leq 1.$$

Lemma 5 Let $n \in \mathbb{Z}$ be an integer with $0 \leq n \leq 5^k$. Then we have

$$(n,0)^T \in \mathsf{T}^{(2k-1)} \cup \left((5^k,0)^T + \mathsf{T}^{(2k-1)}\right) \tag{5}$$

$$(-n,0)^T \in \mathsf{T}^{(2k-1)} \cup \left((-5^k,0)^T + \mathsf{T}^{(2k-1)}\right)$$

$$(0,n)^T \in \mathsf{T}^{(2k-1)} \cup \left((0,5^k)^T + \mathsf{T}^{(2k-1)}\right) \tag{6}$$

$$(0,-n)^T \in \mathsf{T}^{(2k-1)} \cup \left((0,-5^k)^T + \mathsf{T}^{(2k-1)}\right).$$

And hence

$$(\pm n,0)^T, (0,\pm n)^T \in \mathsf{T}^{(2k)}.$$

This implies these points can be expressed in the form

$$\mathbf{a}_0 + B\mathbf{a}_1 + B^2\mathbf{a}_2 + \cdots + B^{2k}\mathbf{a}_{2k} \tag{7}$$

where $\mathbf{a}_i \in \mathscr{D}$ for $0 \leq i \leq 2k$.

PROOF: By Lemma 4, $(n,0)$ can be expressed in the form,

$$(n,0)^T = (a_0,0)^T + 5I(a_1,0)^T + \cdots + 5^{k-1}I(a_{k-1},0)^T + 5^k I(a_k,0)^T. \tag{8}$$

where $a_i \in \{0,\pm 1,\pm 2\}$ and $|a_k| \leq 1$. Substituting $(2,0)^T = (0,-1)^T + B(1,0)^T$ to (8), we have the form (7). Thus (5) is proved. The point $(0,n)$ with $n \leq 5^k$ can be expressed in the form,

$$(0,n)^T = (0,b_0)^T + 5I(0,b_1)^T + \cdots + 5^{k-1}I(0,b_{k-1})^T + 5^k I(0,b_k)^T, \tag{9}$$

where $b_i \in \{0,\pm 1,\pm 2\}$ and $|b_k| \leq 1$. Substituting $(0,2)^T = (0,-1)^T + B(0,-1)^T$ to (9), we have (7). Thus (6) is proved. □

Theorem 6 The family $\{P_r\}_{r=1}^{2m}$ forms a broadcasting with the total path length $\Delta(G)$.

PROOF: We first show $\{P_r\}_{r=1}^{2m}$ satisfies the condition 5, that is, any two paths in P_r do not intersect each other. Using the fact $I(P,r-1) = L^{(2m-r+1)}$, $V(G)$ can be partitioned into disjoint subsets,

$$V(G) = \bigcup_{\mathbf{v} \in I(P,r-1)} \left(\mathbf{v} + \mathsf{T}^{(2m-r)}\right).$$

For a vertex $\mathbf{v} \in I(P,r-1) = L^{(2m-r+1)}$ we define $P_r(\mathbf{v}) = \{w \in P_r \mid s(w) = \mathbf{v}\}$, that is, $P_r(\mathbf{v})$ consists of paths in P_r starting from the vertex $\mathbf{v}$. We then show all of the vertices on the paths in $P_r(\mathbf{v})$ is contained in $\mathbf{v} + \mathsf{T}^{(2m-r)}$.

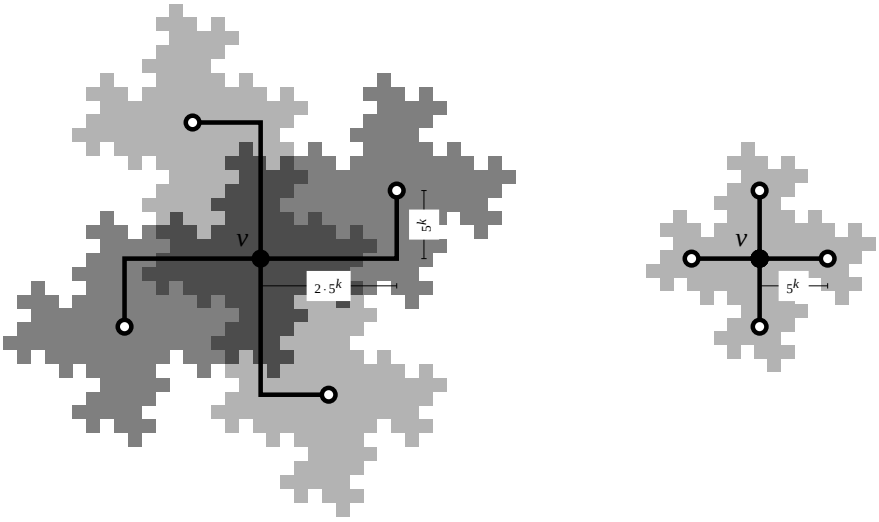


Figure 5: $P_r(v)$ is contained in $\mathbf{v} + \mathsf{T}^{(2m-r)}$

- When $r \equiv 0 \bmod 2$, it follows from Lemma 5 that all of the vertices on $P_r(\mathbf{v})$ are contained in the tile $\mathbf{v} + \mathsf{T}^{(2m-r)}$.
- When $r \equiv 1 \bmod 2$, we can show the vertices on $P_r(\mathbf{v})$ are contained in $\mathbf{v} + \mathsf{T}^{(2m-r)}$ as follows. There exists a integer $k \in \{0,1,\cdots,m-1\}$, such that $2k+1 = 2m-r$. First we check the three points

  $$\mathbf{v}_1 = \mathbf{v} + (5^k,0)^T,\ \ \mathbf{v}_2 = \mathbf{v} + (2\cdot 5^k,0)^T,\ \ \mathbf{v}_3 = \mathbf{v} + (2\cdot 5^k, 5^k)$$

  on the path $w = (\mathbf{v}, X^{2u}Y^u)$ are contained in $\mathbf{v} + \mathsf{T}^{(3k+1)} = \mathbf{v} + \mathsf{T}^{(2m-r)}$. These three points are expressed as

  $$\begin{aligned} \mathbf{v}_1 &= \mathbf{v} + B^{2k}(1,0)^T, \\ \mathbf{v}_2 &= \mathbf{v} + B^{2k+1}(1,0)^T + B^{2k}(0,-1)^T, \\ \mathbf{v}_3 &= \mathbf{v} + B^{2k+1}(1,0)^T \end{aligned}$$

  and hence are contained in the tile $\mathbf{v} + \mathsf{T}^{(2m-r)}$. The difference $\mathbf{v}_{i+1} - \mathbf{v}_i$ are of the form $(5^k,0)$ or $(0,5^k)$. And so all of the points on the path $w = (\mathbf{v}, X^{2u}Y^u)$ are contained in the tile $\mathbf{v} + \mathsf{T}^{(2m-r)}$. In the same manner we can show other vertices on the paths in $P_r(\mathbf{v})$ are contained in $\mathsf{T}^{(3k+2)}$.

We then prove the last part of the statement. Every vertex of the 2-dimensional torus networks have the degree 4. Hence the round number $2m = \log_{4+1} 5^{2m}$ is optimal. When r is odd, all of the paths contained in P_r are of length $3 \cdot 5^{(2m-r-1)/2}$ and when r is evern paths in P_r are of length $5^{(2m-r)/2}$. Hence the total path length is

$$3 \cdot 5^{m-1} + 5^{m-1} + 3 \cdot 5^{m-2} + 5^{m-2} + \cdots + 3 + 1 = 5^{m-1} - 1,$$

which is equal to the diameter of the network. □

4 Three-dimensional cases

We can extend this broadcasting alogrithm to 3-dimensional cases using

$$B = \begin{pmatrix} -1 & 1 & -1 \\ -2 & -1 & 0 \\ 1 & 1 & 2 \end{pmatrix}, \quad \mathscr{D} = \left\{ (0,0,0)^T, \pm(1,0,0)^T, \pm(0,1,0)^T, \pm(0,0,1)^T \right\}.$$

But from this setting, we have broadcastings with total path length $11/9\Delta(G)$. The authors do not know whether there exists a broadcasting with shorter total path length.

Acknowledgment

The authors would like to thank Joseph Peters for helpful discussions and suggestions.

References

[1] Joseph G. Peters and Michel Syska, *Circuit switched Broadcasting in Torus Networks* IEEE Transactions on Parallel and Distributed Systems **7** (1996), 246–255

[2] Andrew Vince *Digit tiling of Euclidean space* Directions in mathematical quasi-crystals, 329–370, CRM Monogr. Ser., **13**, Amer. Math. Soc., Providence, RI, (2000)

[3] Takahiko Fujita, Shunji Ito, and Syoiti Ninomiya *The generalized van der Corput sequence and its application to numerial integrations*, Monte Carlo Methods and Applications, Vol8, No2,2002,149-158

Laminar Covering Problem

MARIKO SAKASHITA

Graduate School of Informatics,
Kyoto University,
Kyoto, 606-8501, Japan.
sakasita@amp.i.kyoto-u.ac.jp

KAZUHISA MAKINO

Graduate School of Engineering Science,
Osaka University,
Toyonaka, Osaka, 560-8531, Japan.
makino@sys.es.osaka-u.ac.jp

SATORU FUJISHIGE

Research Institute for Mathematical Sciences,
Kyoto University,
Kyoto, 606-8502, Japan.
fujishig@kurims.kyoto-u.ac.jp

Abstract: Let V be a finite set with $|V| = n$. A family $\mathscr{F} \subseteq 2^V$ is called *laminar* if for all two sets $X, Y \in \mathscr{F}$, $X \cap Y \neq \emptyset$ implies $X \subseteq Y$ or $X \supseteq Y$. Given a laminar family $\mathscr{F}$, a demand function $d : \mathscr{F} \to \mathbb{R}_+$, and a monotone concave cost function $F : \mathbb{R}_+^V \to \mathbb{R}_+$, we consider the problem of finding a minimum-cost $x \in \mathbb{R}_+^V$ such that $x(X) \geq d(X)$ for all $X \in \mathscr{F}$. We show that the problem can be solved in $O(n^2q)$ time if F can be decomposed into monotone concave functions by the partition of V that is induced by the laminar family $\mathscr{F}$, where q is the time required for the computation of $F(x)$ for any $x \in \mathbb{R}_+^V$. We also prove that if F is given by an oracle, then it takes $\Omega(n^2q)$ time to solve the problem, which implies that our $O(n^2q)$ time algorithm is optimal in this case. Furthermore, we propose an $O(n \log^2 n)$ algorithm if F is the sum of linear cost functions with fixed setup costs. Finally, we show that in general our problem requires $\Omega(2^{\frac{n}{2}}q)$ time when F is given implicitly by an oracle, and that it is NP-hard if F is given explicitly.

Keywords: Graph algorithm, Source location problem, Laminar family.

1 Introduction

Let V be a finite set with $|V| = n$. A family $\mathscr{F} \subseteq 2^V$ is called *laminar* if for arbitrary two sets $X, Y \in \mathscr{F}$, $X \cap Y \neq \emptyset$ implies $X \subseteq Y$ or $X \supseteq Y$. Given a laminar family $\mathscr{F}$, a demand function $d : \mathscr{F} \to \mathbb{R}_+$, and a monotone concave function $F : \mathbb{R}_+^V \to \mathbb{R}_+$, the problem to be considered in this paper is given as

$$\begin{array}{lll} \text{Minimize} & F(x) & \\ \text{subject to} & x(X) \geq d(X) & (X \in \mathscr{F}), \\ & x(v) \geq 0 & (v \in V), \end{array} \tag{1}$$

where $\mathbb{R}_+$ denotes the set of all nonnegative reals, and $x(X) = \sum_{v \in X} x(v)$ for any $X \subseteq V$. The present problem has various applications, since laminar families represent hierarchical structures in many organizations. Moreover, such applications include the source location problem and the edge-connectivity augmentation problem in undirected graphs, which do not seemingly have laminar structures. We shall show in Section 3 that they can be formulated as (1) by using extreme sets in given networks.

In this paper, we study the following three cases, in which the cost functions F are expressed as

$$\text{(i)} \quad F_1(x) = \sum_{X \in \mathscr{F}} f_{\Delta X}(x[\Delta X]) \quad \text{(laminar sum)},$$

$$\text{(ii)} \quad F_2(x) = \sum_{v \in V} f_v(x(v)) \quad \text{(separable)},$$

$$\text{(iii)} \quad F_3(x) = \sum_{v \in V : x(v) > 0} (a_v x(v) + b_v) \quad \text{(fixed-cost linear)},$$

where $\Delta X = X - \bigcup \{Y \mid Y \in \mathscr{F},\ Y \subsetneq X\}$, $x[\Delta X]$ denotes the projection of x on ΔX, $f_{\Delta X} : \mathbb{R}_+^{\Delta X} \to \mathbb{R}_+$ and $f_v : \mathbb{R}_+^{\{v\}} \to \mathbb{R}_+$ are monotone concave, and a_v and b_v are nonnegative constants. It is clear that F_2 is a special case of F_1, and F_3 is a special case of F_2 (and hence of F_1). We shall show that if $F = F_1$, the problem can be solved in $O(n^2q)$ time, where q is the time required for the computation of $F(x)$ for each $x \in \mathbb{R}_+^V$. The problem requires $\Omega(n^2q)$ time if $F (= F_2)$ is given by an oracle.

This implies that our $O(n^2q)$ algorithm is optimal if $F(=F_1)$ is given by an oracle. Moreover, we show that the problem can be solved in $O(n\log^2 n)$ and $O(n(\log^2 n+q))$ time if F is explicitly and implicitly given as F_3, respectively, and the problem is intractable in general. Table 1 summarizes the complexity results obtained in this paper.

Table 1: Summary of the results obtained in this paper

	F_1	F_2	F_3	general
explicit	$O(n^2q)$	$O(n^2q)$	$O(n\log^2 n)$	NP-hard
implicit (oracle)	$\Theta(n^2q)$	$\Theta(n^2q)$	$O(n(\log^2 n+q))$	$\Omega(2^{\frac{n}{2}}q)$

q: the time required for computing $F(x)$ for each $x \in \mathbb{R}_+^V$.

Our results for functions of types F_1, F_2, and F_3 can be applied to the source location problem and the edge-connectivity augmentation problem, which shows extensions of the existing complexity results (see Section 3 for details).

Due to the space limitation, we only discuss two applications of our covering problem and investigate the case in which F is a laminar sum.

2 Definitions and Preliminaries

Let V be a finite set with $|V|=n$. A family $\mathscr{F} \subseteq 2^V$ is called *laminar* (or *nested*) if for arbitrary two sets $X,Y \in \mathscr{F}$, at least one of the three sets $X \cap Y$, $X-Y$, and $Y-X$ is empty, i.e., $X \cap Y \neq \emptyset$ implies $X \subseteq Y$ or $X \supseteq Y$.

For a laminar family $\mathscr{F}=\{X_i \mid i \in I\}$, define a directed graph $T=(W,A)$ with a vertex set W and an arc set A by

$$\begin{aligned} W &= \{w_i \mid i \in I \cup \{i_0\}\} \\ A &= \{a_i=(w_i,w_j) \mid X_i \subsetneq X_j, \mathscr{F} \text{ contains no set } Y \text{ with } X_i \subsetneq Y \subsetneq X_j\} \\ &\quad \cup \{a_i=(w_i,w_{i_0}) \mid X_i \text{ is a maximal set in } \mathscr{F}\}, \end{aligned}$$

where i_0 is a new index not in I. Since $\mathscr{F}$ is laminar, the graph $T=(W,A)$ is a directed tree toward the root w_{i_0} and is called the *tree representation* of $\mathscr{F}$. For each $X_i \in \mathscr{F}$, let us respectively define the family $\mathscr{S}(X_i)$ of the children, the incremental set ΔX_i, and the depth $h(X_i)$ by

$$\begin{aligned} \mathscr{S}(X_i) &= \{X_j \mid a_j=(w_j,w_i) \in A\}, \\ \Delta X_i &= X_i \setminus \bigcup_{X_j \in \mathscr{S}(X_i)} X_j, \\ h(X_i) &= |\{X_j \mid X_j \in \mathscr{F} \text{ with } X_j \supseteq X_i\}|. \end{aligned}$$

A function $F:\mathbb{R}^V \to \mathbb{R}$ is called *monotone nondecreasing* (simply *monotone*) if $F(x) \leq F(y)$ holds for arbitrary two vectors $x,y \in \mathbb{R}^V$ with $x \leq y$, and *concave* if

$$\alpha F(x)+(1-\alpha)F(y) \leq F(\alpha x+(1-\alpha)y) \tag{2}$$

holds for arbitrary two vectors $x,y \in \mathbb{R}^V$ and real α with $0 \leq \alpha \leq 1$.

Now, we formulate the problem of minimizing a monotone concave function with laminar covering constraints. Given a laminar family $\mathscr{F} \subseteq 2^V$, a monotone concave function $F:\mathbb{R}_+^V \to \mathbb{R}_+$, and a demand function $d:\mathscr{F} \to \mathbb{R}_+$, we consider the problem given by

$$\begin{aligned} (\mathbf{P}) \qquad \text{Minimize} \quad & F(x) && && (3) \\ \text{subject to} \quad & x(X) \geq d(X) && (X \in \mathscr{F}), && (4) \\ & x(v) \geq 0 && (v \in V), && (5) \end{aligned}$$

where $x(X)=\sum_{v \in X} x(v)$. We assume without loss of generality that $F(\mathbf{0})=0$.

For a function $d:\mathscr{F} \to \mathbb{R}$ we also define the increment Δd by $\Delta d(X)=d(X)-\sum_{Y \in \mathscr{S}(X)} d(Y)$. If $\Delta d(X) \leq 0$, we can remove constraint $x(X) \geq d(X)$ from (4). Hence we assume that every set $X \in \mathscr{F}$ satisfies

$$\Delta d(X) > 0. \tag{6}$$

We consider Problem (**P**) when the cost function F is given either explicitly or implicitly. Here "implicitly" means that F is given by an oracle, i.e., we can invoke the oracle for the evaluation of $F(x)$ for any x in $\mathbb{R}_+^V$ and use the function value $F(x)$. In either case (explicitly or implicitly), we assume that $F(x)$ can be computed for any $x \in \mathbb{R}_+^V$ in $O(q)$ time.

In this paper we study the following three cases, in which the objective functions F are, respectively, given as

$$F_1(x) = \sum_{X \in \mathscr{F}} f_{\Delta X}(x[\Delta X]), \tag{7}$$

$$F_2(x) = \sum_{v \in V} f_v(x(v)), \tag{8}$$

$$F_3(x) = \sum_{v \in V : x(v) > 0} (a_v x(v) + b_v), \tag{9}$$

where $f_{\Delta X}$ is a nonnegative monotone concave function on $\mathbb{R}_+^{\Delta X}$, $x[\Delta X]$ denotes the restriction of x on ΔX, f_v is a nonnegative monotone concave function on $\mathbb{R}_+^{\{v\}}$, and a_v and b_v are nonnegative constants.

3 Applications of Our Covering Problem

In this section we introduce two network problems as examples of our problem.

3.1 Source Location Problem in Undirected Networks

Let $\mathscr{N} = (G = (V, E), u)$ be an undirected network with a vertex set V, an edge set E, and a capacity function $u : E \to \mathbb{R}_+$. For convenience, we regard $\mathscr{N}$ as a symmetric directed graph $\widehat{\mathscr{N}} = (\widehat{G} = (V, \widehat{E}), \widehat{u})$ defined by $\widehat{E} = \{(v,w), (w,v) \mid \{v,w\} \in E\}$ and $\widehat{u}(v,w) = \widehat{u}(w,v) = u(\{v,w\})$ for any $\{v,w\} \in E$. We also often write $u(v,w)$ instead of $u(\{v,w\})$.

A flow $\varphi : \widehat{E} \to \mathbb{R}_+$ is *feasible* with a supply $x : V \to \mathbb{R}_+$ if it satisfies the following conditions:

$$\partial\varphi(v) \stackrel{\text{def}}{=} \sum_{(v,w) \in \widehat{E}} \varphi(v,w) - \sum_{(w,v) \in \widehat{E}} \varphi(w,v) \le x(v) \quad (v \in V), \tag{10}$$

$$0 \le \varphi(e) \le \widehat{u}(e) \qquad (e \in \widehat{E}). \tag{11}$$

Here (10) means that the net out-flow value $\partial\varphi(v)$ at $v \in V$ is at most the supply at v.

Given an undirected network $\mathscr{N}$ with a demand $k > 0$ and a cost function $F : \mathbb{R}_+^V \to \mathbb{R}_+$, the source location problem is to find a minimum-cost supply x such that for each $v \in V$ there is a feasible flow φ such that the sum of the net in-flow value and the supply at v is at least k. The problem is rewritten as follows.

$$\begin{aligned} &\text{Minimize} && F(x) \\ &\text{subject to} && \forall\, v \in V,\ \exists \text{ a feasible flow } \varphi_v \text{ in } \mathscr{N} \text{ with a supply } x\text{:} \\ & && -\partial\varphi_v(v) + x(v) \ge k, \qquad (12) \\ & && x(v) \ge 0 \qquad (v \in V). \qquad (13) \end{aligned}$$

Note that the flow φ_v $(v \in V)$ in (12) may depend on $v \in V$.

The above-mentioned source location problem was investigated in [1, 15] in a special case where the cost function is given as

$$F(x) = \sum_{v \in V : x(v) > 0} b_v.$$

Namely, the cost function depends only on the fixed setup cost of the facilities at vertices $v \in V$ with $x(v) > 0$, and is independent of the positive supply value $x(v)$. Some variants of the problem such as non-uniform demand and directed network cases are also examined in [1, 7, 8, 16].

We can show that (12) can be represented by a laminar covering constraint (4), where $\mathscr{F}$ denotes the family of all extreme sets in $\mathscr{N}$. Hence the source location problem can be formulated as (**P**) in Section 2.

We remark that given an undirected network $\mathscr{N} = (G = (V, E), u)$ and a demand $k\,(> 0)$, the family $\mathscr{F}$ of all extreme sets, as well as the deficiency $d : \mathscr{F} \to \mathbb{R}_+$, can be computed in $O(n(m + n\log n))$ time [12]. Therefore, our results for the laminar covering problem immediately imply the ones for the source location problem considered in [1, 15]. In particular, if the cost function is a separable monotone concave function, i.e., the sum of fixed setup costs and concave running costs for facilities at $v \in V$, the source location problem can be solved in $O(nm + n^2(q + \log n))$ time. Moreover, we can solve the problem in $O(n(m + n\log n))$ time if the cost is the sum of fixed setup costs and linear running costs.

3.2 Edge-connectivity Augmentation in Undirected Networks

Let $\mathscr{N} = (G = (V,E), u)$ be an undirected network with a capacity function $u : E \to \mathbb{R}_+$. We call $\mathscr{N}$ is k-edge-connected if for every two nodes $v, w \in V$ the maximum flow value between v and w is at least k. Given an undirected network $\mathscr{N}$, a positive real k, and a node-cost function $F : \mathbb{R}_+^V \to \mathbb{R}_+$, the edge-connectivity augmentation problem [4, 12] is to find a set D of new edges with capacity $\mu_D : D \to \mathbb{R}_+$ for which $\mathscr{N}' = (G' = (V, E \cup D), u \oplus \mu_D)$ is k-edge-connected and $F(\partial\mu_D)$ is minimum, where $\partial\mu_D(v) = \sum_{e \in D : e \ni v} \mu_D(e)$ $(v \in V)$, and $u \oplus \mu_D$ is the direct sum of u and μ_D. From the max-flow min-cut theorem, we can see that $x = \partial\mu_D$ must satisfy $\kappa(X) + x(X) \geq k$ for any nonempty $X \subsetneq V$, which implies

$$x(X) \geq d(X) \qquad (X \in \mathscr{F}), \tag{14}$$

where $d(X) = \max\{k - \kappa(X), 0\}$, and $\mathscr{F}$ is the family of all extreme sets in $\mathscr{N}$.

On the other hand, it is known that any x satisfying (14) can create a capacity function $\mu_D : D \to \mathbb{R}_+$ for which $\mathscr{N}'$ is k-edge-connected [9, 10, 11] and moreover, such an x of minimum $x(V)$ can be found in $O(n(m + n\log n))$ time [13].

The edge-connectivity augmentation problem has been studied only when $F(x) = \sum_{v \in V} c(v)x(v)$ for some $c : V \to \mathbb{R}_+$ [4, 12] (see [2, 3, 14] for some other kinds of connectivity augmentation problems). Note that, if $c(v) = \frac{1}{2}$ for all $v \in V$, then the cost $F(\partial\mu_D)$ with $\mu_D(e) = 1$ $(e \in D)$ is equal to the number of edges in D.

Our results extend the existing ones for the augmentation problem. Especially when F is given by (7), the algorithm proposed in this paper together with the ones in [12, 13] solves the augmentation problem in $O(nm + n^2(q + \log n))$ time. Moreover, if F is given by (9), we can solve the problem in $O(n(m + n\log n))$ time.

4 The Laminar Cost Case

In this section we consider the problem whose cost function is given by F_1, i.e.,

$$\begin{array}{lll} (\mathbf{P}_1) \quad \text{Minimize} & \displaystyle\sum_{X \in \mathscr{F}} f_{\Delta X}(x[\Delta X]), & \\ \text{subject to} & x(X) \geq d(X) & (X \in \mathscr{F}), \\ & x(v) \geq 0 & (v \in V), \end{array} \tag{15}$$

where $f_{\Delta X}$ is a monotone concave function on ΔX with $f_{\Delta X}(\mathbf{0}) = 0$.

We shall present an $O(n^2 q)$ time algorithm for the problem and show the $\Omega(n^2 q)$ time bound when the cost function is given by an oracle.

4.1 Structural Properties of Optimal Solutions

This section reveals structural properties of optimal solutions of Problem $(\mathbf{P}_1)$ in (15), which makes it possible for us to devise a polynomial algorithm for Problem $(\mathbf{P}_1)$.

Let $\mathscr{F}$ be a laminar family on V, and $T = (W, A)$ be the tree representation of $\mathscr{F}$. Consider the problem projected on $Y \in \mathscr{F}$ that is given as

$$\begin{array}{lll} (\mathbf{P}_Y) \quad \text{Minimize} & \displaystyle\sum_{X \in \mathscr{F} : X \subseteq Y} f_{\Delta X}(x[\Delta X]) & \\ \text{subject to} & x(X) \geq d(X) & (X \in \mathscr{F}, X \subseteq Y), \\ & x(v) \geq 0 & (v \in V). \end{array} \tag{16}$$

We first show properties of optimal solutions of $(\mathbf{P}_Y)$, from which we derive properties of optimal solutions of $(\mathbf{P}_1)$.

Lemma 1 For a minimal $Y \in \mathscr{F}$, Problem $(\mathbf{P}_Y)$ has an optimal solution $x = z_v$ for some $v \in Y$ such that

$$z_v(t) = \begin{cases} d(Y)\ (= \Delta d(Y)) & (t = v) \\ 0 & (t \in V \setminus \{v\}). \end{cases} \qquad \square$$

Lemma 2 Let Y be a non-minimal set in $\mathscr{F}$. Then there exists an optimal solution x of Problem $(\mathbf{P}_Y)$ such that for some $v \in \Delta Y$

$$\begin{array}{rll} x(t) & = \begin{cases} \Delta d(Y) & (t = v) \\ 0 & (t \in (V \setminus Y) \cup (\Delta Y \setminus \{v\})), \end{cases} & \\ x(X) & = d(X) & (X \in \mathscr{S}(Y)), \end{array}$$

or for some $X \in \mathscr{S}(Y)$

$$\begin{aligned} x(Z) &= \begin{cases} d(X)+\Delta d(Y) & (Z=X) \\ d(Z) & (Z \neq X,\ Z \in \mathscr{S}(Y)), \end{cases} \\ x(v) &= 0 \qquad\qquad (v \in (V \setminus Y) \cup \Delta Y). \end{aligned}$$ □

Let $W^* = \{w_i \mid X_i \in \mathscr{F}\}$. A partition $\mathscr{P} = \{P_1, \cdots, P_k\}$ of W^* is called a *path-partition* of W^* if each $P_j = \{w_{j_0}, w_{j_1}, \cdots, w_{j_{r_j}}\} \in \mathscr{P}$ forms a directed path $w_{j_0} \to w_{j_1} \to \cdots \to w_{j_{r_j}}$ in $T = (W, A)$ with $\Delta X_{j_0} \neq \emptyset$.

We are now ready to describe our structure theorem.

Theorem 3 Problem $(\mathbf{P}_1)$ in (15) has an optimal solution x^* that can be obtained from a path-partition $\mathscr{P} = \{P_1, \cdots, P_k\}$ of W^* together with $v_j \in \Delta X_{j_0}$ $(j = 1, \cdots, k)$ as follows.

$$x^*(t) = \begin{cases} \sum\limits_{w_{j_i} \in P_j} \Delta d(X_i) & (t = v_j,\ j = 1, \cdots, k) \\ 0 & (t \in V \setminus \{v_j \mid j = 1, \cdots, k\}). \end{cases}$$ □

4.2 A Polynomial Algorithm

In this section we present a polynomial algorithm for Problem $(\mathbf{P}_1)$ in (15). The algorithm applies dynamic programming to compute an optimal path-partition of W^*.

For any $Y \in \mathscr{F}$, we denote by w_Y the node in W corresponding to Y, and by $w_{j_0}(= w_Y), w_{j_1}, \cdots, w_{j_{h(Y)-1}}, w_{j_{h(Y)}}(= w_{i_0})$ the directed path from w_Y to the root w_{i_0}. Our dynamic programming solves the following $h(Y)$ problems for each $Y \in \mathscr{F}$.

$$(\mathbf{P}(Y,k)) \qquad \text{Minimize} \quad \sum_{X \in \mathscr{F}: X \subseteq Y} f_{\Delta X}(x[\Delta X]) \tag{17}$$

$$\text{subject to} \quad x(Y) \geq d(Y) + \sum_{i=1}^{k} \Delta d(X_{j_i}), \tag{18}$$

$$x(X) \geq d(X) \qquad (X \in \mathscr{F}, X \subsetneq Y), \tag{19}$$

$$x(v) \geq 0 \qquad (v \in V), \tag{20}$$

where $Y \in \mathscr{F}$ and $k = 0, 1, \cdots, h(Y) - 1$. Let $\alpha(Y,k)$ denote the optimal value of Problem $(\mathbf{P}(Y,k))$. By Theorem 3, these problems $(\mathbf{P}(Y,k))$ have optimal solutions based on a path-partition $\mathscr{P}$ of $\{w_i \mid X_i \in \mathscr{F}, X_i \subseteq Y\}$. For $P_j \in \mathscr{P}$ containing $w_Y \in W$ (that corresponds to Y), let v_j be the node in ΔX_{j_0} given in Theorem 3. We store v_j as $\beta(Y,k)$. It follows from Lemmas 1 and 2 that $\alpha(Y,k)$ and $\beta(Y,k)$ can be computed as follows.

For each minimal $Y \in \mathscr{F}$ (which corresponds to a leaf in T) the following z_v^k for some $v \in Y$ gives an optimal solution, due to Lemma 1.

$$z_v^k(t) = \begin{cases} \sum_{i=0}^{k} \Delta d(X_{j_i}) & (t = v) \\ 0 & (t \in \Delta Y \setminus \{v\}). \end{cases}$$

Hence we have

$$(\alpha(Y,k), \beta(Y,k)) = \left(\min_{v \in Y} f_Y(z_v^k), \arg\min_{v \in Y} f_Y(z_v^k) \right) \tag{21}$$

for $k = 0, \cdots, h(Y)-1$, where $\arg\min_{v \in Y} f_Y(z_v^k)$ denotes a vertex $v^* \in Y$ satisfying $f_Y(z_{v^*}^k) = \min_{v \in Y} f_Y(z_v^k)$.

For a non-minimal $Y \in \mathscr{F}$, Lemma 2 validates the following recursive formulas.

$$\alpha(Y,k) = \min \left\{ \min_{X \in \mathscr{S}(Y)} \left\{ \alpha(X,k+1) + \sum_{\substack{Z \in \mathscr{S}(Y) \\ Z \neq X}} \alpha(Z,0) \right\}, \min_{v \in \Delta Y} \left\{ f_{\Delta Y}(z_v^k) + \sum_{X \in \mathscr{S}(Y)} \alpha(X,0) \right\} \right\}, \tag{22}$$

$$\beta(Y,k) \begin{cases} \beta(X,k+1) \text{ if } \alpha(Y,k) = \alpha(X,k+1) + \sum\limits_{\substack{Z \in \mathscr{S}(Y) \\ Z \neq X}} \alpha(Z,0), \\ v \qquad\qquad \text{if } \alpha(Y,k) = f_{\Delta Y}(z_v^k) + \sum\limits_{X \in \mathscr{S}(Y)} \alpha(X,0). \end{cases} \tag{23}$$

By using (21), (22), and (23), our algorithm first computes each α and β from the leaves toward root w_{i_0} of T. Then we obtain an optimal value $\sum_{X \in \mathscr{S}(X_{i_0})} \alpha(X,0)$ of Problem $(\mathbf{P}_1)$ in (15). Next, we compute an optimal solution x^* by using β from the root toward the leaves of T.

Our algorithm is formally described as follows.

Algorithm DP

Input: A laminar family $\mathscr{F}$, a demand function $d : \mathscr{F} \to \mathbb{R}_+$, and a cost function F as in (15).

Output: An optimal solution x^* for Problem $(\mathbf{P}_1)$ in (15).

Step 0. $\widetilde{W} := W$.

Step 1. (Compute α and β) While $\widetilde{W} \neq \{w_{i_0}\}$ **do**

Choose an arbitrary leaf $w \in \widetilde{W}$ of $T[\widetilde{W}]$.

/* Let Y be the set in $\mathscr{F}$ corresponding to w. */

(1-I) Compute $\alpha(Y,k)$ and $\beta(Y,k)$ for $k = 0, \cdots, h(Y)-1$ by using either (21) or ((22) and (23)).

(1-II) $\widetilde{W} := \widetilde{W} \setminus \{w\}$.

Step 2. $\widetilde{W} := W \setminus \{w_{i_0}\}$, and $x^*(v) := 0$ for all $v \in V$.

Step 3. (Compute an optimal x^*) While $\widetilde{W} \neq \emptyset$ **do**

Choose an arbitrary node w of $T[\widetilde{W}]$ having no leaving arc.

/* Let Y be the set in $\mathscr{F}$ corresponding to w, w_{j_0} be the node in W corresponding to X_{j_0} such that $\beta(Y,0) \in \Delta X_{j_0}$ and let $w_{j_0} \to w_{j_1} \to \cdots \to w_{j_l} (= w)$ be a directed path in $T[\widetilde{W}]$. */

(3-I) $x^*(\beta(Y,0)) := \sum_{i=0}^{l} \Delta d(X_{j_i})$.

(3-II) $\widetilde{W} := \widetilde{W} \setminus \{w_{j_0}, \cdots, w_{j_l}\}$.

Step 4. Output x^* and halt. □

Here $T[\widetilde{W}]$ denotes the subtree of T induced by $\widetilde{W}$.

We now have the following theorem.

Theorem 4 Algorithm DP computes an optimal solution for Problem $(\mathbf{P}_1)$ in $O(n^2q)$ time. □

4.3 The Lower Bound for the Time Complexity When F is Given by an Oracle

In this section we consider a lower bound for the time complexity of our problem when F is given by an oracle. We shall show that the oracle has to be invoked $\Omega(n^2)$ times even if we know in advance that F is given in the form of (8), i.e., $F = \sum_{v \in V} f_v(x(v))$. This, together with Theorem 4, implies that Algorithm DP is optimal if F is given by an oracle.

Suppose n is a positive even number. Let $g_0 : \mathbb{R}_+ \to \mathbb{R}_+$ be a monotone increasing and strictly concave function (e.g., $g_0(x) = \frac{-1}{x+1} + 1$ $(x \geq 0)$), and for each $i = \frac{n}{2}+1, \frac{n}{2}+2, \cdots, n$ define $g_i : \mathbb{R}_+ \to \mathbb{R}_+$ by

$$g_i(\xi) = \begin{cases} g_0(\frac{n}{2}+1) - g_0(i-\frac{n}{2}) & (\xi > 0) \\ 0 & (\xi = 0) \end{cases}.$$

Then, let $V = \{v_1, \cdots, v_n\}$ and consider a problem instance I obtained by

$$\begin{aligned} &\text{a laminar family } \mathscr{F} = \left\{ X_i = \{v_1, \cdots, v_{\frac{n}{2}+i}\} \;\middle|\; i = 0, \cdots, \tfrac{n}{2} \right\}, \\ &\text{a demand function } \; d : d(X_i) = i+1 \quad (i = 0, 1, \cdots, \tfrac{n}{2}), \\ &\text{a cost function } F(x) = \sum_{v \in V} f_v(x(v)), \end{aligned} \tag{24}$$

where

$$f_{v_i}(\xi) = \begin{cases} g_0(\xi) & (v_i \in X_0) \\ g_i(\xi) & (v_i \in V - X_0). \end{cases}$$

For this problem instance, we have the following lemma.

Lemma 5 The problem instance I defined as above requires at least $\frac{n}{2}(\frac{n}{2}+1)$ calls to the oracle for F. □

This implies the following theorem.

Theorem 6 If F is given by an oracle, then Problem (15) requires $\Omega(n^2q)$ time. □

We can easily see that Lemma 5 still holds even if each f_v is given by an oracle.

Theorem 7 Let F be a separable monotone concave function (i.e., $F = \sum_{v \in V} f_v$ with monotone concave functions $f_v : \mathbb{R}_+ \to \mathbb{R}_+$ $(v \in V)$). If each f_v is given by an oracle, then Problem (15) requires $\Omega(n^2q)$ time. □

Notice that Algorithm DP given in Section 4.2 is optimal, due to this theorem.

Corollary 8 If the cost function $F = \sum_{X \in \mathscr{F}} f_{\Delta X}$ is given by an oracle, then Problem (15) requires $\Theta(n^2q)$ time. □

5 Concluding Remarks

We have considered the problem of minimizing monotone concave functions with laminar covering constraints. Our results can be summarized by Table 1 in Section 1.

In this paper we have assumed that the objective function F is monotone nondecreasing. It should be noted that this monotonicity assumption can be removed if we impose that the sum $x(V)$ be equal to a constant.

References

[1] K. ARATA, S. IWATA, K. MAKINO, AND S. FUJISHIGE, Locating sources to meet flow demands in undirected networks, *J. Algorithms*, **42** (2002), 54–68.

[2] A. A. BENCZÚR AND D. R. KARGER, Augmenting undirected edge connectivity in $\widetilde{O}(n^2)$ time, *J. Algorithms*, **37** (2000), 2–36.

[3] G.-R. CAI AND Y.-G. SUN, The minimum augmentation of any graph to k-edge-connected graph, *Networks*, **19** (1989), 151–172.

[4] A. FRANK, Augmenting graphs to meet edge-connectivity requirements, *SIAM J. Discrete Mathematics*, **5** (1992), 25-53.

[5] A. FRANK AND T. JORDÁN, Minimal edge-coverings of pairs of sets, *J. Combin. Theory B*, **65** (1995), 73–110.

[6] S. FUJISHIGE, *Submodular Functions and Optimization*, North-Holland (1991).

[7] H. ITO AND M. YOKOYAMA, Edge connectivity between nodes and node-subset, *Networks*, **31** (1998), 157–164.

[8] H. ITO, K. MAKINO, K. ARATA, S. HONAMI, Y. ITATSU, AND S. FUJISHIGE, Source location problem with flow requirements in directed networks, *Optimization Methods and Software*, **18** (2003), 427–435.

[9] L. LOVÁSZ, *Combinatorial Problems and Exercises*, North-Holland (1979).

[10] W. MADER, A reduction method for edge-connectivity in graphs, *Ann. Discrete Mathematics*, **3** (1978), 145–164.

[11] W. MADER, Konstruktion aller n-fach kantenzusammenhangenden Digraphen, *European J. Combin.*, **3** (1982), 63–67.

[12] H. NAGAMOCHI, Computing extreme sets in graphs and its applications, *Proc. of the 3rd Hungarian-Japanese Symposium on Discrete Mathematics and Its Applications* (January 21–24, 2003, Tokyo, Japan) 349–357.

[13] H. NAGAMOCHI AND T. IBARAKI, Augmenting edge-connectivity over the entire range in $\widetilde{O}(nm)$ time, *J. Algorithms*, **30** (1999), 253–301.

[14] D. NAOR, D. GUSFIELD AND C. MARTEL, A fast algorithm for optimally increasing the edge connectivity, *SIAM J. Computing*, **26** (1997), 1139–1165.

[15] H. TAMURA, M. SENGOKU, S. SHINODA, AND T. ABE, Some covering problems in location theory on flow networks, *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, **E75-A** (1992), 678–683.

[16] H. TAMURA, H. SUGAWARA, M. SENGOKU, AND S. SHINODA, Plural cover problem on undirected flow networks, *IEICE Trans.*, **J81-A** (1998), 863–869 (in Japanese).

The Packing Clutter of the Positive Cocircuits of an Oriented Matroid Whose Rank is $\leqq 4$

KENJI KASHIWABARA

Dept. of Systems Science,
Graduate School of Arts and Sciences,
The University of Tokyo,
3-8-1 Komaba, Meguro-Ku, Tokyo
153-8902, JAPAN
kashiwa@graco.c.u-tokyo.ac.jp

TADASHI SAKUMA[†]

Dept. of Advanced Social and International Studies,
Graduate School of Arts and Sciences,
The University of Tokyo,
3-8-1 Komaba, Meguro-Ku, Tokyo
153-8902, JAPAN
sakuma@waka.c.u-tokyo.ac.jp

Abstract: The class of clutters of the positive cocircuits of oriented matroids is an important class which generalizes both the dicut clutters and the dicycle clutters of directed graphs, together. In this article, we will show that the clutter of the positive cocircuits of an oriented matroid whose rank $\leqq 4$ has the packing property if and only if it has none of the five minimally non-packing minors C_3^2, C_5^3, Q_6, $Q_6 \otimes 1$ and $Q_6 \otimes \{1,2\}$.

Keywords: oriented matroid, positive cocircuit, clutter, ideal-ness, packing property

1 Dicuts, dicycles, and positive cocircuits of oriented matroids

The *clutter of circuits* of a digraph $D := (V,A)$ is the clutter such that its ground set is $V \cup A$ and that its edge is the set of vertices and arcs of a circuit of D. Note that this clutter belongs to the class of the arc-wise dicycle clutters.
Recently, B. Guenin [4] proved the following.

Theorem 1 ([4]) The clutter of the circuits of a digraph is the MFMC-property if and only if it has no odd hole minor C_n^2 (n is odd and $\geqq 3$).

Clearly, the above theorem guarantees this clutter is ideal if and only if it has the MFMC-property.
On the other hand, it is well known that the clutter of dicut is always ideal and has the MFMC-property. And if a digraph is planar, this fact guarantees both the MFMC-property and the ideal-ness of the arc-wise dicycle clutters of planar digraphs.
Actually, both of the arc-wise dicycle clutters and the dicut clutters are somewhat special subclasses of the positive (co-)circuit clutters of the oriented matroids (for details about oriented matroids, please see [1].)
Quite recently, concerned with this, M. Hachimori and M. Nakamura [5] proved the following:

Theorem 2 ([5]) The clutter of the positive circuits of an oriented matroid whose rank is $\leqq 2$ has the MFMC-property if and only if it does not have C_5^3 as its minor.

Thus one may expect that the class of the clutters of the positive (co-)circuits of oriented matroids possibly has the good property "MFMC=IDEAL".
In this article, concerned with the above, we prove the following.

Theorem 3 The clutter of the positive cocircuits of an oriented matroid whose rank is $\leqq 4$ has the packing property if and only if it has none of the five minimally non-packing minors C_3^2, C_5^3, Q_6, $Q_6 \otimes 1$ and $Q_6 \otimes \{1,2\}$.
(For the definition of the operator $\otimes$, please see [3].)

Hence we know that, on this general situation, the ideal-ness of a clutter does not necessarily warrant its packing property or its MFMC-property.
This theorem clearly implies the following.

Corollary 4 The clutter of the positive cocircuits of an oriented matroid whose rank is $\leqq 4$ is ideal if and only if it has none of the two clutter-minors C_3^2 and C_5^3.

Nevertheless, the authors guess that the following statements will be affirmative.

[†]Research is supported by Grant-in-Aid for Young Scientists (B)

Conjecture 5 The arc-wise dicycle clutter of a directed graph has MFMC-property if and only if it has no odd hole minor C_n^2 (n is odd and $\geqq 3$).

Conjecture 6 The clutter of the positive (co-)circuits of an oriented matroid is ideal if and only if it has neither C_n^2 (n is odd and $\geqq 3$) nor C_5^3 as its clutter-minor.

Remark 7 *It is known that the blockers of the dicut clutters (and also the blockers of the arc-wise dicycle clutters) can contain only ideal minimally non-packing clutters as their minimally non-packing minors. One of these examples, found by A. Schrijver [6], is $Q_6 \otimes \{1,3,5\}$. Two additional examples are found by G. Cornuéjols and B. Guenin [2].*

2 Assumption to point configurations

We can assume the following on oriented matroids without loss of generality.
Concerning oriented matroids of rank 3 or 4, the class of the clutters of positive cocircuits of all oriented matroids equals that of the clutters of positive cocircuits of all realizable oriented matroids. We may assume oriented matroids realizable in the sequel.
When there are elements which do not belong to any cocircuit, we can delete such elements from the clutter (or equivalently, contract them from the oriented matroid) without changing the packing property of the clutter. Particularly, it means that we can assume oriented matroids acyclic.
Moreover, when there are elements which belong to every cocircuit, we can contract such elements from the clutter (or equivalently, delete them from the oriented matroid) without changing the packing property of the clutter.
Then we can represent such an oriented matroid of rank 3 (4, respectively,) a point configuration on the surface of a polygon (a 3-dimensional polytope, respectively.) Here, let us call the vertices (0-faces) of a facet of the polytope together with all the points in the relative interior of each faces of the facet a *facet* of the oriented matroid, which is corresponding to the complement of a positive cocircuit of the oriented matroids.
Now, by using the above term 'facets', we will mention here some useful observations about the relation between our oriented matroid and our clutter.:

- *The packing number is at least 2 if and only if there exist two facets such that every point belongs to either facet.*
- *The packing number is at least 3 if and only if there exist three facets such that every point belongs to two facets among the three.*
- *The packing number is 4 if and only if there exist four facets such that every point belongs to three facets among the four.*
- *The blocking number is at most k if and only if there exist k points such that no facet contains all of the k points together.*

Note that a deletion of the clutter of the positive cocircuits of an oriented matroid $\mathscr{M}$ is corresponding to the clutter of the positive cocircuits of some contraction of the oriented matroid, because a contraction of the clutter of the positive circuits of the dual oriented matroid $\mathscr{M}^*$ is equivalent to the clutter of the positive cocircuits of some deletion of $\mathscr{M}$.

3 Proof of Theorem 3 for oriented matroids of rank 3

Lemma 8 The clutter of the positive cocircuits of an oriented matroid whose rank is 3 has the packing property if and only if it has none of the two minimally non-ideal minors C_3^2, C_5^3 as its clutter-minor.

PROOF: Because of the acyclicity and the realizability of the oriented matroid, this oriented matroid can be represented by a point configuration on the plane. We separate the cases according to the number of vertices of the polygon of the convex hull of the point configuration on the plane.
When the convex hull of the configuration is a hexagon or a polygon with more vertices, contract all the points other than three vertices to every one vertex. Then you will have C_3^2.
When the convex hull of the configuration is a pentagon, contract the points other than the five vertices. You will have C_5^3 as a minor.
Now, consider the case of a quadrangle. In this case, the clutter of positive cocircuits does not have the packing property if and only if there is at least one point on the interior of each edge for some adjacent pair of edges of the quadrangle. Because when such a pair of edges exists, contract all the points other than three points; two points on the interior of such two edges and one point is the vertex which is not adjacent to such two edges. Then you will have C_3^2. Otherwise, you can easily check that all the minors pack.

Now, consider the case of a triangle. In this case, the clutter of the positive cocircuits does not have packing property if and only if there is a point on the interior of every edge. Because when such points exist, contract all the points other than such three points. Then you will have C_3^2 as a minor. Otherwise every minor packs. □

4 Proof of Theorem 3 for oriented matroids of rank 4

If the clutter has such a minor in Theorem 3, it is clear that it does not have the packing property. So we show the inverse direction as a sketch of the proof.
The proof in this section consists of two parts; one is the case that the packing number of our clutter is at least 2, the other is the case that the packing number of point configurations is 1.

4.1 Conditions on polyhedral embeddings

Here, we will translate the conditions on a clutter of positive cocircuits of our acyclic oriented matroid to the conditions of 'the 1-skeleton' of the convex hulls of the points in the point configuration of our oriented matroid.
It is well known that an abstract graph can be isomorphic to a 1-skeleton of some 3-dimensional polytope if and only if it is simple, planar, and 3-connected (Steinitz' Theorem.) Moreover, for every simple planar 3-connected graph, its embedding in S^2 is uniquely determined.
Now, let G be a 3-connected plane graph embedded in S^2 (embedding, for short,) such that our 1-skeleton of the convex hull is homotopic to G.
Thus, first of all, we have the following condition for the above G:

Condition 1: G is a 3-connected simple plane graph (embedding in S^2).

Since a deletion of the clutter of the positive cocircuits of an oriented matroid is corresponding to the clutter of the positive cocircuits of some contraction of the oriented matroid, the deletion of a clutter can be regarded as the new clutter of the positive cocircuit of some oriented matroid with lower rank. Then we can apply Lemma 8 to it.
Combining the above with the fact that we cannot obtain neither C_5^3 nor C_3^2 by deleting at least 1 element and contracting at least 0 element from the clutter, we have:

Condition 2: The degree of each vertex of G is at most four. Moreover, for any vertex x with degree 4, every edge incident with x is a 1-face of some triangle (triangular facet) containing x. In the same way, for any vertex y of degree 3, at least one of the three 2-cells (facets) containing y is a triangle.

The requirement that we do not have C_3^2 as a contraction minor leads to the following.

Condition 3: For any three vertices of G, there exist two vertices in the same facet among the three.

The requirement that we do not have C_5^3 as a contraction minor leads to the following.

Condition 4: There do not exist five points such that, by joining every pair of points on the same facet, we have a chordless 5-cycle. (If such a 5-cycle exists, then we have C_5^3 by contracting all the points other than the five points.)

Thus we have only to consider point configurations satisfying Conditions 1 to 4. When every point of the configuration is a vertex of the convex hull of the points, then Conditions 1 to 4 together are equivalent to the condition so that the clutter of positive cocircuits has neither C_3^2 nor C_5^3 as its minor.

4.2 The embedding G of a clutter whose packing number $\geqq 2$

In this subsection, we assume that the packing number of our clutter is $\geqq 2$. Then we enumerate all possible forms of the embedding G of Condition 1 which also satisfies Conditions 2 to 4 together.
If the packing number of the clutter is at least two, then all the points must belong to either two facets. Hence Conditions 3 and 4 are always satisfied.
As far as the packing numbers of our clutters are $\geqq 2$, we specify all the graphs satisfying Conditions 1 to 4. That is, the five infinite sequences of graphs and four additional graphs.
The next lemma solves Theorem 3 when the packing number of a clutter is at least 2.

Lemma 9 Consider the point configuration such that its packing number is at least two and it satisfies Conditions 1 to 4. Then the clutter of positive cocircuits has the packing property.

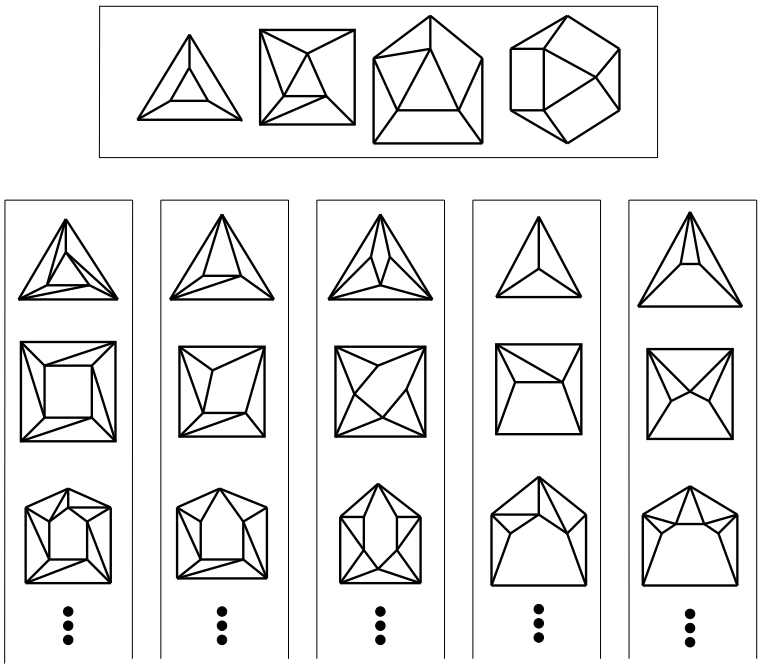


Figure 1: The graphs satisfying Conditions 1 to 4 such that the packing number of each corresponding clutter is $\geqq 2$.

This lemma is proved by the list of graphs above.
If all the points are the vertices of the convex hull, then Conditions 1 to 4 together shows us the complete situation which must be considered. It is not difficult to extend and remake the observations and discussions of ours in order to apply to the more general case that there are points on relative interiors of some faces of the convex hull. Here, however, for simplicity of this article, we omit the proof for that case.

4.3 The embedding G of a clutter whose packing number is 1

Now we assume that the packing number of our clutter is 1 and the graph (the embedding) G of Condition 1 satisfies Conditions 2 to 4, also. Clearly, such a clutter does not pack. Thus we investigate what minimally non-packing clutter are contained in such a clutter of positive cocircuits.
For a given plane graph G, *Y-*Δ *transformation* for a vertex v of degree 3 is the following graph-operation:

1. Join every pair of neighbors of v by an edge if these vertices are not already adjacent;
2. Then delete the vertex v.

Now we will apply the above operation to the embeddings satisfying Conditions 1 and 2. If such an embedding has a vertex x of degree 3, then, from Condition 2, at least one of the three 2-cells (facets) containing x is a triangle. Thus, there are only three types of Y-Δ transformations; the vertex of degree 3 belongs to either one triangle or two triangles or three triangles. We call these three transformations *Y-*Δ *transformations (1) (2)* and *(3)*, respectively.

Lemma 10 Except for a tetrahedron, the graph obtained by Y-Δ transformation to a graph satisfying Conditions 1 and 2 also satisfies Conditions 1 and 2.

By repeating Y-Δ transformations, we decrease the number of vertices. Eventually, it reaches the graph of tetrahedron or a 4-regular graph satisfying Conditions 1 and 2.
Now we consider the inverse operation of Y-Δ transformation, called Δ-*Y transformation*. By applying such transformations to the 4-regular graphs satisfying all of Conditions 1 to 4, we can get every graph satisfying all of Conditions 1 to 4 and whose minimum degree is 3.
Note that the dual graph of a plane 4-regular graph is a graph of quadrangulation, so it is a bipartite graph.
A *crown* is a graph embedded in S^2 which consists of two disjoint k-gons $\{x_0,x_0x_1,x_1,x_1x_2,x_2,\ldots,x_{k-1},x_{k-1}x_0\}$ and $\{y_0,y_0y_1,$ $y_1,y_1y_2,y_2,\ldots,y_{k-1},y_{k-1}y_0\}$ plus $2k$ additional edges x_iy_i and x_iy_{i+1} for $i=0,1,\ldots,k-1\ (\mathrm{mod}\,k)$.
Then we have;

Lemma 11 Let G be a 4-regular graph satisfying Conditions 1 to 4. If each color class of its dual bipartite graph G^* has a vertex whose degree is $\geqq 4$, then G is a crown.

Next we will enumerate all 4-regular graphs satisfying all of Conditions 1 to 4 such that they are not crowns. First we consider what kind of 4-regular graphs satisfies Conditions 1 and 2 together and is not a crown.
By easy calculation, the number of vertices of this type of 4-regular graphs is a multiple of 3 because its dual bipartite graph has a color class whose every vertex is degree 3.

Lemma 12 Let G be a 4-regular graph satisfying Conditions 1 and 2, and be not a crown. If the dual bipartite graph G^* of G has a color class consisting of vertices with degree 3 and if the size of the color class is $\geqq 12$, then G cannot satisfy Condition 3.

We cannot apply Δ-Y transformation to crowns except for octahedra. Since the size of our graph should be a multiple of 3 and cannot exceed 11, the size of the graph must be either 9 or 6.
Every 4-regular graph with 9 vertices satisfying Conditions 1 to 4 is a unique graph H with 8 triangles and 3 quadrangles. By applying Δ-Y transformation iteratively to H, we turns out to obtain only such graphs that the clutters corresponding to them always have C_5^3 as a contraction minor. For example, here we show H and a terminal resultant of iterative applications of Δ-Y transformations to H, namely, a truncated triangular prism with 18 vertices. See Figure 2. Contracting the complement of the big vertices from each corresponding polytope, we have C_5^3 as its clutter minor.

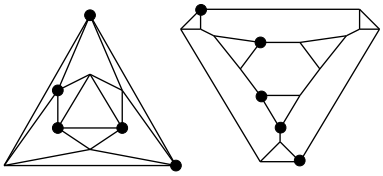


Figure 2: H and a truncated triangular prism with 18 vertices.

If the size of our graph is 6, then it is an octahedron. If the resultant of the Δ-Y transformations to an octahedron satisfies all of Conditions 1 to 4, then it turns out to be either a graph G such that the packing number of the clutter corresponding to G is $\geqq 2$, or one of the 7 graphs each of which has a corresponding ideal clutter whose every minimally non-packing minor is Q_6 or $Q_6 \otimes 1$ or $Q_6 \otimes \{1,2\}$. See Figure 3. By contracting the big vertices from each of the 7 polytopes, we have one of Q_6, $Q_6 \otimes 1$ and $Q_6 \otimes \{1,2\}$ as its clutter minor.

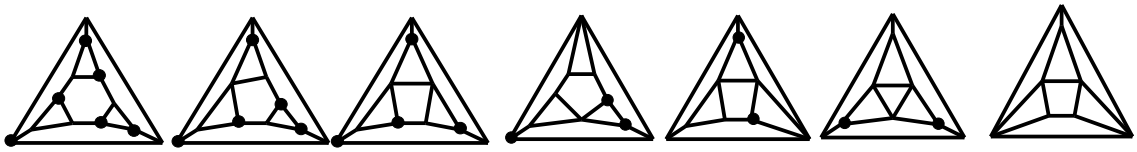


Figure 3: The 7 polytopes whose clutters have only ideal minimally non-packing minors.

We have completed the sketch of the proof.

References

[1] A. Bjőrner, M. Las Vergnas, B. Sturmfels, N. White, and G. Ziegler. *ORIENTED MATROIDS – SECOND EDITION* ENCYCLOPEDIA OF MATHEMATICS AND ITS APPLICATIONS **Vol. 46**, CAMBRIDGE UNIVERSITY PRESS, Cambridge, 1993, 1999.

[2] G. Cornuéjols and B. Guenin. On Dijoins, *Discrete Mathematics* **243** (2002), 213–216.

[3] G. Cornuéjols, B. Guenin and F. Margot. The packing property, *Mathematical Programming*, **Ser. A 89** (2000), 113–126.

[4] B. Guenin, Circuit Mengerian directed graphs, in: *Integer Programming and Combinatorial Optimization* (Proceedings 8th International IPCO Conference, Utrecht, 2001; K. Aardal, B. Gerards, eds.) [Lecture Notes in Computer Science 2081], Springer, Berlin, 2001, 185–195.

[5] M. Hachimori and M. Nakamura, Rooted circuits of closed-set systems and the Max-Flow Min-Cut property, preprint 2003.

[6] A. Schrijver, A counterexample to a conjecture of Edmonds and Giles, *Discrete Mathematics* **32** (1980), 213–214.

Spanning Tree Optimization Problems with Degree Based Objective Functions

GÁBOR SALAMON*

Department of Computer Science and Information Technology
Budapest University of Technology and Economics
1117 Budapest, Magyar tudósok körútja 2., Hungary
gsala@cs.bme.hu

Abstract: Design of communication networks may need to find a spanning tree that satisfies certain optimality criteria. We consider the case when the quality of a solution depends only on the degree-distribution of the obtained spanning tree. A known result for this class of problems provides a 2-approximation for the MAXIMUMLEAFSPANNINGTREE problem.

We consider the MINIMUMBRANCHINGSPANNINGTREE problem, when the goal is to minimize the number of (≥ 3)-degree nodes, and the MAXIMUMINTERNALSPANNINGTREE problem, where the goal is to maximize the number of non-leaf nodes. Despite their importance no approximation results were known for either of these problems. For the MINIMUMBRANCHINGSPANNINGTREE problem we present an algorithm that produces a spanning tree with at most $\mathcal{O}(\log n)$ branching nodes whenever each node of the input graph has a degree at least cn for some c. We also show that the optimum of this problem is unlikely to be better approximated than with a ratio of $\mathcal{O}(\log n)$ for general graphs. For the MAXIMUMINTERNALSPANNINGTREE problem, we give a 2-approximation algorithm.

Keywords: spanning trees, branching nodes, leaves, approximation

1 Introduction

The design process of optical networks raises various new graph theoretical problems. The progress of different multiplexing technologies can guarantee a high bandwidth over a single physical link. To obtain this, different connections must be join and then forked by specific routing devices. However the currently high price of such devices makes us to minimize their quantity in a network. Given a physical topology of an optical network, and the quantity of demands between its nodes, our research aims to install as few devices as possible, and satisfy all demands using them. As the general problem, being a combination of a facility location and a routing problem is extremely complex, we deal in this paper with an important special case. We suppose that the network topology to be designed is a tree, and that the specific routing devices must be installed to the branching nodes of the tree.

We present some positive and negative approximability results for the following problems: MINIMUMBRANCHINGSPANNINGTREE problem to find a spanning tree of a minimum number of branching (degree ≥ 3) nodes; MAXIMUMINTERNALSPANNINGTREE problem to find a spanning tree of a maximum number of non-leaf (degree ≥ 2) nodes.

Searching a spanning tree that satisfies some constraints and/or optimizes a goal function has a large literature. However, only a very few paper deal with the case when the quality of a solution depends only on the degree of nodes in the spanning tree. The MAXIMUMLEAFSPANNINGTREE problem, or the equivalent CONNECTEDDOMINATINGSET problem was already approximated by a factor of 2 [10], and $\log n$ [6], respectively.* Our results presented in this paper fit into this class of optimization problems.

2 Problem Formulation

Before formally defining the problems under consideration, we need some basic notations. By a graph $G = (V,E)$ we mean an undirected simple graph of $n = |V|$ nodes and $m = |E|$ edges. $N(v)$, or $N_G(v)$ denotes the set of neighbors of v in G. A

*Research is supported by Grant Nos. 042559, and 044733 of the Hungarian National Science Foundation (OTKA), and by the Grant No. 2003-5044438 of the European MCRTN Adonet Contract, while visiting the Graph Theory and Combinatorial Optimization group in Grenoble.

*A connected dominating set is a set of nodes which together with its neighbors contains all nodes of the graph, and in addition which spans a connected subgraph. Finding such a set of minimum cardinality is equivalent to find a spanning tree with a maximum number of leaves, however as the goal functions are duals of each other, they have different approximability properties.

degree of a node v in a subgraph H is denoted by $d_H(v)$. If it is not confusing, we use $d(v)$ instead of $d_G(v)$. If $A \subseteq V$ we use $d(A,v)$ to denote $|N(v) \cap A|$. Let $V_1(H)$, $V_2(H)$, and $V_{\geq 3}(H)$ denote the set of nodes of degree 1, 2, and at least 3 in H. The elements of $V_{\geq 3}(H)$ are called *branching nodes of* H. By a weighted graph we mean a triple $G = (V,E,c)$, where V and E are defined as above, and $c : E \to \mathbf{R}_+$ is a non-negative cost-function on the edges ($c(H) = \sum_{e \in E(H)} c(e)$, as usual). If it is not stated otherwise, our graphs are not weighted. For a variety of spanning tree optimization problems, and their approximations see [11].

We deal with the following problems, all of them are $\mathcal{NP}$-hard:

Problem 1 MINIMUMBRANCHINGSPANNINGTREE (MINBST) problem
Given an undirected graph $G = (V,E)$. The goal is to give a spanning tree H of G minimizing the number of branching nodes of H, namely the function: $cost(H) = |V_{\geq 3}(H)|$.

Problem 2 MINIMUMWEIGHTEDBRANCHINGSPANNINGTREE (MINWBST) problem
Given an undirected weighted graph $G = (V,E)$. The goal is to give a spanning tree H of G minimizing the sum of the number of branching nodes of H and of the total edge-cost of H, namely the function: $cost(H) = |V_{\geq 3}(H)| + c(H)$.[†]

According to our knowledge, this is the first paper considering these problems in such a form, although the following similar problems have already been widely investigated [1, 8, 10].

Problem 3 MAXIMUMLEAFSPANNINGTREE (MAXLST) problem
Given an undirected graph $G = (V,E)$. The goal is to give a spanning tree H of G maximizing the number of leaves of H, namely the function: $cost(H) = |V_1(H)|$.

Problem 4 MINIMUMLEAFSPANNINGTREE (MINLST) problem
Given an undirected graph $G = (V,E)$. The goal is to give a spanning tree H of G minimizing the number of leaves of H, namely the function: $cost(H) = |V_1(H)|$.

Although for MAXLST there is a 2-approximation algorithm [10], and LP-based algorithms also exist [3, 4], it is very unlikely that a constant approximation exists for the MINLST problem [7]. This makes us to examine the following problem, where the optimal solutions are the same as in the MINLST problem, but the negative approximability results do not hold as the cost function is dualized.

Problem 5 MAXIMUMINTERNALSPANNINGTREE (MAXIST)
Given an undirected graph $G = (V,E)$. The goal is to give a spanning tree H of G maximizing the number of internal nodes of H, namely the function: $cost(H) = |V_2(H)| + |V_{\geq 3}(H)| = n - |V_1(H)|$.

In this paper we present non-approximability results on the MINBST and the MINWBST problems. We give an approximation algorithm for "evenly dense" Hamiltonian graphs for the former one, and some approximability bounds on general graphs for the latter one. A 2-approximation algorithm for the MAXIST problem is also given.

3 Complexity and Approximability

Each problem of the previous section has an $\mathcal{NP}$-complete decision version. The HAMILTONIANPATH problem easily reduces to either the MINBST, or the MINWBST, or the MINLST, or the MAXIST problem. The $\mathcal{NP}$-completeness of the MAXLST problem was mentioned in [5].

The following theorem is our main negative approximability result on the MINIMUMBRANCHINGSPANNINGTREE problem. We recall that the MINSETCOVER (MINSC) problem is as follows: given a ground set $\mathcal{S}$, and a set $\Sigma = \{S_j\}_{j=1}^{s}$ of its subsets. The goal is to find a minimum number of subsets whose union contains the whole $\mathcal{S}$. This problem was shown to be non-approximable within a multiplicative factor of $(\log|\mathcal{S}|)$ unless $\mathcal{NP} \subseteq \mathcal{DTIME}(n^{\mathcal{O}(\log\log n)})$[2]. Note that the proof of this fact uses not more than $|\mathcal{S}|$ subsets, which is a crucial point of the polynomiality of our reduction below.

Theorem 6 For any $g(n) < \mathcal{O}(\log n)$ there is no $g(n)$-approximation algorithm for the MINBST problem unless $\mathcal{NP} \subseteq \mathcal{DTIME}(n^{\mathcal{O}(\log\log n)})$.

PROOF: We give an approximation preserving reduction of the MINSC problem to the MINBST problem, thus the approximability bound of MINSC yields directly the theorem.

Given an instance of the MINSC problem (defined as above), we construct the following graph G (Fig. 1)[‡].

[†]Taking a convex combination instead of the sum is an equivalent problem formulation as far as one can use a rescaling of the function c

[‡]We suppose that each element belongs to at least 2 subsets.

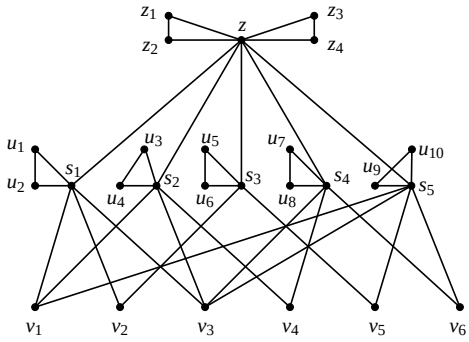


Figure 1: Reduction of the MINSC problem to the MINBST problem. The graph derived from the set $\mathscr{S} = \{1,2,3,4,5,6\}$, and its subsets $\Sigma = \{\{1,2,3\},\{1,3,4\},\{2,5\},\{3,4,6\},\{1,3,5,6\}\}$

For each element e_i of $\mathscr{S}$ we take a node v_i. For each subset S_j in Σ we take a node s_j, and connect it to a node v_i iff $e_i \in S_j$. We take additional nodes $\{u_j : j = 1,\ldots,2|\Sigma|\}$, z, z_1, z_2, z_3, and z_4. Then we connect both u_{2j-1} and u_{2j} to s_j and to each other. Furthermore, we make an edge between each s_j and z. Finally z is connected to all z_is, and z_1 is connected to z_2 while z_3 to z_4.

If S can be covered by k subsets $S_{j_1}, S_{j_2}, \ldots S_{j_k}$ in Σ, then taking all edges incident to $s_{j_1}, s_{j_2}, \ldots s_{j_k}$ covers all nodes v_i. We drop some edges if needed to set the degree of each node v_i to 1. Then one can easily form a spanning tree with $k+1$ branching nodes by adding all edges incident to z, and connecting u_{2j-1} to u_{2j} and to s_j. Thus we obtained that $cost(OPT(\text{MINBST})) \leq cost(OPT(\text{MINSC})) + 1$.

If we have a spanning tree of G with l branching nodes, then we got the followings: z must be a branching node in every spanning tree. Clearly, because of connection requirements, each v_i has at least one neighbor which is a branching node. Considering the sets $S_{j_1}, S_{j_2}, \ldots S_{j_r}$ corresponding to branching nodes $s_{j_1}, s_{j_2}, \ldots s_{j_r}$ we got a set covering of $\mathscr{S}$. Thus $r \leq l-1$, and if l was minimal, using the above we got $k = l-1$.

Thus if we had a spanning tree H which is a good approximation to the MINBST problem, we could get an approximative solution S of the MINSC problem with the following ratio: $cost(S) = cost(H) - 1 \leq g(n)cost(OPT(\text{MINBST})) = g(n)\,(cost(OPT(\text{MINSC})) + 1) \leq 2g(n)cost(OPT(\text{MINSC}))$. This forms a contradiction modulo $\mathscr{NP} \not\subseteq \mathscr{DTIME}(n^{\mathscr{O}(\log\log n)})$.

Note that we supposed in the proof that the number of the subsets is $\mathscr{O}(|\mathscr{S}|)$, however this does not influence our hardness result as the proof in [2] implicitly uses the same precondition. □

Then for the edge-weighted case we have the following corollary, whose proof is omitted here.

Corollary 7 For any $h(n) < \mathscr{O}(\log n)$ there is no $h(n)$-approximation algorithm for the MINWBST problem unless $\mathscr{NP} \subseteq \mathscr{DTIME}(n^{\mathscr{O}(\log\log n)})$.

Despite the above results the approximation ratio can be improved if the edge-weights are bounded. We show here a positive and a negative result that apply if this is the case.

The following upper bound on the approximation ratio is useful if the edge-weights are comparable to the cost of a branching node (close to 1 after a possible rescaling).

Proposition 8 There exists an $\left(1+\frac{1}{2s}\right)$-approximation algorithm for the MINWBST, where $s = \min_e c(e)$.

PROOF: We get the desired approximation ratio taking any minimal weight spanning tree T, as $cost(OPT(\text{MINWBST})) \geq c(T) \geq (n-1)s$ and $cost(T) = c(T) + |V_{\geq 3}(T)| \leq c(T) + \frac{n-2}{2}$. Here we used a trivial upper bound of $|V_{\geq 3}(T)| \leq \frac{n-2}{2}$ for the unweighted case. Thus,

$$\frac{cost(T)}{cost(OPT(\text{MINWBST}))} \leq \frac{c(T)+\frac{n-2}{2}}{c(T)} = 1 + \frac{n-2}{2c(T)} \leq 1 + \frac{n-2}{2(n-1)s} \leq 1 + \frac{1}{2s}.$$

□

When the edge-weights are negligible comparing to the cost of a branching node, the problem reduces to the search of a Hamiltonian path. Using this observation, we have the following negative result for the approximation ratio:

Proposition 9 There is no α-approximation algorithm for the MINWBST problem, if $\alpha < ((n-1)r)^{-1}$ and $r = \max_e c(e)$, unless $\mathscr{P} = \mathscr{NP}$.

PROOF: Let G be an arbitrary graph, and r, α are defined as above. If an α-approximation algorithm exists and gives a solution T we have the following:

If G has a Hamiltonian path H, $cost(OPT(\text{MINWBST})) \leq cost(H) = c(H) \leq (n-1)r$ and thus $cost(T) \leq \alpha(n-1)r < 1$. Contrary, in the case when G has no Hamiltonian path then $cost(OPT(\text{MINWBST})) \geq c(\text{MinSpanTree}) + 1 \geq 1$, which means that we would be able to decide the HAMILTONIANPATH problem in polynomial time. □

4 Approximation Algorithms

In this section we give an approximation algorithm for that special case of the MINBST problem when each node has a high degree, namely, when $d_G(v) \geq c|V|$ for all $v \in V$, and for a suitable constant c. Later, we also provide a 2-approximation for the MAXIST problem.

4.1 Approximating the MINBST problem for "evenly dense" graphs

Our approximation algorithm for the MINBST problem works as follows: We build a spanning tree H by subsequently adding edges to it starting from $H = (V, \emptyset)$. In iteration i we select a node p_i with the maximal number of such neighbors that are still isolated in H. Then we add as many edges of p_i to H as possible without forming a cycle. This process ends when a spanning forest without isolated vertices is formed. Then as a second phase, we connect the components of the forest by additional edges to obtain a spanning tree.

To do this, we will maintain three disjoint node sets: C containing the already selected nodes, B containing their neighbors, and A containing all other nodes (that are the currently isolated nodes of H). Initially all nodes are in A, and they are moved to B or C during the run of the algorithm. The first phase ends when A becomes empty.

Note that our algorithm shows some similarity with one of the algorithms for the CONNECTEDDOMINATINGSET problem in [6]. However they use it to prove a multiplicative approximation ratio of $(\log n)$, while in our context it is used to find a (not necessary connected) dominating set of cardinality $\leq \mathcal{O}(\log n)$. Thus we give here a different formulation, a different node-selection method, a different objective function, and a different proof.

We will need the following definition for the formal description.

Definition 10 The *Importance* of a node v in a subgraph H of G is defined as $imp_H(v) = comp_H(v \cup N_G(v)) - 1$, where $comp_H(X)$ is the number of components of H covering X. For some $Y \subseteq V$ we define the *Y-importance of v (in H)* as $comp_H((v \cup N_G(v)) \cap Y) - 1$ and denote it by $imp_{H,Y}(v)$. A set of edges $E' \subseteq E$ is called an *important edge set (of v in H)* if $|E'| = imp_H(v)$, and each edge of E' connects a different component (not containing v) of H to v.

That is, our algorithm always selects the node p_{i+1} having the highest A-importance (i.e., maximizing $imp_A(.)$) among all nodes of $A \cup B$. This means, that at first the node with the highest degree Δ is selected. In each iteration, the algorithm adds to H an important edge set of p_{i+1}, while moving p_{i+1} into C and its neighbors being in A into B.

We use the following simplification of notations throughout this section. X_i denotes the actual value of the structure X after the i^{th} iteration.[§] Furthermore, $imp(v) = imp_{H_i}(v)$, $imp_X(v) = imp_{H_i,X}(v)$, and l is the number of "while" iterations during the forest-building phase.

Algorithm 4:

Init: $H_0 := (V, \emptyset)$; $A_0 := V$; $B_0, C_0 := \emptyset$; $i := 0$; $\forall v \in V : imp(v) := d_G(v)$
Building a forest:
while $A_i \neq 0$ **do**
  $p_{i+1} := v \in A_i \cup B_i$ which maximizes $imp_{A_i}(.)$
  $C_{i+1} := C_i + p_{i+1}$; $B_{i+1} := (B_i \cup N_G(p_{i+1})) \setminus C_{i+1}$; $A_{i+1} := V \setminus (B_{i+1} \cup C_{i+1})$
  Let E' be an important edge set of p_{i+1} in H_i
  $H_{i+1} := H_i + E'$
  $\forall v \in A_{i+1} \cup B_{i+1} : imp_{A_{i+1}}(v) := comp_{H_{i+1}}((v + N_G(v)) \cap A_{i+1}) - 1$
  $i := i + 1$
Connecting the components: Add edges to H_i to obtain a spanning tree.

Theorem 11 Given an undirected graph $G = (V, E)$ with node degrees $\forall v \in V : d(v) \geq cn$. Then the above algorithm yields a spanning tree with at most $3\left(\frac{\log_2 n}{\log_2(\frac{1}{1-c})}\right) + 1$ branching nodes in $\mathcal{O}(n+m)\log^2 n$ time, where $n = |V|$, and $m = |E|$.

Moreover, if $\forall v \in V : d(v) \geq \frac{n}{2}$ $(*)$ then the obtained spanning tree has at most $\frac{3}{2}\log_2 n + 1$ branching nodes.

[§]For example A_i and A_{i+1} are not different variables, only the values of the same variable A after different iterations.

We will need the following lemmas:

Lemma 12 $(*) \Rightarrow comp_{H_i}(B_i \cup C_i) = 1$, namely, H_i is connected in $B_i \cup C_i$, and hence when A_i becomes empty, H_i forms a spanning tree of G.

PROOF: By induction, if $p_{i+1} \in B_i$ then $B_{i+1} \cup C_{i+1}$ is connected, even if $(*)$ does not hold. If $p_{i+1} \in A_i$, then $(*)$ implies that $|C_i \cup B_i| \geq |C_1 \cup B_1| \geq 1 + \frac{n}{2}$, and as $|N_G(p_{i+1})| \geq \frac{n}{2}$, $B_{i+1} \cup C_{i+1}$ is connected. □

Lemma 13 $\forall 1 \leq i \leq l : |A_i| \leq (1-c)^{i-1}(n-\Delta-1)$

PROOF: First we show that $imp_{A_i}(p_{i+1}) \geq c|A_i|$. Suppose indirectly that $imp_{A_i}(p_{i+1}) < k_i := \frac{|A_i|nc}{n-i}$. Then using that p_{i+1} has maximum A_i-importance, that H_i has no edge incident to A_i, and a double counting of the edges between A_i and B_i we obtain that:

$$k_i|A_i| > \sum_{w \in A_i} imp_{A_i}(w) = \sum_{w \in A_i} d(A_i, w) = \sum_{w \in A_i} d(w) - \sum_{w \in A_i} d(B_i, w) \geq$$
$$|A_i|nc - \sum_{w \in A_i} d(B_i, w) = |A_i|nc - \sum_{v \in B_i} d(A_i, v) = |A_i|nc - \sum_{v \in B_i} imp_{A_i}(v) > |A_i|nc - |B_i|k_i.$$

Thus $|A_i|nc < (|A_i| + |B_i|)k_i = (|V| - |C_i|)k_i = (n-i)k_i < |A_i|nc$ gives a contradiction, and so shows that $imp_{A_i}(p_{i+1}) \geq \frac{|A_i|nc}{n-i} \geq c|A_i|$. Using $|A_1| = n - \Delta - 1$ this directly proves the lemma by the observation, that at least $c|A_i|$ nodes of A_i are added to $B_{i+1} \cup C_{i+1}$. □

We now turn to the proof of Theorem 11.

PROOF: At first, we show that we need only $l \leq \mathcal{O}(\log n)$ iterations to build a spanning forest having $\mathcal{O}(\log n)$ branching nodes. Then we prove that connecting these components does not produce more than $\mathcal{O}(\log n)$ new branching nodes. This latter part is not necessary if $(*)$ holds, as in this particular case, the original spanning forest is already a spanning tree.

Clearly, the definition of A_i, B_i, and C_i implies that there is no edge of H_i incident to a node $w \in A_i$, that $|C_i| = i$, that $comp_{G_i}(B_i \cup C_i) \leq i$, and that there is no edge of H_i spanned by B_i.

By Lemma 13 l iteration is enough to build the spanning forest (that is to move all nodes of V into $B_l \cup C_l$, or equivalently to make C_l a dominating set of V) if

$$(n-\Delta-1)(1-c)^{l-1} < 1 \Leftrightarrow l \geq \left\lfloor \frac{\log_2(n-\Delta-1)}{\log_2\left(\frac{1}{1-c}\right)} \right\rfloor + 2.$$

Using $nc \leq \Delta$ this shows that there are

$$l \leq \frac{\log_2 n}{log_2 \frac{1}{1-c}} + 1 \qquad (**)$$

iterations.

We now turn to counting the branching nodes in the obtained spanning forest. On one hand, in C_l we have $b_1 \leq l$ of them. On the other hand, as B_l is a stable set in H_l we have $\sum_{v \in B_l} d_{H_l}(v) \leq \frac{1}{2}E(H_l) = n - comp(B_l \cup C_l)$. Hence the number of branching nodes in B_l before connecting the components of the forest is $b_2 \leq \frac{1}{2}\left(\sum_{v \in B_l} d_{H_l}(v) - |B_l|\right) \leq \frac{1}{2}(l - comp(B_l \cup C_l))$. This means that if $(*)$ holds, and H_l must be a spanning tree by Lemma 12, then we have a solution $H = H_l$ with $|V_{\geq 3}(H)| = b_1 + b_2 \leq \frac{3}{2}l - \frac{1}{2} \leq \frac{3}{2}(\log_2 n) + 1$ branching nodes. This proves the second part of the theorem.

If H_l has more than one components, they must be connected by $comp(B_l \cup C_l)$ edges (say E'') that produces $b_3 \leq 2\left[comp(B_l \cup C_l) - 1\right]$ new branching nodes. Thus we have a solution $H = H_l \cup E''$ with b branching nodes such that $b = b_1 + b_2 + b_3 \leq \frac{3}{2}l + \frac{3}{2}comp(B_l \cup C_l) - 2 \leq 3l - 2$. Adding $(**)$ proves the theorem.

If a union-lookup data structure is used for maintaining components we get the following bounds for the time complexity of the algorithm. A single iteration step is composed of searching for the maximum A-importance node ($\mathcal{O}(n)$), joining the components of the node and its neighbors ($\mathcal{O}(\Delta)$), and updating importance values. This latter needs $\mathcal{O}(n+m)$ LOOKUP calls in $\mathcal{O}((n+m)\log n)$ time. The number of iterations is $\mathcal{O}(\log n)$, hence the spanning forest of $\mathcal{O}(\log n)$ components can be found in $\mathcal{O}(\log^2 n(n+m))$ time. Components can be joined to form a spanning tree by calling LOOKUP $\mathcal{O}(m)$ times. Thus the total time complexity of the algorithm is $\mathcal{O}(n+m)\log^2 n$. □

4.2 Approximating the MAXIST problem

In the followings we give a 2-approximation algorithm for the MAXIST problem. Its approximation ratio is proved by a primal-dual technique. Note that the similar MAXIMUMLEAFSPANNINGTREE problem has been examined by linear programming techniques recently [3, 4].

Let us consider an undirected graph $G=(V,E)$. Defining a vector $x\in\{0,1\}^{|E|}$ over the edge set of G, the convex hull of the characteristic vectors of spanning trees is described by the above spanning tree polyhedron [9]:

$$\mathscr{SP}(G)=\{x \mid \forall S\subseteq V : x(S)\le |S|-1, x(V)=|V|-1, \forall e\in E : 0\le x(e)\le 1\},$$

where $x(S)=\sum_{e\in\gamma(S)} x(e)$ is the sum of x over all edges spanned by S.

Let us consider the following linear program ($\gamma(S)$ are the edges spanned by S, and $\delta(v)$ are the edges incident to v):

$$\begin{aligned}
&\max \sum_{v\in V} q(v) - \sum_{v\in V} w(v) \\
&\text{s. t.,} \\
\forall S\subseteq V : &\sum_{e\in\gamma(S)} x(e) \le |S|-1 && (1)\\
&\sum_{e\in E} -x(e) \le -(|V|-1) && (2)\\
\forall v\in V : -&\sum_{e\in\delta(v)} x(e) - \sum_{e\in\delta(v)} w(v) \le -2 && (3)\\
\forall v\in V : &\; q(v) = 1 \\
\forall e\in E, v\in V : &\; x(e), w(v) \ge 0 && (4)
\end{aligned}$$

Here (1), (2), and (4) define the spanning tree polyhedron as above, while (3) is to set an indicator $w(v)=1$ whenever v is a leaf of the spanning tree, and q is a fake variable only used for dualizing the goal function. Although this is not an integer polyhedron, each minimal integer solution defines a spanning tree of $w(v)$ leaves. To see this, one has to notice that in any feasible solution $\forall e : x(e)\le 1$, and $w(v)=0$ iff v is not a leaf.

The dual of the above program is:

$$\begin{aligned}
&\min \sum_{S\subseteq V} (|S|-1)y(S) - (|V|-1)t - \sum_{v\in V} 2z(v) + \sum_{v\in V} r(v) \\
&\text{s. t.,} \\
\forall e : &\sum_{e\in\gamma(S)} y(S) - t - \sum_{e\in\delta(v)} z(v) \ge 0 \\
\forall v : &-z(v) \ge -1 \\
\forall v : &\; r(v) = 1 \\
\forall S, v : &\; y(S), t, z(v) \ge 0
\end{aligned}$$

Informally we are looking for a subset Z of nodes (characterized by $z(v)=1$) and a cover $\mathscr{Y}$ by subsets (defined by y), such that each edge is spanned by at least as many elements of $\mathscr{Y}$ as many of its ends are in Z.

Let us have a spanning tree T whose leaves form a stable set of G. Then the corresponding primal solution has a value $V_2(T)+V_3(T)$. Define dual variables as follows: $z(v)=1$ iff z is a leaf of T, $y(V)=1$, $r(v)=1$ for all v. All others are 0. Then we have a feasible dual solution, as each edge is spanned by V, and no edge has more than one ends in Z. The value of this solution is $(|V|-1)-2V_1(T)+|V|\le 2(|V_2(T)|+|V_3(T)|)$ ensuring the approximation ratio of 2.

Now we have to prove that one can find a spanning tree whose leaves are independent in G.

Let us have any spanning tree T. If there is no edges in G between the leaves of T, we are done. Suppose that there is an edge $e=(u,v)\in E$ between two leaves of T. $T+e$ contains a unique cycle C. Either this is a Hamiltonian cycle (and then T is a Hamiltonian path, so an optimal solution), or C has an edge $f=(x,y)$ such that $d_T(x)\ge 3$. In this case $T'=T+e-f$ forms a spanning tree having more leaves than T. One can see now by induction that repeatedly applying this we obtain either a spanning tree with independent leaves or a Hamiltonian path.

So we have the following theorem:

Theorem 14 There exists a 2-approximation algorithm for the MAXIST problem, which runs in $\mathscr{O}(nm)$ time.

PROOF: The correctness and the approximation ratio are immediate consequences of the above argument. An initial spanning tree can be built together with finding "leaf-leaf" edges in $\mathscr{O}(n+m)$ time. Adding an edge according to the algorithm and

finding one to remove from the unique cycle needs $\mathcal{O}(n)$ time. One can refresh the list of "leaf-leaf" edges in $\mathcal{O}(m)$ time. As by each iteration the number of leaves decreases, we need at most n iterations, that yields a total running time of $\mathcal{O}(nm)$. □

The following example shows that the analysis of the algorithm is tight. Take a graph $G = (V,E)$ where $V = \{v_1, v_2, \ldots, v_{n/2}, u_1, u_2, \ldots, u_{n/2}\}$, and connect each v_i to v_{i+1} forming a path P of length $n-1$. Furthermore connect each u_i to v_i and to v_{i+1}. This graph has a Hamiltonian path, so an optimal solution contains $n-2$ non-leaf nodes. Taking the spanning tree T such that $E(T) = P \cup \{(u_i, v_i)\}$, T has $n/2$ non-leaf nodes, and their leaves form a stable set.

5 Conclusions and Final Remarks

We considered two spanning tree optimization problems with a degree-based goal function. For the MINIMUMBRANCHINGSPANNINGTREE problem we gave an $\mathcal{O}(\log n)$ approximation for input graphs having minimum degree cn. This is a result which holds for a subclass of Hamiltonian graphs. For the general case, we have shown that it is very unlikely that a better than $\mathcal{O}(\log n)$ approximation exists. An interesting direction of further research should fill the gap between these results either by giving an approximation of $\mathcal{O}(\log n)$ for general graphs, or by establishing a tighter negative result. For the MAXIMUMINTERNALSPANNINGTREE problem we gave the first approximation algorithm having an approximation ratio of 2. As this bound was shown to be sharp our forthcoming research aims to design better algorithms for improving this ratio.

Acknowledgements. The author thanks to András Recski and to András Sebő for their valuable comments and remarks.

References

[1] Y. Caro, D. B. West, and R. Yuster. Connected domination and spanning trees with many leaves. *SIAM Journal on Discrete Mathematics*, 13(2):202–211, 2000.

[2] U. Feige. A threshold of ln *n* for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

[3] L. M. Fernandes and L. Gouveia. Minimal spanning trees with a constraint on the number of leaves. *European Journal of Operational Research*, 104:250–261, 1998.

[4] T. Fujie. The maximum-leaf spanning tree problem: Formulations and facets. *Networks*, 43(4):212–223, 2004.

[5] M. R. Garey and D. S. Johnson. *Computers and intractability: a guide to NP-completeness*. Freeman, 1979.

[6] S. Guha and S. Khuller. Approximation algorithms for connected dominating sets. In *Proc. of European Symposium on Algorithms*, pages 179–193, 1996.

[7] H.-I. Lu and R. Ravi. The power of local optimization: approximation algorithms for maximum-leaf spanning tree. In *Proc. of 30th Annual Allerton Conference on Communication, Control and Computing*, pages 533–542, 1992.

[8] H.-I Lu and R. Ravi. Approximation for maximum leaf spanning trees in almost linear time. *J. Algorithms*, 29(1):132–141, 1998.

[9] A. Schrijver. *Combinatorial optimization, Polyhedra and efficiency. Vol. B*, chapter 50, pages 855–876. Springer-Verlag, 2003.

[10] R. Solis-Oba. 2-approximation algorithm for finding a spanning tree with maximum number of leaves. In *Proc. of 6th ESA Symposium, Springer LNCS Vol. 1461*, pages 441–452, 1998.

[11] B. Y. Wu and K.-M. Chao. *Spanning Trees and Optimization Problems*. Chapman & Hall / CRC, 2004.

Sharpenings of Sauer's Bound

RICHARD ANSTEE*

Mathematics Department
The University of British Columbia
Vancouver, B.C., Canada V6T 1Z2
anstee@math.ubc.ca

BALIN FLEMING*

Mathematics Department
The University of British Columbia
Vancouver, B.C., Canada V6T 1Z2
balinf@interchange.ubc.ca

ZOLTÁN FÜREDI†

Department of Mathematics
University of Illinois at Urbana-Champaign
1409 W. Green Street Urbana, Illinois 61801-2975, USA
and
Alfréd Rényi Instiute of Mathematics
Hungarian Academy of Sciences
Budapest, P.O.Box 127 H-1364 Hungary
z-furedi@math.uiuc.edu

ATTILA SALI‡

Alfréd Rényi Instiute of Mathematics
Hungarian Academy of Sciences
Budapest, P.O.Box 127 H-1364 Hungary
sali@renyi.hu

Abstract: The present paper gives sharpenings of Sauer's bound on forbidden configurations. We define a matrix to be *simple* if it is a (0,1)-matrix with no repeated columns. Let F be a $k \times l$ (0,1)-matrix (the forbidden configuration). Assume A is an $m \times n$ simple matrix which has no submatrix which is a row and column permutation of F. We define $\text{forb}(m,F)$ as the best possible upper bound on n, for such a matrix A, which depends on m and F. It is known that $\text{forb}(m,F) = O(m^k)$ for any F, and Sauer's bond states that $\text{forb}(m,F) = O(m^{k-1})$ fore *simple* F. We give sufficient condition for non-simple F to have the same bound using linear algebra methods. $\text{forb}(m,\mathscr{F})$ for *families* of forbidden configurations is also discussed.

Keywords: forbidden configuration, extremal hypergraph, linear algebra method

1 Introduction

The study of forbidden configurations is a problem in extremal set theory. The language we use here is matrix theory which conveniently encodes the problems. We define a *simple* matrix as a (0,1)-matrix with no repeated columns. Such a matrix can be thought of a set of subsets of $\{1,2,\ldots,m\}$ with the columns encoding the subsets and the rows indexing the elements. Assume we are give a $k \times l$ (0,1)-matrix F. We say that a matrix A has no *configuration* F if no submatrix of A is a row and column permutation of F and so F is referred to as a *forbidden configuration* (sometimes called *trace*). A variety of combinatorial objects can be defined by forbidden configurations. For a simple $m \times n$ matrix A which is assumed to have no configuration F, we seek an upper bound on n which will depend on m,F. We denote the best possible upper bound as $\text{forb}(m,F)$. Many results have been obtained about $\text{forb}(m,F)$ including [2],[3],[5].

At this point all values known for $\text{forb}(m,F)$ are of the form $\Theta(m^e)$ for some integer e. We completed the classification for $2 \times l$ matrices F in [2] and for $3 \times l$ matrices F in [6]. We also put forward a conjecture on what properties of F drive the exponent e. Roughly speaking, we proposed a set of constructions and guessed that these constructions are sufficient to deduce the exponent e in the expression $\Theta(m^e)$.

We use the notation K_k to denote the $k \times 2^k$ simple matrix of all possible columns on k rows. The basic result for $\text{forb}(m,F)$ is as follows.

Theorem 1 [Sauer [12], Perles and Shelah [13], Vapnik and Chervonenkis [14]] We have that $\text{forb}(m,K_k)$ is $\Theta(m^{k-1})$.

In fact Theorem 1 is usually stated with $\text{forb}(m,K_k) = \binom{m}{k-1} + \binom{m}{k-2} + \cdots + \binom{m}{0}$ but the asymptotic growth of $\Theta(m^{k-1})$ was what interested Vapnik and Chervonenkis.

*Research is supported in part by NSERC
†Research is supported in part by Hungarian National Research Fund (OTKA) numbers T034702 and T037846
‡Research is supported in part by Hungarian National Research Fund (OTKA) numbers T034702 and T037846 and by NSERC of the first author

One easy observation is that if we let A^c denote the 0-1-complement of A then $\text{forb}(m,F^c) = \text{forb}(m,F)$. Another observation is that if F' is a submatrix of F, then $\text{forb}(m,F) \geq \text{forb}(m,F')$. We let K_k^s denote the $k \times \binom{k}{s}$ simple matrix of all possible columns of column sum s.

We use the notation $[A|B]$ to denote the matrix obtained from concatenating the two matrices A and B. We use the notation $k \cdot A$ to denote the matrix $[A|A|\cdots|A]$ consisting of k copies of A concatenated together. We give precedence to the operation $\cdot$ (multiplication) over concatenation so that for example $[2 \cdot A|B]$ is the matrix consisting of the concatenation of B with the concatenation of two copies of A.

Below is a conjecture [6] that focused our attention while exploring results in this paper. Let A_i be an $m_i \times n_i$ simple matrix for $1 \leq i \leq k$. Denote $A_1 \times A_2 \times \cdots \times A_k$ as the $(\sum m_i) \times (\Pi n_i)$ simple matrix whose columns are formed in all possible ways by putting a column of A_1 in the first m_1 rows and putting a column of A_2 in the next m_2 rows etc. Let T_h denote the $h \times h$ *triangular* matrix

$$T_h = \begin{bmatrix} 1 & & 1's \\ & 1 & \\ & \ddots & \\ 0's & & 1 \end{bmatrix}.$$

Let F be a $k \times l$ (0,1)-matrix. Let $X(F)$ be the smallest p so that F is a configuration in $A_1 \times A_2 \times \cdots \times A_p$ for every choice of A_i as either $K^1_{m/p}$, $K^{(m/p)-1}_{m/p}$ or $T_{m/p}$. We assume m is large and divisible by p, in particular that $m \geq (k+1)(kl+1)$ so that $m/p \geq kl+1$. Divisibility by p does not affect the asymptotics since we can use a simple submatrix of a simple matrix that avoids F for construction purposes. We are using the fact that we need only consider p-fold products for $p \leq k+1$, since we can find F as a configuration in $A_1 \times A_2 \times \cdots \times A_{k+1}$ by taking 1 row from each of the first k products (each row has [01]) and then, since we are taking zero rows from the final A_{k+1}, we get the configuration $(m/(k+1)) \cdot K_k$ in the product and F is a configuration in $l \cdot K_k$.

If F is a configuration in the p-fold product $A_1 \times A_2 \times \cdots \times A_p$, assume that a_i rows of A_i are used with $\sum_{i=1}^p a_i = k$. If we form the submatrix of A_i of a_i rows, then we would be interested in at most l copies of a given column on these rows (F has l columns) if this is possible. Now for $t \geq k+l$, any a_i rows of K_t^1 contains l columns of 0's as well as a copy of $K^1_{a_i}$. The analogous result is true for K_t^{t-1}. Also for $t \geq kl+l$, the a_i rows of T_t consisting of rows $l+1, 2l+1, 3l+1, \ldots, kl+1$ have l columns of 0's and $l \cdot T_{a_i}$. Thus as long as $m \geq (k+1)(kl+1)$ we are able to use the matrices A_i as if they were arbitrarily large.

Note that the definition of $X(F)$ ensures $\text{forb}(m,F)$ is $\Omega(m^{X(F)-1})$, although for $X(F) = 1$ a little care must be taken.

Conjecture 2

$$\text{forb}(m,F) = \Theta(m^{X(F)-1}).$$

According to a result of Füredi [10] $\text{forb}(m,F) = O(m^k)$ for *arbitrary* $k \times l$ configuration F. The goal of this paper is to give sufficient conditions that ensure $\text{forb}(m,F) = O(m^{k-1})$.

2 The boundary between m^{k-1} and m^k

Theorem 1 implies that simple configurations all have $\text{forb}(m,F) = O(m^{k-1})$, thus we investigate f's with multiple columns. First, we show that which configurations F have $\text{forb}(m,F) = \Omega(m^k)$ using the direct product construction. Let $A(k,2)$ be defined as a minimal matrix with the property that any pair of rows has $\begin{bmatrix}1\\1\end{bmatrix}$ has both with 1's in some column and such that deleting a column of $A(k,2)$ would violate this property.

Lemma 3 Let F be a $k \times l$ configuration. $\text{forb}(m,F) = \Omega(m^k)$ if F contains $2 \cdot K_k^l$ for $2 \leq l \leq k-2$ and $l = 0,k$ or if F contains $[2 \cdot K_k^1 | A(k,2)]$.

PROOF: We find that $\text{forb}(m,F)$ is $\Omega(m^k)$ if F contains $2 \cdot K_k^l$ for $0 \leq l \leq k$ and $l \neq 1, k-1$. This follows since $2 \cdot K_k^l$ is not contained in the k-fold product of l $K^1_{m/k}$'s and $k-l$ $K^{(m/k)-1}_{m/k}$'s and so may deduce $\text{forb}(m, 2 \cdot K_k^l)$ is $\Omega(m^k)$. To verify this for $2 \leq l \leq k-2$, we note that any pair of rows of K_k^l has $\begin{bmatrix}1&0\\1&0\end{bmatrix}$ and so if we have a submatrix that is a row and column permutation of K_k^l, we can only choose one row from either $K^1_{m/k}$ or from $K^{(m/k)-1}_{m/k}$. The verification for K_k^0 or K_k^k is easier.

For $l = 1$ (the case $l = k-1$ is the (0,1)-complement) we can no longer assert that any pair of rows of K_k^l has $\begin{bmatrix}1&0\\1&0\end{bmatrix}$ merely $\begin{bmatrix}0\\0\end{bmatrix}$ and so can choose two rows from the copy of $K^1_{m/k}$, one row from each of $k-2$ of the $K^{(m/k)-1}_{m/k}$ terms and generate a copy

of $2\cdot K_k^1$. (Theorem 5.1 of [6] shows that $\mathrm{forb}(m,K_k^1)$ is $\Theta(m_{k-1})$). This is fixed by considering a minimal matrix $A(k,2)$ with the property that any pair of rows has $\begin{bmatrix}1\\1\end{bmatrix}$ has both with 1's in some column and such that deleting a column of $A(k,2)$ would violate this. As above, we have that if F contains $[2\cdot K_k^1|A(k,2)]$, then $\mathrm{forb}(m,F)$ is $\Omega(m^k)$. □

Lemma 3 leaves two possibilities if we want $\mathrm{forb}(m,f)$ be bounded away from m^k. Either F is contained in a matrix $F_B=[K_k\,|\,t\cdot(K_k-B)]$ for an $k\times(k+1)$ matrix B consisting of one column of each possible column sum or F is contained in a matrix $[K_k^0|\,t\cdot C]$ where C is a k-rowed simple matrix consisting of all columns which do not have 1's in both rows 1 and 2 and also with at least one 1. Note, that these are not mutually exclusive cases. Our main theorem is that in the first case $\mathrm{forb}(m,F)=O(m^{k-1})$.

Theorem 4 Let F be contained in $F_B=[K_k\,|\,t\cdot(K_k-B)]$ for an $k\times(k+1)$ matrix B consisting of one column of each possible column sum. Then $\mathrm{forb}(m,F)=O(m^{k-1})$.

PROOF:

Let A be an $m\times n$ simple 0-1 matrix, and B be a $k\times(k+1)$ matrix consisting of one column of each possible column sum. Suppose that A does not have $F_B=[K_k\,|\,t\cdot(K_k-B)]$ as configuration. This implies that on a given k-tuple L of rows either K_k is missing, or if all possible columns of size k occur on L, then $t\cdot(K_k-B)$ must be missing. This latter means, that for some $0\le s\le k$, two columns of column sum s occur at most $t-1$ times on L, respectively. Let $\mathscr{K}$ be the set of k-tuples of rows where the latter happens. Using Lemma 5 a set of columns of size $O(m^{k-1})$ can be removed from A to obtain A', so that for all $L\in\mathscr{K}$ a column (in fact two) is missing on L in A'. However, this implies that K_k is not a configuration in A', thus by Theorem 1 A' has at most $O(m^{k-1})$ columns. □

Let $\mathscr{K}$ be a system of k-tuples of rows such that $\forall K\in\mathscr{K}$ there are two $(k\times 1)$ columns, $\alpha_K\neq\beta_K$ specified. We say that a column x of A *violates* (K,α_K), if $x|_K=\alpha_K$, similarly, x *violates* (K,β_K), if $x|_K=\beta_K$.

Lemma 5 Assume, that for every $K\in\mathscr{K}$ there are at most $t-1$ columns of A that violate (K,α_K), and at most $t-1$ columns of A violate (K,β_K). Then there exists a subset X of columns of A, such that $|X|=O(m^{k-1})$ and no column of $A-X$ violates any of (K,α_K) or (K,β_K).

PROOF: It can be assumed without loss of generality that for all $K\in\mathscr{K}$ $\alpha_K=\alpha$ and $\beta_K=\beta$ independent of K. Indeed, there are $2^k\times 2^k$ possible α_K,β_K pairs, that is a constant number of them .Thus, $\mathscr{K}$ can be partitioned into a constant number of parts, so that in each part $\alpha_K=\alpha$ and $\beta_K=\beta$ holds. We apply induction on k using the simplification given above. $k=1$ is obvious.

Consider now $k\times 1$ columns $\alpha\neq\beta$. Assume first, that $\alpha\neq\overline{\beta}$. That is, there is a coordinate where α and β agree, say both have 1 as their ℓth coordinate. The case of a common 0 coordinate is similar. For the ith row of A we count how many columns have violation so that for some $K\in\mathscr{K}$ the ℓth coordinate in K is exactly row i. Let $\mathscr{K}_{i,\ell}$ be the set of these k-tuples from $\mathscr{K}$. Columns that have violation on k-tuples from $\mathscr{K}_{i,\ell}$ have 1 in the ith row, let $A_{i,1}$ denote matrix formed by the set of columns that have 1 in row i. If row i is removed from $A_{i,1}$, the remaining matrix $A'_{i,1}$ is still simple. Let $\mathscr{K}'_{i,\ell}$ denote the set of $(k-1)$-tuples obtained from k-tuples of $\mathscr{K}_{i,\ell}$ by removing their ℓth coordinate, i, furthermore let α' (β', respectively) denote the $(k-1)\times 1$ column obtained from α (β) by removing the ℓth coordinate, 1. Note, that $\alpha'\neq\beta'$. A column of A has a violation on $K\in\mathscr{K}_{i,\ell}$ iff its counterpart in $A'_{i,1}$ has a violation on the corresponding $K'\in\mathscr{K}'_{i,\ell}$. The number of those columns is at most cm^{k-2} by the inductive hypothesis. Since $\mathscr{K}=\cup_{i=1}^m\mathscr{K}_{i,\ell}$, we obtain that the number of columns of A having violation on some $K\in\mathscr{K}$ is at most $m\cdot cm^{k-2}$.

Let us assume now, that $\alpha=\overline{\beta}$. A subset $\mathscr{J}\subseteq\mathscr{K}$ is called *independent* if there exists an ordering $J_1,J_2,\dots J_g$ of the elements of $\mathscr{J}$ such that for every $J_i\in\mathscr{J}$ there exists an $m\times 1$ 0-1 column that violates J_i and does not violate any $J_j\in\mathscr{J}$ for $j<i$. Let us call a *maximal* independent subset $\mathscr{B}$ of $\mathscr{K}$ a *basis* of $\mathscr{K}$. If a column of A has a violation on $K\in\mathscr{K}$, then it has a violation on some $B\in\mathscr{B}$, as well. Indeed, either $K\in\mathscr{B}$ holds, or if $K\notin\mathscr{B}$, then by the maximality of $\mathscr{B}$, K cannot be added to it as a $|\mathscr{B}|+1$st element in the order, so the column having violation on K must have a violation on $B\in\mathscr{B}$, for some B. By Theorem 6 for a basis $\mathscr{B}$ we have

$$|\mathscr{B}|\le\binom{m}{k-1}+\binom{m}{k-2}+\ldots+\binom{m}{0},$$

since a column violating a k-tuple B_i from $\mathscr{B}$, but none of B_j for $j<i$, gives an appropriate partition of the set of rows.

Thus, there could be at most $(2t-2)\left[\binom{m}{k-1}+\binom{m}{k-2}+\ldots+\binom{m}{0}\right]$ columns violating some $K\in\mathscr{K}$. □

The following theorem is of independent interest.

Theorem 6 Let $\mathscr{E}\subseteq\binom{[m]}{k}$ be a k-uniform set system on an underlying set X of m elements. Let us fix an ordering $E_1,E_2,\dots E_t$ of $\mathscr{E}$ and a prescribed partition $A_i\cup B_i=E_i$ ($A_i\cap B_i=\emptyset$) for each member of $\mathscr{E}$. Assume that for all $i=1,2,\dots,t$ there exists

a partition $C_i \cup D_i = X$ ($C_i \cap D_i = \emptyset$), such that $E_i \cap C_i = A_i$ and $E_i \cap D_i = B_i$, but $E_j \cap C_i \neq A_j$ and $E_j \cap C_i \neq B_j$ for all $j < i$. (That is, the ith partition cuts the ith set as it is prescribed, but does not cut any earlier set properly.) Then

$$t \leq \binom{m}{k-1} + \binom{m}{k-2} + \ldots + \binom{m}{0}. \tag{1}$$

PROOF: We define a polynomial $p_i(\underline{x}) \in \mathbb{R}[x_1, x_2, \ldots, x_m]$ for each E_i as follows.

$$p_i(x_1, x_2, \ldots, x_m) = \prod_{a \in A_i} (1 - x_a) \prod_{b \in B_i} x_b + (-1)^{k+1} \prod_{a \in A_i} x_a \prod_{b \in B_i} (1 - x_b) \tag{2}$$

Polynomials defined by (2) are multilinear of degree at most $k-1$, since the product $\prod_{e \in E_i} x_e$ cancels by the coefficient $(-1)^{k+1}$. Thus, they are from the space generated by monomials of type $\prod_{j=1}^{r} x_{i_j}$, for $r = 0, 1, \ldots k-1$. The dimension of this space over $\mathbb{R}$ is $\binom{m}{k-1} + \binom{m}{k-2} + \ldots + \binom{m}{0}$.

We shall prove that polynomials $p_1(\underline{x}), p_2(\underline{x}), \ldots p_t(\underline{x})$ are linearly independent over $\mathbb{R}$, which implies (1). Assume that

$$\sum_{i=1}^{t} \lambda_i p_i(\underline{x}) = 0 \tag{3}$$

is a linear combination of the $p_i(\underline{x})$'s that is the zero polynomial. Consider the partition $C_t \cup D_t = X$, and substitute $x_c = 0$ if $c \in C_t$ and $x_d = 1$ if $d \in D_t$ into (3). Then $p_t(\underline{x}) = 1$, but it is easy to see that $p_k(\underline{x}) = 0$ for $k < t$. This implies that $\lambda_t = 0$. Now assume by induction on j, that $\lambda_t = \lambda_{t-1} = \ldots = \lambda_{t-j+1} = 0$. Take the partition $C_{t-j} \cup D_{t-j} = X$ and substitute into (3) $x_c = 0$ if $c \in C_{t-j}$ and $x_d = 1$ if $d \in D_{t-j}$. Then, as before, $p_{t-j}(\underline{x}) = 1$, but $p_k(\underline{x}) = 0$ for $k < t - j$. This implies $\lambda_{t-j} = 0$, as well. Thus, all coefficients in (3) must be 0, hence the polynomials are linearly independent. □

3 Discussion

It is natural to ask forb$(m, \mathscr{F})$, where $\mathscr{F} = \{F_1, F_2, \ldots, F_t\}$ is a family of forbidden configuration. In this case forb$(m, \mathscr{F})$ is the maximum n such that there exists an $m \times n$ simple 0-1 matrix A with none of the configurations $F_i \in \mathscr{F}$. Balogh and Bollobás [7] made the first step. They proved, although in a different setting, that

$$\text{forb}(m, \{I_k, I_k^c, T_k\}) = O(1). \tag{4}$$

This was independently discovered by Anstee, and also a shorter proof with better constant was given by Keevash and Sudakov. (4) justifies the use of I_k, I_k^c, T_k in Conjecture 2 in the sense that allows us to consider direct products where all terms are one of the three matrices above. Also, (4) fits into the general pattern of Conjecture 2, since $X(\mathscr{F}) = 1$ for $\mathscr{F} = \{I_k, I_k^c, T_k\}$. However, the analogy stops here. Indeed, the nest step would be the nine element set of configurations $\mathscr{F}_2 = \{A_1 \times A_2 \colon A_i \in \{I_k, I_k^c, T_k\} \text{ for } i = 1, 2\}$. It is clear that $X(\mathscr{F}_2) = 2$, so one expects forb$(m, \mathscr{F}_2) = \Theta(m)$. However, that is not true. For $k = 2$ we have $I_2 = I_2^c$ as configurations, thus only three products need to be considered: $I_2 \times I_2$, $T_2 \times I_2$ and $T_2 \times T_2$.

Proposition 7 We have that forb$(m, \{I_2 \times I_2, T_2 \times I_2, T_2 \times T_2\}) = \Omega(m^{\frac{3}{2}})$.

PROOF: It is well known [11], that the Turán number ex$(m, C_4) = \Theta(m^{\frac{3}{2}})$, see [8] for details. Let A be the $m \times n$ incidence matrix of a maximal C_4-free graph on m vertices. The number of columns of A is $n = \Theta(m^{\frac{3}{2}})$. $T_2 \times I_2$ and $T_2 \times T_2$ have columns with at least three 1's, thus they do not occur as configuration in an incidence matrix of a graph. $I_2 \times I_2$ is the incidence matrix of the 4-cycle C_4, so it is not a configuration of A. □

It is easy to see that the "real trouble maker" is $I_2 \times I_2$. Indeed,

Proposition 8 We have that forb$(m, \{T_2 \times I_2, T_2 \times T_2\}) = \Theta(m^2)$.

PROOF: The lower bound follows from the fact that both of the two forbidden configurations contain columns of at least three 1's, so they are not configurations of $I_k \times I_k$.

Let A be a simple $m \times n$ matrix with no configurations $T_2 \times I_2$ and $T_2 \times T_2$. Consider the standard decomposition of A

$$A = \begin{bmatrix} 1 \cdots 1 & 0 \cdots 0 \\ B_1 B_2 & B_2 B_3 \end{bmatrix}, \tag{5}$$

where $[B_1B_2B_3]$ is simple. Observe that

$$T_2 \times I_2 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \quad \text{and} \quad T_2 \times T_2 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}. \tag{6}$$

Applying the standard decomposition with respect to the second row for $T_2 \times I_2$, we obtain that $\begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$ is not a configuration of B_2. That is, considering columns of B_2 as sets, if two of them intersect, then one must contain the other. Similarly, the standard decomposition of $T_2 \times T_2$ with respect to the second row gives that $\begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 0 & 1 \end{bmatrix}$ is not a configuration of B_2. That is, if one set in B_2 contains another one, then the smaller set has at most one elements. These two observations imply that $|B_2| = O(m)$, hence by induction $n = O(m^2)$ $\square$

References

[1] R.P. ANSTEE, Some Problems Concerning Forbidden Configurations, preprint.

[2] R.P. ANSTEE, J.R. GRIGGS, A. SALI, Small Forbidden Configurations, *Graphs and Combinatorics* **13**(1997),97-118.

[3] R.P. ANSTEE, R. FERGUSON, A. SALI, Small Forbidden Configurations II, *Electronic J. Combin.* **8**(2001), R4 (25pp)

[4] R.P. ANSTEE, Z. FÜREDI, Forbidden Submatrices, *Discrete Math.* **62**(1986),225-243.

[5] R.P. ANSTEE, N. KAMOOSI, Small Forbidden Configurations III, preprint.

[6] R.P. ANSTEE, A. SALI Small forbidden configurations IV: The 3-rowed case, *Combinatorica*, to appear.

[7] J. BALOGH, B. BOLLOBÁS, Unavoidable Traces of Set Systems, to appear, Combinatorica.

[8] B. BOLLOBÁS Extremal graph theory. London Mathematical Society Monographs, **11** Academic Press, Inc. [Harcourt Brace Jovanovich, Publishers], London-New York, 1978.

[9] P. ERDŐS, A.H. STONE, On the Structure of Linear Graphs, *Bull. Amer. Math. Soc.* **52**(1946), 1089-1091.

[10] Z. FÜREDI, personal communication.

[11] T. KŐVÁRI, V.T. SÓS, P. TURÁN On a problem of K. Zarankiewicz, *Colloquium Math.* **3** (1954) 50-57.

[12] N. SAUER, On the density of families of sets, *J. Combin. Th. Ser A* **13**(1973), 145-147.

[13] S. Shelah, A combinatorial problem: Stability and order for models and theories in infinitary languages, *Pac. J. Math.* **4**(1972), 247-261.

[14] V.N. VAPNIK, A.YA. CHERVONENKIS, On the uniform convergence of relative frequencies of events to their probabilities, *Th. Prob. and Applics.* **16**(1971), 264-280.

On the local chromatic number of graphs

GÁBOR SIMONYI*

Alfréd Rényi Institute of Mathematics,
Hungarian Academy of Sciences,
1364 Budapest, POB 127, Hungary
simonyi@renyi.hu

GÁBOR TARDOS†

Alfréd Rényi Institute of Mathematics,
Hungarian Academy of Sciences,
1364 Budapest, POB 127, Hungary
tardos@renyi.hu

Abstract: We survey our recent results on the local chromatic number of graphs.

Keywords: chromatic number, topological method, quadrangulation

1 Introduction

The local chromatic number of graphs is a coloring type graph parameter that was introduced about 20 years ago by Erdős, Füredi, Hajnal, Komjáth, Rödl, and Seress [3]. Despite this long time there are very few papers about it so far, and we beleive that this is not because the concept would not deserve more attention, but rather for it somehow did not become known enough. This talk would like to be a modest attempt towards changing this situation. Some of our recent results are to be presented that can be found in detail in the papers [11, 12, 13].

The definition of the local chromatic number is as follows.

Definition 1 ([3]) The *local chromatic number* $\psi(G)$ of a graph G is

$$\psi(G) := \min_c \max_{v \in V(G)} |\{c(u) : u \in N(v)\}| + 1,$$

where $N(v) = \{u : uv \in E(G)\}$ and the minimum is taken over all proper colorings c of G.

In short, $\psi(G)$ is the fewest number of colors we can have in the most colorful closed neighborhood of a vertex in a proper coloring of the graph. It is obvious that the chromatic number $\chi(G)$ is an upper bound on $\psi(G)$. At first sight it is quite surprising, however, that $\psi(G) < \chi(G)$ is also possible, moreover, the gap between these two parameters can be arbitrarily large, cf. [3].

In [4] it was observed that the fractional chromatic number $\chi_f(G)$ can serve as a lower bound for $\psi(G)$, i.e., $\chi_f(G) \le \psi(G)$ always holds. (For the definition and basic properties of the fractional chromatic number we refer to [9].)

This motivates the study of the local chromatic number of such graphs where the fractional and ordinary chromatic numbers are far apart. Not very many different families of such graphs are known. Here we discuss the local chromatic number of some of the standard examples for this gap. These standard examples have the other common feature that the topological technique introduced by Lovász [5] to bound the chromatic number from below is relevant for them in the sense that the bound it gives is sharp for these graphs. It turns out that the same kind of topological information that results in a lower bound for the chromatic number can also be used to bound the local chromatic number from below, and this bound is also sharp in many cases.

We mention that the results in [11] have implications also for the circular chromatic number (cf. [16] for definitions) which we do not discuss here.

2 Local chromatic number of graphs with topologically bounded chromatic number

The main examples of graphs with a large gap between their fractional and ordinary chromatic number given in the book [9] are Kneser graphs and Mycielski graphs. More important for us are two variants of these families that clearly provide at least the same large gap between the two mentioned coloring parameters. The first of these variants is the family of Schrijver graphs $SG(n,k)$ discovered by Schrijver [10] as vertex color-critical induced subgraphs of Kneser graphs, see the defintion below. The second is the family of so-called generalized Mycielski graphs, see their definition, e.g., in [7] or [14].

*Research is partially supported by the Hungarian Foundation for Scientifi c Research Grant (OTKA) Nos. T037846, T046376, and AT048826.
†Research is partially supported by the Hungarian Foundation for Scientifi c Research Grant (OTKA) Nos. T037846, T046234, and AT048826.

Definition 2 ([10]) The *Schrijver graph* $SG(n,k)$ is defined as follows. Their vertices are those k-element subsets of the cyclically arranged set $[n]=\{1,\dots,n\}$ that do not contain consecutive elements $i,i+1$ or $n,1$. Two such vertices are adjacent if they represent disjoint k-subsets.

The chromatic number of $SG(n,k)$ is determined by Schrijver [10] to be $n-2k+2$ by generalizing the topological argument of Bárány [2] that provided a short proof for the earlier result of Lovász [5] determining the chromatic number of Kneser graphs.

For the local chromatic number of Schrijver graphs we have the following result.

Theorem 3 ([11]) If $t=n-2k+2>2$ is odd and $n\geq 4t^2-7t$ then

$$\psi(SG(n,k))=\left\lceil\frac{t}{2}\right\rceil+1.$$

This theorem easily implies that for even $t=n-2k+2>2$ and large enough n the value of $\psi(SG(n,k))$ is one of $t/2+1$ and $t/2+2$.

The following proposition shows that some lower bound on n is really needed in Theorem 3.

Proposition 4 ([11]) $\psi(SG(n,2))=n-2=\chi(SG(n,2))$ for every $n\geq 4$.

The lower bound part of Theorem 3, i.e., the inequality $\psi(SG(n,k))\geq\lceil\frac{t}{2}\rceil+1$ is proved using topological methods. The same argument applies to all graphs that satisfy a certain topological criterion which implies that the chromatic number of the graph is at least t. The upper bound part of Theorem 3 is given by a combinatorial construction that also can be formulated in a more general setting. As a result we can prove similar statements determining the local chromatic number of generalized Mycielski graphs and Borsuk graphs (for the definition of the latter see [6]) of certain parameters. We refer to [11] for further details, as well as, for some topological consequences.

3 4-chromatic graphs and surface quadrangulations

Theorem 3 leaves open the question whether (large enough) 4-chromatic Schrijver graphs have local chromatic number 3 or 4. In other words, Theorem 3 does not decide whether the smallest chromatic number t for which a t-chromatic Schrijver graph with smaller local than ordinary chromatic number exists is 4 or 5. In [12] we have shown that this smallest number is 5, thus the following holds.

Theorem 5 ([12])

$$\psi(SG(2k+2,k))=4.$$

This theorem is again true in a more general setting, namely, for all graphs G satisfying a somewhat stronger topological criterion (than the one mentioned, though not defined above) that implies $\chi(G)\geq 4$, we also have $\psi(G)\geq 4$. See [12] for the details where the different implications of the two kinds of topological criteria are also discussed.

It turns out that 4-chromatic Schrijver graphs are closely related to quadrangulations of the Klein bottle. The chromatic number of surface quadrangulations is a widely investigated topic, see [1, 8, 15], and the above mentioned connection suggests that analogs of Theorem 5 may be true for certain quadrangulations of non-orientable surfaces. Indeed, one can show that non-bipartite quadrangulations of the projective plane have local chromatic number 4, generalizing a celebrated result of Youngs [15] stating that such graphs are never 3-chromatic. In [13] we also prove that certain quadrangulations of the Klein bottle that are shown to be 4-chromatic in [1] and [8] have local chromatic number 4. Surprisingly, however, one can construct graphs that quadrangulate other non-orientable surfaces, have chromatic number 4, and local chromatic number only 3. For further details we refer the reader to [13].

References

[1] D. Archdeacon, J. Hutchinson, A. Nakamoto, S. Negami, K. Ota, Chromatic numbers of quadrangulations on closed surfaces, *J. Graph Theory*, **37** (2001), no. 2, 100–114.

[2] I. Bárány, A short proof of Kneser's conjecture *J. Combin. Theory Ser. A*, **25** (1978), no. 3, 325–326.

[3] P. Erdős, Z. Füredi, A. Hajnal, P. Komjáth, V. Rödl, Á. Seress, Coloring graphs with locally few colors, *Discrete Math.*, **59** (1986), 21–34.

[4] J. Körner, C. Pilotto, G. Simonyi, Local chromatic number and Sperner capacity, to appear in *J. Combin. Theory Ser B.*

[5] L. Lovász, Kneser's conjecture, chromatic number, and homotopy, *J. Combin. Theory Ser. A*, **25** (1978), no. 3, 319–324.

[6] L. Lovász, Self-dual polytopes and the chromatic number of distance graphs on the sphere, *Acta Sci. Math. (Szeged)*, **45** (1983), 317–323.

[7] J. Matoušek, *Using the Borsuk-Ulam Theorem, Lectures on Topological Methods in Combinatorics and Geometry*, Springer-Verlag, Berlin etc., 2003.

[8] B. Mohar, P. D. Seymour, Coloring locally bipartite graphs on surfaces, *J. Combin. Theory Ser. B*, **84** (2002), no. 2, 301–310.

[9] E. R. Scheinerman, D. H. Ullman, *Fractional Graph Theory*, Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley and Sons, Chichester, 1997.

[10] A. Schrijver, Vertex-critical subgraphs of Kneser graphs, *Nieuw Arch. Wisk. (3)*, **26** (1978), no. 3, 454–461.

[11] G. Simonyi, G. Tardos, Local chromatic number, Ky Fan's theorem, and circular colorings, submitted, also available at arXiv:math.CO/0407075.

[12] G. Simonyi, G. Tardos, Local chromatic number and distinguishing the strength of topological obstructions, submitted, also available at arXiv:math.CO/0502452.

[13] G. Simonyi, G. Tardos, On the local chromatic number of quadrangulations of surfaces, manuscript in preparation.

[14] C. Tardif, Fractional chromatic numbers of cones over graphs, *J. Graph Theory*, **38** (2001), 87–94.

[15] D. A. Youngs, 4-chromatic projective graphs, *J. Graph Theory* **21** (1996), 219–227.

[16] X. Zhu, Circular chromatic number: a survey, *Discrete Math.*, **229** (2001), no. 1–3, 371–410.

Some results on the degree prescribed factor problem

JÁCINT SZABÓ*

Dept. of Operations Research,
Eötvös University, Pázmány P. s. 1/C,
Budapest, Hungary H-1117.
The author is a member of the
Egerváry Research Group (EGRES).
e-mail: jacint@elte.hu

Abstract: The degree prescribed factor problem is to decide if a graph has a subgraph satisfying given degree prescriptions at each vertex. Lovász, and later Cornuéjols, gave structural descriptions on this problem in case the prescriptions have no two consecutive gaps. We state the Edmonds-Gallai type structure theorem of Cornuéjols which is only implicit in his paper. By extending a result of Loebl, we then prove that a degree prescription can be reduced to the edge and factor-critical graph packing problem by a 'gadget' if and only if all of its gaps have the same parity. This technique makes it possible to obtain a characterization of the critical components in case all prescriptions can be represented by a gadget. Finally, we show two matroidal results. First, we give three proofs that the up hull of the distance vectors of all subgraphs forms a contra-polymatroid. Second, we prove that the vertex sets coverable by subgraphs F satisfying the degree prescriptions for all $v \in V(F)$ form a matroid, in case 1 is contained in all prescriptions.

Keywords: Edmonds-Gallai decomposition, matroid

1 Introduction

The *$\mathscr{H}$-factor problem* is the following. Let G be an undirected graph and let $\emptyset \neq H_v \subseteq \mathbb{N}$ be a degree prescription for each $v \in V(G)$. For a subgraph F of G define $\delta^F(v) = \text{dist}(\deg_F(v), H_v)$ where $\text{dist}(I,J) = \min\{|i-j| : i \in I,\ j \in J\}$ for $I, J \subseteq \mathbb{N}$. Let $\delta_F = \sum\{\delta^F(v) : v \in V(G)\}$. The minimum δ_F among the subgraphs F is denoted by $\delta_{\mathscr{H}}(G)$. A subgraph F is called *$\mathscr{H}$-optimal* if $\delta_F = \delta_{\mathscr{H}}(G)$, and it is an *$\mathscr{H}$-factor* if $\delta_F = 0$, i.e. if $\deg_F(v) \in H_v$ for all $v \in V(G)$. The *$\mathscr{H}$-factor problem* is to determine the value of $\delta_{\mathscr{H}}(G)$. An integer h is called a *gap* of $H \subseteq \mathbb{N}$ if $h \notin H$ but H contains an element less than h and an element greater than h. Lovász [13] gave a structural description on the $\mathscr{H}$-factor problem in case H_v has no two consecutive gaps for all $v \in V(G)$. He showed that the problem is NP-complete without this restriction. However, the issue of polynomiality remained open. Later, Cornuéjols [3] gave two polynomial algorithms in the case when no H_v contains two consecutive gaps. One of them is an Edmonds type alternating forest algorithm, which implies an Edmonds-Gallai type structure theorem for the $\mathscr{H}$-factor problem. The existence of such a structure theorem is mentioned in Cornuéjols [3] but it is not stated explicitly. We state this result of Cornuéjols, then briefly point out the connection of the decompositions formulated by Lovász and Cornuéjols.

For $v \in V(G)$ we denote $l(v) = \min H_v$, $u(v) = \max H_v$ and $H_v^{\downarrow} = \{0, 1, \ldots, u(v)\}$. Wlog. we assume that $0 \leq l(v) \leq u(v) \leq deg_G(v)$ for all $v \in V(G)$.

Definition 1 [3] A graph G with $|V(G)| \geq 2$ is called *$\mathscr{H}$-critical* if G does not have an $\mathscr{H}$-factor but for all $v \in V(G)$ there exists a subgraph F of G with the property that $\deg_F(v) + 1 \in H_v$ and $\deg_F(w) \in H_w$ for all vertices $w \neq v$. Call G *non-trivial* and define $def(G) = 1$. Moreover, G is *H-critical* if $V(G) = \{v\}$ and $l(v) \geq 1$. In this case G is said to be *trivial* and let $def(G) = l(v)$. $def(G)$ is called the *deficiency* of G.

Definition 2 [3] G_{sub} is the graph what we get from G after subdividing each edge $e = xy$ with two new vertices e_x and e_y (resulting in three new edges xe_x, e_xe_y and e_yy). Let the set of these new vertices be V_E and let the degree prescription on the new vertices be $H_{e_x} = H_{e_y} = \{1\}$.

Observation 3 Let $v \in V(G)$. For any subgraph F of G_{sub}, G has a subgraph F' such that $\delta_{F'} \leq \delta_F$ and $\deg_F(v) = \deg_{F'}(v)$. Besides, trivially, if F' is a subgraph of G, then G_{sub} has a subgraph F such that $\delta_{F'} = \delta_F$ and $\deg_F(v) = \deg_{F'}(v)$. Hence $\delta_{\mathscr{H}}(G) = \delta_{\mathscr{H}}(G_{sub})$.

*Research is supported by OTKA grants T 037547, TS 049788, by European MCRTN Adonet, Contract Grant No. 504438 and by the Egerv´ary Research Group of the Hungarian Academy of Sciences.

A splendid idea of Cornuéjols is that his Edmonds type algorithm should work on the subdivided graph G_{sub}. Thus the Edmonds-Gallai type theorem, implicit in [3], considers the $\mathscr{H}$-factor problem of G_{sub}. We need to state this result for a slightly more general class of graphs.

Definition 4 A simple graph S is called *subdivided* if it is an induced subgraph of a graph of form G_{sub}. Let $S_V = V(S) \cap V(G)$ and $S_E = V(S) \cap V_E$.

Theorem 5 (Cornuéjols) Let S be a subdivided graph. Let $D \subseteq V(S)$ consist of vertices v with the property that there exists an $\mathscr{H}$-optimal subgraph F of S such that $\deg_F(v) \in H_v^{\downarrow} \setminus H_v$. Let A be the set of neighbors of D in S and let $C = V(S) - (D \cup A)$. Let $def(D)$ denote the sum of the deficiencies of the H-critical components of $S[D]$. Then

1. the components of $S[D]$ are $\mathscr{H}$-critical,
2. $\delta_{\mathscr{H}}(S) = def(D) - u(A)$,
3. $\sum\{def(K) : K \text{ is a component of } S[D] \text{ adjacent to } A'\} \geq u(A') + 1$ for all $\emptyset \neq A' \subseteq A$,
4. for all $\mathscr{H}$-optimal subgraphs F of S there is no edge of F between A and C and $F[C]$ is an $\mathscr{H}$-factor of $S[C]$.

For sake of completeness, we briefly describe the relation of Thm. 5 to the decomposition formulated by Lovász, see [13]. This is defined on the original graph G. It consists of four vertex sets.

$$C_L = \{v \in V(G) : \deg_F(v) \in H_v \text{ for all } \mathscr{H}\text{-optimal subgraphs } F \text{ of } G\},$$

$$A_L = \{v \in V(G) \setminus C_L : \deg_F(v) \geq u(v) \text{ for all } \mathscr{H}\text{-optimal subgraphs } F \text{ of } G\},$$

$$B_L = \{v \in V(G) \setminus C_L : \deg_F(v) \leq l(v) \text{ for all } \mathscr{H}\text{-optimal subgraphs } F \text{ of } G\},$$

and finally

$$D_L = V(G) - (A_L \cup B_L \cup C_L).$$

We omit the structure theorem of Lovász [13], it follows from Thm. 5 using the considerations below. By choosing $S = G_{sub}$, let $V(G_{sub}) = D \dot\cup A \dot\cup C$ be the partition by Thm. 5. First observe that Thm. 5 easily implies another characterization of this decomposition.

$$C = \{v \in V(G_{sub}) : \deg_F(v) \in H_v \text{ for all } \mathscr{H}\text{-optimal subgraphs } F \text{ of } G_{sub}\},$$

$$A = \{v \in V(G_{sub}) \setminus C : \deg_F(v) \geq u(v) \text{ for all } \mathscr{H}\text{-optimal subgraphs } F \text{ of } G_{sub}\}, \text{ and}$$

$$D = V(G_{sub}) - (C \cup A).$$

Observation 3 yields that $A_L = V(G) \cap A$ and $C_L = V(G) \cap C$. Hence, by Thm. 5 *1.* and *3.* it is not hard to see that

$$B_L = \{v \in V(G) : \{v\} \text{ is a trivial component of } G_{sub}[D]\} \text{ and}$$

$$D_L = \{v \in V(G) : v \text{ is contained in a non-trivial component of } G_{sub}[D]\}.$$

Let K' be a component of $G[D_L]$. If C' is a circuit of K' then $A' := A \cap \{e_x, e_y : e = xy \in E(C')\} = \emptyset$ by Thm. 5 *3.* On the other hand, assume that e_x, e_y belong to a (non-trivial) component K of $G_{sub}[D]$ for a cut edge $e = xy$ of K'. Then $e_x e_y$ is also a cut edge of K hence the $\mathscr{H}$-criticality of K gives rise to an $\mathscr{H}$-factor of K, which is impossible. So the maximal 2-edge connected subgraphs of the components of $G[D_L]$ correspond to the non-trivial components of $G_{sub}[D]$. It is interesting that with the help of the subdivision of the edges of G, Cornuéjols was able to encode the two sets B_L and D_L in one set D.

Thm. 5 immediately implies a Berge type minimax formula for the $\mathscr{H}$-factor problem in subdivided graphs S. Note that $\geq$ is trivial.

Theorem 6 (Cornuéjols) [3]

$$\delta_{\mathscr{H}}(S) = \max_{A \subseteq V(S)} def(S - A) - u(A).$$

Here $def(S - A)$ denotes the sum of the deficiencies of the $\mathscr{H}$-critical components of $S - A$.

Observe that the $\mathscr{H}$-critical components in this theorem are subdivided even when $S = G_{sub}$. We will give a characterization on subdivided $\mathscr{H}$-critical graphs in terms of perfect matchings in the next section, however, only in case all gaps of H_v have the same parity for each $v \in V(G)$.

2 Main results

As an application of Thm. 5 we deduce an Edmonds-Gallai type structure theorem for the $(1,f)$-*odd factor problem.* This was introduced by Amahashi [1] who gave a Tutte type characterization for those graphs which have an $\mathscr{H}$-factor in case $H_v = \{1,3,5,\dots,2k+1\}$ for some $k \in \mathbb{N}$ for all $v \in V(G)$. Let $f : V(G) \to \mathbb{N}$ be a function with odd values. For the case when $H_v = \{1,3,5,\dots,f(v)\}$, a Tutte type theorem was proved by Cui and Kano [5], and a Berge type minimax formula by Kano and Katona [8]. These results are generalized by the following theorem, which was also obtained by Kano and Katona [9] by a direct inductive proof.

Theorem 7 Let $f : V(G) \to \mathbb{N}$ be a function with odd values and let $H_v = \{1,3,5,\dots,f(v)\}$ for all $v \in V(G)$. Let $D_f \subseteq V(G)$ consist of those vertices v for which there exists an $\mathscr{H}$-optimal subgraph F of G with $\deg_F(v) \in \{0,2,4,\dots,f(v)-1\}$. Let A_f be the set of neighbors of D_f in G and let $C_f = V(G) - (D_f \cup A_f)$. Then

1. the components of $G[D_f]$ have odd size,
2. $\delta_{\mathscr{H}}(G) = c(D_f) - f(A_f)$,
3. $|\{K : K \text{ is a component of } G[D_f] \text{ adjacent to } A'\}| \geq f(A')+1$ for all $\emptyset \neq A' \subseteq A_f$,
4. for all $\mathscr{H}$-optimal subgraphs F of G there is no edge of F between A_f and C_f and $F[C_f]$ is an $\mathscr{H}$-factor of $G[C_f]$.

PROOF: Let $V(G_{sub}) = D \dot\cup A \dot\cup C$ be the Edmonds-Gallai decomposition of G_{sub} by Thm. 5. Observation 3 yields that $D_f = D \cap V(G)$. By parity reasons, the $\mathscr{H}$-critical components of $G_{sub}[D]$ have odd size. Denote $D' = D \cup (V_E \cap A)$. Thm. 5 *3.* implies that each component of $G_{sub}[D']$ contains k vertices of $V_E \cap A$ and $k+1$ components of $G_{sub}[D]$ for some $k \in \mathbb{N}$. Hence these components have odd size, too. It is clear that the components of $G[D_f]$ are obtained from the components of $G_{sub}[D']$ by the following operation: if a vertex of type $e_x \in D'$ then delete e_x if $x \in A$ and contract the edge xe_x if $x \in D$. This proves *1.* Properties *2., 3.* and *4.* follow from the corresponding properties of Thm. 5. □

Observe that if $f \equiv 1$ then the components of $G[D_f]$ are factor-critical by the classical Edmonds-Gallai theorem. However, for general f, these components are only of odd size.

In the rest of this section we characterize those degree prescriptions which can be represented by *gadgets*, which are auxiliary graphs reducing the behavior of a prescription to the edge and factor-critical graph packing problem of Cornuéjols, Hartvigsen and Pulleyblank [4].

Definition 8 Let G be an undirected graph and let $\mathscr{F}$ consist of factor-critical subgraphs of G. A subgraph Q of G is called an $\mathscr{F}$*-packing* if each connected component of Q is either isomorphic to K_2 or is contained in $\mathscr{F}$. Q is *maximum* if it covers a maximum number of vertices and Q is an $\mathscr{F}$*-factor* if it covers all vertices of G. $d_{\mathscr{F}}(G)$ denotes the number of vertices of G missed by a maximum $\mathscr{F}$-packing. G is $\mathscr{F}$*-critical* if it has no $\mathscr{F}$-factor, but $G-v$ has one for each $v \in V(G)$.

Cornuéjols, Hartvigsen and Pulleyblank [4] showed that the $\mathscr{F}$-packing problem is polynomial if the cardinality of $\mathscr{F}$ is polynomial. They also characterized $\mathscr{F}$-critical graphs.

Lemma 9 [4] A graph G is $\mathscr{F}$-critical if and only if it is factor-critical and does not have a subgraph $K \in \mathscr{F}$ such that $G-K$ has a perfect matching.

We make use of the edge and factor-critical graph packing problem in gadgets as follows. See some gadgets in Figs. 1–2.

Definition 10 $(T,U,\mathscr{F})$ is said to be a *gadget representing* the degree prescription $H \subseteq \mathbb{N}$ if T is a graph, $U \subseteq V(T)$ and $\mathscr{F}$ is a set of factor-critical subgraphs of T with the property that $h \in H$ if and only if there exists an h-element set $U' \subseteq U$ such that $T-U'$ has an $\mathscr{F}$-factor.

Definition 11 Let S be a subdivided graph. Suppose $(T_v,U_v,\mathscr{F}_v)$ represents H_v for $v \in V_S$. Let S_{aux} be the graph with vertex set

$$V(S_{aux}) = S_E \cup \bigcup_{v \in S_V} V(T_v)$$

and edge set

$$E(S_{aux}) = \{e_x e_y : e_x, e_y \in S_E\} \cup \{e_x u : e_x \in S_E, x \in S_V, u \in U_x\} \cup \bigcup_{v \in S_V} E(T_v).$$

Moreover, let $\mathscr{F} = \bigcup_{v \in S_V} \mathscr{F}_v$. $(G_{sub})_{aux}$ is denoted simply by G_{aux}.

Clearly, if H_v has a representing gadget for all $v \in V(G)$ then G has an $\mathscr{H}$-factor if and only if G_{aux} has an $\mathscr{F}$-factor. It can also be proved that $\delta_{\mathscr{H}}(G) = \delta_{\mathscr{H}}(G_{sub}) = d_{\mathscr{F}}(G_{aux})$ holds. Moreover, any maximum $\mathscr{F}$-packing of G_{aux} can be transformed to an $\mathscr{H}$-optimal subgraph of G.

It is well known that a parity interval $\{p, p+2, \ldots, p+2r\}$ can be represented by a gadget with $\mathscr{F} = \emptyset$, see Fig. 2. With the help of these gadgets Cornuéjols [3] gave a non-Edmonds type algorithm for the $\mathscr{H}$-factor problem using the local augmenting property of jump systems. By answering a question of Pulleyblank, Loebl [11] then proved that a prescription H can be represented by a gadget $(T, U, \mathscr{F})$ such that $\mathscr{F}$ *contains only triangles* if and only if H is a parity interval or

$$H = I \cap \{p, p+2, p+3, \ldots, p+2r-2, p+2r\},\ r \geq 1$$

for some interval I. Loebl and Poljak [12] mentioned that more representable prescriptions may exist using reductions to more general graph packing problems. Indeed, Thm. 12 shows that many new representable prescriptions arise if we allow any factor-critical graphs in $\mathscr{F}$ not just triangles.

Theorem 12 A degree prescription can be represented by a gadget if and only if all of its gaps have the same parity.

PROOF: *Necessity.* Suppose $(T, U, \mathscr{F})$ is a gadget representing the degree prescription H. Let $p, q \in H,\ p < q$. We prove that

i. $\{p+1, p+2, q-1\} \cap H \neq \emptyset$ and

ii. $\{p+1, q-1\} \cap H \neq \emptyset$ if $p \not\equiv q \mod 2$.

Now *i.* implies that there are no two consecutive gaps in H, and hence *ii.* gives that all gaps have the same parity. Let Q_p, Q_q be $\mathscr{F}$-factors of $T - U_p$, $T - U_q$ resp., with $U_p, U_q \subseteq U$ and $|U_p| = p$, $|U_q| = q$. Let $V_p = V(Q_p) = V(T) - U_p$ and define V_q similarly. Choose Q_p, Q_q with $V_p \cap V_q$ maximal. $|V_p| > |V_q|$ so let $v \in V_p \setminus V_q$. Let P be a longest alternating path starting at v with edges alternately being K_2 components of Q_p and Q_q. Note that P cannot end in $V_q \setminus V_p$ because of the maximality of $V_p \cap V_q$. So three possibilities can occur.

1. If P ends in a factor-critical component of Q_p then we can modify Q_p to an $\mathscr{F}$-factor of $T - U_p - v$.
2. If P ends in a factor-critical component of Q_q then we can modify Q_q to an $\mathscr{F}$-factor of $T - U_q + v$.
3. If P ends in $u \in V_p \setminus V_q$ then we can modify Q_p to an $\mathscr{F}$-factor of $T - U_p - u - v$.

Hence *i.* is proved. Also *ii.* is proved if there exists $v \in V_p \setminus V_q$ for which possibility 1. or 2. occurs. Suppose otherwise. The paths of type 3. pair the elements of $V_p \setminus V_q$ implying that $|V_p \setminus V_q|$ is even and is clearly at least 2. Let P be such an alternating path with end vertices $u, v \in V_p \setminus V_q$ and let P_p (resp. P_q) consist of the K_2 components of P belonging to Q_p (resp. Q_q). The oddness of $q - p$ implies that $|V_q \setminus V_p|$ is odd. For each $w \in V_q \setminus V_p$ let R_w be a longest alternating path starting at w with edges alternately being K_2 components of Q_q and Q_p. Observe that R_w and P are disjoint. As above, R_w either ends in a factor-critical component or it ends in $V_q \setminus V_p$. Since $|V_q \setminus V_p|$ is odd, for at least one vertex $w \in V_q \setminus V_p$, either

1. R_w ends in a factor-critical component of Q_p in which case we can modify $Q_p - P_p + P_q$ to an $\mathscr{F}$-factor of $T - U_p + w - u - v$, or
2. R_w ends in a factor-critical component of Q_q in which case we can modify $Q_q - P_q + P_p$ to an $\mathscr{F}$-factor of $T - U_q - w + u + v$.

This completes the proof of necessity.

Definition 13 Let $l = \min H$, $u = \max H$ and assume that all gaps of H have the same parity and that H is not an interval of length at least 2. Define H-*parity* to be 0 (resp. 1) if all even (resp. odd) integers in $[l, u]$ belong to H.

Sufficiency. Let H be a prescription with no gaps of different parity. If H is an interval $\{p, p+1, \ldots, p+r\}$ then it is well known that H can be represented by a gadget T consisting of $p+r$ isolated vertices, with $U = V(T)$ and $\mathscr{F}$ consisting of r of these vertices as one vertex factor-critical subgraphs, see Fig. 2. Otherwise let $H^0 = \{h - l :\ h \in H\}$. If $(T, U, \mathscr{F})$ is a gadget representing H^0 then adding l isolated vertices to T which belong to U results in a gadget representing H. Hence we may assume that $l = 0$. Construct a gadget $(T, U, \mathscr{F})$ in the following way. If u has H-parity then define $n = 2u$, otherwise let $n = 2u+1$. Let $U = \{y_i :\ 1 \leq i \leq u\}$, $V(T) = U \cup \{x_1, x_2, \ldots, x_n = x_0\}$ and let

$$E(T) = \{x_{i-1}x_i :\ 1 \leq i \leq n\} \cup \{x_{2i-2}x_{2i} :\ 1 \leq i \leq n/2\} \cup \{x_{2i-1}y_i :\ 1 \leq i \leq u\}.$$

If $u - r \in H$ has non H-parity then add the odd circuit $x_0x_2 \ldots x_{2r}x_{2r+1} \ldots x_{n-1}$ to $\mathscr{F}$. Observe that an $\mathscr{F}$-packing of T can have at most one factor-critical component since $x_0 \in V(F)$ for all $F \in \mathscr{F}$. So it is easy to see that $(T, U, \mathscr{F})$ represents H. See an example in Fig. 1. □

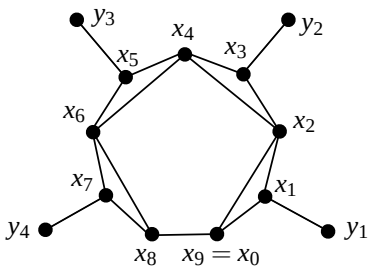


Figure 1: A gadget representing $\{0,1,3,4\}$. $U = \{y_1, y_2, y_3, y_4\}$, $\mathscr{F} = \{x_0x_2x_4x_6x_8,\ x_0 \dots x_8\}$.

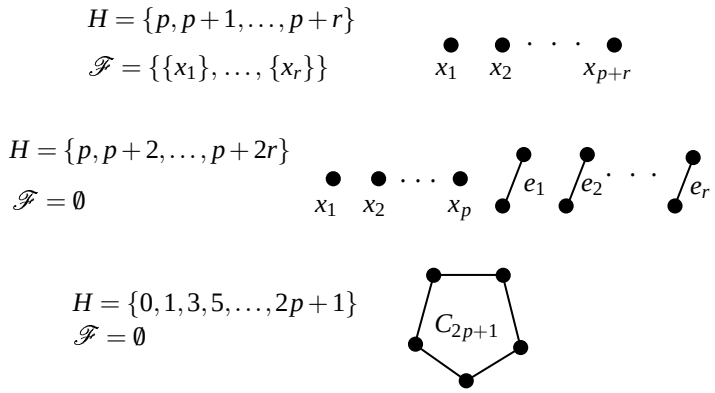


Figure 2: Some simpler gadgets. $U = V(T)$ in all cases.

Of course in some special cases one can give much simpler gadgets, see Fig. 2. All gadgets of Fig. 2 were already known. Note that the addition of l isolated vertices to U shifts H upwards l units.

Checking $\mathscr{H}$-criticality in Theorem 6 is almost as difficult as the original $\mathscr{H}$-factor problem. Actually, we can check $\mathscr{H}$-criticality by any algorithm of Cornuéjols [3]. However, beside an algorithmic proof, we want a nice description of the subdivided $\mathscr{H}$-critical components. E.g. it can be proved that if S is $\mathscr{H}$-critical then S_{aux} is factor-critical and hence $|V(S_{aux})|$ is odd. It also holds that the sum of the H_v-parities is odd in every $\mathscr{H}$-critical graph in the representable cases. Lemma 9 of $\mathscr{F}$-critical graphs can be translated to $\mathscr{H}$-critical graphs as follows.

Theorem 14 Let S be a subdivided graph. Assume that all gaps of H_v have the same parity for all $v \in S_V$ and that $|V(S)| \geq 2$. Then S is $\mathscr{H}$-critical if and only if

1. H_v is not an interval of length at least 2 for all $v \in S_V$,
2. for each $w \in V(S)$, S has a subgraph F such that $\deg_F(w) \notin H_w$ has non H_w-parity, $\deg_F(w)+1 \in H_w$ and $\deg_F(v) \in H_v$ has H_v-parity for all $v \neq w$, and
3. for each $w \in S_V$ and $d \in H_w$ with non H_w-parity it holds that S has no $\mathscr{H}$-factor F such that $\deg_F(w) = d$ and $\deg_F(v)$ has H_v-parity for all $v \neq w$.

Recall that parity intervals can be reduced to the matching problem. Hence it is possible to check $\mathscr{H}$-criticality by solving at most $(u(V) - l(V) + 2|V|)/2$ matching problems (with $V = V(S)$), in case all prescriptions can be represented. Thm. 14 justifies that, in some way, the representable prescriptions form the broadest class of the degree prescribed factor problem where parity plays a role.

Using the description of $\mathscr{H}$-critical graphs, this gadget technique also offers the theoretical possibility of deriving exact Tutte, Berge and Edmonds-Gallai type theorems in the representable cases. Such a reduction can easily imply structure theorems like Thm. 7.

3 Matroidal results

This section contains two matroidal results on the $\mathscr{H}$-factor problem, Thm. 16 and Thm. 23. We give three proofs for the first one. Usually base polyhedrons and contra-polymatroids are integer polyhedra. We are interested only in the integer points of these polyhedra so we define them in the following way.

Definition 15 For $v \in V(G)$ let $e^v \in \mathbb{N}^{V(G)}$ be the unit vector of coordinate v. For a subgraph F of G let $\delta^F \in \mathbb{N}^{V(G)}$ be the vector with component $\delta^F(v)$ for $v \in V(G)$.

$P \subseteq \mathbb{Z}^V$ is a *base polyhedron* if for all $a, b \in P$ and $v \in V$ with $a(v) > b(v)$ there exists $u \in V$ such that $a(u) < b(u)$ and $a - e^v + e^u \in P$. The up hull of a base polyhedron (i.e. $P + \mathbb{N}^{V(G)}$) is called a *contra-polymatroid*.

Theorem 16 $C = \{\delta^F + \mathbb{N}^{V(G)} : F \text{ is a subgraph of } G\}$ is a contra-polymatroid.

PROOF: *1.* It is enough to prove that if $a, b \in C$ and $v \in V(G)$ with $a(v) > b(v)$ then either $a - e^v \in C$ or there exists $u \in V(G)$ such that $a(u) < b(u)$ and $a - e^v + e^u \in C$. We prove this by induction on $|E(F_a) \triangle E(F_b)|$ where F_a, F_b are subgraphs such that $\delta^{F_a} \leq a,\ \delta^{F_b} \leq b$. If $\delta^{F_a}(v) < a(v)$ then we are done, so suppose equality. Thus $\delta^{F_a}(v) > \delta^{F_b}(v)$ so there exists an edge $e = vu \in E(F_a) \triangle E(F_b)$ such that $\delta^{F'}(v) < \delta^{F_a}(v)$ holds with the notation $F' = F_a \triangle e$. If $\delta^{F'}(u) \leq a(u)$ or $\delta^{F'}(u) = a(u) + 1 \leq b(u)$ then F' shows that we are done. Otherwise $\delta^{F'}(u) > b(u)$. Now $|E(F') \triangle E(F_b)| < |E(F_a) \triangle E(F_b)|$ so the statement holds for $\delta^{F'}$ and b by our induction hypothesis. Apply it to $u \in V(G)$. □

In the next two proofs it is enough to show that $P_{\mathscr{H}}(G) := \{\delta^F : F \text{ is an } \mathscr{H}\text{-optimal subgraph of } G\}$ is a base polyhedron by the next lemma.

Lemma 17 For any subgraph F of G there exists an $\mathscr{H}$-optimal subgraph F_0 of G such that $\delta^{F_0} \leq \delta^F$.

PROOF: The $\mathscr{H}$-optimal subgraph F_0 minimizing $|E(F) \triangle E(F_0)|$ will do. Otherwise $\delta^{F_0}(v) > \delta^F(v)$ for some $v \in V(G)$ so there exists an edge $e = vu \in E(F) \triangle E(F_0)$ such that $\delta^{F_0 \triangle e}(v) < \delta^{F_0}(v)$ holds. But then $F_0 \triangle e$ is $\mathscr{H}$-optimal again, contradicting to the choice of F_0. □

We need some preliminaries for the next two proofs. Jump systems were introduced by Bouchet and Cunningham [2].

Definition 18 [2] For $a, b \in \mathbb{Z}^V$ we say that a' is a *step from a to b* if either $a' = a + e^v$ and $a(v) < b(v)$ or $a' = a - e^v$ and $a(v) > b(v)$, for some $v \in V$. $J \subseteq \mathbb{Z}^V$ is a *jump system* if for all $a, b \in J$ and a' step from a to b, either $a' \in J$ or some step from a' to b is contained in J.

If $J_i \subseteq \mathbb{Z}^{V_i}$ are jump systems for $i = 1, 2$ then let $J_1 \wedge J_2 = \{a^1 \wedge a^2 \in \mathbb{Z}^{V_1 \triangle V_2} : a^1 \in J_1,\ a^2 \in J_2,\ a^1|_{V_1 \cap V_2} = a^2|_{V_1 \cap V_2}\}$ where $(a^1 \wedge a^2)_j = a^i_j$ if $j \in V_i$ for $i = 1, 2$. If V_1 and V_2 are disjoint then $J_1 \times J_2 = J_1 \wedge J_2$ is called the *direct sum* of J_1, J_2. For $J \subseteq \mathbb{Z}^V$ and $c \in \{-1, 0, 1\}^V$ let J_c consist of the elements of J minimizing cost function c. J has *constant sum* if $a(V) = b(V)$ for all $a, b \in J$.

Proposition 19 [2] If $J \subseteq \mathbb{Z}^V$, $J_1 \subseteq \mathbb{Z}^{V_1}$, $J_2 \subseteq \mathbb{Z}^{V_2}$ are jump systems and $c \in \{-1, 0, 1\}^V$ then $J_1 \wedge J_2$ and J_c are jump systems. A constant sum jump system is a base polyhedron. The degree sequences of all subgraphs of a graph G is a jump system, denoted by J_G.

This proposition will be used throughout in the next two proofs.

PROOF: *2.* (of Thm. 16) Note that it is enough to prove that $P_{\mathscr{H}}(G_{sub})$ is a base polyhedron since $P_{\mathscr{H}}(G) = \{a|_{V(G)} : a \in P_{\mathscr{H}}(G_{sub}), a(v) = 0 \text{ for all } v \in V_E\}$. Let $V(G_{sub}) = D \dot\cup A \dot\cup C$ be the decomposition by Thm. 5. Shrink all non-trivial components of $G_{sub}[D]$, delete the edges induced by A and delete C resulting in the bipartite graph B. For $a \in \mathbb{N}^{V(B)}$ let $a'(v) = a(v) - u(v)$ if $v \in A$, $a'(v) = l(v) - a(v)$ if $\{v\}$ is a trivial component of $G_{sub}[D]$ and $a'(v) = a(v)$ otherwise. Let $J' = \{a' \geq 0 : a \in J_B\}$ which is clearly a jump system. For a non-trivial component K of $G_{sub}[D]$ define a jump system $J_K = \{e^K, e^v : v \in V(K)\}$ on ground set $\{K\} \cup V(K)$. Let $J_D = \times\{J_K : K \text{ is a non-trivial component of } G_{sub}[D]\}$, J_D is a jump system again. Using Thm. 5 it is not hard to see that $J' \wedge J_D = \{\delta^F|_{D \cup A} : F \text{ is an } \mathscr{H}\text{-optimal subgraph of } G_{sub}\}$. Finally, $\delta^F(v) = 0$ for all $v \in C$ and $\mathscr{H}$-optimal subgraphs F. Thus $P_{\mathscr{H}}(G_{sub})$ is a (constant sum) jump system and hence a base polyhedron. □

PROOF: *3.* (of Thm. 16) We use some results of Lovász [13]. For the definitions of A_L, B_L, C_L and D_L, see page 325.

Definition 20 For $v \in V(G)$ let $I_{\mathscr{H}}(v) = \{\deg_F(v) : F \text{ is an } \mathscr{H}\text{-optimal subgraph of } G\}$. $[I_{\mathscr{H}}(v)]$ denotes the minimal interval containing $I_{\mathscr{H}}(v)$.

Lemma 21 [13] If $v \in D_L$ then H_v contains either exactly the odd or exactly the even numbers of $[I_{\mathscr{H}}(v)]$.

For $v \in C_L$ define $J_v = \{(i, 0) : i \in H_v\}$, for $v \in A_L$ let $J_v = \{(i, i - u(v)) : i \geq u(v)\}$, and for $v \in B_L$ let $J_v = \{(i, l(v) - i) : i \leq l(v)\}$. Finally, for $v \in D_L$ define $J_v = \{(i, 0) : i \in [I_{\mathscr{H}}(v)] \cap H_v\} \cup \{(i, 1) : i \in [I_{\mathscr{H}}(v)] \setminus H_v\}$. Observe that J_v is a jump system for all $v \in V(G)$. Let $J' = \times\{J_v : v \in V(G)\}$ and $J = J' \wedge J_G$. It is clear that if $c \in \mathbb{N}^{V(G)}$ is the constant 1 vector then $J_c = \{\delta^F : F \text{ is an } \mathscr{H}\text{-optimal subgraph of } G\}$. J_c has constant sum and thus a base polyhedron. □

We remark that Thm. 16 holds for all jump systems with ground set V, namely, if $J \subseteq \mathbb{N}^V$ is a jump system and $H_v \subseteq \mathbb{N}$ for all $v \in V$ then $\{\delta^a + \mathbb{N}^V : a \in J\}$ is a contra-polymatroid, where $\delta^a(v) = \text{dist}(a(v), H_v)$. Indeed, the first proof of Thm. 16 directly generalizes to this case.

Definition 22 A subgraph F of G is called an $\mathscr{H}$*-subgraph* if $\deg_F(v) \in H_v$ for all $v \in V(F)$. Let $\mathscr{M}$ consist of those subsets of vertices of G which can be covered by $\mathscr{H}$-subgraphs.

Theorem 23 Suppose $1 \in H_v$ and H_v has no two consecutive gaps for all $v \in V(G)$. Let $\mathscr{M}$ consist of those vertex sets of G which can be covered by $\mathscr{H}$-subgraphs. Then $\mathscr{M}$ is a matroid.

PROOF: Modify proof *2.* of Thm. 16 in the following way. J' should be replaced by $\{a \in J' : a_v = 0 \ \forall v \in A\}$ and J_K by $\{e^K, e^v : K - v \text{ has an } \mathscr{H}\text{-factor}\}$. Thus the dual of $\mathscr{M}$ is a matroid. □

The already known special cases of Thm. 23 is the matching case by Edmonds and Fulkerson [6] (let $H \equiv \{1\}$), the packing by a sequential set of stars by Las Vergnas [10] ($H_v = \{1,2,\dots,u(v)\}$) and the $(1,f)$-odd subgraph case proved by Kano and Katona [8, 9] ($H_v = \{1,3,5,\dots,f(v)\}$).

It is easy to see that Thm. 23 is also true if $\{0,1\} \cap H_v \neq \emptyset$ for all $v \in V(G)$. Otherwise $\mathscr{M}$ is not necessarily a matroid: subdivide each edge of K_4 with one vertex and let the prescription be $\{2\}$ on all vertices.

An application of Thm. 23 is that the 'superstar packing problem' is matroidal, see [7].

The author would like to thank András Sebő for helpful discussions.

References

[1] A. AMAHASHI On factors with all degrees odd. *Graphs and Combin.* (1985) **1** 111–114.

[2] A. BOUCHET, W. H. CUNNINGHAM Delta-matroids, jump systems, and bisubmodular polyhedra. *SIAM J. Discrete Math.* (1995) **8** 17–32.

[3] G. CORNUÉJOLS General factors of graphs. *J. Combin. Theory Ser. B* (1988) **45** 185–198.

[4] G. CORNUÉJOLS, D. HARTVIGSEN, W. PULLEYBLANK Packing subgraphs in a graph. *Oper. Res. Letter* (1981/82) **1** 139–143.

[5] Y. CUI, M. KANO Some results on odd factors of graphs. *J. of Graph Theory* (1988) **12** 327–333.

[6] J. EDMONDS, D. R. FULKERSON Transversals and matroid partition. *J. Res. Nat. Bur. Standards Sect. B* (1965) 69B 147–153.

[7] M. JANATA, J. SZABÓ Generalized star packing problems. *EGRES Technical Reports* 2004-17.

[8] M. KANO, G. Y. KATONA Odd subgraphs and matchings. *Discrete Math.* (2002) **250** 265–272.

[9] M. KANO, G. Y. KATONA Structure theorem and algorithm on $(1,f)$-odd subgraphs. manuscript

[10] M. LAS VERGNAS An extension of Tutte's 1-factor theorem. *Discrete Math.* (1978) **23** 241–255.

[11] M. LOEBL Gadget classification. *Graphs Combin.* (1993) **9** 57–62.

[12] M. LOEBL, S. POLJAK Subgraph packing – a survey. *Topics in combinatorics and graph theory (Oberwolfach, 1990)* 491–503.

[13] L. LOVÁSZ The factorization of graphs. II. *Acta Math. Acad. Sci. Hungar.* (1972) **23** 223–246.

DNA-words and word posets

PÉTER LIGETI *

Department of Computer Science
Eötvös Loránd University
Budapest, Pázmány Péter sétány I/B. H-1117
Hungary
turul@cs.elte.hu

PÉTER SZIKLAI**

Department of Computer Science
Eötvös Loránd University
Budapest, Pázmány Péter sétány I/B. H-1117
Hungary
sziklai@cs.elte.hu

Abstract: In this paper two variants of a combinatorial problem for the set F_q^n of sequences of length n over the alphabet $F_q = \{0,1,..,q-1\}$ are considered, with some applications. The original problem was the following: what is the smallest k such that every word $w \in F_q^n$ is uniquely determined by the set of its subwords of length up to k. This problem was solved by Lothaire [1]. We consider the following variant of this problem: the n-letter word $f = f_1...f_n$ (which is called a *DNA-word*) is composed over an alphabet consisting of q *complement pairs*:$\{i,\bar{i} : i = 0,..,q-1\}$; and denote by f^* its *reverse complement*, i.e. $f^* = \bar{f_n}...\bar{f_1}$. A DNA-word g is called a subword of f if it is a subword of either f or f^*. As above, we are looking for the smallest k for which every DNA-word w of length n is uniquely determined by the set of its subwords of length up to k. We give a simple proof for $k \leq n-1$, and apply this result for determining the automorphism group of the poset of DNA-words of length at most n, partially ordered by the above subword relation. Furthermore, we give a sharp result $k \sim 2n/3$, which is an analogue of the former result [1].

Keywords: poset, reconstruction, subword

1 Introduction

Consider the q-element alphabet $F_q = \{0,1,...,q-1\}$. In this paper we examine the elements of the set F_q^n of sequences of length n called *words*. A subsequence u of a given word w is called a *subword*, denoted by $u \subseteq w$. Consider a given word $w \in F_q^n$.

Definition 1 Let $s_k(w) = u \in F_q^k : u \subseteq w$, the multiset of all of the $\binom{n}{k}$ subwords of w of length k. Let $s_k^*(w) = u \in F_q^k : u \subseteq w$, the set of all of the different subwords of w, of length k.

In other words, the set $s_k^*(w)$ is the set $s_k(w)$ without multiplicities. Here is a simple example to show the difference between s_k and s_k^*.

Example 2 Let $w = 00011$. Then $s_4(w) = \{0011, 0011, 0011, 0001, 0001\}$ and $s_4^*(w) = \{0011, 0001\}$.

There are two types of the *reconstruction problem*: for a given word w of length n what is the smallest k, such that we can reconstruct w from the set $s_k(w)$ or from the set $s_k^*(w)$. In Section 2.1 we give a short overview concerning the known results of this problem.

It is relatively easy to prove (see [2]), that $s_{n-1}^*(w)$ is enough for the reconstruction. Using this result, Erdős, Sziklai and Torney [2] determined the automorphism group of the partially ordered set (or poset) containing all words of length at most n over a q-letter alphabet. Similarly, we consider two other posets and determine their automorphism groups.

Let $u_{m,n}$ denote the word $a_1...a_n$ where $m \geq 2$, $a_1 = 0$ and $a_{i+1} \equiv a_i + 1 \bmod (m)$, ie. for $n \equiv l \bmod (m)$

$$u_{m,n} = 012...(m-1)012...(m-1)...012...(l-1),$$

furthermore, let $B^{m,n}$ denote the set of all subsequences of $u_{m,n}$ partially ordered by the subsequence relation. This is a notable word: among the n-long words over the m-element alphabet, $u_{m,n}$ has maximum number of subwords. Burosch et al. [3] determined $\mathrm{Aut}(B^{m,n})$ by algebraic way, in Section 2.2 using the result of [2] we give a significantly shorter proof of the theorem of Burosch et al.

In Section 2.3 we consider the well-known *DNA-words* and we define a new type of the reconstruction problem. Let $\Gamma_q = \{i,\bar{i} : i = 0,1,...,q-1\}$ be an alphabet of q pairs of symbols (called *complement pairs*); and denote by Γ_q^n the set of all

*Research is supported by ETIK Grant
**Research is supported by OTKA F-043772, T-043758, T-049662 and TÉT Hungarian-Spanish bilaterial grants

sequences of length n over the alphabet Γ_q. The elements of Γ_q^n are called *DNA-words.* Define $\overline{\overline{i}} = i$ for all i and for a word $f = f_1 f_2 ... f_n \in \Gamma_q^n$ let $f^* = \overline{f_n} ... \overline{f_1}$ be the *reverse complement* of f. Note that $(f^*)^* = f$.

Denote by $g \prec f$ if g is a subword of either f or $\widetilde{f}$. Let $d_m^*(f)$ denote the set of all words g of length at most m which $g \prec f$. The *DNA reconstruction* problem is the following: for a given DNA-word f of length n what is the smallest m such that we can reconstruct f from the set $d_m^*(f)$? We prove in a simple way that $d_{n-1}^*(f)$ is enough for reconstruction ([4]). Furthermore, we give a sharp bound for this problem in Section 2.4, for proof and more see Erdős et al. [5].

Let $D^{q,n}$ denote the poset of all DNA-words of length at most n over an alphabet of q complement pairs, partially ordered by the $\prec$ relation. As an application of the previous results we determine $\mathrm{Aut}(D^{q,n})$.

2 Main results

2.1 Known results

The original problem was first considered by Kalashnik in 1973: what is the smallest k such that we can reconstruct any word w of length n from $s_k(w)$, i.e. from the multiset of its $\binom{n}{k}$ subwords of length k?

An upper bound for k was find independently by Leon'tev and Smetanin [6] and Manvel et al [7]. Furthermore, in [7] the authors find a lower bound as well:

Theorem 3 We can reconstruct any w of length n from $s_k(w)$ for $k \geq \frac{n}{2}$; and for $k < \log_2 n$ we cannot.

In these papers the authors use some simple combinatorial ideas and show lot of examples.

Later Krasikov and Roditty [8] found an essentially better upper bound using new results on a problem of the classical Diofantine analysis:

Theorem 4 We can reconstruct w from $s_k(w)$ for $k \geq \lfloor \frac{16}{7}\sqrt{n} \rfloor$.

However the precise upper bound for k is still an open problem.

The second type of the reconstruction problem is the following: for a given word w of length n what is the smallest k such that we can reconstruct w from $s_k^*(w)$, i.e. from the set of its *different* subwords of length k. The following result was proved independently by Levenshtein [9] and Lothaire [1]:

Theorem 5 Every word w of length at most $2m-1$ is uniquely determined by its length and by the set $s_m^*(w)$.

In contrast with the previous problem this result is sharp:

Example 6 Consider the periodic words $u = 0101...01$ and $v = 1010...10$ of length $2n$. It is easy to see, that

$$s_n^*(u) = s_n^*(v) = F_2^n$$

ie. all the binary words of length n.

From Theorem 5 we get the following result:

Corollary 7 Every word $f \in F_q^n$ is uniquely characterized by its length and by subwords of length at most $\lfloor \frac{n+1}{2} \rfloor$.

In our proofs for the automorphism groups we used the following very weak version of Theorem 5 proved by Erdős, Sziklai and Torney [2] in a constructive way:

Lemma 8 If $3 \leq n$ then every word w of length n is uniquely determined by $s_{n-1}^*(w)$ i.e. its $(n-1)$-subwords.

2.2 A short proof of the theorem a Burosh et al.

Before the theorem let's see some remarks. It is clear, that the levels of the poset are invariant under an automorphism. Also homogeneity (i.e. all letters of the word are the same) and total inhomogeneity (i.e. all the letters of the word are different) are kept by every automorphism.

The basic idea of our proof is the following: we consider the action of an arbitrary automorphism on the first two levels of the poset. If an automorphism fixes these levels, then inductively, because of Lemma 8, it is the identity on the whole poset. Then it is enough to examine the automorphisms on the letters and on the two-letter subwords. The theorem was first proved by Burosch et al [3]:

Theorem 9 (i) if $1 \leq n \leq m$, then $\mathrm{Aut}(B^{m,n}) = Sym_n$;
(ii) if $m+1 \leq n \leq 2m-1$, then $\mathrm{Aut}(B^{m,n}) = Z_2 \otimes Sym_{2m-n}$;
(iii) if $2m \leq n$, then $\mathrm{Aut}(B^{m,n}) = Z_2$.

PROOF: We give here a short presentation of our arguments, for proofs and more see Ligeti and Sziklai [4].
(i) Now $u_{m,n} = 012...(n-1)$, i.e. it is a totally inhomogeneous word. Take an arbitrary automorphism $\sigma_0 \in \mathrm{Aut}(B^{m,n})$, and consider its action on the first level of the poset. Thus, this is a permutation π on $\{0,1,2,...,n-1\}$, take its inverse π^{-1}. This permutation induces an automorphism $\sigma_{\pi^{-1}}$ on the poset. Let $\sigma_1 = \sigma_0\sigma_{\pi^{-1}}$. Then σ_1 fixes all of the letters. Furthermore, σ_1 fixes all sequences of form ij where $i < j$ because $\sigma_1(ij) \neq (ji)$ as ji is not a subword of $u_{m,n}$. Then σ_1 is the identity on the two lowest levels of the poset and, by Lemma 8, on the whole poset.
(ii) In this case $u_{m,n} = 01...(m-1)01...(k-1)$ where $n = m+k$, $1 \leq k \leq m-1$ and let σ_0 be an arbitrary automorphism. We prove that we have strong restrictions for the images of the letters $0,1,...,k-1$, but we are free to choose the images of the remaining $2m-n$ letters (this yields the factor Sym_{2m-n}).

Remark 10 Let e be an element of the third level of the poset such that e contains the letters i, j only and suppose that ii is a subword of e. Then we can read from the poset whether j is the middle letter or not.

In that case $e = iij, jii$, or iji. The first two words have two subwords of length two, but the third word has three.

Remark 11 Let $j_1 < j_2 \leq k-1$ and $i \leq k-1$, $i \neq j_1, j_2$, then we can tell the difference between the $j_1 i i j_2$-type subwords and the $j_1 j_2 ii$-type or $ii j_1 j_2$-type subwords in the poset.

From these remarks we get the following:

Lemma 12 For $i = 0,1,2,...,k-1$ the image of the letter i is i or $(k-i-1)$ by any automorphism.

Now we define a mapping ρ: given a word $w = x_1x_2...x_sy_1y_2...y_tz_1z_2...z_u$, where $0 \leq x_i, z_i \leq k-1; k \leq y_i \leq m-1$; let

$$\rho(w) = z_u z_{u-1}...z_1 y_1 y_2...y_t x_s x_{s-1}...x_1.$$

Let ν be the mapping that changes all the letters i $(0 \leq i \leq k-1)$ for $k-1-i$ in each word (and does not changes the letters j for $k \leq j \leq m-1$). Clearly neither ρ nor ν is an automorphism but $\rho\nu$ is an involution in $\mathrm{Aut}(B^{m,n})$.

Now let σ_0 be an arbitrary automorphism, and consider its action on the letters $k,...,m-1$, this induces a permutation π on these letters (still on the first level), take its inverse π^{-1}. This permutation induces an automorphism $\sigma_{\pi^{-1}}$ on the poset. Let $\sigma_1 = \sigma_0\sigma_{\pi^{-1}}$. Then σ_1 is the identity on the letters $k,...,m-1$ and, as above, σ_1 fixes all sequences of form ij where $k \leq i < j$. Finally, if $\sigma_1(0) = (k-1)$ then let $\sigma = \rho\nu\sigma_1$ and if $\sigma_1(0) = 0$ then let $\sigma = \sigma_1$. Hence $\sigma(0) = 0$.

Lemma 13 If an automorphism fixes 0 then it fixes the two lowest levels of the poset.

Now by Lemma 8 and Lemma 13 we get the part **(ii)** of Theorem 9.

(iii) Now the word is of the following

$$u_{m,n} = 012...(m-1)012...(m-1)...012...(l-1)$$

for $n \equiv l \bmod (m)$. The Lemma 12 is clearly true here, furthermore

Lemma 14 For the letters $k \leq j \leq m-1$ the image of the letter j is j or (k+m-j-1).

Now let's describe the involutory automorphism of $B^{m,n}$. Let σ^* be the mapping that reverses all the words, and let $\nu_{k,m}$ be the mapping that changes the letters in the words in the following way: for $0 \leq i \leq k-1$ the letter i is changed for $k-1-i$, and for $k \leq j \leq m-1$ the letter j is changed for $(m+k-1-j)$. Clearly neither σ^* nor $\nu_{k,m}$ is an automorphism of $B^{m,n}$, but $\sigma^*\nu_{k,m} \in \mathrm{Aut}(B^{m,n})$.

Now let σ_0 be an arbitrary automorphism, furthermore let σ be $\sigma^*\nu_{k,m}\sigma_0$ if $\sigma_0(0) = (k-1)$ and let σ be σ_0 if $\sigma_0(0) = 0$. Now $\sigma(0) = 0$. Similarly to part **(ii)**, Lemma 13 is true which proves the Theorem 9. □

2.3 The DNA poset

The motivation of this analysis is coming from the biology: based on some basic properties of DNA strands we can build a mathematical model, which is easy to handle. DNA is composed of units called *nucleotides* : A,C,G and T, these letters are the elements of the alphabet. The letters form two complement pairs: A-T and C-G. Furthermore, DNA is double-stranded, i.e. each sequence occurs together with its reverse complement (we get the reverse complement in two steps: replacing each letter by its complement and reverse this sequence). For example the reverse complement of AACCGT is ACGGTT.

We can generalize the above properties for q complement pairs, and consider a reconstruction problem (see Section 1). It is easy to see that it makes no difference how many complement pairs build up the DNA-word:

Lemma 15 We can solve a reconstruction problem of all DNA strands over an alphabet with q complement pairs iff we can do it for the similar problem for $q = 2$, i.e. iff we can reconstruct all DNA strands over the alphabet $\{\{A,T\},\{C,G\}\}$.

It is clear that if we can reconstruct all strands over an alphabet with k complement pairs, then we can reconstruct them over ACGT. Conversely, suppose that we can reconstruct all strands over ACGT. Then replace the first complement pair with A-T, and all the others with C-G. Now we can reconstruct the strand, and so we find the places of letters from the first complement pair in the original strand (now A-T-s are there); then we can repeat the procedure in order to find the other complement pairs.

Using this, similar to Lemma 8 we proved the following :

Lemma 16 If $3 \leq n$ then every DNA-word f of length n is uniquely determined by $d^*_{n-1}(f)$.

Now we can determine $\mathrm{Aut}(D^{q,n})$. One can see easily two types of automorphisms: (1) a permutation $\pi \in Sym_q$ on the complement pairs induces an automorphism σ_π on $D^{q,n}$. Denote also by Sym_q the automorphism group generated by these σ_π-s. (2) Furthermore, consider a map which interchanges the elements of the i-th complement pair. This induces an automorphism σ_i^* on $D^{q,n}$. Denote by Z_2 the automorphism group generated by σ_i^*. Surprisingly, in most cases there are no more automorphisms. (Note that the automorphism that reverse the order of the letters, which is a natural one, is $\sigma_1^*\sigma_2^*...\sigma_k^*$; e.g. $\sigma_1^*\sigma_2^*(ab) = \overline{b}\overline{a}$, which is identified to its reverse complement, i.e. ba.)

Theorem 17 (i) if $n = 1$, then $\mathrm{Aut}(D^{q,n}) = Sym_q$;

(ii) if $n = 2$, then $\mathrm{Aut}(D^{q,n}) = Sym_q \times Sym_3^q \times Sym_4^{\binom{q}{2}}$;

(iii) if $n \geq 3$, then $\mathrm{Aut}(D^{q,n}) = Sym_q \times Z_2^q$.

PROOF: The proof of the theorem is similar to Theorem 9: if an automorphism fixes the two lowest levels of the poset, then because of Lemma 16 it is the identity on the whole poset (we can apply Lemma 16 only for $n \geq 3$). The case $n = 1$ is considered only for the sake of completeness. In **(ii)** the poset has only two levels. It is clear that an automorphism transfers complement pairs to complement pairs. Take an arbitrary automorphism $\sigma_0 \in \mathrm{Aut}(D^{q,n})$ and consider its action on the set of complement pairs. Thus, this is a permutation on q elements, take its inverse π^{-1}. This permutation induces an automorphism $\sigma_{\pi^{-1}}$ on the poset $D^{q,n}$. Let $\sigma_1 = \sigma_0\sigma_{\pi^{-1}}$. Then σ_1 fixes all of the complement pairs. Now one can partition the second level into $q + \binom{q}{2}$ blocks: we have q blocks of size 3 with elements $\{ii \equiv \overline{ii}, i\overline{i}, \overline{i}i\}$; and $\binom{q}{2}$ blocks of size 4, with elements $\{ij \equiv \overline{ji}, i\overline{j} \equiv \overline{j}i, \overline{i}j \equiv j\overline{i}, \overline{ij} \equiv \overline{j}\overline{i}\}$ for all $i \neq j$, each block is fixed by σ_1 (setwise). This means q copies of Sym_3 and $\binom{k}{2}$ copies of Sym_4, and these automorphisms differ and commute, which proves the second part of the theorem.

For the case **(iii)** we can prove the following easily:

Remark 18 The automorphism σ_1 fixes all sequences in form of ii.

To the contrary suppose that $\sigma_1(ii) = i\overline{i}$ (or equivalently $\overline{i}i$). Then we can not define $\sigma_1(iii)$.

Let σ_i^* be the automorphism which interchanges the elements of the i-th complement pair. Denote by σ_2 the product of σ_1 and those σ_i^*'s for which $\sigma_1(i\overline{i}) = \overline{i}i$. Then σ_2 fixes all elements in the 3-blocks. Furthermore:

Remark 19 The automorphism σ_2 fixes all sequences in form of ij for all $i \neq j$.

Now by Remark 18 and Remark 19 we have that the two lowest levels of the poset are fixed, which completes the proof. □

2.4 Sharp bound for DNA words

In this section we give the solution of the following problem: for a given DNA-word f of length n what is the smallest m such that we can reconstruct f from the set $d^*_m(f)$. The case of *one* complement pair differs a little bit from the general case.

Consider the following words:

$$f = \bar{A}^{2k+\varepsilon}\bar{C}CA^k \quad \text{and} \quad g = \bar{A}^{2k+\varepsilon-1}\bar{C}CA^{k+1}, \tag{1}$$

where $\varepsilon \in \{0,1,2\}$ and $k \geq 1$. The length of both words are $3k+2+\varepsilon$. On the one hand the subword $\bar{A}^{2k+\varepsilon}$ of f satisfies $\bar{A}^{2k+\varepsilon} \not\prec g$. On the other hand it is easy to verify that

$$d^*_{2k+\varepsilon-1}(f) = d^*_{2k+\varepsilon-1}(g).$$

Therefore the following result is sharp.

Theorem 20 Every word $f \in \Gamma^*$ of length at most $3m+1$ built up with two complement pairs (and containing letters from both of them) is uniquely determined by its length and by the set $d^*_{2m}(f)$.

If there is only one letter pair occurring in the words then the situation is slightly different. One can consider the following example:

$$f = \bar{A}^{2k+\varepsilon}A^k \quad \text{and} \quad g = \bar{A}^{2k+\varepsilon-1}A^{k+1}, \tag{2}$$

where $\varepsilon \in \{0,1,2\}$ and $k \geq 1$. The length of both words are $3k+\varepsilon$. On the one hand the subword $\bar{A}^{2k+\varepsilon}$ of f satisfies $\bar{A}^{2k+\varepsilon} \not\prec g$. On the other hand it is easy to check that

$$d^*_{2k+\varepsilon-1}(f) = d^*_{2k+\varepsilon-1}(g).$$

Therefore in the case of "homogenous" words, we have a slightly weaker result than in Theorem 20:

Theorem 21 Every word $f \in \{A,\bar{A}\}^*$ of length at most $3m-1$ is uniquely determined by its length and by the set $d^*_{2m}(f)$

Using induction on the number k of different complement pairs we get the following:

Theorem 22 Theorem 20 remains valid if the word f contains letters from k different complement pairs.

Finally we can collect the main results of this section in the following corollary, which is an analogue of Corollary 7:

Corollary 23 Every word $f \in \Gamma^n_q$ is uniquely characterized by its length and by subwords of length at most
(i) $\lfloor \frac{2n-1}{3} \rfloor$, if at least 2 letters from distinct complement pairs occur in f;
(ii) $\lfloor \frac{2n}{3} \rfloor$, for $q = 1$.

References

[1] M. LOTHAIRE, Combinatorics on words, *Encyclopedia of Mathematics and its Applications* (1983) **17** pp. 119-120.

[2] P.L. ERDŐS, P. SZIKLAI AND D. TORNEY, The word poset and insertion-deletion codes, *Electronic Journal of Combinatorics* (2001) **8** No. 2, 10 pp. (electronic)

[3] G. BUROSCH, H-D. O.F. GRONAU, J-M. LABORDE, The automorphism group of the subsequence poset $B_{m,n}$, *Order* (1999) **16** No. 2, pp. 179-194.

[4] P. LIGETI, P. SZIKLAI, Automorphisms of subword-posets, submitted to *Discrete Math.* (2003)

[5] P.L. ERDŐS, P. LIGETI, P. SZIKLAI, D.C. TORNEY, Subwords in reverse-complement order, submitted to *SIAM J. Disc. Math.* (2004)

[6] V.K. LEONT'EV, YU G. SMETANIN, Problems of information on the set of words, *J. Math. Sci. (New York)* (2002) **108** No. 1, pp. 49-70.

[7] B. MANVEL, A. MEYEROWITZ, A. SCHWENK, K. SMITH. P. STOCKMEYER, Reconstruction of sequences, *Discrete Math* (1994) **94** No. 3 pp. 209-219.

[8] I. KRASIKOV, Y. RODITTY, On a reconstruction problem for sequences , *J. Combin. Theory Ser. A* (1997) **77** No. 2, pp. 344-348.

[9] V. I. LEVENSHTEIN, Efficient reconstruction of sequences from their subsequences or supersequences , *J. Combin. Theory Ser. A* (2001) **93** No. 2, pp. 310-332.

Chomp with Poison-Strewn Chocolates

HIRO ITO

Department of Communications and Computer Engineering,
School of Informatics,
Kyoto University.
Kyoto.606-8501, Japan
itohiro@i.kyoto-u.ac.jp

GISAKU NAKAMURA

Research Institute of Educational Development,
Tokai University.
Tokyo.151-8677, Japan

SATOSHI TAKATA

Department of Communications and Computer Engineering,
School of Informatics,
Kyoto University.
Kyoto.606-8501, Japan
takata@lab2.kuis.kyoto-u.ac.jp

Abstract: Chomp is one of combinatorial games. This game starts with an m-by-n chocolate bar, two players in turn chomp this bar and whoever eats the square on the top-left corner, the poisonous chocolate, loses. Though the existence of the first player's winning strategy was proved by a beautiful argument, nobody knows how to win. We extend this game and consider "Poison-Strewn Chomp." We have found how to decide which player has a winning strategy in this game. We also have shown general consequences for poset games.

Keywords: Chomp, poset games, Nim, winning strategy, combinatorial games

1 Introduction

We studied a well-known combinatorial game called *Chomp*. This game starts with an m-by-n chocolate bar. The square on the top-left corner is a unique poisonous chocolate. Two players in turn choose a square. If one chooses a square, he/she should eat it together with all the squares below and/or to the right of it. Whoever eats the poisonous chocolate loses.

In this game, either the first player should win or the second one should win. That the m-by-n position, the starting position, is the first player's winning position has been proved. We will show it. See Fig.1. We assume that the starting position is the second player's winning one. The first player takes a bottom-right square (Fig. 1 (1)). If there exists the second player's winning strategy, he/she chooses a square according to it (Fig. 1 (2)). But the move must be available for the first player before the turn and if he/she move according to it, he/she should win (Fig. 1 (3)). So there exists the first player's winning strategy. This argument is called *strategy-stealing*.

But it does not show how to win. The prototype of Chomp, "divisors" appeared in 1952. Then soon Chomp appeared and has been studied, but nobody knows the polynomial-time winning strategy of Chomp.

The position of Chomp has a poisonous square on the top-left corner. We generalize it and introduce *Poison-Strewn Chomp*, in which some poisonous squares can be located on arbitrary locations not only on the top-left corner. We can apply this extension to general poset games and introduce a new game *Poison-Strewn Poset Game*.

Before we explain the application, we will explain poset games. A poset consists of a set and a partial order. When a poset A is given, we play a poset game as follows. Two player in turn choose an element a in A. Then he/she deletes a together with all larger elements than a. Whoever is unable to move loses. As the definition of a poset, this game is expressed by a dag (directed acyclic graph). See Fig. 2 (1).

We add a vertex and a directed edge from the new vertex to all vertices with no parents. We define an added vertex as the poisonous vertex. See Fig. 2 (2). We define this game as:

1. Two players in turn choose a vertex.
2. When one chooses a vertex, he/she should delete it together with all descendents of it.
3. Whoever chooses the poisonous vertex loses.

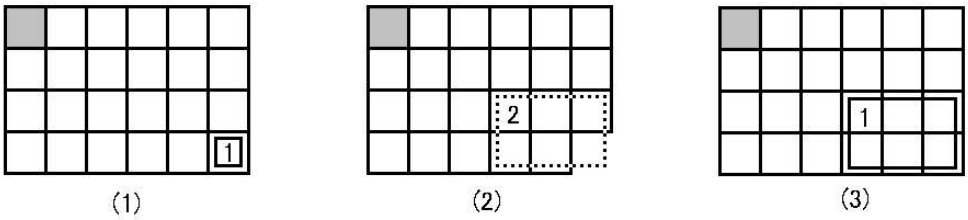


Figure 1: Strategy-stealing : the proof of that the first player has a winning strategy.

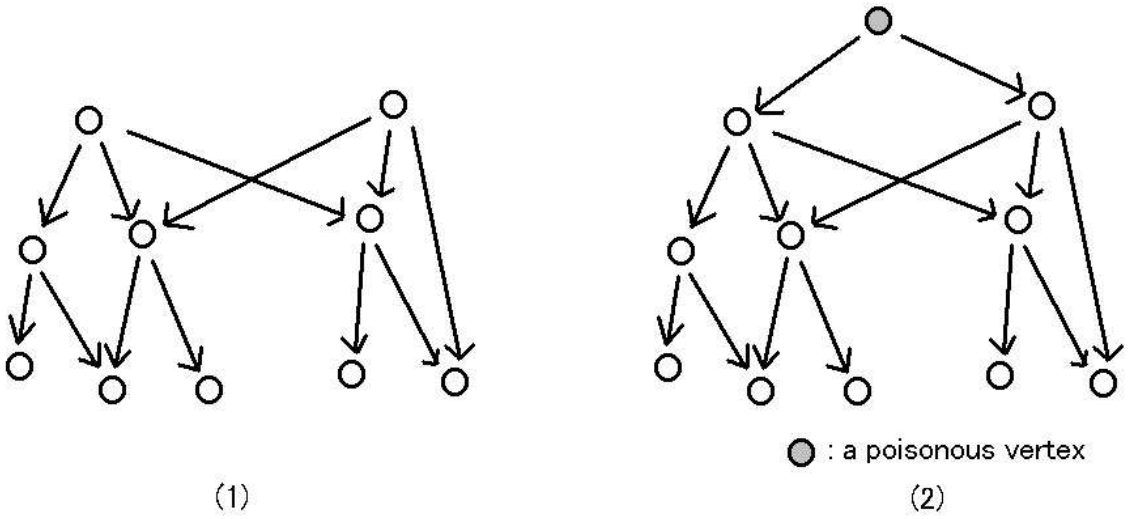


Figure 2: A poset game on a dag

This game is equal to the poset game. So we express a poset game as a game on a dag with a poisonous vertex. We can easily see that Chomp is one of poset games.

As we extend Chomp, we introduce a new game, which uses a dag with some poisonous vertices. We call it *Poison-Strewn Poset Game*. In Poison-Strewn Poset Game, we assume that two players respectively have integral *lives a* and *b*. If one eats poisonous vertices, the lives decrease as many as the number of poisonous vertices he/she ate. Whoever loses his/her lives is a loser.

This paper presents which player has a winning strategy on Poison-Strewn Poset Game on restricted conditions. Applying this argument to Poison-Strewn Chomp, we can show which player has a winning strategy on Poison-Strewn Chomp on any conditions.

In this paper, first we survey past studies and current subjects, in section 2. We show our consequences in section 3. Section 4 is conclusions.

2 Past Studies

Poset games are one field of combinatorial games and have been studied in earnest. The theory of combinatorial games containing poset games has applications in fields such as complexity theory, artificial intelligence, error-correcting codes, algorithms and surreal analysis. So finding winning strategies and knowing the properties of such games are meaningful subjects. Nim, Green Hackenbush on Trees and Chomp are well-known examples of poset games.

First we introduce some definitions often used in game theory.

Given a set of non-negative integers S, *mex*(S) is the smallest non-negative integer that does not belong to S. For example, $\mathrm{mex}(0,1,4,5)=2$, and $\mathrm{mex}(1,4,5)=0$.

Next we introduce the *g-value* (also called grundy-value, nim-value, or Sprague-Grundy function). The g-value of any game position is recursively defined as the mex of the set of g-values of all game positions that can remain after exactly one move. For example, the g-value of the positions where we cannot move is 0. The g-value of the positions from which we can reach the position with g-value 0 and one with g-value 1 after one move is 2. So the g-values of all losing positions are 0 and them of all winning positions are some positive integers. If we can know whether the g-value of a current position is 0 or a positive integer and find the move to change the g-value into 0 when the g-value is positive in polynomial-time, it means that we know the polynomial-time winning strategy.

Nim is a simple and typical poset game, which plays an important role in combinatorial game theory. The game positions of Nim consists of a unique poisonous vertex which has some children and other vertices which has only one parent and has at most one child (See Fig.3). Since Nim has been studied well, we can know which player has winning strategy and how

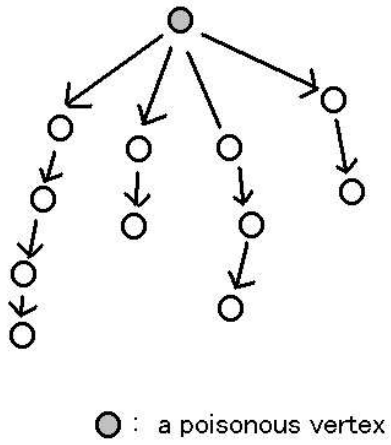


Figure 3: Nim

to win from any game positions in polynomial-time. We will show the winning strategy below by explaining the winning strategy of Green Hackenbush on Trees, which contains Nim.

Green Hackenbush on Trees is the extension of Nim. The game positions of Green Hackenbush on Trees can be represented by an out-tree, ie., it consists of a unique poisonous vertex which has some children and other vertices which has only one parent and has some children (See Fig.4).

We will show the polynomial-time winning strategy of Green Hackenbush on Trees.

The g-value of a vertex can be calculated by using ones of all children. Let $c_1, c_2, \cdots, c_p$ be the children of a vertex v. Then the g-value of v, $g(v)$, is defined as follows.

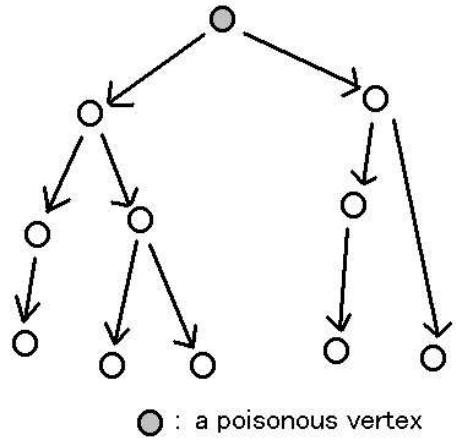


Figure 4: Green Hackenbush on Trees

$$g(v) = ex(g(c_1)+1, g(c_2)+1, \ldots, g(c_p)+1),$$

where $ex(*)$ means applying exclusive-OR to every digit of the binary representation of each value, e.g., $ex(13,8,3,4) = ex(1101,1000,0011,0100) = 0010 = 2$

As you see, we can recursively calculate g-values. The g-value of the root (i.e., the poisonous vertex) is the g-value of the configuration. The calculation obviously needs polynomial-time. We show an example of the calculation of g-values (See Fig.5). By a recursive calculation, we can know that the g-value is 7.

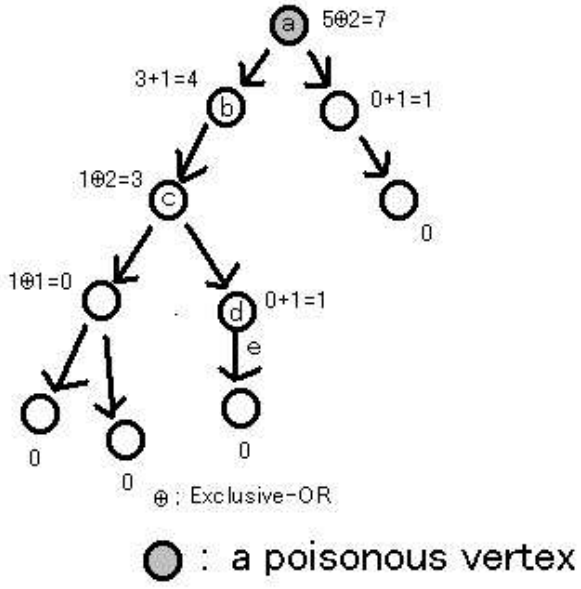


Figure 5: Calculation of g-values and how to find a winning move.

Next we will show how to find the winning move from the position with a positive g-value. The algorithm is easy so we only show an example here.

See Fig.5 again. The winning move is the move which changes a positive g-value into 0. See a vertex *a*. We want to change the g-value of *a* into 0. 5 is 101, 2 is 10 and 7 is 111 by binary representation. 5 has 1 in Most Significant Bit of 7. So we calculate $5 \oplus 7 = 2$. Then we have to change 5 into 2 to change the g-value of *a* into 0.

Next see a vertex *b*. We want to change the g-value of *b* into 1. To do so, we have to change the g-value of *c* into 0.

Next see a vertex c. We want to change the g-value of c into 0. 1 is 1, 2 is 10 and 3 is 11 by binary representation. 2 has 1 in Most Significant Bit of 3. So we calculate $2 \oplus 3 = 1$. Then we have to change 2 into 1 to change the g-value of c into 0.

Next see a vertex d. We want to change the g-value of d into 1. So all we have to do is cutting edge e.

Above two games are called N-free poset games. N-free means that the poset has no four elements a, b, c, d satisfying $a \parallel b$, $a < c$, $a < d$, $b \parallel c$, $b < d$, and $c \parallel d$ ($x \parallel y$ means that the pair is not comparable). (See Fig.6.) A polynomial-time winning strategy for any poset games on an N-free poset is given in [7].

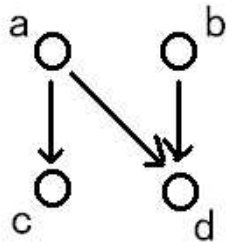


Figure 6: The forbidden construction of N-free poset.

Chomp is also an extension of Nim but is far complicated (not N-free). What do we know about Chomp? To begin with, we know how to win on 1-by-n, 2-by-n and n-by-n (perfect square) chocolate bars. The winning strategy on a 1-by-n bar is trivial. On a 2-by-n bar, you should choose a bottom-right square. On an n-by-n bar, you should choose a square (2,2), which is on right-down of the poisonous chocolate. These strategies are given by Fred Schuh and D. Gale [1]. Now a 3-by-n chocolate bar has been studied. Doron Zeilberger made a computer program to solve three-rowed Chomp and solved 3-by-115 [3]. Xinyu Sun extended this program and found some theories on the positions with the second column having three squares [8]. David Gale offers a prize of $ 100.00 for the first complete analysis of 3D-Chomp [5]. 3D-Chomp is the game on an ℓ-by-m-by-n chocolate bar. But even 2D-Chomp has not been solved yet.

Steven Byrnes found one excellent theorem about all poset games, Poset Game Periodicity Theorem. We will explain the theorem. In an infinite poset X, suppose we have two infinite chains $C(c_1 < c_2 \cdots)$ and $D(d_1 < d_2 < \cdots)$, and a finite subset A, all pairwise disjoint, and assume that no element of C is less than an element of D. Suppose we delete c_{m+1} and d_{n+1}. Then we get a finite poset X' containing $C'(c_1 < c_2 < \cdots < c_m)$, $D'(d_1 < d_2 < \cdots < d_n)$ and A'. We define the poset X' as $A_{m,n}$.

$$A_{m,n} = A \cup C \cup D - \{x \in X | x \geq c_{m+1}\} - \{x \in X | x \geq d_{n+1}\}$$

Then let

$$f_{A,k}(m) = \{n | g(A_{m,n}) = k\}$$ ($g(X)$ is the g-value of a position X).

Poset Game Periodicity Theorem is below.

Poset Game Periodicity Theorem *For any $k \in N_0$, either there are only finitely many positions of the form $A_{m,n}$ with g-value k, or else there exists $N \in N_0, p \in N_0$ such that, for $m \geq N$, $f_{A,k}(m) - m = f_{A,k}(m+p) - (m+p)$.*

Byrnes's theorem have found that when we fix a poset A' and a poset X' has a fixed g-value, a poset X' with a fixed g-value is finite or the value $n - m$ changes ultimately periodicly as m increases. So we can deal with an infinite poset X' with fixed g-value as a finite poset.

Consider that we apply this theorem to Chomp. D is a top row, C is a second row and A is other all squares. Byrnes's theorem asserts that when we fix all but top two rows and consider losing positions, losing positions are finite or the difference between the number of a top row and the number of a second row changes ultimately periodicly as the number of a second row increases. Using this property, we can solve Chomp whose all but top two rows are fixed in polynomial-time [6]. This theorem won the $100,000 scholarship.

Now the application of Byrnes'theorem to other games is an interesting subject. Of course studying the solution of unsolved games, such as Chomp, is also interesting.

3 Which Player Is the Winner?

We will strictly define the game *Poison-Strewn Poset Game*. The starting position is a dag with a poisonous vertex. The two players in turn choose a vertex and delete it together with all descendents of it. Some poisonous vertices exist on a dag. Both two players can identify poisonous vertices. Two players respectively have integral *lives a* and *b*. If one eats poisonous vertices, the lives decrease as many as the number of poisonous vertices he/she ate. Whoever loses his/her lives is a loser. We assume the number of poisonous vertices is greater than or equal to the sum of two players' lives, $a+b$. From this assumption, a game never end in draw. We call the vertex with no children *a maximal vertex*.

Chomp is one of poset games, so the rules of *Poison-Strewn Chomp* are easily obtained from them of Poison-Strewn Poset Game. We have proved the following theorem.

Theorem 1 In the Poison-Strewed Chomp, a player who has a winning strategy is:

1. the player who has more lives if $a \neq b$,
2. the second player if $a = b$ and the bottom-right chocolate is poisonous and
3. the first player if $a = b$ and the bottom-right chocolate is not poisonous.

□

Before proving this theorem, we will show some general consequences.

Lemma 2 The player with more lives has a winning strategy in Poison-Strewn Poset Game. □

For the purpose of proving Lemma 2, we first introduce the following lemma.

Lemma 3 When all maximal vertices are poisonous, the first player can force the second one to take one poisonous vertex by taking only one poisonous vertex.

PROOF: This proof uses a strategy-stealing argument. We consider that all maximal vertices are poisonous and the first player takes one poisonous vertex. We assume that the second player has a move to force the first one to take the next poisonous vertex. But the move was also available for the first player before the turn and if he/she takes the move, he/she can force the second player to eat the next poisonous vertex. □

Now we prove Lemma 2.

PROOF: When $a \neq b$, the player with more lives arbitrarily take vertices without poison whenever he/she can. Then he/she comes to the position where he/she should take a poisonous vertex. The all maximal vertices of the position are poisonous. From Lemma 3, the player with more lives can force his/her opponent to take one poisonous vertex by taking only one poisonous vertex. So the player with more lives can escape till the end of a game keeping the advantage of lives. Then the player with more lives has a winning strategy. □

We solved the case when $a \neq b$, but have not proved which player should win when $a = b$. However, we can easily obtain the following property in a sub-case of $a = b$.

Corollary 4 If $a = b$ and all maximal vertices are poisonous, the second player has a winning strategy.

PROOF: The first player should take a poisonous vertex soon after a game starts. This condition is the same with Lemma 2. So the player with more lives has a winning strategy. Namely, the second player has a winning strategy. □

When some maximal vertices are not poisonous, which player should win? Though we cannot present it, we have found the lemma which simplify a game.

Lemma 5 We construct a new dag as follows. We make all ancestors of poisonous vertices poisoned (Fig. 7 (2)) and then contract all poisonous vertices (Fig. 7 (3)). The winning strategy of the dag is equal to the one of the original dag.

PROOF: When one player takes the first poisonous vertex, his/her opponent has a winning strategy. Because his/her opponent has more lives and can escape till the end of a game, which is followed by Lemma 2. So both players make a move not to take the first poisonous vertex. Then we can see that both players' lives are 1 in the light of which player should win. So we assume that both players' lives are 1.

We consider the changed dag where all ancestors of poisonous vertices are made poisoned (Fig. 7 (2)) and the original dag (Fig. 7 (1)). Both player cannot take the ancestor of a poisonous vertex without taking the poisonous vertex. So the winning strategy on the changed dag is equal to the one on an old dag.

Next we consider the changed dag (Fig. 7 (2)) and a simplified new dag (Fig. 7 (3)). Both players can make the same move on both dags and the influence of the move to the other part of a dag is equal. The losing condition is also equal. So the winning strategy on the changed dag is equal to the one on a new dag.

From the above discussion, the winning strategy of the simplified new dag is equal to the one of the original dag. □

By using the argument, we can reduce the problem to the original (one-poison) poset game.

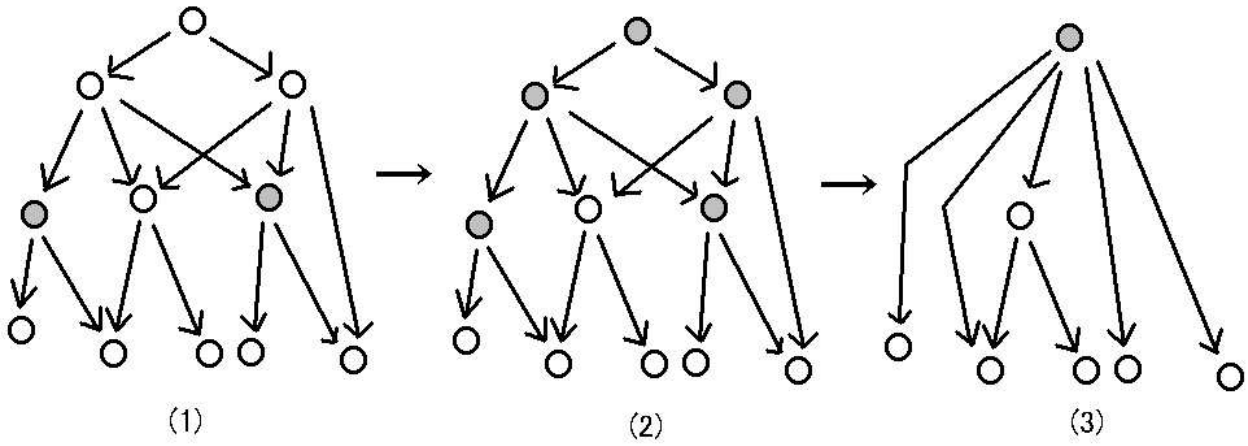


Figure 7: Making a new dag

Applying this argument to Poison-Strewn Chomp, we can show which player has a winning strategy in any case. Now we are ready to prove Theorem 1.

Theorem 1

PROOF:

1. When $a \neq b$.

   That the winning strategy of the player with more lives exists is trivial from Lemma 2.

2. When $a = b$ and the bottom-right chocolate is poisonous.

   That the winning strategy of the second player exists is trivial from Corollary 4.

3. When $a = b$ and the bottom-right chocolate is not poisonous.

   We consider that the first player eats the unique non-poisonous bottom-right chocolate. We assume that the second player has a strategy to force the first one to eat the first poisonous chocolate. But the move was available for the first player before the turn. If he/she took that move, he/she could force his/her opponent to eat the first chocolate. From this argument, the first player can force the second one to eat the first poisonous chocolate. This situation is the same with the case 1, then the first player wins.

□

4 Concluding Remarks

This paper introduce Poison-Strewn Poset Games. It proved that the player with more lives has a winning strategy. It also proved that the second player has a winning strategy when two players' lives are equal and all maximal squares are poisonous. For the remaining case, which includes the general poset games, this paper showed we can reduce the problem to the original (one-poison) poset game.

We also introduce Poison-Strewn Chomp. The paper presented a method to decide which player has a winning strategy: If two players' lives are different, the player with more lives has a winning strategy; If two players' lives are equal, when the bottom-right chocolate is poisonous, the second player has the winning strategy and otherwise the first player has the winning strategy.

When we consider real games, two players of the games may not have equivalent condition. So the extension that two players' lives are difference seems to be natural. This extension may give new arguments in poset games and combinatorial games. They are interesting problems for future work.

5 Acknowledgement

We are very grateful to Prof. Iwama, who helped and advised to this work.

References

[1] ANDRIES E. BROUWER, The game of Chomp, *ONLINE DOCUMENT*, http://www.win.tue.nl/ aeb/games/chomp.html

[2] DORON ZEILBERGER, Chomp, Recurrences, and Chaos, *J. Difference Equations and its Applications* (2004) **10** pp. 1281-1293 .

[3] DORON ZEILBERGER, Three-Rowed CHOMP, *Adv Applied Math* (2001) **26** pp. 168-179 .

[4] ELWYN R. BERLEKAMP, JOHN H. CONWAY AND RICHARD K. GUY, Winning Ways for Your Mathematical Plays 2nd, *A K PETERS LTD* (2001) **1** .

[5] RICHARD J. NOWAKOWSKI, Games of No Chance, *Cambridge Univ Pr (Sd)* (2002) **29** .

[6] STEVEN BYRNES, Poset Game Periodicity, *INTEGER* (2003) **3** pp. 1-16 .

[7] W. DEUBER AND S. THOMASSE, Grundy sets of partial orders, *ONLINE DOCUMENT* , http://citeseer.nj.nec.com/19302.html

[8] XINYU SUN, Improvement on Chomp, *INTEGER* (2002) **2** pp. 1-8 .

A Two-Sided Discrete-Concave Market with Possibly Bounded Side Payments

SATORU FUJISHIGE*

RIMS
Kyoto University
Kyoto 606-8502, Japan
fujishig@kurims.kyoto-u.ac.jp

AKIHISA TAMURA†

Department of Mathematics
Keio University
Yokohama 223-8522, Japan
aki-tamura@math.keio.ac.jp

Abstract: The marriage model due to Gale and Shapley and the assignment model due to Shapley and Shubik are standard in the theory of two-sided matching markets. We give a common generalization of these models by utilizing discrete concave functions and considering possibly bounded side payments. Our main result is the existence of a pairwise stable outcome in our model.

Keywords: Two-sided matching markets, Pairwise stability, Discrete convex analysis.

1 Introduction

The marriage model due to Gale and Shapley [12] and the assignment model due to Shapley and Shubik [23] are standard in the theory of two-sided matching markets. The largest difference between these two models is that the former does not allow side payments or transferable utilities whereas the latter does.

Since Gale and Shapley's paper a large number of variations and extensions have been proposed. Recently, the marriage model was extended to frameworks in combinatorial optimization. Fleiner [7] extended the marriage model to the framework of matroids, and Eguchi, Fujishige and Tamura [3] extended this formulation to a more general one in terms of discrete convex analysis which was developed by Murota [16, 17, 18]. Alkan and Gale [1] and Fleiner [8] also generalized the marriage model to another wide frameworks. The existence of stable matchings in these models are guaranteed.

For the other standard model, the assignment model, Kelso and Crawford [15] proposed a seminal one-to-many variation in which a payoff function of each worker is strictly increasing (not necessarily linear) in a side payment, and a payoff function of each firm satisfies gross substitutability and is linear in a side payment. They showed the existence of a stable outcome.

On the other hand, progress has been made toward unifying the marriage model and the assignment model. Kaneko [14] formulated a general model that includes the two by means of characteristic functions, and proved the nonemptiness of the core. Roth and Sotomayor [22] proposed a general model that also encompasses both and investigated the lattice property for payoffs. Eriksson and Karlander [4] proposed a hybrid model of the marriage model and the assignment model. In the Eriksson-Karlander model, the set of agents is partitioned into two categories, one for "rigid" agents and the other for "flexible" agents. Rigid agents do not get side payments, that is, they behave like agents in the marriage model, while flexible agents behave like ones in the assignment model. Sotomayor [25] also further investigated this hybrid model and gave a non-constructive proof of the existence of a pairwise stable outcome. Fujishige and Tamura [9] proposed a generalization of the hybrid model due to Eriksson and Karlander [4] and Sotomayor [25] by utilizing $\mathrm{M}^\natural$-concave functions in discrete convex analysis.

The model in [9] motivates us to consider a more natural common generalization of the marriage model and the assignment model by utilizing discrete convex analysis. Our goal is to propose such a model which includes models in [3, 4, 7, 9, 12, 23, 25] as special cases, and to verify the existence of a pairwise stable outcome. The characteristic idea of our present model is to adopt a range of a side payment for each pair of agents instead of using the concept of rigid and flexible pairs. Our model can deal with rigidity and flexibility of pairs as ranges $[0,0]$ and $(-\infty,+\infty)$ of side payments respectively as well as any ranges of side payments. This approach is more natural and adaptable than that adopting rigidity and flexibility.

The present extended abstract gives our stability concepts and results, and is organized as follows (see [10] for proofs). Section 2 explains $\mathrm{M}^\natural$-concavity and gives its nice properties from the viewpoint of mathematical economics. Section 3 describes our general model and two concepts of stability, namely "pairwise stability" and "pairwise strict stability," discusses relations between these two concepts, and gives our main theorem about the existence of pairwise stable outcomes.

*Research is supported by a Grant-in-Aid for Scientifi c Research from the Ministry of Education, Culture, Sports, Science and Technology of Japan.
†Research is supported by a Grant-in-Aid for Scientifi c Research from the Ministry of Education, Culture, Sports, Science and Technology of Japan.

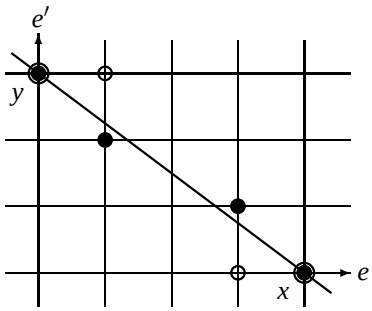


Figure 1: $M^\natural$-concavity for two dimensional case: the sum of function values of black points or that of white points is greater than or equal to that of x and y.

2 $M^\natural$-concavity

In this section we explain the concept of $M^\natural$-concave function, which plays a central role in discrete convex analysis (see [18] for details). Let E be a nonempty finite set, and let 0 be a new element not in E. We denote by $\mathbf{Z}$ the set of integers, and by $\mathbf{Z}^E$ the set of integral vectors $x = (x(e) \mid e \in E)$ indexed by E, where $x(e)$ denotes the e-component of vector x. Also, $\mathbf{R}$ and $\mathbf{R}^E$ denote the set of reals and of real vectors indexed by E, respectively. Let $\mathbf{0}$ and $\mathbf{1}$ be vectors of all zeros and all ones of an appropriate dimension. We define the positive support $\mathrm{supp}^+(x)$ and the negative support $\mathrm{supp}^-(x)$ of $x \in \mathbf{Z}^E$ by

$$\mathrm{supp}^+(x) = \{e \in E \mid x(e) > 0\}, \quad \mathrm{supp}^-(x) = \{e \in E \mid x(e) < 0\}.$$

For each $S \subseteq E$, we denote by χ_S the characteristic vector of S defined by: $\chi_S(e) = 1$ if $e \in S$ and $\chi_S(e) = 0$ otherwise, and write simply χ_e instead of $\chi_{\{e\}}$ for all $e \in E$. We also define χ_0 as the zero vector in $\mathbf{Z}^E$, where we assume $0 \notin E$. For $S \subseteq E$ and $x \in \mathbf{Z}^E$, let $x(S) = \sum_{e \in S} x(e)$. For a vector $p \in \mathbf{R}^E$ and a function $f : \mathbf{Z}^E \to \mathbf{R} \cup \{-\infty\}$, we define functions $\langle p, x\rangle$ and $f[p](x)$ in $x \in \mathbf{Z}^E$ by

$$\langle p, x\rangle = \sum_{e \in E} p(e)x(e), \qquad f[p](x) = f(x) + \langle p, x\rangle \qquad (\forall x \in \mathbf{Z}^E).$$

We also define arg max, the set of maximizers, of f on $U \subseteq \mathbf{Z}^E$ and the *effective domain* of f by

$$\begin{aligned} &\arg\max\{f(y) \mid y \in U\} = \{x \in U \mid \forall y \in U : \ f(x) \geq f(y)\}, \\ &\mathrm{dom}\, f = \{x \in \mathbf{Z}^E \mid f(x) > -\infty\}. \end{aligned}$$

We abbreviate $\arg\max\{f(y) \mid y \in \mathbf{Z}^E\}$ to $\arg\max f$.

A function $f : \mathbf{Z}^E \to \mathbf{R} \cup \{-\infty\}$ with $\mathrm{dom}\, f \neq \emptyset$ is called *$M^\natural$-concave* (Murota [18] and Murota and Shioura [19]) if it satisfies

($M^\natural$) $\forall x, y \in \mathrm{dom}\, f$, $\forall e \in \mathrm{supp}^+(x-y)$, $\exists e' \in \mathrm{supp}^-(x-y) \cup \{0\}$:

$$f(x) + f(y) \leq f(x - \chi_e + \chi_{e'}) + f(y + \chi_e - \chi_{e'}).$$

($(M^\natural)$ is denoted by $(-M^\natural$-EXC) in Murota [18].) Condition ($M^\natural$) says that the sum of the function values at two points does not decrease as the points symmetrically move one or two step closer to each other on the set of integral lattice points of $\mathbf{Z}^E$ (see Figure 1). This is a discrete analogue of the fact that for an ordinary concave function the sum of the function values at two points does not decrease as the points symmetrically move closer to each other on the straight line segment between the two points.

By the definition of $M^\natural$-concavity, if f is $M^\natural$-concave, then $f[p]$ is also $M^\natural$-concave for any $p \in \mathbf{R}^E$. Here are two simple examples of $M^\natural$-concave functions.

Example 1 For the independence family $\mathscr{I} \subseteq 2^E$ of a matroid on E and $w \in \mathbf{R}^E$, the function $f : \mathbf{Z}^E \to \mathbf{R} \cup \{-\infty\}$ defined by

$$f(x) = \begin{cases} \sum_{e \in X} w(e) & \text{if } x = \chi_X \text{ for some } X \in \mathscr{I} \\ -\infty & \text{otherwise} \end{cases} \qquad (\forall x \in \mathbf{Z}^E)$$

is $M^\natural$-concave (see Murota [18]). ∎

Example 2 We call a nonempty family $\mathscr{T}$ of subsets of E a *laminar family* if $X \cap Y = \emptyset$, $X \subseteq Y$ or $Y \subseteq X$ holds for every $X, Y \in \mathscr{T}$. For a laminar family $\mathscr{T}$ and a family of univariate concave functions $f_Y : \mathbf{R} \to \mathbf{R} \cup \{-\infty\}$ indexed by $Y \in \mathscr{T}$, the function $f : \mathbf{Z}^E \to \mathbf{R} \cup \{-\infty\}$ defined by

$$f(x) = \sum_{Y \in \mathscr{T}} f_Y(x(Y)) \qquad (\forall x \in \mathbf{Z}^E)$$

is $\mathrm{M}^\natural$-concave if $\operatorname{dom} f \neq \emptyset$ (see Murota [18]). ∎

An $\mathrm{M}^\natural$-concave function has nice features as a value function from the point of view of mathematical economics. For any $\mathrm{M}^\natural$-concave function $f : \mathbf{Z}^E \to \mathbf{R} \cup \{-\infty\}$, there exists an ordinary concave function $\overline{f} : \mathbf{R}^E \to \mathbf{R} \cup \{-\infty\}$ such that $\overline{f}(x) = f(x)$ for all $x \in \mathbf{Z}^E$ (Murota [16]). That is, any $\mathrm{M}^\natural$-concave function on $\mathbf{Z}^E$ has a concave extension on $\mathbf{R}^E$. An $\mathrm{M}^\natural$-concave function f also satisfies submodularity (Murota and Shioura [20]): $f(x) + f(y) \geq f(x \wedge y) + f(x \vee y)$ for all $x, y \in \operatorname{dom} f$, where $x \wedge y$ and $x \vee y$ are the vectors whose e-components $(x \wedge y)(e)$ and $(x \vee y)(e)$ are, respectively, $\min\{x(e), y(e)\}$ and $\max\{x(e), y(e)\}$ for all $e \in E$.

An $\mathrm{M}^\natural$-concave function satisfies the following two properties which are natural generalizations of the gross substitutability and single improvement property discussed in Kelso and Crawford [15] and Gul and Stacchetti [13].

(GS) For any $p, q \in \mathbf{R}^E$ and any $x \in \operatorname{argmax} f[-p]$ such that $p \leq q$ and $\operatorname{argmax} f[-q] \neq \emptyset$, there exists $y \in \operatorname{argmax} f[-q]$ such that $y(e) \geq x(e)$ for all $e \in E$ with $p(e) = q(e)$.

(SI) For any $p \in \mathbf{R}^E$ and any $x, y \in \operatorname{dom} f$ with $f[-p](x) < f[-p](y)$,

$$f[-p](x) < \max_{e \in \operatorname{supp}^+(x-y) \cup \{0\}} \max_{e' \in \operatorname{supp}^-(x-y) \cup \{0\}} f[-p](x - \chi_e + \chi_{e'}).$$

Here E denotes the set of indivisible commodities, $p \in \mathbf{R}^E$ a price vector of commodities, $x \in \mathbf{Z}^E$ a consumption of commodities, and $f(x)$ a monetary valuation for x. The above conditions are interpreted as follows. Condition (GS) says that when each price increases or remains the same, the consumer wants a consumption such that the numbers of the commodities whose prices remain the same do not decrease. Condition (SI) guarantees that the consumer can bring consumption x closer to any better consumption y by changing the consumption of one or two commodities. The equivalence between gross substitutability and the single improvement condition for set functions was first pointed out by Gul and Stacchetti [13], and the equivalence between the single improvement condition and $\mathrm{M}^\natural$-concavity for set functions was by Fujishige and Yang [11]. Moreover, $\mathrm{M}^\natural$-concavity can be characterized by these properties or their extensions under a natural assumption (see [2, 21] for details).

Fujishige and Tamura [9] showed that an $\mathrm{M}^\natural$-concave function satisfies the following properties.

(S1) Let $z_1, z_2 \in \mathbf{Z}^E$ be such that $z_1 \geq z_2$, $\operatorname{argmax}\{f(y) \mid y \leq z_1\} \neq \emptyset$, and $\operatorname{argmax}\{f(y) \mid y \leq z_2\} \neq \emptyset$. For any $x_1 \in \operatorname{argmax}\{f(y) \mid y \leq z_1\}$, there exists x_2 such that

$$x_2 \in \operatorname{argmax}\{f(y) \mid y \leq z_2\} \quad \text{and} \quad z_2 \wedge x_1 \leq x_2.$$

(S2) Let $z_1, z_2 \in \mathbf{Z}^E$ be such that $z_1 \geq z_2$, $\operatorname{argmax}\{f(y) \mid y \leq z_1\} \neq \emptyset$, and $\operatorname{argmax}\{f(y) \mid y \leq z_2\} \neq \emptyset$. For any $x_2 \in \operatorname{argmax}\{f(y) \mid y \leq z_2\}$, there exists x_1 such that

$$x_1 \in \operatorname{argmax}\{f(y) \mid y \leq z_1\} \quad \text{and} \quad z_2 \wedge x_1 \leq x_2.$$

Suppose that E denotes a set of workers, $y \in \mathbf{Z}^E$ a labor allocation representing labor times of the workers, $f(y)$ a valuation of a firm for labor allocation y, and $z_1, z_2 \in \mathbf{Z}^E$ vectors representing capacities of labor times. Property (S1) says that when each capacity decreases or remains the same, there exists an optimal labor allocation such that for every worker, if his/her original labor time is less than or equal to the new capacity, then the labor time increases or remains the same, and if the original labor time is greater than the new capacity, then the labor time becomes equal to the new capacity. On the other hand, (S2) says that when each capacity increases or remains the same, there exists an optimal labor allocation such that for every worker, if his/her original labor time is less than its original capacity, then the labor time decreases or remains the same. Hence, (S1) and (S2) imply that a choice function $C : \mathbf{Z}^E \to 2^{\operatorname{dom} f}$ defined by $C(z) = \operatorname{argmax}\{f(y) \mid y \leq z\}$ satisfies "substitutability," where $2^{\operatorname{dom} f}$ denotes the set of all subsets of $\operatorname{dom} f$. In fact, if $\operatorname{dom} f \subseteq \{0,1\}^E$ then (S1) and (S2) are equivalent to conditions of substitutability in Sotomayor [24, Definition 4], and if C always gives a singleton (in this case (S1) and (S2) are equivalent), then (S1) and (S2) are equivalent to persistence (substitutability) in Alkan and Gale [1]. Farooq and Tamura [6] showed that $f : \{0,1\}^E \to \mathbf{R} \cup \{-\infty\}$ is $\mathrm{M}^\natural$-concave if and only if $f[-p]$ satisfies (S1) for all $p \in \mathbf{R}^E$, and that f is $\mathrm{M}^\natural$-concave if and only if $f[-p]$ satisfies (S2) for all $p \in \mathbf{R}^E$. Farooq and Shioura [5] extended these characterizations to the case where $\operatorname{dom} f$ is bounded.

3 Model description

We consider a two-sided market consisting of disjoint sets P and Q of agents, in which an agent in P may be called a worker and one in Q a firm. Each worker $i \in P$ can supply multi-units of labor time, and each firm $j \in Q$ can employ workers with multi-units of labor time and pay a salary to worker i if j hires i. We assume possibly bounded side payments, i.e., each pair (i, j) may have lower and upper bounds on a salary per unit of labor time. We also assume that the valuation of each agent $k \in P \cup Q$ on labor allocations is described by a function in monetary terms. We will examine two concepts of stability, namely, *pairwise stability* and *pairwise strict stability*, in a market where the payoff function of each agent is quasi-linear. We will give precise definitions of the two concepts later.

First we describe our model mathematically. Let $E = P \times Q$, i.e., the set of all ordered pairs (i, j) of agents $i \in P$ and $j \in Q$. Also define $E_{(i)} = \{i\} \times Q$ for all $i \in P$ and $E_{(j)} = P \times \{j\}$ for all $j \in Q$. Denoting by $x(i, j)$ the number of units of labor time for which j hires i, we represent a labor allocation by vector $x = (x(i, j) \mid (i, j) \in E) \in \mathbf{Z}^E$. We express lower and upper bounds of salaries per unit of labor time by two vectors $\underline{\pi} \in (\mathbf{R} \cup \{-\infty\})^E$ and $\overline{\pi} \in (\mathbf{R} \cup \{+\infty\})^E$ with $\underline{\pi} \leq \overline{\pi}$. For each $y \in \mathbf{R}^E$ and $k \in P \cup Q$, we denote by $y_{(k)}$ the restriction of y on $E_{(k)}$. For example, for a labor allocation $x \in \mathbf{Z}^E$, $x_{(k)}$ represents the labor allocation of agent k with respect to x. We assume that the valuation of each worker on a labor allocation is determined only by how many units of labor time he/she works in the firms, and that the valuation of each firm is determined only by how many units of labor time it hires the workers. That is, the value function f_k of each $k \in P \cup Q$ is defined on $E_{(k)}$ as $f_k : \mathbf{Z}^{E_{(k)}} \to \mathbf{R} \cup \{-\infty\}$. We assume that each value function f_k satisfies the following assumption:

(A) dom f_k is bounded and hereditary, and has $\mathbf{0}$ as the minimum point,

where heredity means that for any $y, y' \in \mathbf{Z}^{E_{(k)}}$, $\mathbf{0} \leq y' \leq y \in \mathrm{dom}\, f_k$ implies $y' \in \mathrm{dom}\, f_k$. The boundedness of effective domains implies that each value function is implicitly imposed on firm's budget constraint or worker's constraint on labor time. The heredity of effective domains implies that each agent can arbitrarily decrease related labor time (before contract) without any permission from the partner.

A vector $x \in \mathbf{Z}^E$ is called a *feasible allocation* if $x_{(k)} \in \mathrm{dom}\, f_k$ for all $k \in P \cup Q$, and a vector $s \in \mathbf{R}^E$ is called a *feasible salary vector* if $\underline{\pi}(i, j) \leq s(i, j) \leq \overline{\pi}(i, j)$ for all $(i, j) \in E$. We call a pair (x, s) of a feasible allocation $x \in \mathbf{Z}^E$ and a feasible salary vector $s \in \mathbf{R}^E$ an *outcome*.

The payoff functions of agents on outcomes are defined as follows: the payoff of worker $i \in P$ on (x, s) is given by $f_i[+s_{(i)}](x_{(i)}) = f_i(x_{(i)}) + \sum_{j \in Q} s(i, j)x(i, j)$, i.e., the value of i on x plus the income from the firms that hire worker i, and the payoff of firm $j \in Q$ on (x, s) is given by $f_j[-s_{(j)}](x_{(j)}) = f_j(x_{(j)}) - \sum_{i \in P} s(i, j)x(i, j)$, i.e., the value of firm j on x minus the payments to the workers that firm j hires.

An outcome (x, s) is said to satisfy *incentive constraints* if each agent has no incentive to unilaterally decrease the current units x of labor time at the current salary agreements s, that is, if it satisfies

$$f_i[+s_{(i)}](x_{(i)}) = \max\{ f_i[+s_{(i)}](y) \mid y \leq x_{(i)} \} \qquad (\forall i \in P), \tag{1}$$

$$f_j[-s_{(j)}](x_{(j)}) = \max\{ f_j[-s_{(j)}](y) \mid y \leq x_{(j)} \} \qquad (\forall j \in Q). \tag{2}$$

Next we define pairwise (un)stability formally. For any $s \in \mathbf{R}^E$, $\alpha \in \mathbf{R}$, $i \in P$, and $j \in Q$, let $(s^{-j}_{(i)}, \alpha)$ be defined as the vector obtained from $s_{(i)}$ by replacing its (i, j)-component by α, and $(s^{-i}_{(j)}, \alpha)$ be similarly defined. We say that an outcome (x, s) is *pairwise unstable* if it does not satisfy incentive constraints or there exist $i \in P$, $j \in Q$, $\alpha \in [\underline{\pi}(i, j), \overline{\pi}(i, j)]$, $y' \in \mathbf{Z}^{E_{(i)}}$ and $y'' \in \mathbf{Z}^{E_{(j)}}$ such that

$$f_i[+s_{(i)}](x_{(i)}) < f_i[+(s^{-j}_{(i)}, \alpha)](y'), \tag{3}$$

$$y'(i, j') \leq x(i, j') \qquad (\forall j' \in Q \setminus \{j\}), \tag{4}$$

$$f_j[-s_{(j)}](x_{(j)}) < f_j[-(s^{-i}_{(j)}, \alpha)](y''), \tag{5}$$

$$y''(i', j) \leq x(i', j) \qquad (\forall i' \in P \setminus \{i\}), \tag{6}$$

$$y'(i, j) = y''(i, j). \tag{7}$$

For some feasible salary α between i and j, conditions (3) and (4) say that worker i can strictly increase his/her payoff by changing the current units of labor time with j without increasing units of labor time with other firms, and (5) and (6) say that firm j can also strictly increase its payoff by changing the current units of labor time with i without increasing units of labor time with other workers. Moreover, condition (7) requires that i and j agree on units of labor time between them. An outcome (x, s) is called *pairwise stable* if it is not pairwise unstable.

We also consider a stronger pairwise stability, which might be regarded as artificial but plays an important role in showing the existence of a pairwise stable outcome. We say that an outcome (x, s) is *pairwise quasi-unstable* if it does not satisfy incentive constraints or there exist $i \in P$, $j \in Q$, $\alpha \in [\underline{\pi}(i, j), \overline{\pi}(i, j)]$, $y' \in \mathbf{Z}^{E_{(i)}}$ and $y'' \in \mathbf{Z}^{E_{(j)}}$ satisfying (3)$\sim$(6) (but not necessarily (7)). Without requirement (7), conditions (3)$\sim$(6) mean that i and j have an incentive to deviate from (x, s)

without consent to possible labor time between them. An outcome (x,s) is called *pairwise strictly stable* if it is not pairwise quasi-unstable. Since a pairwise unstable outcome is pairwise quasi-unstable, a pairwise strictly stable outcome is pairwise stable. An outcome (x,s) is pairwise strictly stable if and only if (1) and (2) hold and for all $i \in P$, $j \in Q$ and $\alpha \in \mathbf{R}$ with $\underline{\pi}(i,j) \le \alpha \le \overline{\pi}(i,j)$,

$$f_i[+s_{(i)}](x_{(i)}) \ge \max\{ f_i[+(s_{(i)}^{-j},\alpha)](y) \mid y(i,j') \le x(i,j'),\ \forall j' \ne j\}, \tag{8}$$

or

$$f_j[-s_{(j)}](x_{(j)}) \ge \max\{ f_j[-(s_{(j)}^{-i},\alpha)](y) \mid y(i',j) \le x(i',j),\ \forall i' \ne i\}. \tag{9}$$

Conditions (8) and (9) is equivalent to that for each pair $(i,j) \in E$ and each feasible salary between them, both i and j cannot strictly increase their payoffs without increasing labor times with other partners.

The next example illustrates a gap between pairwise stability and pairwise strict stability.

Example 3 Let us consider the case where $E = \{(i,j)\}$ (a singleton),

$$f_i(x) = \begin{cases} x & \text{if } x \in \{0,1,2\} \\ -\infty & \text{otherwise} \end{cases} \qquad (\forall x \in \mathbf{Z}),$$

$$f_j(x) = \begin{cases} x & \text{if } x \in \{0,1,2,3\} \\ -\infty & \text{otherwise} \end{cases} \qquad (\forall x \in \mathbf{Z}),$$

and $\underline{\pi}(i,j) = 0$ and $\overline{\pi}(i,j) = 1/4$. In this case, an outcome $(x,s) = (2,0)$ is not pairwise strictly stable, because $f_i(2) < f_i[+\varepsilon](2)$ and $f_j(2) < f_j[-\varepsilon](3)$ for all $\varepsilon \in (0,1/4]$. However, the outcome is pairwise stable. On the other hand, an outcome $(x,s) = (2,1/4)$ is pairwise strictly stable (and hence, pairwise stable). ∎

The concept of a pairwise strictly stable outcome may be regarded as artificial but, as can be seen from Lemma 4, pairwise strict stability coincides with pairwise stability in some useful special cases: (i) salaries are constant, and (ii) the effective domains of value functions are sets of $\{0,1\}$-vectors. These two cases comprise many known existing models such as the marriage model, the assignment model, and an extension [3] of the marriage model with $\mathrm{M}^\natural$-concave value functions on $\mathbf{Z}^E$.

Lemma 4 If f_k $(k \in P \cup Q)$ are $\mathrm{M}^\natural$-concave functions satisfying (A) and if one of the following conditions

(i) $\underline{\pi} = \overline{\pi}$,

(ii) $\mathrm{dom}\, f_k \subseteq \{0,1\}^{E_{(k)}}$ for all $k \in P \cup Q$

holds, then any pairwise stable outcome is pairwise strictly stable.

Although the concepts of pairwise stability and pairwise strict stability are different in our general model, we have the following theorem.

Theorem 5 Assume that f_k is an $\mathrm{M}^\natural$-concave function satisfying (A) for each $k \in P \cup Q$. If (x,s) is a pairwise stable outcome in our model, then there exists a feasible salary vector s' such that (x,s') is a pairwise strictly stable outcome.

Hence, if we call a feasible allocation x *pairwise* (*strictly*) *stable* if there exists a feasible salary vector s such that (x,s) is pairwise (strictly) stable, then there is no gap between the two concepts of pairwise stability and pairwise strict stability in terms of allocations.

The following is our main theorem that for $\mathrm{M}^\natural$-concave value functions there exists a pairwise strictly stable outcome and hence a pairwise stable outcome in our model.

Theorem 6 For $\mathrm{M}^\natural$-concave functions f_k $(k \in P \cup Q)$ satisfying (A) and for vectors $\underline{\pi} \in (\mathbf{R} \cup \{-\infty\})^E$ and $\overline{\pi} \in (\mathbf{R} \cup \{+\infty\})^E$ with $\underline{\pi} \le \overline{\pi}$, there exists a pairwise strictly stable outcome (x,s), and hence, there exists a pairwise stable outcome. Moreover, if f_k $(k \in P \cup Q)$ are integer-valued on their effective domains, $\underline{\pi} \in (\mathbf{Z} \cup \{-\infty\})^E$, and $\overline{\pi} \in (\mathbf{Z} \cup \{+\infty\})^E$, then the above s can be chosen from $\mathbf{Z}^E$.

To show Theorem 6, we give an alternative characterization of a pairwise strictly stable outcome. (Note that by Theorem 5, the following theorem also gives a characterization of a pairwise stable allocation.)

Theorem 7 Assume that f_k is an $\mathrm{M}^\natural$-concave function satisfying (A) for each $k \in P \cup Q$. Let x be a feasible allocation. There exists a feasible salary vector s forming a pairwise strictly stable outcome (x,s) if and only if there exist $p \in \mathbf{R}^E$, $z_P = (z_{(i)} \mid i \in P) \in (\mathbf{Z} \cup \{+\infty\})^E$, and $z_Q = (z_{(j)} \mid j \in Q) \in (\mathbf{Z} \cup \{+\infty\})^E$ such that

$$x_{(i)} \in \arg\max\{ f_i[+p_{(i)}](y) \mid y \le z_{(i)}\} \qquad (\forall i \in P), \tag{10}$$

$$x_{(j)} \in \arg\max\{ f_j[-p_{(j)}](y) \mid y \le z_{(j)}\} \qquad (\forall j \in Q), \tag{11}$$

$$\underline{\pi} \le p \le \overline{\pi}, \tag{12}$$

$$e \in E,\ z_P(e) < +\infty \ \Rightarrow\ p(e) = \underline{\pi}(e),\ z_Q(e) = +\infty, \tag{13}$$

$$e \in E,\ z_Q(e) < +\infty \ \Rightarrow\ p(e) = \overline{\pi}(e),\ z_P(e) = +\infty. \tag{14}$$

Moreover, for any x, p, z_P, and z_Q satisfying the above conditions, (x,p) is a pairwise strictly stable outcome.

Consider the case where $z_P(i,j) = +\infty$ and $z_Q(i,j) < +\infty$. Condition (10) implies that worker i has no incentive to increase $x(i,j)$ at the current salary. If firm j could strictly increase its payoff by increasing $x(i,j)$ at the current salary, then j would try to increase the salary of worker i to give worker i incentive to increase $x(i,j)$. Condition (14), however, implies that firm j is in an extreme situation where firm j cannot increase the current i's salary any more, i.e., $p(i,j) = \overline{\pi}(i,j)$, and that firm j must give up increasing $x(i,j)$ (and hence $z_Q(i,j)$ is put to be a finite value). Analogously, when $z_P(i,j) < +\infty$ and $z_Q(i,j) = +\infty$, Conditions (11) and (13) imply that if worker i must give up increasing $x(i,j)$, then firm j has no incentive to increase $x(i,j)$ at the current salary and i is in an extreme situation where worker i cannot decrease his/her current salary to give firm j incentive to hire more units of labor time $x(i,j)$. It is of importance that (10)$\sim$(14) give a decentralized characterization of a pairwise (strictly) stable allocation. That is, given appropriate vectors p, z_P and z_Q, a pairwise (strictly) stable allocation can be obtained by individually maximizing each agent's payoff.

To prove our main theorem (Theorem 6), it is convenient to use two aggregated $\mathrm{M}^\natural$-concave functions on $\mathbf{Z}^E$, one for each of P and Q. Let us define f_P and f_Q by

$$f_P(x) = \sum_{i \in P} f_i(x_{(i)}), \quad f_Q(x) = \sum_{j \in Q} f_j(x_{(j)}) \qquad (\forall x \in \mathbf{Z}^E). \tag{15}$$

Since $E_{(i)}$ and $E_{(i')}$ are disjoint for all $i, i' \in P$ with $i \neq i'$, function f_P is $\mathrm{M}^\natural$-concave if all functions f_i ($i \in P$) are $\mathrm{M}^\natural$-concave. Similarly, f_Q is $\mathrm{M}^\natural$-concave if all functions f_j ($j \in Q$) are. Moreover, the following lemma obviously holds.

Lemma 8 Condition (10) holds if and only if $x \in \arg\max\{ f_P[+p](y) \mid y \leq z_P \}$. Condition (11) holds if and only if $x \in \arg\max\{ f_Q[-p](y) \mid y \leq z_Q \}$.

Furthermore, Assumption (A) is rewritten in terms of f_P and f_Q as:

(A′) Effective domains $\mathrm{dom}\, f_P$ and $\mathrm{dom}\, f_Q$ are bounded and hereditary, and have the common minimum point $\mathbf{0} \in \mathbf{Z}^E$.

By Theorem 7 and Lemma 8, Theorem 6 is a direct consequence of the following theorem.

Theorem 9 For $\mathrm{M}^\natural$-concave functions $f_P, f_Q : \mathbf{Z}^E \to \mathbf{R} \cup \{-\infty\}$ satisfying (A′) and for vectors $\underline{\pi} \in (\mathbf{R} \cup \{-\infty\})^E$ and $\overline{\pi} \in (\mathbf{R} \cup \{+\infty\})^E$ with $\underline{\pi} \leq \overline{\pi}$, there exist $x \in \mathbf{Z}^E$, $p \in \mathbf{R}^E$, and $z_P, z_Q \in (\mathbf{Z} \cup \{+\infty\})^E$ such that

$$x \in \arg\max\{ f_P[+p](y) \mid y \leq z_P \}, \tag{16}$$
$$x \in \arg\max\{ f_Q[-p](y) \mid y \leq z_Q \}, \tag{17}$$
$$\underline{\pi} \leq p \leq \overline{\pi}, \tag{18}$$
$$e \in E,\ z_P(e) < +\infty \ \Rightarrow\ p(e) = \underline{\pi}(e),\ z_Q(e) = +\infty, \tag{19}$$
$$e \in E,\ z_Q(e) < +\infty \ \Rightarrow\ p(e) = \overline{\pi}(e),\ z_P(e) = +\infty. \tag{20}$$

Moreover, if f_P and f_Q are integer-valued on their effective domains, $\underline{\pi} \in (\mathbf{Z} \cup \{-\infty\})^E$, and $\overline{\pi} \in (\mathbf{Z} \cup \{+\infty\})^E$, then the above p can be chosen from $\mathbf{Z}^E$.

References

[1] A. Alkan and D. Gale, *Stable schedule matching under revealed preference*, J. Econom. Theory **112** (2003), 289–306.

[2] V. Danilov, G. Koshevoy, and C. Lang, *Gross substitution, discrete convexity, and submodularity*, Discrete Appl. Math. **131** (2003), 283–298.

[3] A. Eguchi, S. Fujishige, and A. Tamura, *A generalized Gale-Shapley algorithm for a discrete-concave stable-marriage model*, Algorithms and Computation: 14th International Symposium, ISAAC2003, LNCS 2906 (T. Ibaraki, N. Katoh, and H. Ono, eds.), Springer-Verlag, Berlin, 2003, pp. 495–504.

[4] K. Eriksson, and J. Karlander, *Stable matching in a common generalization of the marriage and assignment models*, Discrete Math. **217** (2000), 135–156.

[5] R. Farooq and A. Shioura, *A note on the equivalence between substitutability and $M^\natural$-convexity*, Pacific J. Optim. **1** (2005), 243–252.

[6] R. Farooq and A. Tamura, *A new characterization of $M^\natural$-convex set functions by substitutability*, J. Oper. Res. Soc. Japan **47** (2004), 18–24.

[7] T. Fleiner, *A matroid generalization of the stable matching polytope*, Integer Programming and Combinatorial Optimization: 8th International IPCO Conference, LNCS 2081 (B. Gerards and K. Aardal, eds.), Springer-Verlag, Berlin, 2001, pp. 105–114.

[8] T. Fleiner, *A fixed point approach to stable matchings and some applications*, Math. Oper. Res. **28** (2003), 103–126.

[9] S. Fujishige and A. Tamura, *A general two-sided matching market with discrete concave utility functions*, RIMS Preprint 1401, Kyoto University, 2003, `http://www.kurims.kyoto-u.ac.jp/home_page/preprint/list/`.

[10] S. Fujishige and A. Tamura, *A two-sided discrete-concave market with possibly bounded side payments: An approach by discrete convex analysis*, RIMS Preprint 1470, Kyoto University, 2004, `http://www.kurims.kyoto-u.ac.jp/home_page/preprint/list/`.

[11] S. Fujishige and Z. Yang, *A note on Kelso and Crawford's gross substitutes condition*, Math. Oper. Res. **28** (2003), 463–469.

[12] D. Gale and L. S. Shapley, *College admissions and the stability of marriage*, Amer. Math. Monthly **69** (1962), 9–15.

[13] F. Gul and F. Stacchetti, *Walrasian equilibrium with gross substitutes*, J. Econom. Theory **87** (1999), 95–124.

[14] M. Kaneko, *The central assignment game and the assignment markets*, J. Math. Econom. **10** (1982), 205–232.

[15] J. A. S. Kelso and V. P. Crawford, *Job matching, coalition formation, and gross substitutes*, Econometrica **50** (1982), 1483–1504.

[16] K. Murota, *Convexity and Steinitz's exchange property*, Adv. Math. **124** (1996), 272–311.

[17] K. Murota, *Discrete convex analysis*, Math. Programming **83** (1998), 313–371.

[18] K. Murota, *Discrete Convex Analysis*, Society for Industrial and Applied Mathematics, Philadelphia, 2003.

[19] K. Murota and A. Shioura, *M-convex function on generalized polymatroid*, Math. Oper. Res. **24** (1999), 95–105.

[20] K. Murota and A. Shioura *Relationship of M-/L-convex functions with discrete convex functions by Miller and by Favati–Tardella*, Discrete Appl. Math. **115** (2001), 151–176.

[21] K. Murota and A. Tamura, *New characterizations of M-convex functions and their applications to economic equilibrium models with indivisibilities*, Discrete Appl. Math. **131** (2003), 495–512.

[22] A. E. Roth and M. A. O. Sotomayor, *Stable outcomes in discrete and continuous models of two-sided matching: A unified treatment*, Rev. Econom. **16** (1996), 1–24.

[23] L. S. Shapley and M. Shubik, *The assignment game I: The core*, Internat. J. Game Theory **1** (1972), 111–130.

[24] M. Sotomayor, *Three remarks on the many-to-many stable matching problem: Dedicated to David Gale on his 75th birthday*, Math. Social Sci. **38** (1999), 55–70.

[25] M. Sotomayor, *Existence of stable outcomes and the lattice property for a unified matching market*, Math. Social Sci. **39** (2000), 119–132.

Intersection of Random Walks on hierarchical structures*

ANDRÁS TELCS

Department of Computer Science and Information Theory,
Budapest University of Technology and Economics
telcs@szit.bme.hu

Abstract: Hierarchical structures are introduced which serve as discrete counterpart of fractofolds. The paper collects illustrative examples of the application of transient potential theory on this structures.

1) Intersection of Random Walks on hierarchical structures

2) Intersection of RWs with the set of prime numbers.

3) Discrete Frostmann Theorem on hierarchical structures.

1 Introduction and results

This paper generalizes some well known result of Random Walks on the integer lattice Z^d to nearest neighbor random walks on graphs with Greenian index and on hierarchical structures. Particular hierarchical structures are heavily studied in the last two decades. Depending on the approach they called fractals, nested fractals, post critically finite self similar sets etc. (c.f. [8],[4]). This paper introduces a general class of discrete hierarchical structures which contains all classical fractal type graphs, like the variation of the Sierpinski triangular graph, Vicsek tree etc.. They serve as discrete counterparts of continuous structures, the fractafolds recently introduced by Strichartz (c.f. [11],[12]). The paper contains some illustrative example of the application of transient potential theory on hierarchical structures.

Definition 1 Let $G = (V,E)$ a connected, locally finite, infinite graph on the countable vertex set V. For $x,y \in E$ $d(x,y)$ is the shortest path distance on G, denote balls by

$$B_N = B_{xN} = \{y \in V | d(x,y) \leq N\}$$

for $N \geq 0, x \in V$,

$$S_{x,N} = \{y \in V | d(x,y) = N\},$$

Vertex degree is assumed to be finite, i.e., there is a $D > 1$ such that

$$d(x) = |B_{x,1}| - 1 < D$$

for all $x \in V$.

Definition 2 The random walk (or Markov chain) is defined by the one step transition probabilities

$$P(x.y) = \frac{I((x,y) \in E)}{d(x)}$$

where I is the indicator function and $x,y \in V$.

Definition 3 The Green's function for transient random walks is simply

$$G(x,y) = \sum_n P_n(x,y)$$

Definition 4 If there is an index set I and two series $a_i, b_i : i \in I$ then

$$a_i \asymp b_i \tag{1}$$

will be used to denote that there is a constant $c > 1$, for which

$$\frac{1}{c} \leq \frac{a_i}{b_i} \leq c \text{ for all } i \in I. \tag{2}$$

We introduce the Greenian index $\alpha = \alpha(d)$ of random walks with respect to an additive constant d (which is typically the dimension of the graph). This is done in order to maintain consistency with the definition in [9] and also with the notion of $\alpha-$stabile random walks [1].

Definition 5 A random walk has Greenian index $\alpha = \alpha(d)$ if

$$G(x,y) \asymp d(x,y)^{\alpha-d}.$$

Remark 6 It is common to introduce the transition probabilities via conductance function $c : E \to (\frac{1}{a}; a)$ with $1 < a < \infty$ by

$$P(x,y) = \frac{c(x,y)}{c(x)} \tag{3}$$

where $c(x,y) = 0$ if $(x,y) \notin E$ and $c(x) = \sum_{y \in V} c(x,y)$. All the result of the paper can be generalized to this setting but for the seek of simplicity $c \equiv 1$ is assumed.

The defined random walk X_n is a Markov chain starting in a given point $x = X_0 \in V$ and evolving by (3) and reversible with respect to $c(x)$.

1.1 Intersection of Random Walks

The first two topics date back to the classical results of A. Dvoretsky, P. Erdős, S. Kakutan, S.J. Taylor and several other authors on the intersection of two random walks. Recent studies on the grid and in the Euclidean space done by I. Benjamini, G. Lawler, R. Pemantle, Y. Peres (for basic references see [9]). The concept of intersection equivalence and Lyons theorem [7] gave a new impulse to the investigations. The first result generalize Peres' result (Theorem 1.2., 1.3 [9]) to random walk intersection on fractals. A wide class of graphs called buildings will be defined. The graphs will posses some hierarchical structure and their class will cover all finitely ramified fractals. e.g. skeleton of nested fractals (c.f. [6]), or p.c.f. self-similar sets (c.f.[8].)

Intersection of random walk on the integer lattice with Greenian index is investigated by Peres [9], and found that

Proposition 7 Let $G = Z^d, d > 2$. If two random walks X_n and X'_n on G with Greenian index α, β started uniformly in $B_{x,N}$, then

$$P(X_n = X'_m \in B_{x,N} \text{ for } n,m \geq 0) \asymp \begin{cases} 1 \text{ if } \alpha+\beta > d \\ \frac{1}{\log N} \text{ if } \alpha+\beta = d \\ N^{\alpha+\beta-d} \text{ if } \alpha+\beta < d \end{cases}.$$

The first result of this paper generalizes this to building graphs. Building graphs are graphs with finite cut-sets, i.e., any vertex can be cut off the infinity with a finite set of vertices.

The intuitive construction of building graphs is the following. Consider a finite set of finite connected graphs, these are the zero level graphs. The vertex set of the graphs are partitioned into internal and boundary points. The construction is recursive, starts with an arbitrary zero level graph. The next level is built on it replacing the vertices with graphs. Two boundary points of two added graphs are common if the original vertices where joined by an edge. This common vertices and the internal vertices of the graphs form the internal vertices of the new one. The union of the remaining boundary vertices form the new boundary. The resulted graph is a level one building graph. If we have constructed already the k-th level. The k+1 level formed in the same way; a zero level graph is chosen and each vertex replaced with an arbitrary k level graph. The definition of the coincidences and the border works as above. For the exact definition see Section 2.

It is easy to see that all finitely ramified fractal graphs can be constructed in this way and many other.

The main result of the paper is the following. For the elaborated definitions see Section 2.

Theorem 8 Let G be a building graph. Assume that the diameter of the $k-$th level grows as

$$diam(V_k) \asymp q^k$$

and the dimension is $d > 0$

$$|V_k| \asymp \left(q^k\right)^d.$$

If two random walks on G, X_n and X'_n with Greenian index α, β started uniformly in $B_{x,N}$, then

$$P(X_n = X'_m \in B_{x,N} \text{ for } n,m \geq 0) \asymp \begin{cases} 1 \text{ if } \alpha+\beta > d \\ \frac{1}{\log N} \text{ if } \alpha+\beta = d \\ N^{\alpha+\beta-d} \text{ if } \alpha+\beta < d \end{cases}.$$

1.2 Random Dirichlet problem

The original problem came from Dirichlet. Is there a prime number in any arithmetic series? The answer is well-known to be affirmative. J.G. Székely and P. Erdős suggested the random walk counterpart of the problem. Let us consider a transient random walk X_n with positive steps on the natural numbers N. Does the random walk visit the set of primes infinitely often with probability one? In other words is the set of primes recurrent for any random walk? From [1] it follows that for random walk with Greenian index $\alpha(1)$ the answer is yes. To our best knowledge this is the first answer to the problem in this generality.

Theorem 9 For any aperiodic random walk on $\mathbb{N}$ reflected in 0 of index $\alpha = \alpha(1)$, $0 < \alpha < 1$ the set of primes is recurrent.

The proof is simple. One can check that π, the set of primes, is a fractal subset of N with dimension 1 and the statement then follows from Corollary 15. (see definitions and the Corollary 15 in Section 1.3).

The condition ensures that the Green functions vary in a moderate way. This seems necessary, since similar regularity was used in [10]. The dimension notion is not enough sensitive to handle the case when G tends to zero very slowly. A partial result in the opposite direction can be obtained easily from the Borell-Cantelli Lemma.

If for $x > 1$

$$G(0,x) \leq \frac{\omega(x)}{x}$$

for a series $\omega(k)$ with

$$\sum_{k=2}^{\infty} \frac{\omega(k)}{\log(k)} < \infty,$$

then the set of primes is transient for the random walk corresponding to G.

1.3 Frostman Theorem on hierarchical graphs

The problem of this section is based on [1], where a new definition of fractals is given by introducing several dimensions of discrete subset of Z^d. Some major results of that paper can be generalize to buildings. Among others [1] defines the packing dimension $\dim_p(A)$ of a set $A \subset \mathbb{Z}^d$ and the Haussdorf dimension of it $\dim_H(A)$.

Definition 10 ([1]) $A \subset Z^d$ is regular fractal if

$$\dim_p(A) = \dim_H(A).$$

From this it follows that the naive dimension concept (upper and lower mass dimension)

$$\dim_{UM}(A) = \limsup_{x,n} \frac{\log|A \cap B_{x,N}|}{\log N}$$

$$\dim_{LM}(A) = \limsup_{x,n} \frac{\log|A \cap B_{x,N}|}{\log N}$$

coincide with the others; $\dim_H(A) = \dim_{UM}(A) = \dim_p(A)$ provided A is a regular fractal.

Definition 11 ([1]) $A \subset Z^d$ is strongly regular fractal of index d if for all $x \in A$

$$|A \cap B_{x,N}| \asymp N^d.$$

The same has sense for general graphs.

Definition 12 $G = (V,E)$, is a strongly regular fractal of index (or dimension) d if

$$|B_{x,N}| \asymp N^d.$$

From the definition it follows that BG graphs are strongly regular fractals. Along the arguments of [1] Frostman Lemma and Frostman Theorem can be deduced for building graphs as well. The capacity dimension can be defined as follows

Definition 13 Corresponding to [5] Proposition 8.29, the capacity of infinite sets for random walk with Greenian index α for $A \subset V$

$$Cap_\alpha^{(\infty)}(A) = \sup_{|X|<\infty} \{Cap_\alpha(A \cap X)\}$$

if the supremum is finite and

$$\dim_C(A) = \inf\{\alpha > 0 \, Cap_\alpha^{(\infty)}(A) = 0\}.$$

For the definition of $Cap_\alpha(.)$see (5).

Theorem 14 For any $A \subset V$ in a transient hierarchical graph, G,

$$\dim_C(A) = \dim_H(A).$$

This is the discrete version of the classical Frostman Theorem, which is shown by Barlow and Taylor for $V = Z^d$ but, in fact, the same proof works for any hierarchical graph. Here, as in [1], the capacity dimension uses the asymptotic capacity instead of the standard one which can not provide the needed property for infinite sets in the discrete case.

An important consequence of this result is the following.

Corollary 15 Let $d > 2$ is the dimension of the hierarchical graph G. If $A \subset V, \dim_H(A) > d-2$, then

$$\begin{aligned} \dim_H(A) &= \inf\{\alpha > 0 \; A \text{ is transient for random walks of index } d-\alpha\}, \\ \dim_H(A) &= \sup\{\alpha > 0 \; A \text{ is recurrent for random walks of index } d-\alpha\}. \end{aligned}$$

The proofs are step by step reproductions of [1] and hence omitted.

2 Additional definitions and Proof of Theorem 1.

The proof basically follows the method of [2] but needs some definitions and notations.

Definition 16 Bricks.

A finite, locally bounded, connected graph $G = (I \cup B, E)$ called brick if $I \cap B = \emptyset$, and there is a constant c such that

- 1.

$$\frac{\min\{d(x,y) : x,y \in B\}}{diam(V)} \geq c \tag{4}$$

where $diam(V) = \max\{d(x,y) : x,y \in V\}$ for $V = I \cup B$, B is the border, I the internal part, and 4 ensures that the border vertices are relatively far.

In the case of the pre-Sierpinsky gasket $c = 1$.

Definition 17 Building graphs (BG)

Construction of BG is the following.

Let $\mathcal{G}$ be a finite set of finite, connected graphs with maximum degree D. The hierarchical structure is defined recursively.

G_0 is a 1-level graph if $G_0 \in \mathcal{G}$ and is a brick

G_{k+1}, a $k+1-$ level graph, is constructed in the following way. An arbitrary $H_{k+1} = (W,E) \in \mathcal{G}$ is chosen and for all $i \in W$, a $G^{(i)}$ a $k-$ level brick graph is considered. For each $G^{(i)}$ denote $I^{(i)}$ the set of internal points and $B^{(i)}$ the boundary.

1. The cardinality of the boundary points of $G^{(i)}$ coincides with the degree of i.

$$\left|B^{(i)}\right| = d(i).$$

2. there is a map $\varphi^{(i)}$ from the edges joined to $i \in W$ to the boundary points of $B^{(i)}$ and two boundary points of $x \in B^{(i)}$ and $y \in B^{(j)}$ are identified if $(i,j) \in E, \varphi^{(i)}(i,j) = x, \varphi^{(j)}(j,i) = y$.

3. $$B^{(i)} \cap B^{(j)} = V^{(i)} \cap V^{(j)}$$

   and

   $$B^{(i)} \cap B^{(j)} = V^{(i)} \cap V^{(i)} \neq \emptyset \text{ if and only if } (i,j) \in E$$

   The vertices in the union of the intersections called cut-points or k+1-level cut-points, their set is denoted by C_{k+1},

4. $B_{k+1} \subset \cup_{i \in W} B^{(i)} \backslash C_{k+1}$,

5. the new graph is a brick again with $B = B_{k+1}$.

Definition 18 G is a building graphs (BG) with growth rate q if it is BG and the growth rate is controlled by

$$diam(V_{k+1}) \asymp q^{k+1}$$

and has dimension d if

$$|V_k| \asymp \left(q^k\right)^{dz}.$$

Definition 19 The same construction leads to hierarchical graphs (HG) if the assumption (4) is dropped.

1. The assumption that the elements of $\mathscr{G}$ are finite ensures that a HG (or a BG) is finitely ramified.
2. The definition ensures that the graph is locally finite.

The hierarchy of the BG simply correspond to an infinite rooted tree. The following notation will be useful. If, as above, G_{k+1} is built on some $G_k^{(i)}$, then $G_k^{(i)} \prec G_{k+1}$, for $k > 1$ and $\{x\} \prec G_1$ if $x \in V_1$.

There is a rooted tree $\Gamma_k = (W_k, E_k)$ and a bijection $\pi_k : W_k \to H_k$ such that

$$(u,v) \in E_k \text{ if and only if } (\pi_k(u) \prec \pi_k(v) \text{ or } \pi_k(u) \prec \pi_k(v)).$$

It is clear that $\pi_k(\rho) = G_k$ and for the leaf $u \in W$, there is a unique $x \in V_k$ such that $\pi(u) = \{x\}$.

From now on, the subscripts will be omitted if it is not confusing. Each leaf $\pi(u) = \{x\} : x \in V_k$ carries the unique path from the root to it and is denoted by $\overline{x}$. The set of leaves for a tree Γ will be denoted by $\delta\Gamma$.

Two different path have a last common point (farthest from the root) and it is denoted by $\overline{x} \wedge \overline{y}$.

If the function $p : E_k \to [0,1]$ assigns probabilities to the edges of Γ_k, the edge percolation of the tree is the remaining tree if any given edge (u,v) is removed with probability $1 - p(u,v)$ independently of the other edges. The resulting sub-tree is denoted by $\Gamma_k(p)$ and the set accessible leaves by $\delta\Gamma_k(p)$ which, as above, corresponds to the subset of rays $\Omega_k(p)$ and a subset of V_k. A particular percolation is defined. For $\pi(u) \prec \pi(v)$ let

$$p(u,v) = \frac{d(\pi(u))}{d(\pi(v))}.$$

where $d(.)$ denotes the diameter of the subgraph and for $0 < \beta < \infty$

$$p(u,v)^\beta = \left(\frac{d(\pi(u))}{d(\pi(v))}\right)^\beta.$$

This is in full correspondence with the percolation probability used in [2] and gives a simple formula for the probability of nondisconnection of a vertex of the tree.

If $u \in W$, the probability that u is connected to the root is

$$P_\beta(\rho \rightsquigarrow u) = \prod_{e \in \overline{u}} p(e) = \frac{1}{d(\pi(u))^\beta}.$$

Using the bijection, the percolated tree corresponds to sub-hierarchy of G and the image of the accessible leaves $u \in \delta\Gamma_k(p)$ to the random subset of V_k denoted by $Q_k(\beta)$.

The capacity notion is another key object of the result.

For a metric space, M, with metric ρ and a Borel function $K : M^2 \to [0,\infty\}$, the $K-$energy of a finite Borel measure μ is defined as

$$E_K(\mu) = \int\int K(x,y)d\mu(x)d\mu(y),$$

and the K-capacity of M is

$$Cap_K(M) = \left[\inf_{\mu(M)=1} E_K(\mu)\right]^{-1}. \tag{5}$$

If $K(x,y) \asymp \rho(x,y)^\alpha$ the notations E_α and $Cap_\alpha(M)$ are used.

Thereby all the necessary objects are available and the following propositions lead to Theorem 1. with slight modifications of the arguments given in [9]. Only the sketch of the proof is provided, pointing out where changes are needed.

Proposition 20 If X_n is a random walk of index $\alpha = \alpha(d)$ and X_0 is uniformly distributed on V_k, G has dimension d, then for any $\Lambda \subset V_k$

$$P(\exists n \geq 0 : X_n \in \Lambda) \asymp P(Q_k(\alpha - d) \cap \Lambda \neq \emptyset) \tag{6}$$

Proposition 21 If X_n is a random walk of index $\alpha = \alpha(d) < d$ on V_k and it is started with an initial distribution ν with mass on each vertices less than $C/|V_k|$, then for all set $\Lambda \subset V_k$

$$P_\nu(\exists n : X_n \in \Lambda) \asymp |V_k|^{\frac{\alpha}{d}-1} Cap_{d-\alpha}(\Lambda) \tag{7}$$

independently of k and the choice of Λ.

The next proposition (due to Lyons [7] and [9]) uses a distance of rays on the defined tree, Γ_k. If two leaves u, v or the corresponding rays $\overline{u}, \overline{v}$ are considered their distance is defined as

$$\rho(\overline{u}, \overline{v}) = \left(P(\rho \rightsquigarrow \overline{u} \wedge \overline{v})\right)^{-1}.$$

Proposition 22 If the Γ_k tree is percolated with index β, then

$$Cap_\beta(\delta\Gamma_k) \leq P_\beta(\rho \rightsquigarrow \delta\Gamma_k) \ \leq 2Cap_\beta(\delta\Gamma_k). \tag{8}$$

Finally, the connection between the capacity on the tree and on V_k is given by the following proposition.

Proposition 23 With the notations above,

$$Cap_\beta(\delta\Gamma_k) \asymp |V_k|^{-\frac{\beta}{d}} Cap_\beta(\pi\delta\Gamma_k). \tag{9}$$

The above propositions can be summarized in a corollary.

Corollary 24 For any $\Lambda \subset V_k$,

$$P(Q_k(\beta) \cap \Lambda \neq \emptyset) \asymp |V_k|^{-\frac{\beta}{d}} Cap_\beta(\Lambda). \tag{10}$$

PROOF:[Proof of Theorem 8] The proof is immediate from (6) and the independence of the edge percolations (using Lemma 2.2, 2.3) [9]);

$$\begin{aligned} P(\exists n, m \geq 0 : X_n = X_m \in V_k) &\asymp P(Q_k(\alpha - d) \cap Q_k(\beta - d) \neq \emptyset) \\ &= P(Q_k(\alpha + \beta - 2d) \neq \emptyset). \end{aligned}$$

□

Using (10,9,8) for $Q_k(\alpha + \beta - 2d)$ the problem is reduced to the percolation of a $q-$branching tree with percolation $p = q^{\alpha+\beta-2d}$. From the assumption that the hierarchy is $d-$ dimensional, i.e., $d(V_k) \asymp q^k$, and $|V_k| \asymp q^{dk}$, the mean of the $k-$th generation of the branching process is $q^{dk}(q^k)^{\alpha+\beta-2d}$.□

Proposition 20, 21, 22 and Corollary 24. comes as in [9], the proof of Proposition 23 needs minor modifications as follows.

PROOF:[Proof of Proposition 7] The energy of the boundary of a sub-tree can be rewritten as

$$E_k(\mu) = \sum_{x \in \delta\Gamma_k} \sum_{y \in \delta\Gamma_k} \rho^\beta(x,y)\mu(x)\mu(y)$$

$$= \sum_{\sigma \in W_k} P(\rho \rightsquigarrow \sigma)^{-\beta} \sum_{x \in \delta\Gamma_k} \sum_{y \in \delta\Gamma_k} I(\sigma = \overline{x} \wedge \overline{y})\mu(x)\mu(y) \asymp \sum_{\sigma \in W_k} P(\rho \rightsquigarrow \sigma)^{-\beta} \mu[\sigma]^2$$

where $\mu[\sigma]$ is the number of leaves disconnected by σ from the root.

Another useful fact has to be mentioned. By definition if G is a graph at the $i-th$ level of the tree Γ_k, its diameter $d(G) \asymp q^{k-i}$.

On the other hand, the energy of a measure with support of any subset $\Lambda \subset V_k$ has the form

$$\overline{E}_k(\mu) = \sum_{x \in V_k} \sum_{y \in V_k} d^{-\beta}(x,y)\mu(x)\mu(y). \tag{11}$$

Let us recall that $\rho(x,y) = P(\rho \rightsquigarrow \sigma)^{-1} = \frac{d(\pi\rho)}{d(\pi\sigma)}$, where $\sigma = x \wedge y$, then

$$E_k(\mu) \leq \overline{E}_k(\mu)$$

is straightforward. The opposite inequality is based on an observation which is the key of the proof as Peres has also pointed out. If $x \in Q_k$, $q^k \le d(x,y) < q^{k+1}$, then either $y \in Q_k$ or $y \in Q_k'$ which is at most l-th neighbor of Q_k (in the H_{k+1}) and the number of such bricks is bounded. The distance is $l \le \frac{q}{c_1 c_2}$, where the constants are coming from the definition of the BG. With a simple summation by part it follows that

$$E_k(\mu) \asymp \sum_{\sigma \in W_k} P(\rho \rightsquigarrow \sigma)^{-\beta} \mu[\sigma]^2.$$

Let us use (11) :

$$\overline{E}_k(\mu) = \sum_{x \in V_k} \sum_{y \in V_k} d^{-\beta}(x,y)\mu(x)\mu(y).$$

If L_i denotes the i-th level of Γ_k, then

$$\overline{E}_k(\mu) \le \sum_i \sum_{\sigma \in L_i} \sum_{\tau \in L_i} \sum_{x \in \sigma} \sum_{y \in \tau} d^{-\beta}(x,y)\mu(x)\mu(y) I\{d(\pi\sigma) \le d(x,y) < d(\pi\sigma)\}$$

$$\le c_4 \sum_i \sum_{\sigma \in L_i} \sum_{\tau \in L_i} d^{-\beta}(\sigma)\mu(\sigma)\mu(\tau) I\{d(\sigma,\tau) \le l\}$$

$$\le c_5 q^{-\beta k} \sum_i \sum_{\sigma \in L_i} \sum_{\tau \in L_i} P(\rho \rightsquigarrow \sigma)^{-\beta} \frac{1}{2}(\mu(\sigma)^2 + \mu(\tau) I\{d(\sigma,\tau) \le l\}$$

$$\le c_6 L^d q^{-\beta k} \sum_\sigma \sum_{\tau \in L_i} P(\rho \rightsquigarrow \sigma)^{-\beta} \mu(\sigma)^2$$

which leads finally to the relation:

$$E_k(\mu) \asymp q^{-\beta k} \overline{E}_k(\pi\mu).$$

This completes the proof of the propositions. □

References

[1] Barlow, M.T., Taylor, S.J., Defining Fractal Subsets of Z^d, Proc London, Math. Soc. (1991)

[2] Benjamini, I., Peres Y.,Tree-indexed random walks on groups and first passage percolation. Probab. Theory Rel. Fields. 98 (1994), 91-112.

[3] Doyl, Snell L. Doyle, P.G., Snell, J.L., Random walks and electric networks, Carus Math. Monographs, 22, Math. Assoc. Amer., Washington D.C., 1984

[4] Kigami J.,Analysis on Fractals, Cambridge Univ. Press, 2001

[5] Kemeny, J.G., Snell, J.L., Knapp, A.W., Denumerable Markov Chains, Springer 1976

[6] Lindstrøm, T., Random walks and Brownian Motion on Nested Fractals, Mem, Amer. Math, Soc. 420 (1990)

[7] Lyons, R., Random Walks and percolation on trees. Ann. Probab. 20, 1990, 2089-2116

[8] Kigami J., Harmonic calculus on limits of networks and its application to dendrites, J. Funct. Anal., 128 (1995) 48-86.

[9] Peres, Y., Intersection-equivalence of Brownian paths and certain branching processes, Comm. Math. Phys. 177 (1996), 417–434.

[10] Ruzsa, I.Z., Székely, J.G., Intersection of traces of random walks with fixed sets, Annals Probab. 10, 132-136

[11] Strichartz, R., Analysis on fractals, Notices AMS 46, Nov 1999, 1199-1208

[12] Fractafolds based on the Sierpinski gasket and their spectra, Trans. Amer. Math. Soc. 355 (2003), 4019-4043

[13] Telcs, A., Spectra of Graphs and Fractal Dimensions II., J. Theor Probability, 1995, 8. 77-96

Enumeration of Triangles Configuration in Cube Cutting

YOSHINORI TESHIMA

Integrated VCAD System Research Program, RIKEN (The Institute of Physical and Chemical Research), Wako, Saitama 351-0198, Japan
Interdisciplinary Institute of Science, Technology and Art, 2-10-6-B102 Kitahara, Asaka, 351-0036 Japan
kippoh@riken.jp

KIWAMU KASE

Integrated VCAD System Research Program, RIKEN (The Institute of Physical and Chemical Research), Wako, Saitama 351-0198, Japan
kiwamu@riken.jp

AKITAKE MAKINOUCHI

Integrated VCAD System Research Program, RIKEN (The Institute of Physical and Chemical Research), Wako, Saitama 351-0198, Japan
akitake@riken.jp

Abstract: The number of equivalence classes of triangles configuration for cube cutting was investigated in this paper. In Volume Computer-Aided Design (VCAD), the shape of three-dimensional objects is approximated in discrete cubic lattice. The intersection between the edge of a cube of a cubic lattice and three-dimensional shape is called cutting point. With our unique shape approximation method called Kitta cube, three-dimensional shape is approximated by triangles which are held in the cubes of a cubic lattice. The triangle is called cutting triangle and its vertices are cutting points. The combinatorial analysis of the cutting triangles is essential for better shape approximation. This paper describes a study on the enumeration of cutting triangles configuration for the Kitta cube which is applied in VCAD. The purpose of this paper is to provide mathematical foundations to the Kitta cube by enumerating the number of cutting triangle configurations using Pólya's theory of counting. We assume that the number of cutting points on one edge of the cube is at most unity. We enumerated the cutting triangles configuration by considering the symmetry of a rotation group and a reflection group. If symmetry is disregarded, there are $2^{220} \cong 1.685 \times 10^{66}$ different cutting triangle configurations. Our exact enumeration shows 3.510×10^{64} patterns as a result.

Keywords: cube cutting, cutting triangle, cutting point, Kitta cube, equivalence class, Pólya's theory of counting

1 Introduction

Volume CAD (VCAD), a next generation Computer-Aided Design (CAD), can retain not only the shape but also the physical attributes of three-dimensional objects [3][4]. The continuous shape of three-dimensional objects and the physical attribute are retained approximately in local cubes in discrete cubic lattices(Figure 1). A highly precise approximation method called *Kitta cube*[*](KC) was developed for VCAD. With KC, the intersection between a three-dimensional continuous shape and a cube is expressed as shown in Figure 2, where the approximate triangle is called *cutting triangle*, and the intersection between the surface and edge of the cube is called *cutting point*. An advantage of the KC method is that it has superior approximation accuracy to the marching cubes method[5], which is widely used to visualize three-dimensional field values in computer graphics.

Cube cutting in this paper means the arrangement of triangles in cubes as shown in Figure 2. The combinatorial analysis of cube cutting has never been attempted in spite of its importance for better shape approximation. The main purpose of our series of papers is therefore to investigate the combinatorial features of KC. In the present paper, we focused on the enumeration for approximate triangles (= cutting triangles).

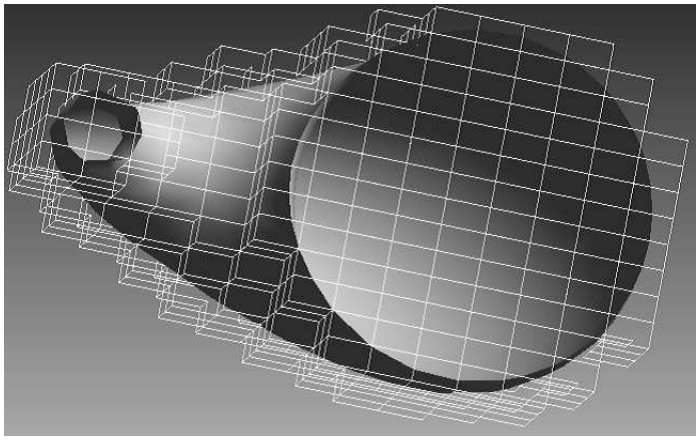
The enumeration for vertices (= cutting points) configuration of triangles was investigated[8][9]. The enumeration results showed 144 patterns for the KC. The results also showed that the set of 14 patterns of cutting points configurations for the MC is a subset of the set of 144 patterns for the KC. Therefore the KC are more expressive than the MC.

[*]In Japanese, Kitta means to cut.

For two-dimensional Kitta cubes, the information measure of simple diagrams constructed by a lot of segments was evaluated[2]. The information measure directly indicates a possibility of condensing of the data. The results also give some information about a length of segments adequate for approximate description of diagrams. For example, the segment length to get stable information measure of a semicircle can be considered as smaller than 0.05 times a radius of the semicircle.

In this paper, we calculate how many equivalence classes there are in all configurations using Pólya's theory of counting[7][6], which is useful for enumeration under group action. In most applications of Pólya theory, only the rotation group is considered, and enantiomorphous pairs which are mirror images of each other are counted individually. However, from the standpoint of pure geometry or pure group theory, such distinction is superfluous and it is proper to add the reflection group for reducing them[1]. Thus we took into account both symmetric groups rotation group and reflection group.

This paper consists of the following sections: the definition of cube cutting (Section 2), enumeration for triangles configuration in a cube using group theory (Section 3), and conclusions (Section 4).



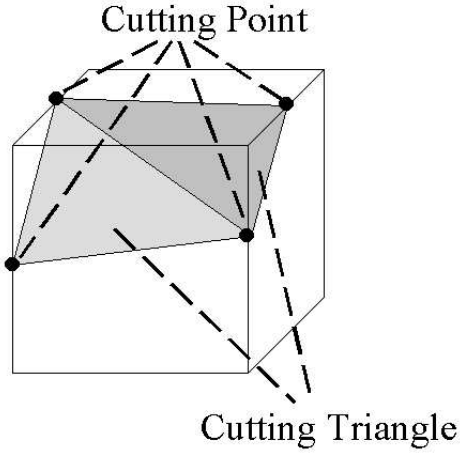


Figure 1(left). Half cyclide in VCAD. VCAD can retain not only shape but also physical attribute in each cube of cubic lattice. Shape is approximated in a local cube like Figure 2.

Figure 2(right). Cutting triangle and cutting point in Kitta cube: The Kitta cube method approximates original shapes by approximate triangles which are held in each cube. The approximate triangle is called cutting triangle, and the intersection between the surface and edge of the cube is called cutting point.

2 Definition of Cube Cutting: Kitta cubes

In section 1, the outline of our cube cutting was explained. We define the cube cutting clearly in this section.

In each cube, the surface of the shape is not retained but surface approximation that satisfies following three conditions are made: (i) Approximation with triangles, (ii) Vertices of the triangle lie on the cube edges, and not on the vertices of cube, (iii) Number of vertices of the triangle on each edge of the cube is at most unity (single cutting point condition). This approximate triangle is called *cutting triangle*, and the intersection between the surface and edge of the cube is called *cutting point* (Figure 2). We call cubes with such approximate triangles *Kitta cubes*. A cutting point itself is a vertex of cutting triangle. The location of a cutting point on an edge is not identified during enumeration.

In KC, we assume that there is only one single cutting point, but generally speaking, two or more cutting points can appear on one edge. They can be unified into a single cutting point by unification disposal. Detail of the disposal shall not be taken up in this paper. After this, we assume that the disposal has already been completed for cutting point on each edge.

The above defines the original KC. If we place cutting points not only on each of the twelve edges but also on each of eight vertices of the cube, in which cutting points on the vertices are independent of cutting points on the edges, other cutting triangles will be obtained. This type of KC is called additional Kitta cube in this paper. Actually, the original KC was applied in version 1 of VCAD, and the additional KC in version 2. The number of patterns of the cutting triangles configuration for the additional KC will also be enumerated in Section 3.

3 Enumeration

This section describes the symmetry group of a cube, taking into account both symmetry groups rotation and reflection. Under the group action, we carry out Pólya's theory of counting. Cutting triangles configuration on twelve edges are investigated in Section 3.2, and then on twelve edges and eight vertices in Section 3.3.

3.1 Symmetry Group on Cube

This section summarizes permutations which place a cube into itself when both the rotation group and reflection group are considered as the permutation group. First, the *order* of the group is checked, that is, the total number of such permutations. The twelve edges $(e_1, e_2, \cdots, e_{12})$ of a cube are sites proposed for cutting points in Figure 3.

Considering rotation and reflection for the cube, we have 12 choices for deciding which edge is situated on the original e_1-position, and four choices for deciding which edge is situated on the original e_3-position because each edge links with four edges. There are therefore $12 \times 4 = 48$ permutations altogether.

The 48 permutations, the permutation group G of a cube, can be classified into the following 10 kinds: $\pi_1, \cdots, \pi_{10}$.

π_1. An identity permutation. We can write down this permutation for the 12 edges of a cube using *cyclic notation* as follows:

$$\pi_1 = (e_1)(e_2)(e_3)(e_4)(e_5)(e_6)(e_7)(e_8)(e_9)(e_{10})(e_{11})(e_{12}) \tag{1}$$

π_2. Permutations arising from a rotation 180° around a four-fold rotational axis of a cube. There are three such permutations according to three four-fold rotational axes shown. One of these three permutations can be written down for the 12 edges as follows.

$$\pi_2 = (e_1 \quad e_6)(e_2 \quad e_5)(e_3 \quad e_7)(e_4 \quad e_8)(e_9 \quad e_{11})(e_{10} \quad e_{12}) \tag{2}$$

π_3. Permutations arising from a rotation 90° around a four-fold rotational axis of a cube. There are six such permutations according to three four-fold rotational axes and two directions of the rotation 90° (clockwise and counterclockwise). One of these six permutations can be written down for the 12 edges as follows.

$$\pi_3 = (e_1 \quad e_5 \quad e_6 \quad e_2)(e_3 \quad e_9 \quad e_7 \quad e_{11})(e_4 \quad e_{10} \quad e_8 \quad e_{12}) \tag{3}$$

π_4. Permutations arising from a rotation 120° around a three-fold rotational axis of a cube. There are eight such permutations according to four three-fold rotational axes and two directions of the rotation 120°(clockwise and counterclockwise). One of these eight permutations can be written down for the 12 edges as follows.

$$\pi_4 = (e_1 \quad e_9 \quad e_3)(e_2 \quad e_{10} \quad e_7)(e_4 \quad e_5 \quad e_{11})(e_6 \quad e_{12} \quad e_8) \tag{4}$$

π_5. Permutations arising from a rotation 180° around a two-fold rotational axis of a cube. There are six such permutations according to six two-fold rotational axes. One of these six permutations can be written down for the 12 edges as follows.

$$\pi_5 = (e_1)(e_6)(e_2 \quad e_5)(e_3 \quad e_{10})(e_4 \quad e_9)(e_8 \quad e_{11})(e_7 \quad e_{12}) \tag{5}$$

π_6. Permutations arising from a reflection against the face whose normal is a four-fold rotational axis. There are three such permutations according to three four-fold rotational axes. One of these three permutations can be written down for the 12 edges as follows.

$$\pi_6 = (e_1)(e_2)(e_5)(e_6)(e_3 \quad e_4)(e_7 \quad e_8)(e_9 \quad e_{10})(e_{11} \quad e_{12}) \tag{6}$$

π_7. Permutations arising from a reflection against the face whose normal is a two-fold rotational axis. There are six such permutations according to six two-fold rotational axes. One of these six permutations can be written down for the 12 edges as follows.

$$\pi_7 = (e_2)(e_5)(e_1 \quad e_6)(e_3 \quad e_{11})(e_4 \quad e_{12})(e_7 \quad e_9)(e_8 \quad e_{10}) \tag{7}$$

π_8. Permutations arising from a reflection against the face whose normal is a four-fold rotational axis and a successive rotation 90° around the same axis. There are six such permutations according to three four-fold rotational axes and two directions of the rotation 90°(clockwise and counterclockwise). One of these six permutations can be written down for the 12 edges as follows.

$$\pi_8 = (e_1 \quad e_5 \quad e_6 \quad e_2)(e_3 \quad e_{10} \quad e_7 \quad e_{12})(e_4 \quad e_9 \quad e_8 \quad e_{11}) \tag{8}$$

π_9. Permutations arising from a reflection against the face whose normal is a three-fold rotational axis and a successive rotation 60° around the same axis. There are eight such permutations according to four three-fold rotational axes and two directions of the rotation 60°(clockwise and counterclockwise). One of these eight permutations can be written down for the 12 edges as follows.

$$\pi_9 = (e_1 \quad e_8 \quad e_9 \quad e_6 \quad e_3 \quad e_{12})(e_2 \quad e_4 \quad e_{10} \quad e_5 \quad e_7 \quad e_{11}) \tag{9}$$

π_{10}**.** Permutations arising from a reflection against the body-center point. Such permutation is caused, for example, by a reflection against the face whose normal is a four-fold rotational axis and a successive rotation 180° around the same axis. Although the permutation is realized via different operations, there is only one such permutation in consequence. This permutation can be written down for the 12 edges as follows.

$$\pi_{10} = (e_1 \quad e_6)(e_2 \quad e_5)(e_3 \quad e_8)(e_4 \quad e_7)(e_9 \quad e_{12})(e_{10} \quad e_{11}) \tag{10}$$

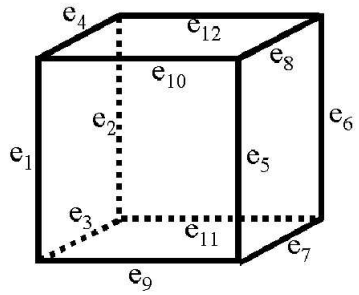


Figure 3. Twelve edges of cube

3.2 Cutting Triangles on Kitta Cubes

The definite operations of the permutation group G was described. The *degree* of the group, that is the number of objects under group action is 220 because the number of different cutting triangles in a cube is $\binom{12}{3} = 220$. The cube will have 220 cutting triangles at maximum, and zero at minimum. Each of the 220 cutting triangles has two possibilities, that is, the existence or non-existence of the cutting triangle. If symmetry is not taken into account, there are $2^{220} \cong 1.685 \times 10^{66}$ different configurations altogether for the cutting triangle configuration.

Under the permutation G, Pólya's theory of counting is applied to our enumeration in this section. Our main purpose is to enumerate the number of the equivalence classes, in other words, to classify the 2^{220} total configurations into the equivalence classes.

The set of cutting triangles is expressed as $D = \{t_1, t_2, \cdots, t_{220}\}$. D means the domain. Either of two states, that is, the existence or non-existence of the cutting triangle is given to each element in D. The set of the states is expressed as $R = \{x, y\}$ (x: non-existence y: existence). R means the range.

When the permutation group G acts on D, the *cycle structure representation* of each permutation can be expressed instead of equations (1) and (10) as follows.

$$\begin{aligned} &\pi_1 : f_1^{220}, \quad \pi_2 : f_2^{110}, \quad \pi_3 : f_4^{55}, \quad \pi_4 : f_1^4 f_3^{72}, \quad \pi_5 : f_1^{10} f_2^{105}, \\ &\pi_6 : f_1^{20} f_2^{100}, \quad \pi_7 : f_1^{10} f_2^{105}, \quad \pi_8 : f_4^{55}, \quad \pi_9 : f_2^2 f_6^{36}, \quad \pi_{10} : f_2^{110} \end{aligned} \tag{11}$$

Therefore, the *cycle index* P_G for permutation group G on D can be expressed as follows.

$$\begin{aligned} &P_G(f_1, f_2, f_3, f_4, f_6) \\ &\quad = \frac{1}{48}(f_1^{220} + 3f_2^{110} + 6f_4^{55} + 8f_1^4 f_3^{72} + 6f_1^{10} f_2^{105} \\ &\qquad + 3f_1^{20} f_2^{100} + 6f_1^{10} f_2^{105} + 6f_4^{55} + 8f_2^2 f_6^{36} + f_2^{110}) \end{aligned} \tag{12}$$

The inventory of a set of patterns is given by substituting the *figure inventory* $f_j = x^j + y^j$ into the cycle index P_G.

$$\begin{aligned} &P_{G_3}(x+y, x^2+y^2, x^3+y^3, x^4+y^4, x^6+y^6) \\ &= \frac{1}{48}((x+y)^{220} + 3(x+y)^{20}\left(x^2+y^2\right)^{100} + 12(x+y)^{10}\left(x^2+y^2\right)^{105} + 4\left(x^2+y^2\right)^{110} \\ &\quad + 8(x+y)^4\left(x^3+y^3\right)^{72} + 12\left(x^4+y^4\right)^{55} + 8\left(x^2+y^2\right)^2\left(x^6+y^6\right)^{36}) \\ &= x^{220} + 9x^{219}y + 568x^{218}y^2 + 36971x^{217}y^3 + \cdots \\ &\quad + 1886222908006572514069691261923905322599325780324861587434171904x^{110}y^{110} \\ &\quad + \cdots + 36971x^3y^{217} + 568x^2y^{218} + 9xy^{219} + y^{220} \end{aligned} \tag{13}$$

Each coefficient of Equation (13) shows the number of patterns in the case the number of cutting triangles is 0, 1, 2, $\cdots$, 220 in sequence.

Total number of the equivalence classes can be obtained directly from Equation (12) as follows.

$$\begin{aligned} & P_{G_3}(|C|,|C|,|C|,|C|,|C|) = P_{G_3}(2,2,2,2,2) \\ &= \frac{1}{48}(2^{220}+3\cdot 2^{110}+6\cdot 2^{55}+8\cdot 2^4\cdot 2^{72}+6\cdot 2^{10}\cdot 2^{105} \\ &\quad +3\cdot 2^{20}\cdot 2^{100}+6\cdot 2^{10}\cdot 2^{105}+6\cdot 2^{55}+8\cdot 2^2\cdot 2^{36}+2^{110}) \\ &= 351040972228523955659726758946503802892698907660584753014914416 64 \\ &\approx 3.510\times 10^{64} \end{aligned} \tag{14}$$

This number is consistent with the sum of coefficients in Equation (13).

3.3 Cutting Triangles on Additional Kitta Cubes

In this section, the number of patterns of additional KC described in Section 2 is enumerated. Each edge and vertex of a cube can have at most one cutting point. Cutting points on the vertices are independent of cutting points on the edges, therefore a cube can have twenty cutting points at maximum.

The degree of the group is 1140 because the number of different cutting triangles in a cube is $\binom{20}{3} = 1140$. The cube will have 1140 cutting triangles at maximum, and zero at minimum. Each of the 1140 cutting triangles has two possibilities, that is, the existence or non-existence of the cutting triangle. If symmetry is not taken into account, there are $2^{1140} \cong 1.493\times 10^{343}$ different configurations altogether for the cutting triangle configurations.

The set of cutting triangles can be expressed as $D = \{t_1, t_2, \cdots, t_{1140}\}$. Either the existence or non-existence of the cutting triangle is given to each element in D. The set of states can be expressed as $R = \{x, y\}$ (x: non-existence y: existence).

In the previous section 3.2, the cycle structure representation for each permutation was expressed by the equation (11). In this section, not all the cycle structure representations are expressed because it is time-consuming to do so for a large order like 1140. Total number of equivalence classes will be calculated approximately but the approximation is very close to the exact number.

The results in the last section are reviewed here. It can be confirmed that only the first term in Equation (12): $f_1^{220}/48$ provides good approximation:

$$2^{220}/48 \approx 3.51041\times 10^{64}. \tag{15}$$

This is almost equal to the exact number(Equation (14)). The ratio of the error from the exact number is merely 2.665×10^{-30}. Thus, the first term in Equation (12) is dominating because the exponent 220 of the term is considerably larger than other exponents.

This approximation is valid for the current enumeration because the exponent 1140 arising from the identity permutation is considerably larger than other exponents. Therefore, an approximate number of equivalence classes for cutting triangle configurations on the additional KC is

$$2^{1140}/48 \approx 3.11138\times 10^{341}. \tag{16}$$

4 Conclusions

The number of equivalence classes of cutting triangles configuration was successfully calculated. It was calculated precisely for the KC and approximately for the additional KC. Each enumerations gave very large nmber. These findings show that the KC are able to express a wide variety of shapes and forms.

Although the KC have a great many patterns, only a small part of KC is appearing in the current VCAD. The approximate surfaces in the current VCAD does not have the crossing of cutting triangles and the branching of the surface. Therefore, a problem to be solved in future is to classify the present patterns into detailed category.

The authors are grateful to Y. Nakashima, H. Ohsugi, T. Soma, N. Ikeda, and VCAD project members for valuable comments and encouragement.

References

[1] Coxeter, H. S. M. (1989) *Introduction to Geometry* (second edition), Section 15.6, John Wiley & Sons, Inc.

[2] Ikeda, N., and Teshima, Y. (2004) “The information measure for diagrams described by cutting segments, ” Proceedings of The fourth international symposium on Human and Artificial Intelligence Systems, (2004), pp.485-489.

[3] Kase, K., Teshima, Y., Usami, S., Ohmori, H., Teodosiu, C., and Makinouchi, A. (2003), "Volume CAD," in *Volume Graphics 2003 Eurographics / IEEE TCVG Workshop Proceedings*, pp. 145-150, pp. 173.

[4] Kase, K., Teshima, Y., Usami, S., Kato, M., Yamazaki, S., Ito, M., and Makinouchi, A., "Volume CAD : CW-complexes based approach, " *Computer-Aided Design*, (In press)

[5] Lorensen, W.E., and Cline, H.E. (1987), "A High-Resolution 3D surface construction algorithm," *SIGGRAPH*, vol. 21-4, pp. 163-169.

[6] Polya, G. (1937) "Kombinatorische Anzahlbestimmungen fur Gruppen, Graphen, und chemische Verbindungen," *Acta Math*. Vol. 68, 145-254.

[7] Redfield, J. H. (1927) "The theory of group reduced distributions," *Amer. J. Math.* Vol. 49, 433-455.

[8] Teshima, Y., Kase, K., Usami, S., Kato, M., Ikeda, N., and Makinouchi, A. (2004) "Self-reversed configurations on cube", The Journal of the International Society for the Interdisciplinary Study of Symmetry, Vol.1-4 (2004), pp. 258-261.

[9] Teshima, Y., Kase, K., Usami, S., Kato, M., Ikeda, N., and Makinouchi, Enumeration of Cuttting Points Configuration in Cube Cutting, Proceedings of The fourth international symposium on Human and Artificial Intelligence Systems, (2004), pp.407-414.

Intersecting families —uniform versus weighted

NORIHIDE TOKUSHIGE*

College of Education, Ryukyu University
Nishihara, Okinawa, 903-2214 JAPAN
e-mail address hide@edu.u-ryukyu.ac.jp

Abstract: What is the maximal size of k-uniform r-wise t-intersecting families? We show that this problem is essentially equivalent to determine the maximal weight of non-uniform r-wise t-intersecting families. Some EKR type examples are included.

Keywords: intersecting families, Erdős–Ko–Rado Theorem, weight

1 Introduction

Throughout this note let n,k,r,t denote positive integers with $t \le k \le n$, and let p and q denote positive reals with $p+q=1$. A family $\mathscr{G} \subset 2^{[n]}$ is called r-wise t-intersecting if $|G_1 \cap \cdots \cap G_r| \ge t$ holds for all $G_1,\ldots,G_r \in \mathscr{G}$. Let us define n-vertex k-uniform r-wise t-intersecting family $\mathscr{F}_i(n,k,r,t)$ as follows:

$$\mathscr{F}_i(n,k,r,t) = \{F \in \binom{[n]}{k} : |F \cap [t+ri]| \ge t+(r-1)i\}.$$

Let $m(n,k,r,t)$ be the maximal size of k-uniform r-wise t-intersecting families on n vertices. Can we extend the Erdős–Ko–Rado Theorem [3] in the following way?

Conjecture 1 $m(n,k,r,t) = \max_i |\mathscr{F}_i(n,k,r,t)|$.

The p-weight of a family $\mathscr{G} \subset 2^{[n]}$, denoted by $w_p(\mathscr{G})$, is defined as follows:

$$w_p(\mathscr{G}) = \sum_{G \in \mathscr{G}} p^{|G|}q^{n-|G|} = \sum_{i=0}^{n} \left|\mathscr{G} \cap \binom{[n]}{i}\right| p^i q^{n-i}.$$

Let $w(n,p,r,t)$ be the maximal p-weight of r-wise t-intersecting families on n vertices. Set $\mathscr{G}_i(n,r,t) = \bigcup_{k=0}^{n} \mathscr{F}_i(n,k,r,t)$.

Conjecture 2 $w(n,p,r,t) = \max_i w_p(\mathscr{G}_i(n,r,t))$.

The aim of this note is to show that roughly speaking $w(n,p,r,t)$ and $m(n,k,r,t)/\binom{[n]}{k}$ are almost the same if $p \approx k/n$. Therefore the above two problems ask essentially the same thing. We list some known results about the conjectures and related problems in the last section.

Our first result says that $w(n,p,r,t)$ can be deduced from $m(n,k,r,t)$ if $\frac{k}{n} \approx p$.

Theorem 3 *Let r,t and p be given. Then (M1) implies (W1).*

(M1) There exist $\varepsilon > 0$ and n_0 such that $m(n,k,r,t) = \binom{n-t}{k-t}$ holds for all $n > n_0$ and k with $|\frac{k}{n} - p| < \varepsilon$.

(W1) $w(n,p,r,t) = p^t$ holds for all $n \ge t$.

Theorem 4 *Let r,t,p and c be given. Then (M2) implies (W2).*

(M2) There exists $\varepsilon > 0$ such that $m(n,k,r,t) = (c+o(1))\binom{n}{k}$ as $n \to \infty$ holds for all n and k with $|\frac{k}{n} - p| < \varepsilon$.

(W2) $w(n,p,r,t) \le c$ holds for all $n \ge t$.

Moreover if there is an r-wise t-intersecting family $\mathscr{G} \subset 2^{[n]}$ with $w_p(\mathscr{G}) = c$ for some n then $w(n,p,r,t) = c$ holds for all $n \ge t$.

*The author was supported by MEXT Grant-in-Aid for Scientifi c Research (B) 16340027.

Assume (M2). We can choose $\delta, \varepsilon' > 0$ sufficiently small so that $|\frac{k}{n} - p| < \varepsilon$ holds for all p' with $|p - p'| < \delta$ and for all n, k with $|\frac{k}{n} - p'| < \varepsilon'$. Then by (M2) we have $m(n,k,r,t) = (c + o(1))\binom{n}{k}$ for all n, k with $|\frac{k}{n} - p'| < \varepsilon'$. Thus by (W2) we have $w(n,p',r,t) \le c$ for all $n \ge t$. This means we can replace (W2) by

(W2′) There exists $\delta > 0$ such that $w(n,p',r,t) \le c$ holds for all $n \ge t$ and p' with $|p - p'| < \delta$.

The next results are the reverses of Theorem 3 and Theorem 4, which say that $m(n,k,r,t)$ can be deduced from $w(n,p,r,t)$ if $\frac{k}{n} \approx p$.

Theorem 5 *Let r,t and p be given. Then (W3) implies (M3).*

(W3) $\lim_{n\to\infty} w(n,p,r,t) = p^t$.

(M3) For all $\varepsilon > 0$ there exists n_0 such that $m(n,k,r,t) \le (1 + o(1))\binom{n-t}{k-t}$ holds for all $n > n_0$ and $\frac{k}{n} < p - \varepsilon$.

Theorem 6 *Let r,t,p and c be given. Then (W4) implies (M4).*

(W4) $\lim_{n\to\infty} w(n,p,r,t) \le c$.

(M4) For all $\varepsilon > 0$ there exists n_0 such that $m(n,k,r,t) \le (c + o(1))\binom{n}{k}$ holds for all $n > n_0$ and $\frac{k}{n} < p - \varepsilon$.

To refine the above result let us introduce non-trivial versions of m and w. An r-wise t-intersecting family $\mathscr{G} \subset 2^{[n]}$ is called non-trivial if $|\bigcap_{F\in\mathscr{F}} F| < t$. Let $m^*(n,k,r,t)$ be the maximal size of k-uniform non-trivial r-wise t-intersecting families on n vertices and let $w^*(n,p,r,t)$ be the maximal p-weight of non-trivial r-wise t-intersecting families on n vertices.

Theorem 7 *Let r,t and p be given. Then (W5) implies (M5).*

(W5) There exists $\gamma > 0$ such that $\lim_{n\to\infty} w^*(n,p,r,t) < (1-\gamma)p^t$.

(M5) For all $\varepsilon > 0$ there exists n_0 such that $m^*(n,k,r,t) < (1-\eta)\binom{n-t}{k-t}$ holds for all $n > n_0$, $0 < \eta < \gamma$ and $\frac{k}{n} < p - \varepsilon$.

Note that $(M5)$ implies that $m(n,k,r,t) = \binom{n-t}{k-t}$. It would be nice to have the reverse of the above result.

Problem 8 *Let r,t and p be given. Then does (M6) imply (W6)?*

(M6) There exists $\eta > 0, \varepsilon > 0$ and n_0 such that $m^*(n,k,r,t) < (1-\eta)\binom{n-t}{k-t}$ holds for all $n > n_0$, and k with $|\frac{k}{n} - p| < \varepsilon$.

(W6) There exists $\gamma > 0$ such that $\lim_{n\to\infty} w^*(n,p,r,t) < (1-\gamma)p^t$.

If this is true then (M6) implies $m(n,k,r,t) \le \binom{n-t}{k-t}$ for all $\frac{k}{n} < p$ and $n > n_0$ by Theorem 7.

2 Proofs

For a family $\mathscr{G} \subset 2^{[n]}$ and a positive integer $\ell < n$, let us define the ℓ-th shadow of $\mathscr{G}$, denoted by $\Delta_\ell(\mathscr{G})$, as follows.

$$\Delta_\ell(\mathscr{G}) = \{F \in \binom{[n]}{\ell} : F \subset \exists G \in \mathscr{G}\}.$$

The complement family $\mathscr{G}^c$ is defined by $\mathscr{G}^c = \{[n] - G : G \in \mathscr{G}\}$.

PROOF OF THEOREM 3: Assume (M1). Since $w(n,p,r,t) \ge w_p(\mathscr{G}_0(n,r,t)) = p^t$ it suffices to show $w(n,p,r,t) \le p^t$. Set an open interval $K = ((p-\varepsilon)n, (p+\varepsilon)n)$. Let $\mathscr{G} \subset 2^{[n]}$ be an r-wise t-intersecting family with $w(n,p,r,t) = w_p(\mathscr{G})$. Then we have

$$\begin{aligned} w(n,p,r,t) &= \sum_{k\in K} \left|\mathscr{G} \cap \binom{[n]}{k}\right| p^k q^{n-k} + o(1) \\ &\le \sum_{k\in K} \binom{n-t}{k-t} p^k q^{n-k} + o(1) \\ &= (1+o(1))p^t \sum_{k=0}^{n} \binom{n}{k} p^k q^{n-k} + o(1). \end{aligned}$$

This implies $\lim_{n\to\infty} w(n,p,r,t) \le p^t$.

Next define $\mathscr{G}' \subset 2^{[n+1]}$ by $\mathscr{G}' = \mathscr{G} \cup \{G \cup \{n+1\} : G \in \mathscr{G}\}$. Then $w_p(\mathscr{G}') = w_p(\mathscr{G})(q+p) = w(n,p,r,t)$, which means $w(n+1,p,r,t) \geq w(n,p,r,t)$. Consequently we have $w(n,p,r,t) = p^t$ for all $n \geq t$. □

PROOF OF THEOREM 4: This is similar to the proof of Theorem 3, and we omit the proof.

PROOF OF THEOREM 5: Assume (W3). Let $0 < \delta < q$ be given. We want to show that $m(n,k,r,t) < (1+\delta)\binom{n-t}{k-t}$ for $n > n_0$. Suppose on the contrary that there exists $\varepsilon > 0$ such that for all n_0 we can find an r-wise t-intersecting family $\mathscr{F} \subset \binom{[n]}{k}$ which satisfies $|\mathscr{F}| \geq (1+\delta)\binom{n-t}{k-t}$ for $n > n_0$ and $\frac{k}{n} = p - \varepsilon$. Let $\mathscr{G} = \{G : G \supset \exists F \in \mathscr{F}\} = \bigcup_{\ell=0}^{n-k} (\Delta_\ell(\mathscr{F}^c))^c$. This family is also r-wise t-intersecting. We will show that $\mathscr{G}$ violates (W3). Set an open interval $I = ((p-\varepsilon)n, (p+\varepsilon)n)$ and set $c = \frac{q-\varepsilon}{q+\varepsilon}$.

Claim 9 $|\Delta_{n-i}(\mathscr{F}^c)| \geq (1+c\delta)\binom{n-t}{n-i}$ *for* $i \in I$.

PROOF: Choose a real x so that $\delta\binom{n-t}{k-t} = \binom{x}{n-k-1}$. Using $\delta < q$ we have $x < n-t-1$. In fact if $x \geq n-t-1$ then we have $\delta \geq \binom{n-t-1}{n-k-1}/\binom{n-t}{k-t} = \frac{1-(k/n)}{1-(t/n)} > 1-(p-\varepsilon) > q$.

Since $|\mathscr{F}^c| = |\mathscr{F}| \geq (1+\delta)\binom{n-t}{k-t} = \binom{n-t}{k-t} + \binom{x}{n-k-1}$, the Kruskal–Katona Theorem [14, 13] implies that $|\Delta_{n-i}(\mathscr{F}^c)| \geq \binom{n-t}{n-i} + \binom{x}{n-i-1}$. Thus it suffices to show that $\binom{x}{n-i-1} \geq c\delta\binom{n-t}{n-i}$, or equivalently,

$$\frac{\binom{x}{n-i-1}}{\binom{x}{n-k-1}} \geq \frac{c\delta\binom{n-t}{n-i}}{\delta\binom{n-t}{k-t}}.$$

Using $i \geq k$ this is equivalent to $\frac{i-t}{x-n+i+1} \cdots \frac{k-t+1}{x-n+k+2} \geq c\frac{n-k}{n-i}$. The LHS is at least 1, in fact, $\frac{i-t}{x-n+i+1} > 1$ follows from $x < n-t-1$. On the other hand, using $i \leq (p+\varepsilon)n$ we have RHS $= c\frac{1-(k/n)}{1-(i/n)} \leq c\frac{1-(p-\varepsilon)}{1-(p+\varepsilon)} = c\frac{q+\varepsilon}{q-\varepsilon} = 1$, which prove the claim. □

Let us finish the proof of Theorem 5. Using the claim we have

$$\begin{aligned}
w_p(\mathscr{G}) &> \sum_{i\in I} \left|\mathscr{G} \cap \binom{[n]}{i}\right| p^i q^{n-i} \\
&= \sum_{i\in I} |\Delta_{n-i}(\mathscr{F}^c)| p^i q^{n-i} \\
&\geq \sum_{i\in I} \Big(1 + \frac{q-\varepsilon}{q+\varepsilon}\delta\Big)\binom{n-t}{n-i} p^i q^{n-i} \\
&= \Big(1 + \frac{q-\varepsilon}{q+\varepsilon}\delta\Big)\left((1-o(1))p^t\left(\sum_{i=0}^{n}\binom{n}{i}p^i q^{n-i} - o(1)\right)\right) \\
&> p^t,
\end{aligned}$$

which contradicts (M3). □

PROOF OF THEOREM 6: This is similar to the proof of Theorem 5, and we omit the proof.

PROOF OF THEOREM 7: The proof is almost identical to the proof of Theorem 5. The only difference is that instead of Claim 9 we use the following fact here:

If $|\mathscr{F}| \geq (1-\eta)\binom{n-t}{k-t}$ *then* $|\Delta_{n-i}(\mathscr{F}^c)| \geq (1-\eta)\binom{n-t}{n-i}$ *holds for* $i \in I$. □

3 Examples

In this section, we list some known results about $m(n,k,r,t)$ and $w(n,p,r,t)$.

3.1 The case $r = 2$

Ahlswede and Khachatrian settled Conjecture 1 for this case.

Example 10 ([1]) $m(n,k,r=2,t) = \max_i |\mathscr{F}_i(n,k,r=2,t)|$.

This together with Theorem 4 gives the following result, which confirms Conjecture 2 for the case $r = 2$.

Example 11 $w(n,p,r=2,t) = \max_i w_p(\mathscr{G}_i(n,r=2,t))$.

Let $i_{\max} = \lfloor \frac{n-t}{2} \rfloor$. We can rephrase the above result more explicitly, that is, we have

$$w(n,p,r=2,t) = w_p(\mathscr{G}_i(n,2,t)) = \sum_{j=t+i}^{t+2i} \binom{t+2i}{j} p^j q^{t+2i-j}$$

for $\frac{i}{t+2i-1} \le p \le \frac{i+1}{t+2i+1}$ where $i = 0,1,\ldots,i_{\max}$, and

$$w(n,p,r=2,t) = w_p(\mathscr{G}_{i_{\max}}(n,2,t))$$

for $p \ge \frac{n-t}{2n-2}$. In particular, for the case $p = 1/2$ we get the Katona Theorem [12], i.e.,

$$w(n,p=1/2,r=2,t) = w_p(\mathscr{G}_{i_{\max}}(n,2,t)) \to 1/2 \quad (n \to \infty).$$

On the other hand, for the case $p > 1/2$ we have

$$w(n,p>1/2,r=2,t) = w_p(\mathscr{G}_{i_{\max}}(n,2,t)) \to 1 \quad (n \to \infty).$$

Let us also mention non-trivial version. We have $w^*(n,p,r=2,t) = w(n,p,r=2,t)$ for $p \ge \frac{1}{t+1}$ and $w^*(n,p,r=2,t) = w_p(\mathscr{G}_1(n,p,r)) = (t+2)p^{t+1} - (t+1)p^{t+2}$ for $p \le \frac{1}{t+1}$.

3.2 The case $t = 1$

In this case both Conjecture 1 and Conjecture 2 are known to be true.

Example 12 ([4]) *We have* $m(n,k,r,t=1) = \binom{n-1}{k-1}$ *for* $\frac{k}{n} \le \frac{r-1}{r}$.

Example 13 ([7]) *We have*

$$w(n,p,r,t=1) = p \quad \text{for } p \le \tfrac{r-1}{r},$$
$$\lim_{n\to\infty} w(n,p,r,t=1) = 1 \quad \text{for } p > \tfrac{r-1}{r}.$$

Let $\mathscr{G} = \{G \subset [n] : |G \cap [r+1]| \ge r\}$. Then this is a non-trivial r-wise intersecting family with $w_p(\mathscr{G}) = p^r(r+1-pr)$. Brace and Daykin proved that $\mathscr{G}$ is the optimal family if $p = 1/2$.

Example 14 ([2]) $w^*(n,p=1/2,r,t=1) = (\frac{1}{2})^r(\frac{r}{2}+1)$.

We can slightly extend the above result as follows.

Example 15 ([20]) *There exists* $\varepsilon > 0$ *such that* $w^*(n,p,r,1) = |w_p(\mathscr{G}))| = p^r(r+1-pr)$ *holds for all* $n \ge t$, $r \ge 11$ *and* p *with* $|p - \frac{1}{2}| < \varepsilon$. *Moreover* $\mathscr{G}$ *is the only optimal confi guration (upto isomorphism).*

The above result fails if $r \le 5$ as follows.

Example 16 ([9]) $\lim_{n\to\infty} w^*(n,p,r=5,t=1) \ge p^3 > p^5(6-5p)$ *holds for* $0 < p < \frac{1+\sqrt{21}}{10}$.

Conjecture 17 *There exists* $\varepsilon > 0$ *such that* $\lim_{n\to\infty} w^*(n,p,r,t=1) = p^r(r+1-pr)$ *holds for all* $n \ge t$, $r \ge 6$ *and* $|p - \frac{1}{2}| < \varepsilon$.

Example 18 ([20]) *Let* $r \ge 11$. *Then there exists* $\varepsilon_r > 0$ *and* n_r *such that*

$$m^*(n,k,r,1) = |\mathscr{F}_1(n,k,r,1)| = (r+1)\binom{n-r-1}{k-r} + \binom{n-r-1}{k-r-1}$$

holds for all $n > n_r$ *and* k *with* $|\frac{k}{n} - \frac{1}{2}| < \varepsilon_r$. *Moreover* $\mathscr{F}_1(n,k,r,1)$ *is the only optimal confi guration (upto isomorphism).*

3.3 The case $r = 3$

Let $p_t = \frac{2}{\sqrt{4t+9}-1}$. Then we have $w_p(\mathscr{G}_0(n,3,t)) \ge w_p(\mathscr{G}_1(n,3,t))$ iff $p \le p_t$. If Conjecture 2 is true then we have $w(n,p,r=3,t) = p^t$ for $p \le p_t$.

Example 19 ([6]) $w(n,p,r=3,t=2) = p^2$ *for* $p \le 0.5018$.

Comparing $p_2 \approx 0.64$, the bound for p in the above example seems to be far from best possible. Theorem 5 and Example 19 with some additional argument give the following.

Example 20 ([8]) $m(n,k,r=3,t=2) = \binom{n-2}{k-2}$ *for* $\frac{k}{n} \le 0.501$ *and* $n > n_0$.

For larger t, we can get the sharp bound for k/n and p.

Example 21 ([18]) $m(n,k,r=3,t) = \binom{n-t}{k-t}$ *for* $t \ge 26$, $\frac{k}{n} \le p_t$ *and* $n > n_0(t)$.

This together with Theorem 4 implies $w(n,p,r=3,t) = w_p(\mathscr{G}_0(n,3,t)) = p^t$ for $t \ge 26$ and $p \le p_t$.

3.4 The case $p \approx 1/2$

Let $T_r = 2^r - r - 1$. Then we have $w_{1/2}(\mathscr{G}_0(n,r,t)) \geq w_{1/2}(\mathscr{G}_1(n,r,t))$ iff $t \leq T_r$. Frankl proved Conjecture 2 for the case $p = 1/2$.

Example 22 ([5]) *$w(n, p = 1/2, r, t) = w_{1/2}(\mathscr{G}_0(n,r,t)) = (1/2)^t$ for $t \leq T_r$.*

Using Theorem 5 we have

$$m(n,k,r,t) = (1+o(1))\binom{n-t}{k-t}$$

for $t \leq T_r$, $\frac{k}{n} < \frac{1}{2}$ and n sufficiently large. Conjecture 1 suggests that the $o(1)$ term could be removed. In fact this was confirmed for $4 \leq r \leq 10$ and smaller t in [19]. Let us define t_r for $4 \leq r \leq 10$ as in the following table.

r	4	5	6	7	8	9	10
t_r	7	18	41	89	184	377	762
T_r	11	26	57	120	247	502	1013

Example 23 ([19]) *For $4 \leq r \leq 10$ there exists $\varepsilon > 0$ and $n_0 = n_0(\varepsilon)$ such that $m(n,k,r,t) = \binom{n-t}{k-t}$ holds for $t \leq t_r$, $|\frac{k}{n} - \frac{1}{2}| < \varepsilon$ and $n > n_0$. Moreover there exists $\gamma = \gamma(\varepsilon) > 0$ such that $m^*(n,k,r,t) < (1-\gamma)\binom{n-t}{k-t}$ holds for $n > n_1(\gamma)$.*

Thus for $4 \leq r \leq 10$ there exists $\varepsilon > 0$ such that $w(n,p,r,t) = p^t$ holds for all $n \geq t$, $t \leq t_r$, $|p - \frac{1}{2}| < \varepsilon$.

3.5 General case

Example 24 ([17]) *$m(n,k,r,t) = \binom{n-t}{k-t}$ if $p = \frac{k}{n}$ satisfies $p < \frac{r-2}{r}$,*

$$q\, p^{\frac{t}{t+1}(r-1)} - p^{\frac{t}{t+1}} + p < 0 \tag{1}$$

and $n > n_0(r,t,p)$.

Let $f(x) = qx^{r-1} - x + p$ and let $\alpha \in (p,1)$ be the root of the equation $f(x) = 0$. Then α can be written in the following form (see [16]):

$$\alpha = \sum_{j \geq 0} \frac{1}{rj+1} p^{(r-1)j+1} q^j.$$

We also note that $f(x) > 0$ for $0 < x < \alpha$ and $f(x) < 0$ for $\alpha < x < 1$. Thus if $\alpha < p^{t/(t+1)}$ then we have $f(p^{t/(t+1)}) < 0$, that is, we have (1). Then it follows that $m(n,k,r,t) = \binom{n-t}{k-t}$ if $\alpha < p^{t/(t+1)}$, i.e., $t \leq \lfloor \frac{-\log \alpha}{\log \alpha - \log p} \rfloor$.

Example 24 and Theorem 3 give $w(n,p,r,t) = p^t$ if (1) holds. On the other hand we have $w_p(\mathscr{G}_0(n,r,t)) \geq w_p(\mathscr{G}_1(n,r,t))$ iff

$$(t+r)p^{r-1} - (t+r-1)p^r - 1 \leq 0, \tag{2}$$

or equivalently, $t \leq \sum_{i=0}^{r-1}(p^{-1} - 1)$. Thus if Conjecture 2 is true then we can replace (1) by (2).

3.6 Intersecting Sperner families

A family $\mathscr{G} \subset 2^{[n]}$ is called a Sperner family if $G \not\subset G'$ holds for all distinct $G, G' \in \mathscr{G}$. Let $s(n,r,t)$ be the maximal size of r-wise t-intersecting Sperner families on n vertices.

Problem 25 *Determine $s(n,r,t)$.*

Milner settled the case $r = 2$.

Example 26 ([15]) *$s(n, r = 2, t) = \binom{n}{\lceil \frac{n+t}{2} \rceil}$.*

Frankl and Gronau settled the case $r = 3$ and $t = 1$.

Example 27 ([4, 10, 11]) *$s(n = 2m, r = 3, t = 1) = \binom{n-1}{m} + 1$ for $m > m_0$ and $s(n = 2m+1, r = 3, t = 1) = \binom{n-1}{m}$ for $m > m_1$.*

Gronau also settled the case $r \geq 4$ and $t = 1$ completely.

Example 28 ([10]) *$s(n, r \geq 4, t = 1) = \binom{n-1}{\lceil \frac{n-1}{2} \rceil}$.*

Based on Example 21, the case $r=3$ and $t=2$ was settled for large n as follows.

Example 29 ([8]) *$s(n=2m,r=3,t=2)=\binom{n-2}{m-1}$ for $m>m_0$ and $s(n=2m+1,r=3,t=2)=\binom{n-2}{m}+2$ for $m>m_1$.*

Problem 30 *Does $s(n,r,t)=\binom{n-t}{\lceil\frac{n-t}{2}\rceil}$ hold for $r\geq 4$, $t\leq 2^r-r-1$ and $n>n_0(r,t)$?*

Example 31 ([19]) *Let r and t be fixed positive integers. Suppose that there exists $\gamma=\gamma(r,t)>0$ and $\varepsilon=\varepsilon(\gamma)>0$ such that $m^*(n,k,r,t)=(1-\gamma)\binom{n-t}{k-t}$ holds for $|\frac{k}{n}-\frac{1}{2}|<\varepsilon$ and $n>n_0(\varepsilon)$. Then we have $s(n,r,t)=\binom{n-t}{\lceil\frac{n-t}{2}\rceil}$ for $n>n_0(\varepsilon)$.*

This together with Example 23 gives the following.

Example 32 ([19]) *For $4\leq r\leq 10$ we have $s(n,r,t)=\binom{n-t}{\lceil\frac{n-t}{2}\rceil}$ for $t\leq t_r$ and $n>n_0$.*

The proof of Example 24 given in [17] can be extended without much changes to prove the following.

Example 33 *For fixed r,t,p with $p<\frac{r-2}{r}$ and (1) there exists $\gamma=\gamma(r,t,p)>0$ such that $m^*(n,k,r,t)=(1-\gamma)\binom{n-t}{k-t}$ holds for $\frac{k}{n}=p$ and $n>n_0(\gamma)$.*

Example 33 for $p\approx 1/2$ and Theorem 31 give the following.

Example 34 *$s(n,r,t)=\binom{n-t}{\lceil\frac{n-t}{2}\rceil}$ for $t\leq\lfloor\frac{-\log\alpha}{\log\alpha-\log 2}\rfloor$ and $n>n_0$, where $\alpha\in(1/2,1)$ is the root of the equation $2x=1+x^{r-1}$.*

This gives an affirmative answer to Problem 30 for $t\leq 2^{r-2}\log 2-1$.

References

[1] R. Ahlswede, L.H. Khachatrian. The complete intersection theorem for systems of finite sets. *European J. Combin.*, 18:125–136, 1997.

[2] A. Brace and D. E. Daykin. A finite set covering theorem. I,II,III,IV. *Bull. Austral. Math. Soc.*, 5:197–202, 1971, 6:19–24, 417–433, 1972, *Infinite and finite sets*, I:199–203, 1975.

[3] P. Erdős, C. Ko, R. Rado. Intersection theorems for systems of finite sets. *Quart. J. Math. Oxford (2)*, 12:313–320, 1961.

[4] P. Frankl. On Sperner families satisfying an additional condition. *J. Combin. Theory (A)*, 20:1–11, 1976.

[5] P. Frankl. Multiply-intersecting families. *J. Combin. Theory (B)*, 53:195–234, 1991.

[6] P. Frankl, N. Tokushige. Weighted 3-wise 2-intersecting families. *J. Combin. Theory (A)* 100:94-115, 2002.

[7] P. Frankl, N. Tokushige. Weighted multiply intersecting families. *Studia Sci. Math. Hungarica* 40:287–291, 2003.

[8] P. Frankl, N. Tokushige. Random walks and multiply intersecting families. *J. Combin. Theory (A)*, 109:121-134, 2005.

[9] P. Frankl, N. Tokushige. Weighted non-trivial multiply intersecting families. *Combinatorica* to appear.

[10] H.-D.O.F. Gronau. On Sperner families in which no k-sets have an empty intersection. *J. Combin. Theory (A)*, 28:54–63, 1980.

[11] H.-D.O.F. Gronau. On Sperner families in which no k-sets have an empty intersection II. *J. Combin. Theory (A)*, 30:298–316, 1981.

[12] G.O.H. Katona. Intersection theorems for systems of finite sets. *Acta Math. Acad. Sci. Hung.*, 15:329–337, 1964.

[13] G.O.H. Katona. A theorem of finite sets, in: Theory of Graphs, Proc. Colloq. Tihany, 1966 (Akademiai Kiadó, 1968) 187–207, MR 45 #76.

[14] J.B. Kruskal. The number of simplices in a complex, in: Math. Opt. Techniques (Univ. of Calif. Press, 1963) 251–278, MR 27 #4771.

[15] E.C. Milner. A combinatorial theorem on systems of sets. *J. London Math. Soc.*, 43:204–206, 1968.

[16] N. Tokushige. A frog's random jump and the Pólya identity. *Ryukyu Math. Journal*, 17:89–103, 2004.

[17] N. Tokushige. Extending the Erdős–Ko–Rado theorem. *preprint*.

[18] N. Tokushige. The maximum size of 3-wise t-intersecting families. *preprint*.

[19] N. Tokushige. EKR type inequalities for 4-wise intersecting families. *preprint*.

[20] N. Tokushige. Non-trivial multiply intersecting families. *preprint*.

Approximation Algorithms for Computing a Highly Dense Subgraph

AKIKO SUZUKI

Graduate School of Information Sciences, Tohoku University, Sendai 980-8579, Japan.
akiko@dais.is.tohoku.ac.jp

TAKESHI TOKUYAMA

Graduate School of Information Sciences, Tohoku University, Sendai 980-8579, Japan.
tokuyama@dais.is.tohoku.ac.jp.

Abstract

We consider the dense subgraph problem that extracts a subgraph with a prescribed number of vertices that has the maximum number of edges (total edge weight for a weighted case) in a given graph. We give approximation algorithms with improved theoretical approximation ratios assuming that the density of the optimal output subgraph is high, where density is the ratio of number of edges (or sum of edge weights) to the number of edges in the clique on the same number of vertices. Moreover, we investigate the case where the input graph is bipartite, and design a pseudo-polynomial time approximation scheme that can become a PTAS (in a randomized sense) even if the size of the optimal output graph is comparatively small. This is a significant improvement in theoretical sense, since no constant-ratio approximation algorithm has been known previously if the output graph has $o(n)$ vertices.

1 Introduction

We consider the weighted *dense subgraph problem* (often called the maximum dispersion problem or dense k-subgraph problem) defined as follows:

> Consider a weighted graph $G = (V,E)$, where $|V| = n$ and each edge e has a nonnegative weight $0 \leq w(e) \leq 1$. Given a natural numbers $k \leq n$, find a subgraph $H = (X,F)$ of G such that $|X| = k$ and $w(F) = \sum_{e \in F} w(e)$ is maximized.

Its bipartite version is as follows:

> Consider a weighted bipartite graph $G = (U,V,E)$, where $|U| = m$, $|V| = n$ and each edge e has a nonnegative weight $0 \leq w(e) \leq 1$. Given two natural numbers $m' \leq m$ and $n' \leq n$, find a subgraph $H = (X,Y,F)$ of G such that $|X| = m'$, $|Y| = n'$ and $w(F) = \sum_{e \in F} w(e)$ is maximized.

We note the condition $0 \leq w(e) \leq 1$ is given since it is convenient for presenting our theoretical results, although we can define each problem without this condition. We say *unweighted dense subgraph problem* if $w(e) = 1$ for each edge. We define the density Δ of the output subgraph H to be $\Delta = \frac{2w(F)}{k(k-1)}$ for the non-bipartite case and $\Delta = \frac{w(F)}{m'n'}$ for the bipartite case. In other words, density is the ratio of the weight sum to the number of edges in a clique (or a bipartite clique) of the same size. We mainly consider the case where the density of the optimal output subgraph is high (e.g., $\Delta = \Omega(1)$), and aim to design efficient approximation algorithms.

Previous work

The densest subgraph problem that finds a subgraph with the maximum average degree without any size constraint of the subgraph can be solved in polynomial time [11]. However, the unweighted dense subgraph problem (where k is given) is *NP*-hard since the max-clique problem is reduced to it.

Therefore, several theoretical approximation algorithms have been proposed [3, 5, 7, 9, 12] in the literature for the dense subgraph problem. We use a convention that an algorithm has an approximation ratio $r > 1$ for a maximization problem if its objective value is at least r^{-1} times the optimal value. Some papers regard r^{-1} as the approximation ratio, and a reader accustomed to that convention should be aware of it. In practice, a greedy algorithm removing the smallest weighted-degree vertex one by one often works well; however, its approximation ratio is $2n/k$ (ignoring smaller terms) if $k < n/3$, and it is asymptotically tight [5]. One nice feature of this algorithm is that it gives a constant approximation ratio if $k = \Omega(n)$; however, the linear dependency of the ratio in n is not satisfactory from the theoretical point of view. The current best algorithm has an approximation ratio that is slightly better than $n^{1/3}$, and it is conjectured that there exists some constant ε such that n^{ε} approximation is hard [7, 9, 4, 12]. As for the lower bound, Feige [8] showed that it is R3SAT-hard to approximate within a ratio better than some constant when $k = \Omega(n)$.

*Research is supported by Science Research Grant 16092202 of Japan

Unweighted dense subgraph problems with some additional density conditions on the input/output graph have been also considered. In particular, when the optimal subgraph has $\Omega(n^2)$ edges (thus, $\Delta = \Omega(1)$ and $k = \Omega(n)$), PTAS algorithms are known [3, 7]. Here, we extend the notion of PTAS such that we allow a Monte-Carlo randomized algorithm running in polynomial time in n and ε^{-1} that finds an $(1+\varepsilon)$-approximate solution with a high probability. However, in many applications of the dense subgraph problem, it is desired to solve the problem with only a density assumption on the optimal output graph. An n^{ε}-approximation algorithm with a time complexity $O(n^{1/\varepsilon})$ is known for the case $\Delta = \Omega(1)$ under an additional condition that the average degree of the input graph is $\Omega(k)$ [9]. It is also claimed in [9] that if the optimal subgraph is a clique, there is an $n^{O((1/\varepsilon)\log(n/k))}$ time algorithm to have an $(1+\varepsilon)$-approximation solution: This time complexity is polynomial only if $k = \Omega(n)$. As far as the authors know, no constant-ratio algorithm is known for $k = o(n)$ even for the unweighted problem. It is an interesting question whether we can relax the requirement $k = \Omega(n)$, and this is one of our motivations of this research. Indeed, one may suspect this is difficult, since the maximum clique problem is known to be hard to approximate if the approximation ratio is measured by the number of vertices of the clique.

Next, let us consider the bipartite dense subgraph problem. In recent applications such as clustering and association rule computation in data mining and Web analysis, bipartite cliques and high-density bipartite subgraphs are frequently considered. Although the bipartite dense subgraph problem looks easier than the general dense subgraph problem, it has not been revealed how easier it is. Indeed, it is not much easier. The problem is $\mathcal{NP}$-hard, since the $\mathcal{NP}$-hard edge-maximizing bipartite clique problem is reduced to this problem. Note that bipartite clique problem is polynomial-time soluble if the criterion is to maximize the number of vertices. (See [10] for the complexities of bipartite clique problems). Moreover, as shown later in Section 2, if we have an approximation algorithm with the approximation ratio r for the bipartite dense subgraph problem, we have an approximation algorithm with the approximation ratio $2r$ (roughly speaking) for the general dense subgraph problem.

Our contribution

For the bipartite dense subgraph problem, let $p = m/m'$, and $q = n/n'$. We have an algorithm with a time complexity $O(mn2^{O(\Delta^{-1}\varepsilon^{-2}\log p\log q)})$ that outputs $H_0 = (X_0, Y_0, F_0)$ such that $|X_0| = m'$, $|Y_0| = n'$ and its weighted density Δ_0 satisfies $(1+\varepsilon)\Delta_0 \geq \Delta$ for any positive $\varepsilon < 1$. The time complexity implies that we have PTAS if either (1) Δ and $\min(p,q)$ are constants, (2) both p and q are constants and $\Delta = \Omega(\log^{-1} n)$, or (3) Δ is a constant and $\max(p,q) = 2^{O(\sqrt{\log mn})}$. We also give a polynomial-time approximation algorithm with an approximation ratio $(\min(p,q))^{\varepsilon}$ for any constant $\varepsilon > 0$ only with the density condition $\Delta = \Omega(1)$. These results imply that Δ is the principal parameter to control the computational complexity. We remark that $\Delta = \Omega(1)$ and $\min(p,q)$ is small in data mining applications as discussed later.

As direct consequences of the bipartite problem, we have the following results for the non-bipartite dense subgraph problem: We give an algorithm with an approximation ratio $(n/k)^{\varepsilon}$ if the optimal solution has $\Omega(k^2)$ edges (or $w(F) = \Omega(k^2)$ for the weighted problem). This improves the previous n^{ε} approximation ratio of [9] when k is large, and also the additional condition on the average degree of the input graph is removed. Moreover, we give a $(2+\varepsilon)$-approximation algorithm for any $\varepsilon \leq 1$ that runs in $O(n^2 2^{O(\Delta^{-1}\varepsilon^{-2}\log^2(n/k))})$ time. This implies that we have a $(2+\varepsilon)$-approximation polynomial-time algorithm if $\Delta = \Omega(1)$ and $n/k < 2^{\sqrt{\log n}}$. As far as the authors know, this is the first constant-ratio polynomial-time algorithm for the dense subgraph problem that works for some $k = o(n)$.

Technically, we apply the framework of Arora *et al.* [3] that was developed to solve combinatorial problems on a dense input graph. Original framework of [3] is as follows: First formulate the bipartite dense subgraph problem into a quadratic integer programming (QIP) problem. Next, take a small sample set of variables, and guess values of sampled variables in order to transform the QIP to an integer programming (IP) problem. Then, solve its LP relaxization and apply randomized rounding to have an approximate solution of the IP. Finally, we restore a solution of the QIP assuming that our guess is correct. For the bipartite problem, the IP can be precisely solved, thus we do not need to handle error caused by the randomized rounding. Moreover, the QIP is bilinear, and hence the error due to the restoration can be more precisely analyzed. This enables to replace the density condition of the input graph with that of the output graph.

Our complexity results are purely theoretical, and not practically useful as they are since the exponents are high and dependent on parameters. However, our contribution includes novel analysis of a core algorithm of common heuristics, and will help to give a guideline to tune such heuristics.

Application to dimension-reducing cluster finding

We briefly explain an application of the bipartite dense subgraph problem with the density condition. We consider a data set S containing m data records (often called *tupples*) on n attributes. A *cluster* in S is a subset X such that each pair of tupples in X "resembles" to each other. In data mining situation, we need a *dimension-reducing cluster*: We consider a subset Q of attributes for the cluster, and each pair of data in the cluster should be strongly correlated to each other in the subspace corresponding to Q. The quality of a cluster depends on $|X|$, $|Q|$, and strength of the correlation. If each attribute is binary or categorical, the dimension-reducing cluster-finding problem is a problem in a bipartite graph between the set U of tupples and the set V of categories of attributes. The output cluster is represented by a subset $X \in U$ and $Y \in V$, and its quality is

measured by $m' = |X|$, $n' = |Y|$, and property (e.g. density) of the induced subgraph. Roughly speaking, Y corresponds to Q mentioned above.

It is popular to formulate the problem as a bipartite clique problem. In Agrawal-Imielinski-Swami's seminal paper [1], Y is called *item set* and $|X|/|U| = m'/m = p^{-1}$ is called the *support* of the item set, where X is the set of vertices in U that are incident to all vertices in Y. Thus, X and Y induces a bipartite clique. An item set whose support p^{-1} is larger than a threshold constant is called a *large item set*. Agrawal and Srikant [2] presented an algorithm (named Apriori) to enumerate all large item sets with a time complexity linear in the output size. Also, there are several algorithms (e.g. [13, 16]) that enumerate all maximal large item sets (named *closed item sets*) to reduce the output size, although there are exponential number of closed item sets in the worst case. Procopiuc *et al.*[15] gave a Monte-Carlo algorithm (named DOC) to find a cluster maximizing a particular objective function in a numeric database. In the heart of the algorithm, it constructs a bipartite graph, and solve a version of the bipartite clique problem. DOC is polynomial time if $p = m/m'$ is a constant, but it requires the β*-balanced condition* that is not always assured.

A dense subgraph can be used as a substitute of a bipartite clique for representing a dimension-reducing cluster, and our approximation algorithm works well in a theoretical sense, especially if the support p^{-1} is not much small. Moreover, it often happens that there exists a dense subgraph with large m' and n' even if there is no large bipartite clique, and thus our approach enables to obtain a high-dimensional large cluster even if there is no large clique.

The weighted bipartite dense subgraph problem is useful to model the cluster finding problem if the database has numeric attributes. For example, if we consider an attribute "Income" and classify the data into three categories "high-income", "middle-income" and "low-income" to have three vertices in V. Here, we can give weights indicating the strength of the classification; We give the weight 1 if the classification is sure (e.g. income is very high) but give lower weights otherwise depending on the value of income (e.g., 0.7 to the edge to the "high-income" and 0.3 to the edge to the "middle-income").

2 Reduction to the bipartite problem

We show that we can reduce the general dense subgraph problem to the bipartite case by increasing the approximation factor by 2. We have a graph $G = (V,E)$ and want to find a dense k-vertex subgraph $H = (X,F)$. We first randomly divide V into two sets U' and V', where each vertex of V is assigned to U' with probability 0.5. Thus, we have a bipartite graph $G' = (U',V',E')$. Consider what happens on the optimal subgraph H. Let $F' = E' \cap F$. Then, we have a bipartite subgraph $W = (U' \cap X, V' \cap X, F')$ of G'. By using the argument of randomized max-cut [14], the expectation of $w(F')$ is $w(F)/2$. Thus, with a nonneglizible probability, $(2+o(1))w(F') \geq w(F)$, and it is easy to see that the density condition inherits. Next, we find an approximation solution W' for the bipartite dense subgraph problem that has an approximation ratio r. Here, although we may try all combinations of m' and n' satisfying $m' + n' = k$, it suffices to consider the case where $n' \approx k/2$ (we omit details in this version). Then, the subgraph in G induced by the vertex set of W' gives an $(2+o(1))r$ approximation solution of the original problem. Hence, we have our results on the dense subgraph problem from those on the bipartite dense subgraph problem.

3 Quasi-polynomial time approximation scheme for the bipartite problem

We give an algorithm named ABDense (Approximate Bipartite-Dense) for computing a dense subgraph with a given combination (m', n') of numbers of vertices in a bipartite graph. In a weighted graph, the weighted degree of a vertex is the sum of weights of edges incident to it. We need to give parameters $\gamma < 1$ and I (the number of iterations of random sampling) to implement the algorithm. We randomly select a small subset X_0 of size $\gamma m'$ from U, and find the subset Y_1 of size n' in V with the largest incidence (i.e., sum of weights of induced edges) with X_0. Then, we find the set X_1 of size m' in U with the largest incidence with Y_1, and output the induced subgraph by (X_1, Y_1). The following is a pseudo-code of the algorithm:

Algorithm *ABDense(G, m′, n′)*
(∗ Output is a subgraph H with density D ∗)
1. $D = 0$;
2. **for** $i \leftarrow 1$ **to** I;
3. **do**
4. Randomly select a subset $X_0 \subset U$ of size $\gamma m'$;
5. Let Y_1 be the set of vertices with n' largest weighted-degrees in the induced subgraph by $X_0 \cup V$ in G;
6. Let X_1 be the set of vertices with m' largest weighted-degrees in the induced subgraph by $U \cup Y_1$ in G;
7. **if** density D_1 of the graph H_1 induced by $X_1 \cup Y_1$ is larger than D;
8. **then** $D = D_1$, $H = H_1$:
9. **return** H;

The algorithm itself is a familiar one. If $\gamma = 1$, it is a common naive algorithm for this problem that is a core of many heuristics (e.g., multi-start local search or evolutional methods), and its deterministic version is utilized as a constituent of a hybrid algorithm of [9] with an $n^{1/3}$ approximation ratio. ABDense with a small γ resembles to the core part of DOC algorithm of [15] to find a bipartite clique, where the output is the maximum subset of V forming a bipartite clique with the sample set X_0.

Our contribution is a precise analysis of this algorithm. The success probability that $D > (1+\varepsilon)^{-1}\Delta$ depends on parameters γ and I. We need a rough estimate of Δ to determine these parameters. If we do not have such an estimate in advance, we can run ABDense by using 2^{-j} as estimates of Δ for $j = 0,1,2,\ldots c$ up to a constant c until we have $D \geq (1+\varepsilon)^{-1}2^{-j}$; otherwise we decide that G does not have a dense subgraph with density 2^{-c}. We analyze the performance of the algorithm to give suitable choice of γ and I, and show the following result: Recall that $p = m/m'$, $q = n/n'$, and $\Delta = \frac{w(F)}{m'n'}$.

Theorem 1 For any $0 < \varepsilon < 1$ ABDense computes an $(1+\varepsilon)$-approximation solution for the bipartite dense subgraph problem in $O(mn2^{O(\Delta^{-1}\varepsilon^{-2}\log p\log q)})$ randomized expected time.

3.1 Framework of the Analysis

We first design another algorithm that is easier to analyze, and then simplify the algorithm to obtain ABDense. The algorithm is a two-step sampling algorithm following a framework given by Arora *et al.* [3] for solving the dense subgraph problem: The algorithm first selects a sample set Γ of size γm and then search in its power set to find a subset of size $\gamma m'$ that leads to an approximate solution with the desired theoretical quality.

We can formulate the problem into a quadratic integer programming problem. We write $U = \{u_1, u_2, \ldots, u_m\}$, $V = \{v_1, v_2, \ldots, v_n\}$. We introduce a matrix $W = (w_{i,j})_{1\leq i\leq m, 1\leq j\leq n}$ indicating the graph structure of G. For the unweighted case, $w_{i,j}$ is binary, and $w_{i,j} = 1$ if and only if $(u_i, v_j) \in E$. For the weighted problem, we regard $w_{i,j}$ as the weight of the edge (u_i, v_j). It is straightforward to see that the bipartite dense subgraph problem is equivalent to the following **QIP**.

$$\textbf{QIP:} \quad \text{Maximize } {}^t\mathbf{x}W\mathbf{y}, \quad \text{subject to} \quad \sum_{1\leq i\leq m} x_i = m', \ \sum_{1\leq j\leq n} y_j = n', \text{ and } x_i, y_j \in \{0,1\}.$$

Here, ${}^t\mathbf{x}W\mathbf{y} = \sum_{1\leq i\leq m}\sum_{1\leq j\leq n} w_{i,j}x_iy_j$ is the matrix product (we consider $\mathbf{x}$ and $\mathbf{y}$ as column vectors and ${}^t\mathbf{x}$ is the transpose of $\mathbf{x}$). A very useful feature is that the objective function is bilinear in x_i and y_j. We note that the integral conditions can be relaxed to $0 \leq x_i \leq 1$ and $0 \leq y_j \leq 1$ for this particular problem, although we do not use this property directly.

Let $(\mathbf{x}^{opt}, \mathbf{y}^{opt})$ be an optimal solution of **QIP** and $z^{\text{opt}} = {}^t\mathbf{x}^{opt}W\mathbf{y}^{opt}$. For a given $\mathbf{x}$, we define an n-dimensional vector $\mathbf{a}(\mathbf{x}) = {}^t\mathbf{x}W$. Then, $z^{\text{opt}} = \mathbf{a}(\mathbf{x}^{opt}) \cdot \mathbf{y}^{opt}$, where $\cdot$ is the inner product operation.

Thus, if $\mathbf{a} = \mathbf{a}(\mathbf{x}^{opt})$ is known, it suffices to solve the following problem:

$$\textbf{IP-y(a):} \quad \text{Maximize } \mathbf{a}\cdot\mathbf{y} \quad \text{subject to } \sum_{1\leq j\leq n} y_j = n' \quad \text{and} \quad y_j \in \{0,1\}.$$

Symmetrically, we have the following **IP-x**:

$$\textbf{IP-x(b):} \quad \text{Maximize } \mathbf{b}\cdot\mathbf{x} \quad \text{subject to } \sum_{1\leq i\leq m} x_i = m' \quad \text{and} \quad x_i \in \{0,1\}.$$

Both of **IP-y(a)** and **IP-x(b)** can be solved by greedy algorithms. Indeed, given $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ (resp. $\mathbf{b} = (b_1, b_2, \ldots, b_m)$), we consider the largest n' (resp. m') entries (breaking tie arbitrary) of $\mathbf{a}$ (resp. $\mathbf{b}$), and let $J(\mathbf{a}) \subset \{1,2,\ldots,m\}$ (resp. $J(\mathbf{b}) \subset \{1,2,\ldots,n\}$) be the set of corresponding indices.

We define the binary vector $\mathbf{y}(\mathbf{a}) = (y_1(\mathbf{a}), y_2(\mathbf{a}), \ldots, y_n(\mathbf{a}))$ such that $y_j(\mathbf{a}) = 1$ if and only if $j \in J(\mathbf{a})$. Similarly, we define a binary vector $\mathbf{x}(\mathbf{b})$ such that $x_i(\mathbf{b}) = 1$ if and only if $i \in J(\mathbf{b})$. Then, it is easy to see that the vectors $\mathbf{y}(\mathbf{a})$ and $\mathbf{x}(\mathbf{b})$ are optimal solutions for **IP-y**$(\mathbf{a})$ and **IP-x**$(\mathbf{b})$, respectively.

Now, we can design an algorithm to find a feasible solution for **QIP**. Start with any nonnegative vector $\mathbf{a}^*$ and solve **IP-y(a*)**. Next, using the output $\mathbf{y}^1 = \mathbf{y}(\mathbf{a}^*)$ of **IP-y(a*)**, we compute the vector $\mathbf{b}^* = \mathbf{b}(\mathbf{y}^1) = W\mathbf{y}^1$, and solve **IP-x (b*)** to have an output vector $\mathbf{x}^1 = \mathbf{x}(\mathbf{b}^*)$. The following lemma is straightforward:

Lemma 2 The pair $(\mathbf{x}^1, \mathbf{y}^1)$ is a feasible solution of **QIP**. Moreover, let $(\mathbf{x}^0, \mathbf{y}^0)$ be any feasible solution of **QIP**, and let $(\mathbf{x}^1, \mathbf{y}^1)$ be the vector obtained by applying the above procedure to $\mathbf{a}(\mathbf{x}^0)$. Then, ${}^t\mathbf{x}^1W\mathbf{y}^1 \geq {}^t\mathbf{x}^0W\mathbf{y}^0$.

Let $z^1 = {}^t\mathbf{x}^1W\mathbf{y}^1$ be the objective function value associated with $(\mathbf{x}^1, \mathbf{y}^1)$. We compare z^1 to z^{opt}. We first claim that if $\mathbf{a}^*$ is a good approximation of $\mathbf{a}^{opt} = \mathbf{a}(\mathbf{x}^{opt})$, then $z^{\text{opt}} - z^1$ is small; in other words, $(\mathbf{x}^1, \mathbf{y}^1)$ gives a good approximation solution for **QIP**. Next, we give a method to obtain an $\mathbf{a}^*$ that approximates $\mathbf{a}^{opt}$.

Lemma 3 Let $J = J(\mathbf{a}^{opt})$ and $J^* = J(\mathbf{a}^*)$. Then, $\sum_{j\in J^*} a^*_j \geq \sum_{j\in J} a^*_j$ and $z^1 \geq z^* = \sum_{j\in J^*} a^{opt}_j$.

PROOF: The first formula is straightforward from the definition of J^*. For the second formula, $z^* = \sum_{j\in J^*} a_j^{opt} = {}^t\mathbf{x}^{opt}W\mathbf{y}^1$ is the objective function value of a feasible solution $(\mathbf{x}^{opt},\mathbf{y}^1)$ of **QIP**. However, if we fix $\mathbf{y}^1$, $\mathbf{x}^1$ is the best possible assignment of x values. Thus, this solution cannot be better than $(\mathbf{x}^1,\mathbf{y}^1)$. □

We define $F_1(\mathbf{a}^*) = \sum_{j\in J} a_j^{opt} - \sum_{j\in J} a_j^*$ and $F_2(\mathbf{a}^*) = \sum_{j\in J^*} a_j^* - \sum_{j\in J^*} a_j^{opt}$.

Lemma 4 $z^{\rm opt} - z^1 \le F_1(\mathbf{a}^*) + F_2(\mathbf{a}^*)$.

PROOF: From the previous lemma, $z^{\rm opt} - z^1 \le \sum_{j\in J} a_j^{opt} - \sum_{j\in J^*} a_j^{opt} \le \sum_{j\in J} a_j^{opt} - \sum_{j\in J^*} a_j^{opt} + \sum_{j\in J^*} a_j^* - \sum_{j\in J} a_j^* = F_1(\mathbf{a}^*) + F_2(\mathbf{a}^*)$. □

Thus, in order to obtain an approximate solution whose objective function value is at least $(1-\varepsilon)z^{\rm opt}$, it suffices to find $\mathbf{a}^*$ such that $F_1(\mathbf{a}^*) + F_2(\mathbf{a}^*) \le \varepsilon z^{\rm opt}$. This gives an approximation ratio $(1-\varepsilon)^{-1}$, and it is routine to replace it by $(1+\varepsilon)$ increasing constants hidden in big-O notations in the time complexity.

Now, we consider a random sample $\Gamma \subset U$ of size $|\Gamma| = \gamma m$. We identify U and the index set $\{1,2,\ldots,m\}$, thus we regard $\Gamma \subset \{1,2,\ldots,m\}$. Let $h_j = \sum_{i\in\Gamma} w_{i,j} x_i^{opt}$, for each $j = 1,2,\ldots,n$. We define $\mathbf{a}(\Gamma) = (a_1(\Gamma), a_2(\Gamma), \ldots, a_n(\Gamma))$ by $a_j(\Gamma) = \gamma^{-1} h_j$. Since $\sum_{1\le i\le m} w_{i,j} x_i^{opt} = a_j^{opt}$, the expected values of h_j and $a_j(\Gamma)$ are represented by $E(h_j) = \gamma a_j^{opt}$ and $E(a_j(\Gamma)) = a_j^{opt}$, respectively.

This $\mathbf{a}(\Gamma)$ is our candidate for $\mathbf{a}^*$, and we estimate $F_1(\mathbf{a}(\Gamma)) + F_2(\mathbf{a}(\Gamma))$. Note that we are not ready to claim that we can compute $\mathbf{a}(\Gamma)$ at this stage, since we do not know $\mathbf{x}^{opt}$.

3.2 Analysis of $F_1(\mathbf{a}(\Gamma)) + F_2(\mathbf{a}(\Gamma))$

Since our analysis (its details are omitted in this paper) is somewhat complicated, we give intuition on how to bound $F_1(\mathbf{a}(\Gamma)) + F_2(\mathbf{a}(\Gamma))$. We have n linear forms corresponding to entries of $\mathbf{a}(\mathbf{x}) = {}^t\mathbf{x}W$ on m variables and the (unknown) m-dimensional binary vector $\mathbf{x}^{opt}$. What we do is to bound the difference between the sum of values of the largest n' linear forms (corresponding to J) at $\mathbf{x}^{opt}$ and the sum for the estimated n' largest elements (corresponding to J^*) obtained from the ranking due to the partial information of $\mathbf{x}^{opt}$ restricted to Γ. Thus, our bounds intuitively follow from standard Clarkson-Shor type theory on random sampling error [6], although we give a rigorous analysis using Chernoff's bounds.

The following is the outline of the precise analysis. Since the corresponding terms in F_1 and F_2 cancels out for indices in $J \cap J^*$, we can remove them from the estimation. Thus, the worst case occurs when $J \cap J^* = \emptyset$, and we assume this situation without loss of generality in our analysis. We denote the expected value $E(h_j)$ by μ_j for $j = 1,2,\ldots n$.

Lemma 5 $Pr[h_j > \mu_j + \delta] < e^{-\delta^2/(2\mu_j+\delta)}$ and $Pr[h_j < \mu_j - \delta] < e^{-\delta^2/2\mu_j}$ for each $1 \le j \le n$. Here, $e = 2.718\ldots$ is the base of natural logarithm.

Corollary 6 If $\mu_j \le f$, $Pr[h_j < \mu_j - rf] < e^{-r^2 f/2}$ and $Pr[h_j > \mu_j + rf] < e^{-r^2 f/(2+r)}$ for any $r > 0$.

From the above corollary, we obtain the following:

Lemma 7 If $\gamma = \frac{cn'\ln q}{z^{\rm opt}\varepsilon^2} = \frac{c\ln q}{\Delta m'\varepsilon^2}$ for a sufficiently large constant c, $F_i(\mathbf{a}(\Gamma)) < \varepsilon z^{\rm opt}/2$ with a probability at least 0.9 for each of $i = 1,2$.

Corollary 8 If $\gamma \ge \frac{cn'\ln q}{z^{\rm opt}\varepsilon^2}$ for a sufficiently large constant c, $F_1(\mathbf{a}(\Gamma)) + F_2(\mathbf{a}(\Gamma)) < \varepsilon z^{\rm opt}$ with a probability at least 0.8.

We consider a sample Γ of size γm suggested in the above corollary. Let $Z(\Gamma)$ be the subset of Γ defined by $Z(\Gamma) = \{i \in \Gamma | x_i^{opt} = 1\}$. $h_j = \sum_{i\in Z(\Gamma)} w_{i,j}$ and $a_j(\Gamma) = \gamma^{-1} h_j$ are computed from $Z(\Gamma)$, thus we obtain a $(1-\varepsilon)$ approximation solution with probability 0.8 if we can correctly guess $Z(\Gamma)$.

We can apply a version of exhaustive search to find $Z(\Gamma)$. However, the number of all subsets of Γ is $2^{O(\Delta^{-1}\varepsilon^{-2}p\ln q)}$. Thus, if we exhaustively search all subsets of Γ to find $Z(\Gamma)$, the computation time is exponential in p. Fortunately, we do not need to examine all subsets, since the expected value of the size of $Z(\Gamma)$ is $\gamma m' = p^{-1}|\Gamma|$, and the following lemma is obtained from Chernoff's bounds.

Lemma 9 The probability that $||Z(\Gamma)| - \gamma m'| > 3\sqrt{\gamma m'}$ is at most $2e^{-2}$.

From the above lemma and Corollary 8, we have the following:

Corollary 10 With probability $0.8 - 2e^{-2}$, we have a sample Γ such that $||Z(\Gamma)| - \gamma m'| < 3\sqrt{\gamma m'}$ and $Z(\Gamma)$ gives an $(1+\varepsilon)$-approximation solution for **QIP**.

The number of subsets of Γ whose cardinality is at most $r = \gamma m' + 3\sqrt{\gamma m'}$ is $O(((p+1)e)^r)$ from the Stirling's formula. Since $r = O(\Delta^{-1}\varepsilon^{-2}\ln q)$, we have the following:

Theorem 11 We can compute an $(1+\varepsilon)$-approximation solution for the bipartite dense subgraph problem in $O(mn2^{O(\Delta^{-1}\varepsilon^{-2}\ln p \ln q)})$ time with high probability.

PROOF: We can enumerate all subsets of Γ whose cardinality is at most $\gamma m' + 3\sqrt{\gamma m'}$ in time that is linear in the output size. For each subset, we can compute its associated feasible solution of **QIP** in $O(mn)$ time. We can do sampling multiple times to increase the success probability. □

3.3 Simplifying the algorithm.

In the above algorithm, we take a sample Γ, and exhaustively search all its subsets of size approximately $\gamma m'$ to find $Z(\Gamma)$. However, it is easy to see that there exists a subset of size exactly $\lfloor \gamma m' \rfloor$ to give an approximation algorithm, if we allow to increase the error ratio ε slightly to $(1+3p^{-0.5})\varepsilon$. Thus, it suffices to find a subset of size exactly $\lfloor \gamma m' \rfloor$, and we can randomly generate such subsets instead of enumerating all of them. Now, instead of two-step sampling, we can randomly select a subset X_0 of size $\gamma m'$ directly from U, and apply **IP-y** and **IP-x**; thus, we obtain the algorithm ABDense. Our analysis given in the previous subsection works to give the same time complexity for ABDense, where the number I of iterations is $O(2^{O(\Delta^{-1}\varepsilon^{-2}\ln p \ln q)})$.

4 Approximation algorithm without size restriction

Now, let us consider the case where both p and q are large. By symmetry, we assume $p \geq q$ without loss of generality. If we could naively set $\varepsilon = \sqrt{\log q}$ in the complexity of our quasi-polynomial time algorithm, it would imply a polynomial time algorithm with an $O(\sqrt{\log q})$ approximation ratio. Unfortunately, the analysis only holds under the condition $\varepsilon < 1$. However, we can prove the following theorem:

Theorem 12 For any $\varepsilon > 0$, we have a q^{ε}-approximation algorithm for the weighted bipartite dense subgraph problem if $\Delta = \Omega(1)$.

We assume q is larger than a sufficiently large constant, since otherwise we have already given a PTAS. For technical reason, we prove $2q^{\varepsilon}$ approximation ratio, since it is easy to remove the factor 2 by decreasing ε slightly. The algorithm is the same as the one in Section 3.1 except the sample size. Here, we take a sample Γ of size $\widetilde{\gamma}m = \frac{cmn'}{z^{\mathrm{opt}}}$, where c is a constant dependent on ε. Using the analysis given in the previous section, the time complexity becomes $O(mn2^{O(\ln p \Delta^{-1})}) = O(mnp^{O(\Delta^{-1})})$, which is polynomial if $\Delta = \Omega(1)$. Therefore, it suffices to estimate the approximation ratio.

Lemma 13 If we take c sufficiently large, and take a random sample Γ such that $|\Gamma| = \widetilde{\gamma}m$, $F_1(\mathbf{a}(\Gamma)) < z^{\mathrm{opt}}/2$ with probability at least 0.9. Also, $F_2(\mathbf{a}(\Gamma)) \leq (q^{\varepsilon}-1)z^*$ with a large probability (say, ≥ 0.9).

Theorem 12 follows from the above lemma. Recall that $z^* = \sum_{j \in J^*} a_j^{\mathrm{opt}}$. The formula $F_2(\mathbf{a}(\Gamma)) \leq (q^{\varepsilon}-1)z^*$ implies that either $z^{\mathrm{opt}} \leq 2q^{\varepsilon}z^*$ or $F_2(\mathbf{a}(\Gamma)) \leq \frac{1-q^{-\varepsilon}}{2} z^{\mathrm{opt}}$. In the first case, we have $2q^{\varepsilon}$-approximation since the objective function value of our solution is at least z^*. In the second case, with probability 0.8, $z^* \geq z^{\mathrm{opt}} - (F_1(\mathbf{a}(\Gamma)) + F_2(\mathbf{a}(\Gamma))) \geq \frac{q^{-\varepsilon}}{2} z^{\mathrm{opt}}$, and hence we also attain $2q^{\varepsilon}$-approximation.

5 Concluding remarks

It is important to seek for more practical algorithms with approximation ratios as good as those given in this paper. Our algorithms are not practically efficient, but we may have good solution by taking a smaller number of instances combined with heuristics so that the process finishes within practical computation time. For example, Lemma 2 implies that we can iterate our procedure given there by applying it to $\mathbf{a} = \mathbf{a}(\mathbf{x}^1)$ to have a new solution $(\mathbf{x}^2, \mathbf{y}^2)$, and continue if we have improvement. This is a kind of local search method that system engineers tend to try without considering its performance guarantee. Our analysis implies a principle to design its multi-start version that has a theoretical guarantee, and analysis considering the effect of the local search is remained as a research problem.

Finally, it would be nice if we can directly approach to the nonbipartite dense bipartite subgraph problem removing the factor of 2 caused by the ratio for the max-cut.

References

[1] R. Agrawal, T. Imielinski, and A. Swami, Mining Association Rules between Sets of Items in Large Database, *Proc. SIGMOD'93* (1993), pp. 207-216.

[2] R. Agrawal and R. Srikant, Fast Algorithms for Mining Association Rules, *Proc. VLDB* (1994), pp. 487-499.

[3] S. Arora, D. Karger, and M. Karpinski, Polynomial Time Approximation Scheme for Dense Instances of $\mathcal{NP}$-hard problems, *Proc. STOC'95* (1995), pp. 284–293.

[4] Y. Asahoro, R. Hassin, and K. Iwama, Complexity of Finding Dense Subgraphs, *Discrete Applied Math.*, **121** (2002), pp. 15–26.

[5] Y. Asahiro, K. Iwama, H. Tamaki, and T. Tokuyama, Greedily Finding Dense Subgraphs, *Journal of Algorithms*, **34** (2000), 203-221.

[6] K. Clarkson, P. Shor, Application of Random Sampling in Computational Geometry II, *Discrete & Computational Geometry* **4** (1989) 387-421.

[7] A. Czygrinow, Maximum Dispersion Problem in Dense Graphs, *Operation Research Letters* **27** (2000) 223-227.

[8] U. Feige, Relations between Average Case Complexity and Approximation Complexity, *Proc. STOC 02* (2002) 534-543.

[9] U. Feige, G. Kortsarz, D. Peleg, The Dense *k*-Subgraph Problem, *Algorithmica* **29** (2001) 410-421.

[10] D. S. Hochbaum, Approximating clique and biclique problems, *J. Algorithms* **29** (1998), 174–200.

[11] G. Gallo, M.D.Grigoriadis, R. E. Tarjan, A Fast Parametric Maximum Flow Algorithm and Applications, *SIAM J. Comput.* **18** (1989) 30-55.

[12] G. Kortsarz, D. Peleg, On Choosing a Dense Subgraph, *Proc. FOCS93* (1993) 692-701.

[13] K. Makino, T. Uno, New Algorithms for Enumerating All Maximal Cliques, *Proc. SWAT04, LNCS 3111* (2004) pp. 260-272.

[14] R. Motowani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, 1995.

[15] C. M. Procopiuc, M. Jones, P.K. Agarwal, and T.M. Murali, A Monte Carlo Algorithm for Fast Projective Clustering, *Proc. ACM SIGMOD* 2002, pp. 418-427.

[16] N. Pasquier, Y. Bastide, R. Taouil, and L. Lanke, Discovering Frequent Closed Itemsets for Association Rules, *Proc. ICDT'99* (1999), pp.398-416.

Primal-dual approach for directed vertex connectivity augmentation and generalizations

LÁSZLÓ A. VÉGH *

Department of Operations Research
Eötvös University,
MTA-ELTE Egerváry Research Group (EGRES)
Budapest, Hungary
veghal@cs.elte.hu

ANDRÁS A. BENCZÚR*

Department of Operations Research
Eötvös University,
Computer and Automation Institute,
Hungarian Academy of Sciences Budapest, Hungary
benczur@cs.elte.hu

Abstract: In their seminal paper, Frank and Jordán show that a large class of optimization problems including certain directed graph augmentation ones fall into the class of *covering supermodular functions over pairs of sets*. They also give an algorithm for such problems, however that relies on the ellipsoid method. Prior to our result, combinatorial algorithms existed only for the 0–1 valued problem. Our key result is a combinatorial algorithm for the general problem that includes directed vertex or $S-T$ connectivity augmentation. The algorithm is based on the second author's previous algorithm for the 0–1 valued case.

Our algorithm uses a primal-dual scheme for finding covers of partially ordered sets that satisfy natural abstract properties as in Frank and Jordán. For an initial (possibly greedy) cover the algorithm searches for witnesses for the necessity of each element in the cover. If no two (weighted) witnesses have a common cover, the solution is optimal. As long as this is not the case, the witnesses are gradually exchanged by smaller ones. Each witness change defines an appropriate change in the solution; these changes are finally unwound in a shortest path manner to obtain a solution of size one less.

Keywords: connectivity augmentation, bisupermodular functions

1 Introduction

Frank and Jordán [8] introduced the problem *covering supermodular functions over pairs of sets*, and showed that it contains various directed connectivity augmentation problems. They give an algorithm that uses the ellipsoid method. In this paper we present a combinatorial algorithm for the general problem, and show how it applies for the connectivity augmentation cases. Previously, combinatorial algorithms existed only for special problems [5, 6] and for the 0–1 valued case [2, 7]. For the problem of increasing directed vertex connectivity to target value k, the best previous combinatorial algorithm has running time polynomial in n but exponential in k [9].

The central example of covering supermodular functions over pairs of sets is finding the minimum number of directed edges that make a directed graph G k-vertex-connected. We may consider all cuts of G with less than k vertices as set pairs (X,Y) of the vertex set where X is the source and Y is the sink side of the cut (recall the graph is directed). For a directed cut with sides X and Y, let

$$p(X,Y) = \max\{0, k-(|V|-|X|-|Y|)\}$$

denote the number of vertices "missing" for a k-connected graph; for all other pairs X, Y let $p(X,Y)=0$. The graph becomes k-connected iff for all X and Y we add at least $p(X,Y)$ edges that lead from X to Y. The running time of our algorithm is $O(k^4 \cdot n^5)$ for this problem.

The above demand function p satisfies the following crossing supermodular property: whenever $X \cap X' \neq \emptyset$, $Y \cap Y' \neq \emptyset$ and $p(X,Y) > 0$, $p(X',Y') > 0$,

$$p(X \cap X', Y \cup Y') + p(X \cup X', Y \cap Y') \geq p(X,Y) + p(X',Y').$$

Another problem that falls into the class of covering set pairs is increasing directed $S-T$ vertex or edge connectivity to target value k by adding a minimum number of edges between S and T [8]. For two possibly overlapping vertex sets S and T, the $S-T$ connectivity is the maximum number of directed vertex (or edge) disjoint paths that connect pairs of vertices with head in S and tail in T. Yet another remarkable problem of this class is Győri's rectangle cover problem [12, 10, 3].

*Research is supported by the Hungarian National Foundation for Scientifi c Research Grant, OTKA T037547 and by European MCRTN ADONET, Grant Number 504438.

In the heart of most results related to covering problems over set pairs we find Dilworth's theorem stating that the minimum number of chains that cover a partially ordered set is equal to the maximum number of pairwise incomparable elements of the set. Both the non-combinatorial algorithm [8] and certain combinatorial ones [5, 6, 7, 3] start with a reduction to chain covers as in Dilworth's theorem.

Similar to most results related to covering problems over set pairs [8, 5, 6, 7, 3] we find Dilworth's chain cover theorem in the heart of our new combinatorial algorithm. However we circumvent the reduction to Dilworth's theorem; instead we give a more general algorithm that, when specialized to posets, gives a slightly modified folklore Dilworth algorithm as described in [2]. The relation between supermodular functions over set pairs and Dilworth's chain covers is based on the following observation. It is easy to show that for each directed edge (x,y) there is a unique set pair (X,Y) with X minimum and Y maximum and another (X',Y') with Y' minimum and X' maximum with $x \in X, X'$ and $y \in Y, Y'$. All other (X'',Y'') satisfy this property iff $X \subseteq X'' \subseteq X'$ and $Y \supseteq Y'' \supseteq Y'$. Thus if we define a partially order with the above "skew" containment relation, we get the problem of covering a partially ordered set by intervals, a direct generalization of Dilworth's problem.

Our algorithm is based on the unweighted one of [2] that directly generalizes a Dilworth algorithm. In the weighted case we start out with a multichain version of Dilworth's problem where poset elements have weights and the total number of chains containing an element must be at least its weight. We consider multiple copies of the same chain instead of weighted chains; our algorithm is pseudo-polynomial in this sense.

We construct an optimum interval cover by starting with an arbitrary greedy cover and gradually improve it in a primal-dual augmenting path manner that mimics standard Dilworth algorithms. Algorithms for Dilworth's theorem are based on a reduction to the bipartite matching problem [11]. When we unfold the reduction of Dilworth's theorem to bipartite matchings, we find that the classical alternating path matching algorithm translates into an algorithm that (i) maintains one element for each chain (one for each copy that may be different if a chain has multiple copies) as a candidate dual optimum; (ii) terminates with optimum if no element occurs more than its weight and they are pairwise incomparable when multiple copies are ignored; finally (iii) otherwise uses these elements to guard exchanges in chain parts such that one of the chains eventually becomes unnecessary for the cover. Such a direct Dilworth algorithm is described by Frank [4]. We remark that the current best bipartite matching algorithm is given in [13] and for Dilworth's problem in [3].

In the bulk of this paper we describe our algorithm that solves certain directed augmentation problems via a reduction to covering a poset by weighted intervals where poset elements are weighted by a supermodular function p. As shown in Section 2, this covering problem is equivalent with that considered by Frank and Jordán [8]. Thus our algorithm applies among others to the task of increasing directed vertex connectivity or directed S–T edge connectivity to a target value.

The rest of the paper is organized as follows. In Section 2 we give the main definitions and state the equivalence of our theorem with that of Frank and Jordán [8]. In Section 3 we first give an overview of the primal-dual procedure, then in separate subsections show the key Procedures PUSHDOWN and REDUCE and in separate subsections show their correctness. Finally in Section 4 we briefly elaborate on the running times for the augmentation problems.

2 Poset properties of the Frank–Jordán set pairs

Frank and Jordán [8] introduce systems of set pairs closed under a certain "skew intersection" operation defined next. Let two members (X^-,X^+) and (Y^-,Y^+) be called **dependent** if both $X^- \cap Y^-$ and $X^+ \cap Y^+$ are nonempty; otherwise they are **independent**. Then for all dependent pairs,

$$(X^- \cap Y^-, X^+ \cup Y^+),\ (X^- \cup Y^-, X^+ \cap Y^+)$$

are also members of the set system. A function p over the system of set pairs satisfies the *crossing supermodular* property if for all dependent (X^-,X^+) and (Y^-,Y^+) with $p(X^-,X^+) > 0$ and $p(Y^-,Y^+) > 0$,

$$p(X^- \cap Y^-, X^+ \cup Y^+) + p(X^- \cup Y^-, X^+ \cap Y^+) \geq p(X^-,X^+) + p(Y^-,Y^+)$$

They prove the following theorem:

Theorem 1 (Frank and Jordán [8]) Let p be a crossing supermodular function over a system of set pairs. The minimum cardinality of an edge multiset $\{e = (v_1,v_2)\}$ such that for all (X^-,X^+) there exist $p(X^-,X^+)$ edges with $v_1 \in X^-$, $v_2 \in X^+$ is equal to the maximum sum of p-values for pairwise independent elements in the system of set pairs.

We give an alternate proof of an equivalent form of this theorem stated as a poset covering problem. The proof is via a combinatorial algorithm.

Definition 2 Consider a poset $\mathscr{P}$; let $u,v \in \mathscr{P}$ be called **dependent** if $\exists m,M$ with $m \leq u \leq M$ and $m \leq v \leq M$; otherwise they are **independent**. For all dependent u and $v \in \mathscr{P}$ two operations $\vee$ and $\wedge$ are uniquely defined as

$$\begin{aligned} s \vee t &= \min\{x : x \geq s, x \geq t\}; \\ s \wedge t &= \max\{x : x \leq s, x \leq t\}. \end{aligned} \tag{1}$$

We say that for a minimal element m and a maximal element M, the set $\{x : m \leq x \leq M\}$ is the **interval** $[m,M]$. Let $\mathscr{P}$ satisfy furthermore the **strong interval property**: for every interval $[m,M]$,

$$u \wedge v \in [m,M] \text{ implies } u \in [m,M] \textbf{ or } v \in [m,M],$$

and the same holds with $u \wedge v$ replaced by $u \vee v$.

The notion of a crossing supermodular function p over the poset follows similar to set pairs: for all dependent x and y with $p(x) > 0$ and $p(y) > 0$ we require

$$p(x \vee y) + p(x \wedge y) \geq p(x) + p(y)$$

We say that $\mathscr{I}$ **covers** the function p if for every x at least $p(x)$ intervals contain x. An element v is called **tight** if we have equality. By the strong interval property all intervals that cover $x \vee y$ or $x \wedge y$ also cover x or y and if they cover both, then they cover all four. This fact has two important consequences.

Lemma 3 If x and y are two dependent tight elements with $p(x) > 0$, $p(y) > 0$, then both $x \vee y$ and $x \wedge y$ are tight. □

Lemma 4 If x and y are two dependent tight elements with $p(x) > 0$, $p(y) > 0$, and the interval $I_j = [m_j, M_j]$ contains x, then I_j contains at least one of $x \vee y$ and $x \wedge y$. In other words, either $y \leq M_j$ or $m_j \leq y$. □

Given the notion of the cover problem for a poset with the strong interval property, we next show its equivalence with the Frank–Jordán set pair cover problem. First we show the equivalence of the poset properties.

Theorem 5 Let $\mathscr{P} \subseteq \{(X^-, X^+) : X^- \subseteq \mathscr{X}^-, X^+ \subseteq \mathscr{X}^+\}$ such that for all dependent $x = (X^-, X^+)$ and $y = (Y^-, Y^+)$,

$$\begin{aligned} x \wedge y &= (X^- \cap Y^-, X^+ \cup Y^+) \in \mathscr{P}, \\ x \vee y &= (X^- \cup Y^-, X^+ \cap Y^+) \in \mathscr{P}. \end{aligned}$$

For any $x = (X^-, X^+)$ and $y = (Y^-, Y^+)$ let $x \leq y$ iff $X^- \subseteq Y^-$ and $X^+ \supseteq Y^+$. Then $\mathscr{P}$ with operations $\vee$, $\wedge$ and $\leq$ over $\mathscr{P}$ satisfies Definition 2. Furthermore subfamilies

$$\{(X^-, X^+) : v_1 \in X^-, v_2 \in X^+\}$$

for pairs $v_1 \in \mathscr{X}^-$, $v_2 \in \mathscr{X}^+$ are either intervals themselves or contained by some intervals of $\mathscr{P}$. Furthermore for all intervals of $\mathscr{P}$ there exist v_1 and v_2 such that the interval can be given in such a form. □

Before giving our algorithm, we state our main result as a min-max formula.

Theorem 6 For a poset $\mathscr{P}$ as in Definition 2 and a crossing supermodular function p, the minimum number of intervals covering $\mathscr{P}$ is equal to the maximum of the sum of p values for pairwise independent elements of $\mathscr{P}$.

Theorem 5 implies that Theorem 1 is a special case of this theorem. It can also be shown, that Theorem 1 implies Theorem 6, hence they are equivalent.

3 The algorithm

Algorithm PUSHDOWN-REDUCE($\mathscr{I}$)
for $j = 1, \ldots, k$ do
  If I_j has no tight elements then
    return reduced cover $\{I_i : i = 1, \ldots, j-1, j+1, \ldots, k\}$
  $u_j^{(1)} \leftarrow$ maximal tight element of I_j
while exist $u_i^{(t)}$ and $u_j^{(t)}$ dependent such that $u_i^{(t)}$ may push $u_j^{(t)}$ down
  for $j = 1, \ldots, k$ do
    $u_j^{(t+1)} \leftarrow$ PUSHDOWN($j, t, \mathscr{I}$)
  $t \leftarrow t + 1$
return optimal dual solution $\{u_1^{(t)}, \ldots, u_k^{(t)}\}$

We give a brief overview of our algorithm for the 0–1 valued case first. The algorithm starts out with a (possible greedy) interval cover $I_1, \ldots, I_k$. In Algorithm PUSHDOWN-REDUCE we maintain a tight element $u_i \in I_i$ for each interval I_i as a witness for the necessity of I_i in the cover. As long as the set of witnesses are non-independent or in other words they do not form a dual solution, in Procedure PUSHDOWN we replace certain u_i by smaller elements. By such steps we aim to arrive in an independent system of witnesses. If witnesses are indeed pairwise independent, they form a dual solution with the same value as the primal cover solution, thus showing both primal and dual optimality. Otherwise the algorithm calls another Procedure REDUCE that exchanges interval endpoints so that we get an interval cover of size one less.

In order to handle weighted posets, technically we need to consider multisets of intervals and witnesses in our algorithm. We assume $I_1, \ldots, I_k$ may contain multiple elements and the same may happen to the set of witnesses. The next lemma shows that if the witnesses are pairwise independent *as a weighted set* instead of a multiset, then the solution is optimal.

Lemma 7 If for every i, j u_i and u_j are either independent or $u_i = u_j$, then the elements $\{u_1, ..., u_k\}$ give a dual optimal solution.

PROOF: It suffices to show that if for some poset element y there exists an i with $y = u_i$, then there exist exactly $p(y)$ such intervals I_j with $u_j = y$. Since $y = u_i$ is tight, there are exactly $p(y)$ intervals I_j with $y \in I_j$. Consider such an u_j now: u_i and u_j are either independent or $u_i = u_j$, but the first case is impossible since both of them are covered by I_j. Hence $u_j = u_i$ for all $p(y)$ values of j. □

3.1 The PUSHDOWN step

Our Algorithm PUSHDOWN-REDUCE (see box) tries to push witnesses down along their intervals in iterations $t = 1, 2, \ldots$ until they satisfy the requirements of Lemma 7; witnesses are superscripted by the iteration value (t). Given two intervals $I_i = [m_i, M_i]$ and $I_j = [m_j, M_j]$ and two tight elements $u \in I_i$ and $v \in I_j$, we say that u may **push** v **down** (with regard to I_i) if u and v are dependent and $v \not\leq M_i$.

Lemma 8 If $u, u' \in I_i$ and $v \in I_j$ are tight with $u' \leq u$ and u may push v down, then u' may also push v down.

Lemma 9 Suppose $u \in I_i$, $v \in I_j$, $v' \in I_h$ are tight elements. Let v and v' be dependent. If u may push $v \vee v'$ down, then it may also push either v or v' down.

PROOF: Since u may push $v \vee v'$ down, we have $v \vee v' \not\leq M_i$, hence by Lemma 4 we have $m_i \leq v \vee v'$. By the strong interval property either $m_i \leq v$ or $m_i \leq v'$. By symmetry let us consider the first case; in this case v and u are also dependent since their common lower bound is m_i and their common upper bound is that of u and $v \vee v'$. If $v \not\leq M_i$, then u may push v down. Suppose now $m_i \leq v \leq M_i$. We have $v' \not\leq M_i$ since u may push $v \vee v'$ down and thus $v \vee v' \not\leq M_i$. Then by applying Lemma 4 for v and v' it follows that $m_i \leq v'$, hence u and v' are dependent. Finally by $v' \not\leq M_i$ we get that u may push v' down. □

Procedure PUSHDOWN$(j, t, \mathscr{I})$
  $V \leftarrow \{x : m_j \leq x \leq u_j^{(t)}$, x tight and $\forall i = 1, \ldots, k$
    $u_i^{(t)}$ may not push x down$\}$
  If $V = \emptyset$ then
    $t^* \leftarrow t$;
    return REDUCE$(j, t^*, \mathscr{I})$
  else return the maximal $x \in V$

The actual change of a witness $u_j^{(t)}$ is performed in Procedure PUSHDOWN (see box). We select all tight elements $x \in I_j$, $x \leq u_j^{(t)}$ into a set V that cannot be pushed down with elements $u_i^{(t)}$. If V is nonempty, we next show that it has a unique maximal element; we use this element as the new witness $u_j^{(t+1)}$. The next lemma follows easily by Lemma 9.

Lemma 10 In Procedure PUSHDOWN either $V = \emptyset$ or else it has a unique maximal element.

If we find no dependent pair of witnesses such that one may push the other down, then we will show that the witnesses are pairwise independent or equal and thus the solution is optimal. And as long as we find pairs such that one may push the other down, in the main loop of Algorithm PUSHDOWN-REDUCE we record a possible interval endpoint change by pushing one witness lower in its interval; these changes are then unwound to a smaller cover as shown in Section 3.3.

3.2 Proof for termination without REDUCE

We turn to the first key step in proving the correctness: we show that if the algorithm terminates without calling Procedure REDUCE, then $u_i^{(t)}$ are pairwise independent or equal; in other words if none of them may be pushed down by another, then the solution is optimal.

Theorem 11 If the algorithm terminates without calling Procedure REDUCE, then $u_i^{(t)}$ and $u_j^{(t)}$ dependent implies $u_i^{(t)} = u_j^{(t)}$.

The theorem is an immediate consequence of the next lemma.

Lemma 12 Assume that $t_1 \le t_2$, and $u_i^{(t_2)}$ and $u_j^{(t_1)}$ are dependent, and $u_j^{(t_1)}$ may not push $u_i^{(t_2)}$ down. Then $u_i^{(t_2)} \le u_j^{(t_1)}$.

This lemma is used not only for proving Theorem 11 but also in showing the correctness of Procedure REDUCE in Section 3.3 via the next immediate corollary.

Corollary 13 If $u_j^{(t)}$ and $u_i^{(t+1)}$ are dependent, then $u_i^{(t+1)} \le u_j^{(t)}$

In the proof of Lemma 12 we need to characterize elements that cause witness u_j move below a certain tight element y. Assume that for some tight $y \in I_j$ and t we have $y \not\le u_j^{(t)}$. Since $u_j^{(1)}$ is maximal tight, we may select the unique t_0 with $y \le u_j^{(t_0)}$ but $y \not\le u_j^{(t_0+1)}$. In step PUSHDOWN$(j, t_0, \mathscr{I})$ we must have an $u_d^{(t_0)}$ that may push y down. We will use this in the following special case:

Lemma 14 Assume that z is tight and dependent with $u_j^{(t)}$. Assume furthermore that $z \not\le u_j^{(t)}$ and $z \le M_j$. Then there exists $t_0 < t$ and d such that $u_d^{(t_0)}$ may push $u_j^{(t)} \vee z$ down. In addition $u_d^{(t_0)}$ may also push z down.

PROOF: The first part follows from the above observations, and the second part can be obtained using Lemma 9. □

PROOF:[Lemma 12] $u_i^{(t_2)} \le M_j$, since $u_j^{(t_1)}$ may not push $u_i^{(t_2)}$ down. If $u_i^{(t_2)} \not\le u_j^{(t_1)}$, then the conditions of Lemma 14 hold with $z = u_i^{(t_2)}$ and $t = t_1$. Thus we have some $t_0 < t_1$ and d such that $u_d^{(t_0)}$ may push $z = u_i^{(t_2)}$ down. But then $u_d^{(t_2-1)}$ may also push $u_i^{(t_2)}$ down by Lemma 8. This latter contradicts the choice of $u_i^{(t_2)}$ as the maximum tight element that may not be pushed down in PUSHDOWN$(i, t-1, \mathscr{I})$. □

3.3 The REDUCE step

Procedure REDUCE$(j, t^*, \mathscr{I})$
  $j_1 \leftarrow j$;
  for $t = t^*, \ldots, 1$ do
    $s \leftarrow t^* + 1 - t$
    $q \leftarrow$ minimal tight element in $[m_{j_s}, M_{j_s}]$
    $j_{s+1} \leftarrow$ value $\ell \neq j_s$ such that $u_\ell^{(t)}$ may push q down
    $m_{j_s} \leftarrow m_{j_{s+1}}$
  return reduced cover $\{[m_i, M_i] : 1 \le i \le k, i \neq j_{t^*+1}\}$.

So far we have proved that if the initial primal solution is optimal, then the algorithm finds a dual optimum proof of this fact. Now we turn to the second scenario when one witness eventually disappears from the dual solution. In this case we unwind the steps to find a cover of size one less in Procedure REDUCE based on interval exchanges at certain pairs of tight poset elements.

Our aim in Procedure REDUCE (see box) is to repeatedly pick an interval $[m_{j_s}, M_{j_s}]$, select its minimal tight element q and try to find another interval $[m_{j_{s+1}}, M_{j_{s+1}}]$ such that if we replace $[m_{j_s}, M_{j_s}]$ by $[m_{j_{s+1}}, M_{j_s}]$, then we obtain another cover. In particular we want q remain covered and the minimum tight element of $[m_{j_{s+1}}, M_{j_s}]$ increase. We ensure the latter by making sure that the new interval $[m_{j_{s+1}}, M_{j_s}]$ adds a new cover to certain witness $u_{j_{s+1}}$.

While the first step of the procedure is well-defined since we call Procedure REDUCE exactly when the minimal tight $q \in I_j$ for $j = j_1$ is pushed down by certain other $u_\ell^{(t^*)}$, the existence of such an ℓ is by no means obvious for all the other

iterations of the main loop. The existence of $\ell = j_{s+1}$ for $s > 1$ is reduced to the properties of the first iteration by a special induction.

We show that if the procedure terminates in a single iteration, then we obtain a cover with one interval less. The first lemma shows that Procedure REDUCE unwinds PUSHDOWN steps with a special property that enables us to make interval endpoint exchanges that, in the particular case of $t^* = 1$, result in replacing two intervals by one in a valid cover. In Procedure REDUCE we define $\ell = j_{s+1}$ such that $u = u_\ell^{(t)}$ may push a *minimal* tight element q down. The lemma below shows that if we replace $[m_\ell, M_\ell]$ by $[m_j, M_\ell]$ for $j = j_s$, then the minimal tight $q \in I_j$ gets an additional cover and hence it becomes no longer tight.

Lemma 15 Let q be the minimal tight element of I_j. If $u \in I_\ell$ may push q down, then $w \le M_j$. □

Lemma 16 In Procedure REDUCE$(j, t^*, \mathscr{I})$ $\mathscr{I}' = \mathscr{I} - [m_{j_1}, M_{j_1}] + [m_{j_2}, M_{j_1}]$ forms an interval cover. □

Lemma 17 If $t^* = 1$, Procedure REDUCE$(j, t^*, \mathscr{I})$ returns an interval cover. □

The existence of all further j_ℓ in Procedure REDUCE as well as the correctness of the algorithm is proved by "rewinding" the algorithm after the first iteration of Procedure REDUCE and showing that each step is repeated identical up to iteration $t^* - 1$.

Notice that Procedure REDUCE exchanges elements in the cover by traversing a virtual "augmenting path". Instead of directly proving augmenting path properties, our proof can be considered as a special induction by executing the main loop of the procedure step by step and after each iteration rewinding the main algorithm. In the analogy of network flow algorithms, this may correspond to analyzing an augmenting path algorithm by choosing path edges backward from the sink, changing the flow along this edge to a preflow, and at each step proving that the remaining path augments the flow.

So far we see that the *last* step of REDUCE results in a new interval cover; unless in the lucky scenario of $t^* = 1$ the size of the cover remains the same. By the next central theorem however we may inductively use the modified cover to re-run the algorithm with a value of t^* one less than before; eventually we reach $t^* = 1$ and a cover of size one less.

Theorem 18 For $t^* > 1$ and $\mathscr{I}'$ as in Lemma 16, Algorithm PUSHDOWN-REDUCE performs the exact same steps as with input $\mathscr{I}$ until iteration $t^* - 1$ when REDUCE$(j_2, t^* - 1, \mathscr{I}')$ is called.

By using the above theorem inductively for $t^*, t^* - 1, \ldots, 1$ we prove that REDUCE finds an interval cover of size one less than before. This completes the correctness analysis of Procedure REDUCE.

To prove Theorem 18 now we define elements that are no longer tight and elements that become tight in the new cover:

Lemma 19 For $\mathscr{I}$ and $\mathscr{I}' = \mathscr{I} - [m_{j_1}, M_{j_1}] + [m_{j_2}, M_{j_1}]$ as in Procedure REDUCE$(j, t^*, \mathscr{I})$, let

$$Z_1 = \{x \text{ tight in } \mathscr{I} \text{ and } x \text{ not tight in } \mathscr{I}'\},$$
$$Z_2 = \{x \text{ not tight in } \mathscr{I} \text{ and } x \text{ tight in } \mathscr{I}'\}.$$

Then

$$Z_1 \subseteq \{x : x \in [m_{j_2}, M_{j_1}], x \not\ge m_{j_1}\} \tag{2}$$
$$Z_2 \subseteq \{x : x \in [m_{j_1}, M_{j_1}], x \not\ge m_{j_2}\}. \tag{3}$$

Hence the same elements are tight in I_{j_1} for $\mathscr{I}$ as in $[m_{j_2}, M_{j_1}]$ for $\mathscr{I}'$.

Next we show that the algorithm proceeds identical for $\mathscr{I}$ and $\mathscr{I}'$ for $t < t^*$. The proof is by induction and is based on the fact that the key elements used in defining $u_i^{(t)}$ may not belong to $Z_1 \cup Z_2$.

Lemma 20 Let $u_i^{\prime(t)}$ denote elements selected by Algorithm PUSHDOWN-REDUCE with input $\mathscr{I}'$ with the convention that $u_{j_1}^{\prime(t)}$ belongs to the modified interval $[m_{j_2}, M_{j_1}]$. Then for all $t < t^*$ we have $u_i^{(t)} = u_i^{\prime(t)}$. □

We complete the proof of Theorem 18 by the following lemma.

Lemma 21 When run with input $\mathscr{I}'$, Procedure REDUCE is called in iteration $t^* - 1$ with $j = j_2$. □

4 Running times

In the application of our general algorithm to connectivity augmentation problems we have to be careful since we typically have an exponential size poset implicitly given as a set of (directed) cuts. For the implementation of the connectivity augmentation problems we need a more technical reformulation of the Procedure PUSHDOWN, which we omit in this paper. This will provide the computation of a new witness in Procedure PUSHDOWN by a sequence of BFS computations, which we will call basic steps.

The number of mincut computations is polynomial in the number of initial intervals j and the length of a longest chain ℓ in the poset: we take $j\ell$ basic steps for one REDUCE while the latter may happen $O(j)$ times. For example for vertex connectivity problems, the former is $O(n^2)$ while the latter is $O(n)$, giving $O(n^7)$ running time for the algorithm.

By approximate augmentation results we may reduce the running time in the following scenarios. When we increase connectivity by only one, the approximate augmentation result of Jordán [14] can be used to reduce the number of initial intervals to $O(n)$ and gives an initial solution such that at most $O(k)$ REDUCE calls are neeed. This gives a running time of $O(k \cdot n^5)$ for this special problem. And for the general case the non-polynomial algorithm of [9] can easily be turned to a polynomial one that finds a solution with $O(k^4)$ more edges than the optimum. This replaces one $O(j)$ term by $O(k^4)$, thus resulting in a running time of $O(k^4 \cdot n^5)$.

Conclusion

We have given a combinatorial algorithm for covering posets satisfying a special property by the minimal number of intervals of the poset. As noticed by Frank and Jordán [8], the result can be applied for certain directed edge augmentation problems. The existence of a strongly polynomial combinatorial algorithm, however, remains open. Another major open problem regards the complexity of undirected augmentation.

One may wonder of how strong the generalizational power of the interval covering problem. Two algorithmically equivalent problems, Dilworth's chain cover and bipartite matching, are special cases of interval covers; our algorithm generalizes the standard augmenting path matching algorithm. One may ask whether the network flow problem as different algorithmic generalization of matchings could also fit into our framework. Or, extending the question of [15], can we at least tell the hierarchy of hardness of the interval cover, Dilworth, (bipartite) matching and maximum flow problems? We might also hope that ideas such as capacity scaling, distance labeling and preflows [1] that give polynomial algorithms for network flows can be used in the construction of a strongly polynomial algorithm for the interval covering problem.

Finally one may be interested in the efficiency of our algorithm for the particular problems that can be handled. Here particular implementations and good oracle choices are needed. We may want to reduce the number of mincut computations needed by polynomial size poset representations. One might also be able to give improvements in the sense of the Hopcroft–Karp matching algorithm [13].

References

[1] Ahuja, R.K., T.L. Magnanti and J.B. Orlin, *Network Flows*, Prentice Hall (1993).

[2] Andr´as A. Bencz´ur, Pushdown-Reduce: An algorithm for connectivity augmentation and poset covering problems. *Discr. Appl. Math*, pp 233-262 vol 129 (2003)

[3] Bencz´ur, A.A., J. F¨orster and Z. Kir´aly, Dilworth's Theorem and its application for path systems of a cycle—implementation and analysis. *Proc. European Symp. Alg., Springer Lecture Notes in Computer Science* 1643:598–509 (1999)

[4] Frank, A., *Combinatorial algorithms, algorithmic proofs.* Doctoral Thesis, Budapest, E¨otv¨os University (1976), in Hungarian

[5] Frank, A., Finding minimum generators of path systems, *JCT B* **75** (1999), pp. 237–244.

[6] Frank, A., Finding minimum weighted generators of a path system, Contemporary Trends in Discrete Mathemaics (eds.: R.L. Graham, J. Kratochvil, J. Nesetril, and F.S. Roberts), *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, Volume **49** (1999), pp. 129–138.

[7] Frank, A., Finding minimum edge-coverings of pairs of subsets, EGRES TECHNICAL REPORT SERIES (2001). Available at `http://www.cs.elte.hu/egres/`

[8] Frank, A. and T. Jord´an, Minimal edge-coverings of pairs of sets, *JCT* B **65** (1995), pp. 73–110.

[9] Frank, A. and Jord´an, T., Directed vertex-connectivity augmentation *Math. Prog. 84*, (1999), pp. 537-553

[10] Franzblau, D.S. and D.J. Kleitman, An algorithm for constructing polygons with rectangles, *Information and Control* **63** (1984), pp. 164–189.

[11] Ford, L.R. and D.R. Fulkerson, *Flows in Networks*, Princeton University Press (1962).

[12] Győri, E., A min-max theorem on intervals, *JCT* B **37** (1984), pp. 1–9.

[13] Hopcroft, J.E. and R.M. Karp, An $n^{5/2}$ algorithm for maximum matching in bipartite graphs, *SIAM J. Comp.* **2** (1973), pp. 225–231.

[14] Jord´an, T., Increasing the vertex-connectivity in directed graphs, In: Proceedings of the First Annual European Symposium on Algorithms, Bad Honnef, 1993, Springer Lecture Notes In Computer Science, Vol 726, pp. 236-247, Springer-Verlag, New York/Berlin, 1993

[15] Karger, D.R. and M.S. Levine, Finding Maximum Flows in Simple Undirected Graphs is Easier than Bipartite Matching, *30th ACM Symposium on Theory of Computing* (1998).

Improving Performance Ratios by Repeatedly Executing Approximation Algorithms for Several Graph Connectivity Related Problems

MAKOTO TAMURA

Graduate School of Engineering
Hiroshima University, Japan
tamura@infonets.hiroshima-u.ac.jp

SATOSHI TAOKA

Graduate School of Engineering
Hiroshima University, Japan
taoka@infonets.hiroshima-u.ac.jp

TOSHIMASA WATANABE

Graduate School of Engineering
Hiroshima University, Japan
watanabe@infonets.hiroshima-u.ac.jp

Abstract: The subject is to show that performance ratios can be improved by repeating approximation algorithms for several graph connectivity related problems that are known to be NP-complete: 2-vertex(2-edge, respectively)-connectivity augmentation for a set S of specified vertices of a given graph having a connected component containing S, $(\lambda+1)$-edge-connectivity augmentation of a given λ-edge-connected graph, extracting a strongly (λ-edge, respectively)-connected spanning subgraph of a given directed (undirected) graph. The paper proposes $(2-\frac{2}{|L|})$-approximation algorithms for the first three problems and $(2-\frac{2}{|V|})$-approximation ones for the last two problems, where L is the set of leaves of a certain tree constructed from a given graph. The common property of these problems is that they can be solved by handling or constructing one or more arborescences, and the point is that repeating an existing algorithm and selecting the best solution among those obtained improve each performance ratio by $2/|L|$ or $2/|V|$.

Keywords: edge-connectivity, vertex-connectivity, augmentation problems, NP-hardness, approximation algorithms

1 Introduction

1.1 Problem Definitions

The following graph connectivity related problems that are known to be NP-comple are considered: (1) the 2-*vertex-connectivity augmentation problem for specified vertices* (2**VCA-SV**); (2) the 2-*edge-connectivity augmentation problem for specified vertices* (2**ECA-SV**); (3) the $(\lambda+1)$-*edge-connectivity augmentation problem* ($(\lambda+1)$**ECA**); (4) the *strongly connected spanning subgraph problem* (**SCSS**); and (5) the *k-edge-connected spanning subgraph problem* (*k***ECSS**). Their definitions are as follows.

[2**VCA-SV** (2**ECA-SV**, respectively)] Given an undirected graph $G=(V,E)$, a spanning subgraph $G'=(V,E')$ of G, specified vertices $S\subseteq V$, and a nonnegative weight function $w:E\rightarrow R^+$, find a set $E''\subseteq E-E'$ with the minimum total weight, such that $G'+E''=(V,E'\cup E'')$ has at least two internally disjoint (edge disjoint) paths between any pair of vertices in S. If $S=V$ then 2**VCA-SV** (2**ECA-SV**) is denoted simply as 2**VCA** (2**ECA**), which is called the 2-*vertex-connectivity* (2-*edge-connectivity*) *augmentation problem*.

[$(\lambda+1)$**ECA**] Given an undirected graph $G=(V,E)$ with a nonnegative weight function $w:E\rightarrow R^+$ and a spanning subgraph $G'=(V,E')$ of G such that G' has λ edge disjoint paths between any pair of vertices, find a set $E''\subseteq E-E'$ with the minimum total weight such that $G'+E''=(V,E'\cup E'')$ has at least $(\lambda+1)$ edge disjoint paths between any pair of vertices.

[**SCSS** (*k***ECSS**, respectively)] Given a directed (undirected) graph $G=(V,E)$ with a nonnegative weight function $w:E\rightarrow R^+$, find a minimum weight spanning subgraph $G'=(V,E')$ of G such that G' has a directed path between any ordered pair of vertices (at least k edge disjoint paths between any pair of vertices).

1.2 Known Results

Best known results on 2**VCA-SV**, 2**ECA-SV**, $(\lambda+1)$**ECA**, **SCSS** and *k***ECSS** are summarized in Table 1. We omit describing how their research has been developed: see [5, 9, 12, 19, 21] for 2**VCA**, [4, 20, 28] for 2**VCA-SV**, [2, 5, 12, 13] for 2**ECA**,

Table 1: The best known results for the five problems and the main resutls of the paper

	The best known results			**Resutls in the paper**	
Problem	ref.	ratio	time complexity	ratio	time complexity
2VCA-SV with $\kappa(S;G')=1$	[20]	2	$O(\|V\|^3+\|V\|\|E\|\alpha(\|V\|,\|V\|))$	$2-\frac{2}{\|L\|}$	$O(\|V\|\|E\|+\|V\|^2\log\|V\|+\|L\|\|V\|^2)$
2ECA-SV with $\lambda(S;G')=1$	[14]	$2-\frac{2}{\|S\|}$	$O(\|V\|^2\log\|V\|)$	$2-\frac{2}{\|L\|}$	$O(\|V\|\|E\|+\|V\|^2\log\|V\|+\|L\|\|V\|^2)$
$(\lambda+1)$ECA	[17]	2	$O(\Delta+\|V\|\|E\|)$ if λ is odd $O(\Delta+\|E\|)$ if λ is even	$2-\frac{2}{\|L\|}$	$O(\Delta+\|L\|^2\|E\|+\|L\|\|V\|\log\|V\|)$ if λ is odd $O(\Delta+\|L\|\|E\|+\|L\|\|V\|\log\|V\|)$ if λ is even
SCSS	[5]	2	$O(\|E\|+\|V\|\log\|V\|)$	$2-\frac{2}{\|V\|}$	$O(\|V\|\|E\|+\|V\|^2\log\|V\|)$
kECSS	[13]	2	$O(k\|V\|(\|E\|+\|V\|\log\|V\|)\log\|V\|)$	$2-\frac{2}{\|V\|}$	$O(k\|V\|^2(\|E\|+\|V\|\log\|V\|)\log\|V\|)$

Note that $|L|\leq|S|\leq|V|$, and Δ is the time complexity of finding a structural graph of a given graph G

[10, 14, 27, 28] for 2**ECA-SV**, [15, 17, 26, 16] for $(\lambda+1)$**ECA**, [2, 5] for **SCSS**, and [13] for k**ECSS**. The algorithms for finding a mimimum arborescence [8] or a minimum packing arboresence [7] has been utilized in designing these algorithms.

1.3 Main Results

Approximation algorihms *R2VS* for **2VCA-SV**, *R2ES* for **2ECA-SV**, *RMW+1* for $(\lambda+1)$ECA, *RSCS* for **SCSS**, and *RkECS* for k**ECSS** are proposed. Their performance ratios and time complexity are summarized in Table 1. The common property of these problems is that they can be solved by handling or constructing one or more arborescences, and the point is that repeating an existing algorithm and selecting the best solution among those obtained improve each performance ratio by $2/|L|$ or $2/|V|$.

Becase of space limitation, the details of *R2VS* for **2VCA-SV** is described, while only the outlines of other algorithms are given due to shortage of space.

2 Preliminaries

Technical terms not specified here can be identified in [3, 27].

An undirected (or directed) graph is often denoted as $G=(V,E)$, V and E are often written as $V(G)$ and $E(G)$, respectively, and a directed graph is often written as $\overrightarrow{G}=(V,\overrightarrow{E})$ or $\overrightarrow{G}=(V(\overrightarrow{G}),E(\overrightarrow{G}))$.

For a weight function $w:E\to R^+$ (nonnegative real numbers), we denote as $w(E')=\sum_{e\in E'}w(e)$ for $E'\subseteq E$. In an undirected graph, an edge with endvertices u,v is denoted by (u,v). A directed edge from u to v is denoted as $\langle u,v\rangle$. For $\langle u,v\rangle$, u is called the *parent* of v and v is called the *child* of u. For a set E' of edges, let $G+E'$ denote the graph $(V(G),E(G)\cup E')$. If $E'=\{e\}$ then we denote $G+e$. For two sets P and Q, let $P-Q=\{x\in P\mid x\notin Q\}$. For $V_1\subset V$ and $E_1\subset E$, let $G-(V_1\cup E_1)=(V-V_1,E-(E_1\cup E(V_1)))$, where $E(V_1)=\{(u,v)\in E\mid\{u,v\}\cap V_1\neq\emptyset\}$. $G-\{x\}$ is simply denoted as $G-x$. For a set $X\subseteq E(G)$ ($Y\subseteq V(G)$, respectively), a subgraph having $\{u,v\in V(G)\mid(u,v)\in X\}$ as its vertex set and X as its edge set (having Y as its vertex set and $\{(u,v)\in E\mid u,v\in Y\}$ as its edge) is called an *induced subgraph* of G by X (by Y).

A path between u and v in an undirected graph G is denoted by $P(u,v;G)$ or $P(u,v)$. A directed (A weakly directed, respectively) path from u to v in a directed graph $\overrightarrow{G}$ is denoted by $P\langle u,v;\overrightarrow{G}\rangle$ or $P\langle u,v\rangle$ ($\widetilde{P}\langle u,v;\overrightarrow{G}\rangle$ or $\widetilde{P}\langle u,v\rangle$).

For a set E of edges, $V(E)$ denotes a set of all endvertices of edges in E. For two vertices $u,v\in V(G)$, let $\kappa(u,v;G)$ ($\lambda(u,v;G)$, respectively) denote the maximum number of pairwise internally disjoint (edge disjoint) paths between u and v. Let us denote $\kappa(S;G)=\min\{\kappa(u,v;G)\mid u,v\in S\}$ ($\lambda(S;G)=\min\{\lambda(u,v;G)\mid u,v\in S\}$) for a set of vertices $S\subseteq V$. If $S=V$ then we simply represent as $\kappa(G)$ ($\lambda(G)$). A *k-vertex-connected graph* (*k-edge-connected graph*) is a graph such that $\kappa(G)\geq k$ ($\lambda(G)\geq k$). A *nonseparable* graph is a connected graph with no cutvertices. A *block* (*2-edge-connected component*) of a graph is the vertex set of a maximal nonseparable (2-edge-connected) subgraph.

An *arborescence* is a weakly connected acyclic directed graph such that it has only one specified vertex r, called the *root*, having no entering edges, and for any vertex v except r, there is a directed path $P\langle r,v\rangle$ and v has exactly one entering edge. For an arborescence with the root r, a graph consisting of edges $\langle u,v\rangle$ such that the oppositely directed edges $\langle v,u\rangle$ are contained in the arborescence is called a *reverse arborescence* with the root r, where the root in the reverse arborescence is a vertex having no leaving edges. If there is a directed path $P\langle u,v\rangle$ in a (reverse) arborescence, then we say that u is an *ancestor* of v and v is a *descendant* of u. For a reverse arborescence with the root r, suppose that u is not a descendant of v and that v is not a descendant of u. Then a *nearest common descendant* of u and v is a common descendant of u and v such that it is the nearest among all such common descendants. A cutvertex (leaf, respectively) in a directed tree means a cutvertex (leaf) in the undirected tree which is obtained by replacing each edge $\langle u,v\rangle$ by an undirected one (u,v).

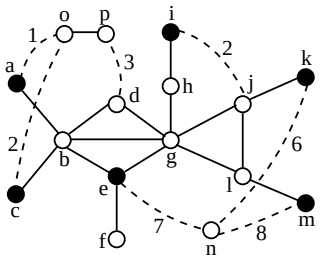


Figure 1: $G = (V,E)$ and $G' = (V,E')$. Solid lines are edges in E', closed vertices are those in $S \subseteq V$, and numbers shown beside edges are weights. Broken lines form an optimum solution $E^* \subseteq E - E'$ and the other edges in $E - E'$ are omitted for convenience.

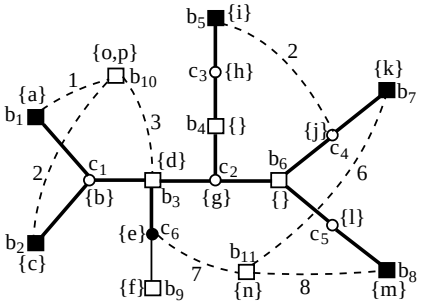


Figure 2: $G_B = (V_B, E_B)$ and a block-cutvertex-forest $T_B = (V_B, E'_B)$ constructed from G and G' shown in Fig. 1. Solid lines are edges in E'_B, broken lines are edges in $E_B - E'_B$, squares are block-vertices, circles are cutvertices, closed squares and circles are vertices in S_B, and $\beta^{-1}(u)$ for each $u \in V_B$ is shown in braces.

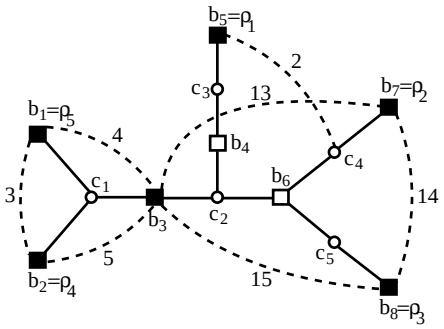


Figure 3: $G_P = (V_P, E_P)$ (to be defined in Step 2 of *R2VS*) and the path-tree $T_P = (V_P, E'_P)$ constructed from G_B and T_B shown in Fig. 2. Solid lines are edges in E'_P, broken lines are edges in $E_P - E'_P$ and closed squares and circles are vertices in S_P.

3 The 2-Vertex-Connectivity Augmentation Problem for Specified Vertices

This section proposes an approximation algorithm *R2VS* for 2**VCA-SV**. The performance ratio is $(2 - \frac{2}{|L|})$ if G' has a connected component containing S (that is, $\kappa(S;G') \geq 1$), where L is the set of leaves of a certain tree constructed from G' and S. Its time complexity is $O(|V||E| + |V|^2 \log|V| + |L||V|^2)$.

3.1 Definitions

3.1.1 Block-Cutvertex-Trees and Its Supergraphs

A *block-cutvertex-tree* described below is used to handle not only blocks but cutvertices as well [5, 9, 12, 21, 28]. As an instance of 2VCA-SV, suppose that $G' = (V,E')$ ($|V| \geq 3$) has a connected component containing S (see Fig. 1). We focus on blocks and cutvertices of G', where for any connected component V' in G' containing no vertices in S, we regard V' as a block of G' (such as the connected component $\{o,p\}$ in Fig. 1).

We construct a block-cutvertex-tree $T_B = (V_B, E'_B)$ from G' as follows (see Fig. 2). Any block or any cutvertex in G' is represented as a new vertex, called a block-vertex or a cutvertex, respectively. Let V_{b_B} or V_{c_B} be the set of block-vertices or of cutvertices (given as new vertices), respectively, and denote as $V_B = V_{b_B} \cup V_{c_B}$. Let E'_B be the set of edges (u,v) such that u is a block-vertex and v represents an individual cutvertex contained in the block corresponding to u in G'. Since all blocks of G' can be found in $O(|V| + |E'|)$ time [1], the block-cutvertex-tree of G' can be constructed in $O(|V| + |E'|)$ time.

Observation 1 T_B consists of a tree and zero or more isolated vertices. Block-vertices and cutvertices appear alternately in the tree. □

Definition 2 For each vertex u of T_B, let $\beta^{-1}(u)$ be the set of vertices of G' (see sets represented as $\{\ldots\}$ in Fig. 2) satisfying the following (i) or (ii). (i) If u is a cutvertex of T_B then $\beta^{-1}(u) = \{u'\}$, where u' is a cutvertex of G' that corresponds to u. (ii) If u is a block-vertex of T_B then $\beta^{-1}(u) = B_u - V_c$, where B_u is a block corresponding to u and V_c is the set of cutvertices of G'. Each vertex $u \in V_B$ is often denoted as $\beta(x)$ for some $x \in \beta^{-1}(u)$. □

Observation 3 Consider the sets associated with the vertices of T_B. Each vertex of G' belongs to exactly one such set. □

Let $S_B = \{u \mid \beta^{-1}(u) \cap S \neq \emptyset\}$ and we partition S_B into the set S_{b_B} of block-vertices and the set S_{c_B} of cutvertices. We construct a supergraph $G_B = (V_B, E_B)$ of T_B as follows (see Fig. 2). For any pair of vertices $u,v \in V_B$, let $\widehat{E_B}$ be the set of edges (u,v) such that there exists an edge whose endvertices belong to $\beta^{-1}(u)$ and $\beta^{-1}(v)$ in $E - E'$. Let $E_B = E'_B \cup \widehat{E_B}$ and $w_B((u,v)) = min\{w((u',v')) \mid (u',v') \in E - E',\ u' \in \beta^{-1}(u),\ v' \in \beta^{-1}(v)\}$ for any $(u,v) \in \widehat{E_B}$.

Definition 4 For a supergraph $\widehat{G_B}$ of T_B, let us denote as $\kappa'(S_B; \widehat{G_B}) \geq 2$ if and only if, for any cutvertex $v \in V_{c_B}$, $\kappa(S_B; \widehat{G_B} - v) \geq 1$ (that is, $\widehat{G_B} - v$ has a connected component containing S_B). □

Note that there may exist a block vertex $v \in V_{b_B}$ such that $\kappa(S_B; \widehat{G_B} - v) = 0$ even if $\kappa'(S_B; \widehat{G_B}) \geq 2$. We can prove the following lemma from the property of a block-cutvertex-tree.

Lemma 5 If $\kappa(S;G) \geq 2$ then $\kappa'(S_B;G_B) \geq 2$. Conversely, for a set $E''_B \subseteq \widehat{E_B}$ if $\kappa'(S_B;T_B+E''_B) \geq 2$ then $\kappa(S;G'+E'') \geq 2$, where $E'' = \{(u',v') \in E - E' \mid (u,v) \in E''_B,\ u' \in \beta^{-1}(u),\ v' \in \beta^{-1}(v)\}$. □

Because of Lemma 5, we can restrict our attention to G_B and T_B.

3.1.2 Path-Trees and Its Supergraphs

Let T'_B be a minimal subtree of T_B containing S_B (denoted by thick bold lines in Fig. 2). Let $T_P = (V_P, E'_P)$ be the tree constructed from T'_B by deleting all leaves $v \in S_{c_B}$ (see c_6 in Fig. 2). T_P is called the *path-tree* [28]. Let us partition V_P as $V_P = V_{b_P} \cup V_{c_P}$, where V_{b_P} and V_{c_P} are the set of block-vertices and of cutvertices, respectively. Let $S_P = (S_B - C_L) \cup \{x \in V_{b_P} \mid x \text{ is adjacent to } u \in C_L \text{ in } T'_B\}$, where C_L is the set of cutvertices deleted in constructing T_P. (In Fig. 2, c_6 is deleted from S_B and b_3 is added to S_P.) Note that any leaf of T_P is contained in $S_P \cap V_{b_P}$ (see Fig. 3).

We construct a supergraph $G_P = (V_P, E_P)$ of T_P as follows (see Fig. 3). We set $w_B((u,v)) \leftarrow 0$ for any $(u,v) \in E'_B - E'_P$. Set $\widehat{E_P} \leftarrow \emptyset$. For any pair of vertices $u,v \in V_P$ with $(u,v) \notin E'_P$, if $G_B - E'_P$ has a path $P(u,v)$ such that $V(P(u,v)) \cap V_P = \{u,v\}$ then set $\widehat{E_P} \leftarrow \widehat{E_P} \cup \{(u,v)\}$ and let $w_P((u,v))$ be the shortest length of such paths with respect to the weight function w_B (see $E_P - E'_P$ whose edges are denoted by broken lines in Fig. 3). Let $E_P = E'_P \cup \widehat{E_P}$.

Definition 6 For a supergraph $\widehat{G_P}$ of T_P, let us denote as $\kappa'(\widehat{G_P}) \geq 2$ if and only if, for any cutvertex $v \in V_{c_P}$, $\kappa(\widehat{G_P} - v) \geq 1$ (that is, $\widehat{G_P} - v$ is connected). □

Note that there may exist a block vertex $v \in V_{b_P}$ such that $\kappa(\widehat{G_P} - v) = 0$ even if $\kappa'(\widehat{G_P}) \geq 2$. Since the path-tree T_P contains all cutvertices each of which separates some pair of vertices of S_B in T_B, we can easily prove the following lemma.

Lemma 7 If $\kappa'(S_B;G_B) \geq 2$ then $\kappa'(G_P) \geq 2$. Conversely, for a set $E''_P \subseteq \widehat{E_P}$ if $\kappa'(T_P+E''_P) \geq 2$ then $\kappa(S_B;T_B+E''_B) \geq 2$, where E''_B be the set of edges obtained by transforming each edge in E''_P. □

3.2 The Proposed Algorithm *R2VS* when $\kappa(S;G') = 1$

The algorithm *2-ABIS* of [24] or the algorithm *R2VS* to be proposed is a 2-approximation or $(2 - \frac{2}{|L|})$-approximation one, respectively, for 2**VCA-SV** when $\kappa(S;G') = 1$. *2-ABIS* utilizes the algorithm of [12] for solving 2**VCA**, while *R2VS* repeats *2-ABIS* as follows: it repeats selecting each leaf of a given path tree T_P and executing the algorithm of [12] to solve 2**VCA**, and then selects the best solution among those obtained. The proposed algorithm *R2VS* is stated in the following.

Algorithm *R2VS*;
/* **Input:** An undirected graph $G = (V,E)$, a set of specified vertices $S \subseteq V$, a spanning subgraph $G' = (V,E')$ of G with $\kappa(S;G') = 1$, and a nonnegative weight function $w : E \rightarrow R^+$. (See Fig. 1.)*/
/* **Output:** A set of edges $E'' \subseteq E - E'$. */

1. Construct a block-cutvertex-tree $T_B = (V_B, E'_B)$ of G' and a supergraph $G_B = (V_B, E_B)$ of T_B and define a nonnegative weight function $w_B : E_B - E'_B \rightarrow R^+$, as discussed in subsection 3.1.1. (See Fig. 2.)
2. Construct a path-tree $T_P = (V_P, E'_P)$ from T_B and a supergraph $G_P = (V_P, E_P)$ of T_P and define a nonnegative weight function $w_P : E_P - E'_P \rightarrow R^+$, as discussed in subsection 3.1.2. (See Fig. 3.)
3. Let $L = \{\rho_1, \cdots, \rho_{|L|}\}$ be the set of leaves of T_P. Set $E'' \leftarrow \emptyset$ and $i \leftarrow 1$ initially and repeat the following Steps 4 through 8.
4. Select a leaf $\rho_i \in L$ as the root and construct a reverse arborescence $\overrightarrow{T_i} = (V_P, \overrightarrow{E'_i})$ with the root ρ_i from T_P. (See solid arrows in Fig. 4 or 5.)
5. Define $\overrightarrow{E_i^+}$ and $w_i : \overrightarrow{E_i^+} \rightarrow R^+$ by executing the following 1 through 3.
    1. Set $\overrightarrow{E_i^+} \leftarrow \overrightarrow{E'_i}$. For each edge $e \in \overrightarrow{E'_i}$, $w_i(e) \leftarrow 0$. For each edge $(u,v) \in E_P - E'_P$, execute the following 2 or 3.
    2. If one of $\{u,v\}$, say u, is a descendant of the other, v, in $\overrightarrow{T_i}$, then set $\overrightarrow{E_i^+} \leftarrow \overrightarrow{E_i^+} \cup \{\langle u,v\rangle\}$ and $w_i(\langle u,v\rangle) \leftarrow w_P((u,v))$.
    3. Otherwise (that is, any one of $\{u,v\}$ is not a descendant of the other), set $\overrightarrow{E_i^+} \leftarrow \overrightarrow{E_i^+} \cup \{\langle t,u\rangle, \langle t,v\rangle, \langle u,v\rangle, \langle v,u\rangle\}$ and $w_i(\langle t,u\rangle) = w_i(\langle t,v\rangle) = w_i(\langle u,v\rangle) = w_i(\langle v,u\rangle) \leftarrow w_P((u,v))$, where t is the nearest common descendant of u and v in $\overrightarrow{T_i}$. (For example, in Fig. 4(a), if $u = b_7$ and $v = b_8$ then $t = b_6$).
6. $\overrightarrow{E_i} \leftarrow \overrightarrow{E_i^+}$ initially, and construct $\overrightarrow{E_i}$ as follows (see Figs. 4(b) and 5(b)): for each edge $\langle u,v\rangle \in \overrightarrow{E_i^+} - \overrightarrow{E'_i}$ such that u is a cutvertex and v is an ancestor of u in $\overrightarrow{T_i}$, set $\overrightarrow{E_i} \leftarrow \overrightarrow{E_i} \cup \{\langle u_v,v\rangle\} - \{\langle u,v\rangle\}$ and $w_i(\langle u_v,v\rangle) \leftarrow w_i(\langle u,v\rangle)$, where u_v is the parent (a block vertex) of u in $V(P\langle v,u;\overrightarrow{T_i}\rangle)$. (For example, in Fig. 4, $\langle c_2,b_8\rangle \in \overrightarrow{E_i^+} - \overrightarrow{E'_i}$ is changed to $\langle b_6,b_8\rangle \in \overrightarrow{E_i}$, where $u_v = b_6$ if $u = c_2$.)

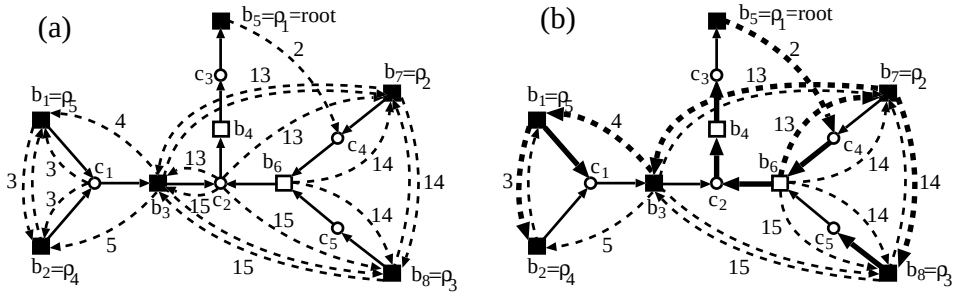


Figure 4: (a) Construction of $\vec{T}_1$ (solid arrows) and $\vec{E}_1^+$ (broken arrows) from G_P and T_P shown in Fig. 3. (b) Construction of $\vec{E}_1$ (broken arrows) from $\vec{E}_1^+$. In Step 6, directed edges emanating from c_2 in (a) are transformed into those which start from b_6 with resulting self-loops deleted. Bold arrows represent a minimum weight arborescence $\vec{T}_1'$ with the root ρ_1. The total weight is 49.

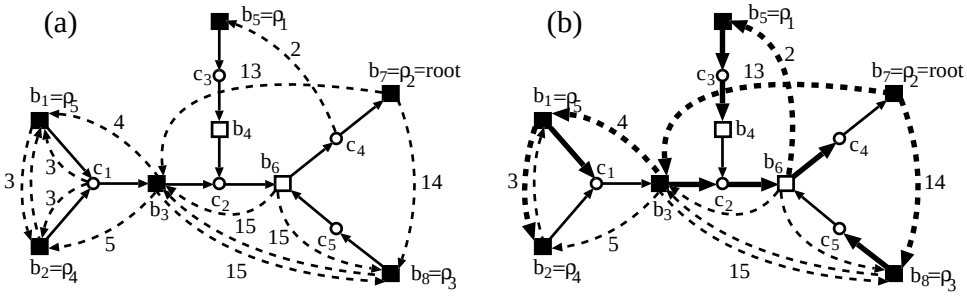


Figure 5: (a) Construction of $\vec{T}_2$ (solid arrows) and $\vec{E}_2^+$ (broken arrows) from G_P and T_P shown in Fig. 3. (b) Construction of $\vec{E}_2$ (broken arrows) from $\vec{E}_2^+$. In Step 6, a directed edge emanating from c_4 in (a) are transformed into that which start from b_6. Bold arrows represent a minimum weight arborescence $\vec{T}_2'$ with the root ρ_2. The total weight is 36.

7. Find a minimum weight arborescence $\overrightarrow{T_i'} = (V_P, \overrightarrow{A_i})$ with the root ρ_i in $\overrightarrow{G_i} = (V_P, \overrightarrow{E_i})$ (see bold arrows in Figs. 4(b) and 5(b)). Set $\overrightarrow{E_i''} \leftarrow \overrightarrow{A_i} - \overrightarrow{E_i'}$. Construct $E_i'' \subseteq E - E'$ by replacing each edge of $\overrightarrow{E_i''}$ by the corresponding undirected edge of G, where multiple edges are changed to a simple one.
8. If $i = 1$ or $w(E'') > w(E_i'')$ then $E'' \leftarrow E_i''$. Set $i \leftarrow i+1$. If $i \leq |L|$ then go back to Step 4. □

The correctness and the performance ratio of the algorithm *R2VS* are going to be shown later. Here we consider its time complexity. Step 1 takes $O(|V|+|E|)$ time. Since a single source shortest path tree can be found in $O(|E_B|+|V_B|\log|V_B|)$ time in $G_B - E_P'$, finding shortest paths between all pairs of vertices in $V_P \subseteq V_B$ of $G_B - E_P'$ can be done in $O(|V_P||E_B| + |V_P||V_B|\log|V_B|)$ time. $|V_P|$ and $|V_B|$ are both $O(|V|)$, and $|E_B|$ is $O(|E|)$. Hence Step 2 takes $O(|V||E|+|V|^2\log|V|)$ time. The loop from Steps 4 through Step 8 is repeated $|L|$ times and each iteration of the loop takes at most $O(|\overrightarrow{E_i}|+|V_P|\log|V_P|)$ time [12]. Since $|\overrightarrow{E_i}|$ is $O(|V|^2)$, the loop takes $O(|L||V|^2)$ time. Thus, the algorithm runs in $O(|V||E|+|V|^2\log|V|+|L||V|^2)$ time.

3.3 Correctness and Performance Ratio of *R2VS*

Theorem 8 If $\kappa(S;G) \geq 2$ then $\kappa(S;G'+E'') \geq 2$.

PROOF: By Lemmas 5 and 7, if $\kappa(S;G) \geq 2$ then $\kappa'(G_P) \geq 2$. Let E_{P_i}'' be the set of edges obtained by transforming each directed edge in $\overrightarrow{E_i''}$ to an undirected edge of G_P. We can prove that if $\kappa'(G_P) \geq 2$ then $\kappa'(T_P + E_{P_i}'') \geq 2$, similarly to the proof of [12, Lemma 4.6]. Hence, by Lemmas 5 and 7 again, we can obtain a set E'' of edges with $\kappa(S;G'+E'') \geq 2$. □

We consider the relationship between the total weight of $\overrightarrow{E_i''}$ and that of an optimum solution $E^* \subseteq E - E'$. Let $E_B^* = \{(\beta(u),\beta(v)) \mid (u,v) \in E^*\}$. We may assume that there are no multiple edges in E_B^* and that $E_B^* \subseteq E_B - E_B'$. Clearly, $w_B(E_B^*) = w(E^*)$. Since E^* is an optimum solution, we have $\kappa'(S_B;T_B+E_B^*) \geq 2$ by Lemma 5. Thus, we consider E_B^* instead of E^* in the rest of the section.

Let $\overline{T_B} = (\overline{V_B}, \overline{E_B'})$ be any fixed minimal subgraph of T_B such that $E_P' \subseteq \overline{E_B'}$ and $\kappa'(S_P;\overline{T_B}+E_B^*) \geq 2$. (In Fig. 2, for example, $(b_3,c_6) \in \overline{E_B'}$, while $(c_6,b_9) \notin \overline{E_B'}$: even if there were an edge $(b_{10},c_1) \in E_B'$, it would not in $\overline{E_B'}$.) We partition $E_B^* \cup (\overline{E_B'} - E_P')$

into the three sets, E_B^1, E_B^2 and E_B^3, as follows:

$$\begin{aligned}
E_B^1 &= \{(u,v) \in E_B^* \cup (\overline{E_B'} - E_P') \mid u,v \in \overline{V_B} - V_P\};\\
E_B^2 &= \{(u,v) \in E_B^* \cup (\overline{E_B'} - E_P') \mid u \in \overline{V_B} - V_P,\\
&\qquad v \in V_P\};\\
E_B^3 &= \{(u,v) \in E_B^* \cup (\overline{E_B'} - E_P') \mid u,v \in V_P\}.
\end{aligned}$$

Let $Z = (\overline{V_B} - V_P, E_B^1)$. Since E^* is an optimum solution, Z is a forest. Let $Z_1, Z_2, \ldots, Z_t$ be its trees, where t is the number of trees in Z. We define a *tent* as follows.

Definition 9 A *tent* is the graph induced by the edge set $E(Z_q) \cup E_{B_q}^2$ for $1 \le q \le t$, where $E_{B_q}^2 = \{(u,v) \in E_B^2 \mid u \in V(Z_q), v \in V_P\}$. If $E_B^3 \neq \emptyset$ then let us define a tent to be a graph consisting of an individual edge in E_B^3 and its endvertices. □

Property 10 A tent is a tree and all the leaves are vertices in V_P. □

Let n be the number of tents (that is, $n = t + |E_B^3|$). We denote a tent as $\mathscr{T}_j$ $(1 \le j \le n)$. We define a *floor* as follows.

Definition 11 A *floor* is the subtree of T_P such that it consists of all paths $P(u,v;T_P)$, one path for each pair $u,v \in V(\mathscr{T}_j) \cap V_P$. □

We denote a floor as $\mathscr{F}_j$. For each j, the subgraph of $\overrightarrow{T_i}$ corresponding to $\mathscr{F}_j$ is denoted as $\overrightarrow{\mathscr{F}_{ij}}$. $\overrightarrow{\mathscr{F}_{ij}}$ is also called a floor. $\overrightarrow{\mathscr{F}_{ij}}$ has exactly one vertex having no leaving edges. The vertex is called the root of $\overrightarrow{\mathscr{F}_{ij}}$ and denoted by r_{ij}. Let x_{ij} be a vertex defined as follows: if r_{ij} is a block vertex then set $x_{ij} \leftarrow r_{ij}$, or if r_{ij} is a cutvertex then select any block-vertex u that is a parent of r_{ij} in $\overrightarrow{\mathscr{F}_{ij}}$ and set $x_{ij} \leftarrow u$ (see x_{ij} in Fig. 6). Let $\mathscr{N}_{ij}$ denote the graph consisting of a tent $\mathscr{T}_j$ and a floor $\overrightarrow{\mathscr{F}_{ij}}$ (see Fig. 6). $\mathscr{N}_{ij}$ is called a *net*.

Lemma 12 For each net $\mathscr{N}_{ij}$, a set of directed edges $\overrightarrow{\mathscr{A}_{ij}} \subseteq \overrightarrow{E_i} - \overrightarrow{E_i'}$ satisfying the following (a) and (b) can be constructed from E_B^*: (a) all vertices of $\overrightarrow{\mathscr{F}_{ij}}$ are reachable from x_{ij} in $\overrightarrow{\mathscr{F}_{ij}} + \overrightarrow{\mathscr{A}_{ij}}$; (b) If r_{ij} is the leaf of $\overrightarrow{\mathscr{F}_{ij}}$ then $w_i(\overrightarrow{\mathscr{A}_{ij}}) \le (2 - \frac{2}{p_j}) w_B(E(\mathscr{T}_j))$ else $w_i(\overrightarrow{\mathscr{A}_{ij}}) \le 2 w_B(E(\mathscr{T}_j))$, where p_j is the number of leaves of $\mathscr{F}_j$.

PROOF: Let $s_j \in V(\mathscr{T}_j)$ be a vertex defined as follows: if r_{ij} is a leaf of $\overrightarrow{\mathscr{F}_{ij}}$ then set $s_j = r_{ij}$, or if r_{ij} is not a leaf of $\overrightarrow{\mathscr{F}_{ij}}$ then select any vertex s_j in $\mathscr{T}_j$. For each $\mathscr{T}_j$, we execute the depth-first-search (DFS) by selecting s_j as the starting vertex, and assign the DFS-number to each vertex. Suppose that L_j is the set of leaves of $\mathscr{F}_j$ and $p_j = |L_j|$. Note that $L \cap V(\mathscr{F}_j) \subseteq L_j$ and that $p_j \ge 2$ since $E(\mathscr{F}_j) \neq \emptyset$. Then, leaves of $\overrightarrow{\mathscr{F}_{ij}}$ are numbered as $l_1^{(j)}, l_2^{(j)}, \cdots, l_{p_j}^{(j)}$, whose indices denote the order in which they are visited by DFS (see Fig. 6), where $l_1^{(j)}$ may be often denoted as $l_{p_j+1}^{(j)}$ for notational simplicity. Each of p_j paths $P(l_k^{(j)}, l_{k+1}^{(j)}; \mathscr{T}_j)$ $(1 \le k \le p_j)$ is called a *bypass* (connecting $l_k^{(j)}$ and $l_{k+1}^{(j)}$). $P(l_k^{(j)}, l_{k+1}^{(j)}; \mathscr{T}_j)$ is often represented as $P_k^{(j)}$ for simplicity. Then the following (1) and (2) hold.

1. For each $e \in E(\mathscr{T}_j)$, there are exactly two bypasses containing e.
2. For some k' with $1 \le k' \le p_j$, there is at least one weakly connected path $\widetilde{P}_{k'} = \widetilde{P}(l_{k'}^{(j)}, l_{k'+1}^{(j)}; \overrightarrow{\mathscr{F}_{ij}})$ such that $r_{ij}, x_{ij} \in V(\widetilde{P}_{k'})$, where $l_{k'+1}^{(j)}$ is an ancestor of x_{ij} in $\overrightarrow{T_i}$.

Let us define a set of directed edges $\overrightarrow{\mathscr{A}_{ij}}$ as in the following (i) or (ii) by appropriately choosing one weakly connected path $\widetilde{P}_{k'}$ such that $r_{ij}, x_{ij} \in V(\widetilde{P}_{k'})$ (x_{ij} may be equal to r_{ij}) in $\overrightarrow{\mathscr{F}_{ij}}$.
(i) If r_{ij} is a leaf of $\overrightarrow{\mathscr{F}_{ij}}$ (see Fig. 6(1), (3)) then $l_1^{(j)} = r_{ij}$. First, let

$$\omega_k^{(j)} = \sum_{(u,v) \in P_k^{(j)}} w_B((u,v)), \quad \text{and}$$

$$\omega_{k''}^{(j)} = \max\{\omega_k^{(j)} \mid 1 \le k \le p_j\} \ \left(\ge \frac{1}{p_j} \sum_{k=1}^{p_j} \omega_k^{(j)}\right).$$

If $k'' \neq 1$ and $k'' \neq p_j$ ($l_{k''} = l_3$ in Fig. 6(1)) then let

$$\begin{aligned}
\overrightarrow{\mathscr{A}_{ij}} = \{&\langle x_{ij}, l_2^{(j)}\rangle, \langle x_{ij}, l_{p_j}^{(j)}\rangle\} \cup \{\langle l_m^{(j)}, l_{m+1}^{(j)}\rangle \mid 2 \le\\
&m \le k'' - 1\} \cup\\
&\{\langle l_{m'+1}^{(j)}, l_{m'}^{(j)}\rangle \mid k'' + 1 \le m' \le p_j - 1\}.
\end{aligned}$$

If $k'' = 1$ (see Fig. 6(3)) then let

$$\overrightarrow{\mathscr{A}_{ij}} = \{\langle x_{ij}, l_{p_j}^{(j)}\rangle\} \cup \{\langle l_{m'+1}^{(j)}, l_{m'}^{(j)}\rangle \mid 2 \le m' \le p_j - 1\}.$$

If $k'' \neq 1$ and $k'' = p_j$ then let

$$\overrightarrow{\mathscr{A}_{ij}} = \{\langle x_{ij}, l_2^{(j)}\rangle\} \cup \{\langle l_m^{(j)}, l_{m+1}^{(j)}\rangle \mid 2 \le m \le p_j - 1\}.$$

(ii) If r_{ij} is not a leaf of $\overrightarrow{\mathscr{F}_{ij}}$ then let

$$\begin{aligned}\overrightarrow{\mathscr{A}_{ij}} = {} & \{\langle x_{ij}, l_{k'+1}^{(j)}\rangle\} \cup \\ & \{\langle l_m^{(j)}, l_{m+1}^{(j)}\rangle \mid 1 \le m \le p_j, m \neq k'\}\end{aligned}$$

($l_{k'} = l_8$ and $l_{k'+1} = l_9$ in Fig. 6(2)).

The following important points (a)–(c) hold from the definitions of $\overrightarrow{\mathscr{A}_{ij}}$, G_P and w_P:
(a) all vertices of $\overrightarrow{\mathscr{F}_{ij}}$ are reachable from x_{ij} in $\overrightarrow{\mathscr{F}_{ij}} + \overrightarrow{\mathscr{A}_{ij}}$;
(b) $\overrightarrow{\mathscr{A}_{ij}} \subseteq \overrightarrow{E_i} - \overrightarrow{E_i'}$;
(c) $w_P((l_k^{(j)}, l_{k+1}^{(j)})) \le \omega_k^{(j)}$ for each k ($1 \le k \le p_j$).

Now, if r_{ij} is a leaf of $\overrightarrow{\mathscr{F}_{ij}}$ then the condition (1) mentioned above in this proof and the fact that $w_i(\langle x_{ij}, l_2^{(j)}\rangle) = w_P((l_1^{(j)}, l_2^{(j)}))$ and $w_i(\langle x_{ij}, l_{p_j}^{(j)}\rangle) = w_P((l_1^{(j)}, l_{p_j}^{(j)}))$ (this follows from Steps 52 and 6 of *R2VS*) show the following inequality

$$\begin{aligned} w_i(\overrightarrow{\mathscr{A}_{ij}}) &\le \sum_{k=1, k\neq k''}^{p_j} w_P((l_k^{(j)}, l_{k+1}^{(j)})) \\ &\le \sum_{k=1, k\neq k''}^{p_j} \omega_k^{(j)} \le \Big(1 - \frac{1}{p_j}\Big) \sum_{k=1}^{p_j} \omega_k^{(j)} \\ &= \Big(1 - \frac{1}{p_j}\Big) \cdot 2 \sum_{(u,v)\in E(\mathscr{T}_j)} w_B((u,v)) \\ &= \Big(2 - \frac{2}{p_j}\Big) w_B(E(\mathscr{T}_j)). \end{aligned}$$

If r_{ij} is not a leaf of $\overrightarrow{\mathscr{F}_{ij}}$ then $w_i(\langle x_{ij}, l_{k'+1}^{(j)}\rangle) = w_P((l_{k'}^{(j)}, l_{k'+1}^{(j)}))$ (this follows from Steps 53 and 6 of *R2VS*) and, therefore,

$$\begin{aligned} w_i(\overrightarrow{\mathscr{A}_{ij}}) &\le \sum_{k=1}^{p_j} w_P((l_k^{(j)}, l_{k+1}^{(j)})) \le \sum_{k=1}^{p_j} \omega_k^{(j)} \\ &= 2 \sum_{(u,v)\in E(\mathscr{T}_j)} w_B((u,v)) = 2 w_B(E(\mathscr{T}_j)). \end{aligned}$$

□

Lemma 13 For some h with $1 \le h \le |L|$, a set of directed edges $\overrightarrow{B_h} \subseteq \overrightarrow{E_h} - \overrightarrow{E_h'}$ satisfying the following (a) and (b) can be constructed from E_B^*: (a) $\overrightarrow{T_h} + \overrightarrow{B_h}$ is strongly connected; (b) $w_h(\overrightarrow{B_h}) \le (2 - \frac{2}{|L|}) w_B(E_B^*)$.

PROOF: We prove that the desired set $\overrightarrow{B_h}$ is obtained from E_B^*. First, for each i with $1 \le i \le |L|$, we show how to construct a set of directed edges $\overrightarrow{B_i}$ such that $\overrightarrow{T_i} + \overrightarrow{B_i}$ is strongly connected (see Fig. 7 and Fig. 8). Initially set $\overrightarrow{B_i} \leftarrow \emptyset$ and $\mathscr{E}_B^* \leftarrow E_B^* \cup (\overrightarrow{E_B'} - E_P')$, and assign "*accessible*" to the root ρ_i of the path-tree $\overrightarrow{T_i} = (V_P, \overrightarrow{E_i'})$ and "*nonaccessible*" to the other vertices of $\overrightarrow{T_i}$. Repeat the following pair of procedures **AL1** and **AL2** until "*accessible*" is assigned to every vertex of V_P:

(AL1) Select an accessible block-vertex x satisfying the following (1) and (2):

1. x is in a net $\mathscr{N}_{ij}$ constructed from $\mathscr{E}_B^*$;
2. in $\overrightarrow{\mathscr{F}_{ij}}$, if r_{ij} is a block-vertex then $x \leftarrow r_{ij}$, otherwise x is set to a parent of r_{ij}.

(AL2) For x and $\mathscr{N}_{ij}$, construct $\overrightarrow{\mathscr{A}_{ij}}$ by using Lemma 12 (in which x is written as x_{ij}), and set $\overrightarrow{B_i} \leftarrow \overrightarrow{B_i} \cup \overrightarrow{\mathscr{A}_{ij}}$. Then assign "*accessible*" to all vertices of the floor $\overrightarrow{\mathscr{F}_{ij}}$ of the net $\mathscr{N}_{ij}$, and set $\mathscr{E}_B^* \leftarrow \mathscr{E}_B^* - E(\mathscr{T}_j)$. □

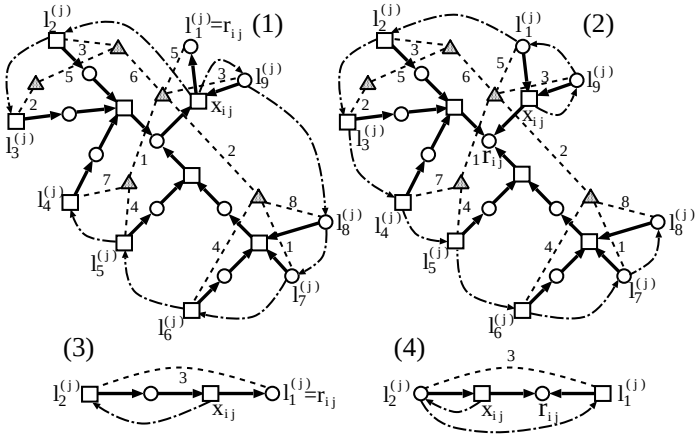


Figure 6: Four examples (1) through (4) of nets $\mathscr{N}_{ij}$. Triangles are vertices in $\overline{V_B} - V_P$. Solid arrows, broken lines and dash-dotted arrows represent a floor $\vec{\mathscr{F}}_{ij}$, a tent $\mathscr{T}_j$ and edges of $\vec{\mathscr{A}}_{ij}$, respectively.

If we assume that there are no block-vertices x satisfying (1) and (2) of **AL1**, while we have a nonaccessible vertex, then we can easily show a contradiction that $\kappa'(S_B; T_B + E_B^*) \geq 2$ is not satisfied. Hence it is concluded that "*accessible*" is assigned to every vertex eventually, implying that $\overrightarrow{T_i} + \overrightarrow{B_i}$ is strongly connected.

Next we show that there is a direct edge set $\overrightarrow{B_h}$ with $w_h(\overrightarrow{B_h}) \leq (2 - \frac{2}{|L|})w(E_B^*)$. Let h be an index such that $w_h(\overrightarrow{B_h}) = min\{w_i(\overrightarrow{B_i}) \mid 1 \leq i \leq |L|\}$. For any $r \in L_j - L$, there is an edge $(r,u) \in E(\mathscr{F}_j)$ such that $T_P - (r,u)$ has a connected component T' with $V(T') \cap V(\mathscr{F}_j) = \{r\}$ and $L \cap V(T') \neq \emptyset$. Hence there is a reverse arborescence, rooted at some $r' \in L \cap V(T')$, such that any path $P\langle u, r'\rangle$ with $u \in V(\mathscr{F}_j)$ passes through r toward r', meaning that r is a root of $\overrightarrow{\mathscr{F}_{ij}}$. That is, each vertex in L_j becomes the root of $\overrightarrow{\mathscr{F}_{ij}}$ at least once in $\overrightarrow{T_1}, \ldots, \overrightarrow{T_{|L|}}$. (For example, b_3, b_7 and b_8 of Fig. 7 become the root of $\overrightarrow{\mathscr{F}_{13}}$ in $\overrightarrow{T_4}$, $\overrightarrow{T_2}$ and $\overrightarrow{T_3}$, respectively.) Since $E(\mathscr{T}_{j'}) \cap E(\mathscr{T}_{j''}) = \emptyset$ if $j' \neq j''$, Lemma 12 gives us

$$
\begin{aligned}
w_h(\overrightarrow{B_h}) &\leq \frac{1}{|L|} \sum_{i=1}^{|L|} w_i(\overrightarrow{B_i}) \\
&= \frac{1}{|L|} \sum_{i=1}^{|L|} \sum_{j=1}^{n} w_i(\overrightarrow{\mathscr{A}_{ij}}) \\
&\leq \frac{1}{|L|} \sum_{j=1}^{n} \Big(p_j \cdot \Big(2 - \frac{2}{p_j}\Big) w_B(E(\mathscr{T}_j)) + \\
&\qquad (|L| - p_j) \cdot 2 w_B(E(\mathscr{T}_j)) \Big) \\
&= \frac{1}{|L|} \cdot 2(|L| - 1) \sum_{j=1}^{n} w_B(E(\mathscr{T}_j)) \\
&= \Big(2 - \frac{2}{|L|}\Big) w_B(E_B^*).
\end{aligned}
$$

□

Since $\overrightarrow{T_h} + \overrightarrow{B_h}$ is strongly connected, it has a subgraph $T_H = (V_P, \overrightarrow{E_H})$ which is an arborescence rooted at ρ_h. Since $w_h(\langle u, v\rangle) = 0$ for any directed edge $\langle u, v\rangle \in \overrightarrow{E'_h}$, we have $w_h(\overrightarrow{E_H}) \leq w_h(\overrightarrow{B_h})$. (Note that $\overrightarrow{B_h}$ may have some edges not contained in $\overrightarrow{E_H}$.) Hence we obtain the next corollary.

Corollary 14 $\overrightarrow{T_h} + \overrightarrow{B_h}$ of Lemma 13 contains, as a subgraph, an arborescence $T_H = (V_P, \overrightarrow{E_H})$, with the root ρ_h, such that $w_h(\overrightarrow{E_H}) \leq w_h(\overrightarrow{B_h})$.

We obtain the next theorem from the above discussion.

Theorem 15 If $\kappa(S; G') = 1$ then the proposed algorithm *R2VS* generates a $(2 - 2/|L|)$-approximate solution to 2VCA-SV in $O(|V||E| + |V|^2 \log|V| + |L||V|^2)$ time, where L is the set of leaves of the path-tree of G'.

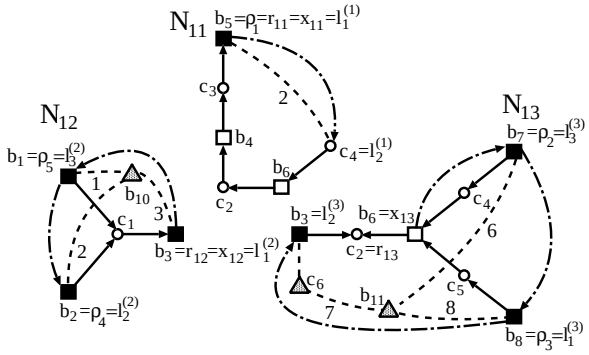


Figure 7: Three nets $\mathscr{N}_{11}$, $\mathscr{N}_{12}$ and $\mathscr{N}_{13}$ (denoted by solid arrows and broken lines) constructed from T_B and E_B^* shown in Fig. 2. Triangles are vertices in $\overline{V_B} - V_P$ and dash-dotted arrows are directed edges $\vec{B}_1$.

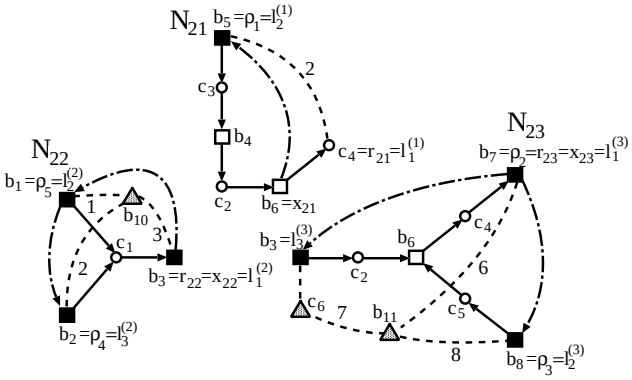


Figure 8: Three nets $\mathscr{N}_{21}$, $\mathscr{N}_{22}$ and $\mathscr{N}_{23}$ (denoted by solid arrows and broken lines) constructed from T_B and E_B^* shown in Fig. 2. Triangles are vertices in $\overline{V_B} - V_P$ and dash-dotted arrows are directed edges $\vec{B}_2$.

PROOF: Since the time complexity of the algorithm has already been given in Section 3.2, we consider the weight of any approximate solution E'' given by *R2VS*. $\overrightarrow{T_h'}$ constructed in Step 7 is a minimum weight arborescence in a directed graph $(V_P, \overrightarrow{E_h})$ with respect to the weight function w_h. That is, $w_h(\overrightarrow{E_h''}) \leq w_h(\overrightarrow{E_H})$. By considering the rooted tree $T_H = (V_P, \overrightarrow{E_H})$ mentioned in Lemma 13 and Corollary 14, we obtain

$$\begin{aligned} w(E'') &= min\{w(E_i'') \mid 1 \leq i \leq |L|\} \\ &\leq w(E_h'') \leq w_h(\overrightarrow{E_h''}) \\ &\leq w_h(\overrightarrow{E_H}) \leq w_h(\overrightarrow{B_h}) \leq (2 - \frac{2}{|L|}) w(E^*). \end{aligned}$$

□

4 The 2-Edge-Connectivity Augmentation Problem for Specified Vertices

This section proposes an approximation algorithm *R2ES* for 2**ECA-SV**. The performance ratio is $(2 - \frac{2}{|L|})$ if $\lambda(S;G') = 1$. Its time complexity is $O(|V||E| + |V|^2 \log |V| + |L||V|^2)$.

R2ES utilizes the algorithm of [12] for solving 2**ECA** in order to improve the time complexity. The differences from [12] are the following (1) and (2): (1) we select each leaf of T as the root in Step 4 and execute the algorithm of [12], and then select the best solution; (2) we have modified construction of $\overrightarrow{E_i}$ in Step 63 and add Steps 2, 3 in order to extend 2**ECA** to 2**ECA-SV**. The proposed algorithm *R2ES* is stated in the following.

Algorithm *R2ES*;
/* **Input:** An undirected graph $G = (V,E)$, a set of specified vertices $S \subseteq V$, a spanning subgraph $G' = (V,E')$ of G with $\lambda(S;G') = 1$, and a nonnegative weight function $w : E \to R^+$. */
/* **Output:** A set of edges $E'' \subseteq E - E'$. */

1. Construct a graph $G'_s = (V_s, E'_s)$ from G' by shrinking each 2-edge-connected component of G' into an individual vertex, where any connected component not containing S is regarded as a 2-edge-connected component in this construction. For $u, v \in V_s$, let $\widehat{E}_s = \{(u,v) \mid (u',v') \in E - E', u' \in \gamma^{-1}(u), v' \in \gamma^{-1}(v)\}$, $w_s((u,v)) = \min\{w((u',v')) \mid (u',v') \in E - E', u' \in \gamma^{-1}(u), v' \in \gamma^{-1}(v)\}$, where $\gamma^{-1}(x)$ is the component represented by $x \in V_s$. Let $E_s = E'_s \cup \widehat{E}_s$ and $G_s = (V_s, E_s)$.

2. Let $S' = \{u \in V_s \mid \gamma^{-1}(u) \cap S \neq \emptyset\}$ and $T = (V_T, E'_T)$ (called a path-tree) be a subgraph consisting of those edges in $P(u,v;G'_s)$ for any pair of $u, v \in S'$.

3. Set $E_T \leftarrow E'_T$ and construct $G_T = (V_T, E_T)$ as follows: we set $w_s((u,v)) \leftarrow 0$ for any $(u,v) \in E'_s - E'_T$. For any pair of vertices $u, v \in V_T$ if $G_s - E'_T$ has a path $P(u,v)$ such that $V(P(u,v)) \cap V_T = \{u,v\}$ then set $E_T \leftarrow E_T \cup \{(u,v)\}$ and let $w_T((u,v))$ be the shortest length of such paths with respect to the weight function w_s.

4. Let $L = \{\rho_1, \cdots, \rho_{|L|}\}$ be the set of leaves of T. Set $E'' \leftarrow \emptyset$ and $i \leftarrow 1$ initially and repeat the following Steps 5 through 8.

5. Select a leaf ρ_i of T as the root and construct a reverse arborescence $\overrightarrow{T_i} = (V_T, \overrightarrow{E'_i})$ with the root ρ_i from T.

6. Define $\overrightarrow{E_i}$ and $w_i : \overrightarrow{E_i} \to R^+$ by executing the following 1 through 3.

   1. Set $\overrightarrow{E_i} \leftarrow \overrightarrow{E'_i}$. For each edge $e \in \overrightarrow{E'_i}$, $w_i(e) \leftarrow 0$. For each edge $(u,v) \in E_T - E'_T$, execute the following 2 or 3.
   2. If one of $\{u,v\}$, say u, is a descendant of the other, v, in $\overrightarrow{T_i}$, then set $\overrightarrow{E_i} \leftarrow \overrightarrow{E_i} \cup \{\langle u,v\rangle\}$ and $w_i(\langle u,v\rangle) \leftarrow w_T((u,v))$.
   3. Otherwise (that is, any one of $\{u,v\}$ is not a descendant of the other), set $\overrightarrow{E_i} \leftarrow \overrightarrow{E_i} \cup \{\langle t,u\rangle, \langle t,v\rangle, \langle u,v\rangle, \langle v,u\rangle\}$ and $w_i(\langle t,u\rangle) = w_i(\langle t,v\rangle) = w_i(\langle u,v\rangle) = w_i(\langle v,u\rangle) \leftarrow w_T((u,v))$, where t is the nearest common descendant of u and v in $\overrightarrow{T_i}$.

7. Find a minimum weight arborescence $\overrightarrow{T'_i} = (V_T, \overrightarrow{A_i})$ with the root ρ_i in $\overrightarrow{G_i} = (V_T, \overrightarrow{E_i})$. Set $\overrightarrow{E''_i} \leftarrow \overrightarrow{A_i} - \overrightarrow{E'_i}$. Construct $E''_i \subseteq E - E'$ by replacing each edge of $\overrightarrow{E''_i}$ by the corresponding undirected edge of G, where multiple edges are changed to a simple one.

8. If $i = 1$ or $w(E'') > w(E''_i)$ then $E'' \leftarrow E''_i$. Set $i \leftarrow i+1$. If $i \leq |L|$ then go back to Step 5. □

Note that, in solving 2**ECA-SV**, we may restrict paths $P(u,v)$ in Step 3 to those having $V(P(u,v)) \cap V_T = \{u,v\}$, even though it is usual to require $P(u,v) \cap E'_T = \emptyset$.

We can prove the following theorem. The details of the proof is omitted due to shortage of space.

Theorem 16 If $\lambda(S;G') = 1$ then the proposed algorithm *R2ES* generates a $(2 - \frac{2}{|L|})$-approximate solution to 2ECA-SV in $O(|V||E| + |V|^2 \log|V| + |L||V|^2)$ time, where L is the set of leaves of the path-tree of G'. □

5 The $(\lambda+1)$-Edge-Connectivity Augmentation Problem

This section proposes an approximation algorithm *RMW+1* for $(\lambda+1)$**ECA**. The performance ratio of *RMW+1* is $(2 - \frac{2}{|L'|})$, where L' is the set of leaves of a certain tree constructed from G'. Its time complexity is $O(\Delta + |L'|^2|E| + |L'||V|\log|V|)$ if λ is even, or $O(\Delta + |L'||E| + |L'||V|\log|V|)$ if λ is odd, where Δ is the time complexity of constructing a structural graph of G'.

5.1 Structural Graphs

A *cactus* is an undirected connected graph in which any pair of simple cycles share at most one vertex. An edge of a cactus is called a *cycle edge* if it is contained in a cycle; otherwise it is called a *tree edge*. A *cut* of a given connected graph $G' = (V, E')$ is a set $K \subseteq E'$ such that $G' - K$ is disconnected. A *minimum cut* is a cut of minimum cardinality among those of G'.

A *structural graph* G'_s of a given graph $G' = (V, E')$ with edge-connectivity $\lambda(G') = \lambda$ is a representation of all minimum cuts of G' (see Figures 9 and 10). G'_s is an edge-weighted cactus of $O(|V|)$ nodes and edges such that each tree edge has weight λ and each cycle edge has weight $\frac{\lambda}{2}$. Particularly if λ is odd then G'_s is a weighted tree. There is a one-to-one function $\delta : V(G') \to V(G'_s)$, and $V(G'_s)$ may have some other vertices, called *empty vertices*, to which no vertices of G' are mapped (see *l*, *m*, *n* in Fig. 10). Karzanov and Timofeev [11] first showed that $F(G)$ can be constructed in polynomial time. It is shown that $F(G)$ can be constructed in $O(|V||E|\log(|V|^2/|E|))$ time [6] or in $O(|V||E| + |V|^2\log|V|)$ time [18]. Note that if λ is even then replacing each tree edge by a pair of multiple edges of weight $\frac{\lambda}{2}$ preserves the properties of structural graphs and makes their handling easy because the resulting graphs have no bridges. This graph, as well as a tree in the case where λ is odd, is called a *modified cactus*. In this paper, G'_s denotes a modified cactus unless otherwise stated. Note that $\lambda(G'_s) = 1$ if λ is odd and $\lambda(G'_s) = 2$ if λ is even.

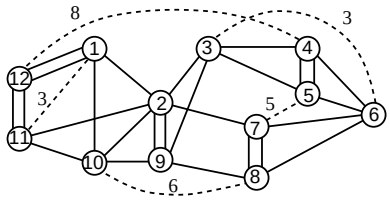


Figure 9: $G=(V,E)$ and $G'=(V,E')$ with $\lambda(G')=4$. Solid lines are edges in E', and numbers shown beside edges are weights. Broken lines are an optimum solution $E^*\subseteq E-E'$ to 5ECA and the other edges in $E-E'$ are omitted for convenience.

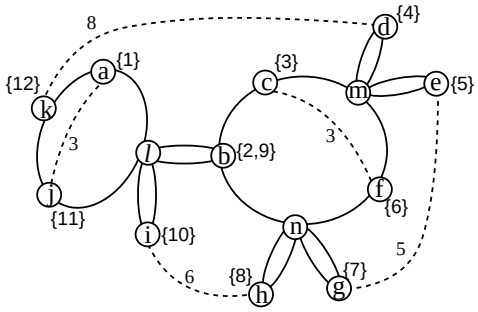


Figure 10: $G_s=(V_s,E_s)$ and a cactus $G'_s=(V_s,E'_s)$ constructed from G and G' shown in Fig. 9, where solid lines are edges in E'_s and broken lines are edges in $E_s-E'_s$.

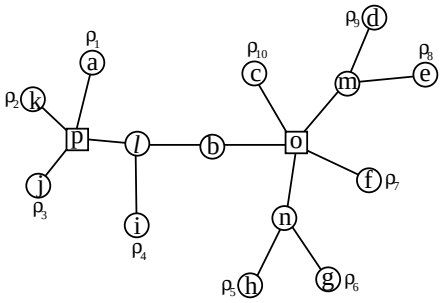


Figure 11: A tree G'_b constructed from G'_s shown in Fig. 10 by procedure *REMAKE*. Squares are dummy vertices.

5.2 The Proposed Algorithm *RMW+1*

RMW+1 is based on the existing algorithm *MW*[17] for $(\lambda+1)$**ECA** (see also [26, 16]). *MW* is a 2-approximation one and utilizes a minimum weight arborescence algorithm [8].

MW first construct a stractual graph G'_s of G' (as in Figs. 9 and 10) and then obtain a tree G'_b from G'_s (as shown in Fig. 11). Similarly to *R2VS* and *R2ES*, G'_b is changed into a rooted tree by choosing a leaf as the root, and we find an approximate solution by means of a minimum arborescence. The details of *MW* is can be found in [17].

The difference between *RMW+1* and *MW* is the following: *RMW+1* repeats selection of each leaf in G'_b as the root, execution of Steps 3–7 of *MW*, and then selects the best solution among those obtained. The proposed algorithm *RMW+1* is stated in the following.

Algorithm *RMW+1*;
/* **Input:** An undirected graph $G=(V,E)$, a spanning subgraph $G'=(V,E')$ of G (Fig. 9), and a nonnegative weight function $w:E\to R^+$. */
/* **Output:** A set of edges $E''\subseteq E-E'$. */

1. Construct G'_s and G_s by Step 1 of *MW*, respectively. Set $E''_s\leftarrow\emptyset$ and $m\leftarrow 1$ initially. Repeat the following Steps 2 and 3.
2. Obtain a set of edges $E''_s(m)$ with respect to a leaf $\rho_m\in L'$ by Steps 3–7 of *MW*.
3. If $m=1$ or $w_s(E''_s)>w_s(E''_s(m))$ then $E''_s\leftarrow E''_s(m)$. Set $m\leftarrow m+1$. If $|L'|\geq m$ then go to Step 2, otherwise output $E''\leftarrow\{b(e)\mid e\in E''_s\}$ and terminate the algorithm. □

5.3 Correctness and Performance Ratio of *RMW+1*

We can easily prove the following lemma.

Lemma 17 If $\lambda(G)\geq\lambda(G')+1$ then $\lambda(G'+E'')\geq\lambda(G')+1$ holds. □

Next, we prove that the proposed algorithm *RMW+1* produces the worst approximation no greater than $(2-\frac{2}{|L'|})$ times the optimum (The proof is omitted due to shortage of space).

Lemma 18 Let E^* be an optimum solution for $(\lambda+1)$**ECA**. Then $w(E'')\leq(2-\frac{2}{|L'|})w(E^*)$. □

Constructing G_s, G'_s, w_s and b can be done in $O(\Delta+|E|)$ time, where Δ is time complexity of obtaining a structural graph of G' and $\Delta=\min\{|V||E'|\log(|V|^2/|E'|),|V||E'|+|V|^2\log|V|\}$ [6, 18]. Since G'_s has $O(|V|)$ vertices and edges, Step 2 of *RMW+1* takes $O(|L'||E|+|V|\log|V|)$ time if λ is even, and $O(|E|+|V|\log|V|)$ if λ is odd [17]. The loop of Steps 2 and 3 is repeated $|L'|$ times. Hence we obtain the next theorem.

Theorem 19 The proposed algorithm *RMW+1* generates a $(2-\frac{2}{|L'|})$-approximate solution to $(\lambda+1)$**ECA** in $O(\Delta+|L'|^2|E|+|L'||V|\log|V|)$ time if λ is even, or $O(\Delta+|L'||E|+|L'||V|\log|V|)$ time if λ is odd, where Δ is the time complexity of obtaining a structural graph of G'. □

6 The Minimum Weight Strongly Connected Spanning Subgraph Problem

In this section, we propose an $O(|V||E|+|V|^2\log|V|)$ time $(2-\frac{2}{|V|})$-approximation algorithm *RSCS* for **SCSS**.

6.1 The Proposed Algorithm *RSCS*

Our algorithm is based on a minimum weight arborescence algorithm [8] and utilizes a 2-approximation algorithm proposed in [5]. *RSCS* repeats selection of each vertex in G as the root and execution of the algorithm in [5], and then selects the best solution among those obtained.

Algorithm *RSCS*;
/* **Input:** A directed graph $G=(V,E)$ ($V=\{v_1,v_2,\ldots,v_n\}, n=|V|$) and a nonnegative weight function $w: E \to R^+$. */
/* **Output:** A set of edges $A \subseteq E$. */

1. Set $A \leftarrow \emptyset$ and $i \leftarrow 1$ initially.
2. Set $E_i \leftarrow E$ and let $G_i=(V,E_i)$. For each edge $\langle x,y\rangle \in E$, let $w_i(\langle x,y\rangle) \leftarrow w(\langle x,y\rangle)$. Find a minimum weight reverse arborescence $T_i'=(V,E_i')$ with the root v_i in $G_i=(V,E_i)$.
3. If $\langle x,y\rangle \in E_i'$ then set $w_i(\langle x,y\rangle) \leftarrow 0$. Find a minimum weight arborescence $T_i''=(V,E_i'')$ with the root v_i in $G_i=(V,E_i)$.
4. Set $A_i \leftarrow E_i' \cup E_i''$. If $i=1$ or $w(A) > w(A_i)$ then $A \leftarrow A_i$. Set $i \leftarrow i+1$. If $i \leq n$ then go to Step 2. □

6.2 Correctness and Performance Ratio of *RSCS*

Lemma 20 ([5]) If $G=(V,E)$ is strongly connected then $G_i'=(V,A_i)$ is strongly connected. □

Clearly, if $G=(V,E)$ is strongly connected then so is $G'=(V,A)$. We can prove the following lemma. The proof is omitted due to shortage of space.

Lemma 21 Let A^* be an optimum solution. Then $w(A) \leq (2-\frac{2}{|V|})w(A^*)$. □

Finding a minimum weight arborescence or a reverse arborescence can be done in $O(|E|+|V|\log|V|)$ time by using the algorithm in [8]. The loop in the algorithm is repeated $|V|$ times and each iteration of the loop takes $O(|E|+|V|\log|V|)$ time. Hence the total time spent by the algorithm is $O(|V||E|+|V|^2\log|V|)$.

Theorem 22 The proposed algorithm *RSCS* generates a $(2-\frac{2}{|V|})$-approximate solution to **SCSS** in $O(|V||E|+|V|^2\log|V|)$ time. □

7 The Minimum Weight k-Edge-Connected Spanning Subgraph Problem

In this section, we propose an $O(k|V|^2(|E|+|V|\log|V|)\log|V|)$ time $(2-\frac{2}{|V|})$-approximation algorithm *RkECS* for k**ECSS**.

7.1 The Proposed Algorithm *RkECS*

Our algorithm is based on a minimum weight packing arborescence algorithm [7] and utilizes a 2-approximation algorithm proposed in [13]. *RkECS* repeats selection of each vertex in G as the root and execution of the algorithm of [13], and then selects the best solution among those obtained.

Algorithm *RkECS*;
/* **Input:** An undirected graph $G=(V,E)$ ($V=\{v_1,v_2,\ldots,v_n\}, n=|V|$), a nonnegative weight function $w: E \to R^+$ and a positive integer k. */
/* **Output:** A set of edges $E' \subseteq E$. */

1. Set $E' \leftarrow \emptyset$, $A \leftarrow \emptyset$ and $i \leftarrow 1$ initially. For each edge $(x,y) \in E$, let $A \leftarrow A \cup \{\langle x,y\rangle, \langle y,x\rangle\}$ and $w'(\langle x,y\rangle) = w'(\langle y,x\rangle) \leftarrow w(x,y)$.
2. Find a minimum weight directed subgraph $H_i=(V,A_i)$ of $G_D=(V,A)$ such that H_i has k-edge-disjoint directed paths from the root v_i to each vertex.
3. Set $E_i' \leftarrow \emptyset$. If $\langle x,y\rangle \in A_i$ or $\langle y,x\rangle \in A_i$ then $E_i' \leftarrow E_i' \cup \{(x,y)\}$.
4. If $i=1$ or $w(E') > w(E_i')$ then $E' \leftarrow E_i'$. Set $i \leftarrow i+1$. If $i \leq n$ then go to Step 2. □

7.2 Correctness and Performance Ratio of *RkECS*

Lemma 23 ([13]) If $G=(V,E)$ is k-edge-connected then $G_i'=(V,E_i')$ is k-edge-connected. □

Clearly, if $G=(V,E)$ is k-edge-connected then so is $G'=(V,E')$. We can prove the following lemma. The proof is omitted due to shortage of space.

Lemma 24 Let E^* be an optimum solution. Then $w(E') \leq (2-\frac{2}{|V|})w(E^*)$ holds. □

Finding a minimum weight k-packing-arborescence can be done in $O(k|V|(|E|+|V|\log|V|)\log|V|)$ time by means of the algorithm in [7]. The loop in the algorithm is repeated $|V|$ times and each iteration of the loop takes $O(k|V|(|E|+|V|\log|V|)\log|V|)$ time. Hence the total time taken by the algorithm is $O(k|V|^2(|E|+|V|\log|V|)\log|V|)$.

Theorem 25 The proposed algorithm *RkECS* generates a $(2-\frac{2}{|V|})$-approximate solution to k**ECSS** in $O(k|V|^2(|E|+|V|\log|V|)\log|V|)$ time. □

Acknowledgements

This research was partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research (C), No. 15500011, 2003.

References

[1] A. Aho, J. Hopcroft, and J. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.
[2] K. P. Eswaran and R. E. Tarjan, "Augmentation problems," *SIAM J. Comput.*, Vol. 5, pp. 653–655, 1976.
[3] S. Even, *Graph Algorithms*, Pitman, London, 1979.
[4] L. Fleischer, "A 2-approximation for minimum cost {0,1,2} vertex connectivity," *Lecture Notes in Computer Science*, Vol. 2081, pp. 115–129, 2001.
[5] G. N. Frederickson and J. Ja'ja', "Approximation algorithms for several graph augmentation problems," *SIAM J. Comput.*, Vol. 10, No. 2, pp. 270–283, 1981.
[6] H. N. Gabow, "A representation for crossing set families with applications to submodular flow problems," in *Proc. 4th ACM Symposium on Discrete Algorithms*, pp. 202–211, 1993.
[7] H. N. Gabow, "A matroid approach to finding edge connectivity and packing arborescences," *Journal of Computer and System Sciences*, Vol. 50, pp. 259–273, 1996.
[8] H. N. Gabow, Z. Galil, T. Spencer, and R. E. Tarjan, "Efficient algorithms for finding minimum spanning trees in undirected and directed graphs," *Combinatorica*, Vol. 6, No. 2, pp. 109–122, 1986.
[9] T. Hsu and V. Ramachandran, "On finding a smallest augmentation to biconnect a graph," *SIAM J. Comput.*, Vol. 22, No. 5, pp. 889–912, 1983.
[10] K. Jain, "A factor 2 approximation algorithm for the generalized Steiner network problem," *Combinatorica*, Vol. 21, pp. 39–60, 2001.
[11] A. V. Karzanov and E. A. Timofeev, "Efficient algorithm for finding all minimal edge cuts of a nonoriented graph," *Cybernetics*, pp. 156–162, March-April 1986, Translated from Kibernetika, 2 (1986), 8–12.
[12] S. Khuller and R. Thurimella, "Approximation algorithms for graph augmentation," *Journal of Algorithms*, Vol. 14, pp. 214–225, 1993.
[13] S. Khuller and U. Vishkin, "Biconnectivity approximations and graph carvings," *Journal of the ACM*, Vol. 41, No. 2, pp. 214–235, 1994.
[14] P. Klein and R. Ravi, "When cycles collapse: A general approximation technique for constrained two-connectivity problems," in *Proceedings of the Third MPS Conference on Integer Programming and Combinatorial Optimization*, pp. 39–55, 1993.
[15] T. Mashima, S. Taoka, and T. Watanabe, "Approximation algorithms for the k-edge-connectivity augmentation problem," Tech. Rep. COMP92-24, IEICE of Japan, pp. 11–20, 1992.
[16] T. Mashima, S. Taoka, and T. Watanabe, "A 2-approximation algorithm to $(k+1)$-edge-connect a specified set of vertices in a k-edge-connected graph," *IEICE Trans. Fundamentals*, Vol. E88-A, No. 5, May 2005, (to appear).
[17] T. Mashima and T. Watanabe, "Approximation Algorithms for the k-Edge-Connectivity Augmentation Problem," in *1995 IEEE International Symposium on Circuits and Systems*, pp. 155–158, May 1995.
[18] H. Nagamochi, S. Nakamura, and T. Ishii, "Constructing a cactus for minimum cuts of a graph in $O(mn+n^2\log n)$ time and $O(m)$ space," *IEICE Trans. Fundamentals*, Vol. E86-D, No. 2, pp. 179–185, 2003.
[19] M. Penn and H. Shasha-Krupnik, "Improved approximation algorithms for weighted 2- and 3-vertex connectivity augmentation problems," *Journal of Algorithms*, Vol. 22, pp. 187–196, 1997.
[20] R. Ravi and D. P. Williamson, "An approximation algorithm for minimum-cost vertex-connectivity problems," *Algorithmica*, Vol. 18, pp. 21–43, 1997.
[21] A. Rosenthal and A. Goldner, "Smallest augmentations to biconnect a graph," *SIAM J. Comput.*, Vol. 6, No. 1, pp. 55–66, 1977.
[22] M. Tamura, "A study on approximation algorithms for graph connectivity augmentation problems," M.S. thesis, Graduate School of Engineering, Hiroshima University, 2003.
[23] M. Tamura, S. Taoka, and T. Watanabe, "A $(2-2/|L|)$-approximation algorithm *R2VS* or *R2ES* to 2-vertex- or 2-edge-connect specified vertices in a graph," IPSJ SIG Notes AL87-3, IPSJ, pp. 17–24, Nov. 2002.
[24] M. Tamura, S. Taoka, and T. Watanabe, "A 2-approximation algorithm 2-ABIS for 2-vertex-connectivity augmentation of specified vertices in a graph," *IEICE Trans. Fundamentals*, Vol. E86–A, No. 4, pp. 105–111, Apr. 2003.
[25] M. Tamura, S. Taoka, and T. Watanabe, "$(2-\frac{2}{|V|})$-approximation algorithms for several graph connectivity related problems," Tech. Rep. COMP2003-6, IEICE of Japan, pp. 39–46, Apr. 2003.
[26] S. Taoka, T. Mashima, and T. Watanabe, "A 2-approximation algorithm FSA+1 to $(\lambda+1)$-edge-connect a specified set of vertices in a λ-edge-connected graph," in *Proc. of the 2003 IEEE International Symposium on Circuits and Systems (ISCAS 2003)*, pp. 236–239, May 2003.
[27] R. E. Tarjan, *Data Structures and Network Algorithms*, CBMS-NSF Regional Conference Series in Applied Mathematics, SIAM, Philadelphia, 1983.
[28] T. Watanabe, Y. Higashi, and A. Nakamura, "Construction robust networks by means of graph augmentation problems," *IEICE Trans. Fundamentals (Japanese Edition)*, Vol. J73-A, No. 7, pp. 1242–1254, 1990, Also see Electronics and Communications in Japan, Part 3, Vol. 74, No. 2, pp. 79–96 (1991).

Approximately Separating Systems

GÁBOR WIENER

Department of Computer Science and Information Theory
Budapest University of Technology and Economics
wiener@cs.bme.hu

Abstract: The concept of approximately separating systems is introduced and several results concerning separating systems (like the bounds of Katona and Wegener about separating systems of sets of at most a fixed k elements) are generalized.

Keywords: separating system, combinatorial search, approximate search

1 Introduction

A set system $\mathscr{H} \subseteq 2^S$ is said to be a *separating system* if for any two elements $x,y \in S$ there exists a set $H \in \mathscr{H}$, such that $|\{x,y\} \cap H| = 1$. Separating systems play an important role in Combinatorial Search: one can find a hidden element asking a sequence of question sets fixed before any of the questions was answered (such a sequence is called a *successful non-adaptive search algorithm*) if and only if they form a separating system. (About Combinatorial Search see the book by Aigner [1] and the papers of Rényi [5] and Katona [3], [4].)

If we do not have to determine the hidden element exactly, that is, it is enough to find a "small" set that contains x, then we speak of *approximate search*. The most general definition of being "small" is that the set system of the "small" sets is a given hereditary set system $\mathscr{G} \subseteq 2^S$. If $\mathscr{G}$ consists of only the one-element sets of the underlying set and the empty set, then we obtain the standard model of Combinatorial Search. We would like to generalize the concept of separating systems to the approximate case, that is, to describe those set systems that are successful non-adaptive approximate search algorithms.

Let us introduce the following notations. Let S be a finite set, $\mathscr{H} \subseteq 2^S$. A set $H \in \mathscr{H}$ is called a *minimal set* of $\mathscr{H}$ if there is no non-empty set $G \in \mathscr{H}$ such that $G \subsetneq H$. The set system of minimal sets of $\mathscr{H}$ is denoted by $\min \mathscr{H}$. We define set systems

$$\mathscr{H}^{k\cap} = \{ \cap H_i \mid H_i \in \mathscr{A} \subseteq \mathscr{H},\ |\mathscr{A}| = k \}, \quad \text{for } k \in \mathbb{N},$$
$$\mathscr{H}^{\cap} = \bigcup_{i \in \mathbb{N}} \mathscr{H}^{i\cap},$$
$$\mathscr{H}^{k\cup} = \{ \cup H_i \mid H_i \in \mathscr{A} \subseteq \mathscr{H},\ |\mathscr{A}| = k \}, \quad \text{for } k \in \mathbb{N},$$

and

$$\mathscr{H}^{\cup} = \bigcup_{i \in \mathbb{N}} \mathscr{H}^{i\cup},$$
$$\mathscr{H}^{-} = \{ \overline{H} \mid H \in \mathscr{H} \},$$

Now we define approximately separating systems.

Definition 1 A set system $\mathscr{H} \subseteq 2^S$ is a separating system on S with respect to hereditary set system $\mathscr{G}$ if $\min(\mathscr{H} \cup \mathscr{H}^-)^{\cap} \subseteq \mathscr{G}$.

Remark 2 It is easy to see that the elements of $\min(\mathscr{H} \cup \mathscr{H}^-)^{\cap}$ are disjoint and that every element of the underlying set is contained in some set of $\min(\mathscr{H} \cup \mathscr{H}^-)^{\cap}$, for any (nonempty) set system $\mathscr{H}$. Thus the condition $\min(\mathscr{H} \cup \mathscr{H}^-)^{\cap} \subseteq \mathscr{G}$ is equivalent to the condition that $\min(\mathscr{H} \cup \mathscr{H}^-)^{\cap}$ is a partition of the underlying set into sets of $\mathscr{G}$. This latter condition is useful, because *completely separating systems* with respect to some $\mathscr{G}$ can be defined similarly. A set system $\mathscr{C} \subseteq 2^S$ is completely separating, if for any elements $x,y \in S, x \neq y$ there exists a set $C \in \mathscr{C}$ such that $x \in C$ and $y \notin C$. It can be easily verified that $\mathscr{C}$ is completely separating if and only if $\min \mathscr{C}^{\cap}$ consists of the one-element subsets of the underlying set. This is generalized to the approximate case in the following way: a set system $\mathscr{C}$ is said to be completely separating with respect to a set system $\mathscr{G}$ if $\min \mathscr{C}^{\cap}$ is a partition of the underlying set into sets from $\mathscr{G}$.

It is easy to verify that $\mathscr{H} \subseteq 2^S$ is a separating system on S with respect to $\mathscr{G}$ if and only if it is a successful non-adaptive approximate search algorithm (with respect to $\mathscr{G}$), that is, given the answers of all of the questions $x \in H?$ $(H \in \mathscr{H})$ we can show a set $G \in \mathscr{G}$ that contains the hidden element x.

2 Separating systems of $\mathscr{R}_{\leq k}$

The set system $\mathscr{R}_{\leq k}$ is defined as the system of those subsets of the underlying set S that contain at most k elements.

We generalize the well-known bounds of Katona [1],[2] and Wegener [6] and an important theorem of Katona [2] concerning separating systems of $\mathscr{R}_{\leq k}$.

Definition 3 $L_{pre}(R,\mathscr{H},\mathscr{G})$ is the smallest size of a separating system $\mathscr{R}\subseteq\mathscr{H}$ with respect to $\mathscr{G}$.

Theorem 4 For a set S of n elements and $k,l\in\mathbb{Z}^+$, $k<\frac{n}{2}$, $l\leq k$, we have

$$L_{pre}(S,\mathscr{R}_{\leq k},\mathscr{R}_{\leq l})\geq\frac{n}{k}\cdot\frac{\log\frac{n}{l}}{\log e\frac{n}{k}}.$$

In fact, a similar method can be used to give a lower estimate on $L_{pre}(S,\mathscr{R}_{\leq k},\mathscr{G})$ for an arbitrary set system $\mathscr{G}$.

Definition 5 Let S be an arbitrary finite set and $\mathscr{P}=\{P_1,P_2,\ldots,P_r\}$ be a partition of S. The *distribution induced by* $\mathscr{P}$ is $X_{\mathscr{P}}:\{1,2,\ldots,r\}\to\mathbb{R}^+$,

$$X_{\mathscr{P}}(i)=\frac{|P_i|}{|S|}.$$

Theorem 6 Let S be a set of n elements, let $\mathscr{G}$ be an arbitrary set system on S and let $k<\frac{n}{2}$. Denote the number $\min\limits_{\substack{\mathscr{P}\subseteq\mathscr{G}\\ \text{partition}}}H(X_{\mathscr{P}})$ by $H(\mathscr{G})$. Then

$$L_{pre}(S,\mathscr{R}_{\leq k},\mathscr{G})\geq\frac{n}{k}\cdot\frac{H(\mathscr{G})}{\log e\frac{n}{k}}.$$

Now we give an upper bound on $L_{pre}(S,\mathscr{R}_{\leq k},\mathscr{R}_{\leq l})$ by generalizing a theorem of Wegener [6]:

Theorem 7 For a set S of n elements and $k,l\in\mathbb{Z}^+$, $k<\frac{n}{2}$, $l\leq k$, we have

$$L_{pre}(S,\mathscr{R}_{\leq k},\mathscr{R}_{\leq l})\leq\left\lceil\frac{\lceil n/l\rceil}{\lfloor k/l\rfloor}-1\right\rceil\cdot\left\lceil\frac{\log\frac{n}{l}}{\log\frac{n}{k}}\right\rceil.$$

3 Constructing approximately separating systems of sets of size k and at most k

We give a generalization of a theorem of Katona [2].

Theorem 8 For a set S of n elements and $k,l\in\mathbb{Z}^+$, $k<\frac{n}{2}$, $l\leq k$, $L_{pre}(S,\mathscr{R}_{\leq k},\mathscr{R}_{\leq l})$ is equal to the least number $m\in\mathbb{Z}^+$, for which there exists a system of non-negative integers $s_0,s_1,\ldots,s_m$ satisfying the following three conditions:

$$m\cdot k=\sum_{i=0}^{m}i\cdot s_i, \tag{1}$$

$$n=\sum_{i=0}^{m}s_i, \tag{2}$$

$$s_i\leq l\cdot\binom{m}{i}\quad i=0,1,\ldots,m. \tag{3}$$

Remark 9 This theorem may be applied to construct small completely separating systems of sets of at most (or precisely) k elements. This is important, because determining the size of the smallest completely separating system of sets of at most (or precisely) k elements is an open problem.

References

[1] M. AIGNER, Combinatorial Search, *John Wiley, Chichester and Teubner, Stuttgart*, 1988

[2] G. KATONA, On separating systems of a finite set, *J. of Combinatorial Theory* **1**, (1966), 174-194.

[3] G. KATONA, Combinatorial search problems, *Survey of Combinatorial Theory, ed. by J. Srivastava et al., North-Holland, Amsterdam*, 1973, 285-308.

[4] G. KATONA, Rényi and the combinatorial search problems, *Stud. Sci. Math. Hung.*, **26**, (1991), 363-378.

[5] A. RÉNYI, Lectures on the theory of search, *Univ. North Carolina, Mimeo Series*, 1969

[6] I. WEGENER, On separating systems whose elements are sets of at most k elements, *Discrete Mathematics*, **28**, (1979), 219-222.

Coding Floorplans with Fewer Bits

(Extended Abstract)

KATSUHISA YAMANAKA

Department of Computer Science
Gunma University
1–5–1 Tenjin-Cho, Kiryu, 376–8515, Japan
yamanaka@msc.cs.gunma-u.ac.jp

SHIN-ICHI NAKANO

Department of Computer Science
Gunma University
1–5–1 Tenjin-Cho, Kiryu, 376–8515, Japan
nakano@msc.cs.gunma-u.ac.jp

Abstract: A naive coding of floorplans needs $2m$ bits for each floorplan, where m is the number of edges, in this paper we give a new simple coding of floorplans. Our coding needs only $5m/3$ bits for each floorplan.

Keywords: graph, floorplan, coding, removing sequence

1 Introduction

In this paper we consider the problem of encoding a given floorplan R into a binary string S so that S can be decoded to reconstruct R. Furthermore we wish to minimize the length of S.

Succinct representation of graphs are studied for many classes of graphs, for instance, trees[5, 10] and plane graphs[2, 3, 4, 8].

The well known naive coding of ordered trees is as follows. Given a ordered trees T we traverse T starting at the root with depth first manner. If we go down an edge then we code it with 0, and if we go up an edge then we code it with 1. Thus any n-vertex ordered tree T has a code with $2(n-1) = 2m$ bits, where m is the number of edges in T. Some examples are shown in Fig. 1.

On the other hand, the number of ordered trees with n vertices is known as the Catalan number C_{n-1}, and it is defined as follows[9, p.145].

$$C_n = \frac{1}{(n+1)}\frac{2n!}{n!n!}$$

For example, the number of ordered trees with four vertices is $C_{4-1} = 5$, as depicted in Fig. 1. We need at least $\log C_{n-1}$ bits to code those ordered trees with n vertices. Because the Catalan number can be denoted as follows[1, p495],

$$C_n = \frac{4^n}{(n+1)\sqrt{\pi n}}\left(1 - \frac{1}{8n} + \frac{1}{128n^2} + \frac{5}{1024n^3} - \frac{21}{32768n^4} + O(n^{-5})\right)$$

we need at least $\log C_{n-1} = 2n - o(n) = 2m - o(n)$ bits to code a ordered tree with n vertices. So the naive coding using $2m$ bits for each tree is asymptotically optimal.

In this paper we wish to code floorplans with a small number of bits.

A floorplan is a partition of a rectangle into a set of rectangles. For example, all floorplans with three faces are depicted in Fig. 2.

A naive coding of floorplans needs $2m$ bits for each floorplan, as we explain in Section 3. In this paper we give a new simple coding of floorplans, which needs only $5m/3$ bits for each floorplan.

Note that we cannot treat floorplans simply as plane graphs. See two floorplans in Fig. 3. They are identical as plane graphs, however different as floorplans. Because in Fig. 3(a) the two faces F_a and F_b shares a horizontal line, however in Fig. 3(b) they shares a vertical line. We need to store the direction (horizontal or vertical) for each edge in a given floorplan.

It is interesting that we need less bits for floorplans than ordered trees.

The rest of the paper is organized as follows. Section 2 gives some definitions. Section 3 explains the naive coding using $2m$ bits for each floorplan. Section 4 introduces "the removing sequence" of a floorplan, which is the main idea of our coding. Section 5 gives our new coding using $5m/3$ bits for each floorplan. Finally Section 6 is a conclusion.

2 Preliminaries

In this section we give some definitions.

Let G be a connected graph. A *tree* is a connected graph with no cycle. A *rooted* tree is a tree with one vertex r chosen as its *root* . A *ordered* tree is a rooted tree with fixed orderings for siblings.

A drawing of a graph is *plane* if it has no two edges intersect geometrically except at a vertex to which they are both incident. A plane drawing divides the plane into connected regions called *faces*. The unbounded face is called *the outer face*, and other faces are called *inner faces*. Two faces F_1 and F_2 are *ns-adjacent* (north-south adjacent) if F_2 is located to the bottom of F_1 and they share a horizontal line segment. Two faces F_1 and F_2 are *ew-adjacent* (east-west adjacent) if F_2 is located to the left of F_1 and they share a vertical line segment.

A *floorplan* is a plane drawing in which every face (including the outer face) is a rectangle. A *based* floorplan is a floorplan with one designated bottom line segment on the contour of the outer face. The designated bottom line segment is called *the base*, and we always draw the base as the lowermost line segment of the drawing. For examples, all based floorplans with three faces are shown in Fig. 2, in which each base is depicted by a thick line. If two based floorplans P_1 and P_2 have a one-to-one correspondence between faces preserving ns- and ew-adjacency, and in which each base corresponding to the other, then we say P_1 and P_2 are isomorphic.

Let n be the number of vertices of a floorplan, m the number of edges, and f the number of faces (including the outer face). In this paper we only consider based floorplans having no vertex shared by four (or more) rectangles. Thus G has $n-4$ vertices with degree three, and 4 vertices with degree two (at the four corner of the outer face), and we have $2m = 3(n-4)+2\cdot 4$. The equation and the Euler's formula $n-m+f=2$ gives $n=2f$ and $m=3f-2$.

A vertex with degree three is *w-missing* (west missing) if it has edges to top, bottom and right. We denote the number of w-missing vertices as n_W. Similarly we define *e-missing* (east missing), *n-missing*, *s-missing*, n_E, n_N and n_S. Note that, since each w-missing vertex is the left end of a maximal horizontal line segment, and each e-missing vertex is the right end of a maximal horizontal line segment, $n_W = n_E$ holds in any floorplan. Similarly $n_N = n_S$ holds. Thus $n_W + n_N = (n-4)/2$.

An inner face F of a floorplan is *U-active* if F shares a line segment with the uppermost horizontal line segment of the contour of the outer face. Intuitively, a face is U-active if it touches the uppermost line segment. For convenience, we regard the outer face also as U-active.

A face F is *Uw-active* if F has a U-active face to the left with sharing a vertical line segment. Intuitively, a face is Uw-active if it touch some U-active face at the left (or west) boundary. Note that "some U-active face" may be the outer face.

3 The Naive Coding

In this section we sketch a naive code for floorplans, based on the depth first search of a tree. The code needs $2m+3$ bits for each floorplan.

Given a based floorplan R, we replace the lower right corner vertex of each inner face with two vertices as depicted in Fig. 4. See an example in Fig. 6. Note that since we only break each cycle corresponding to an inner face at the lower right corner, the resulting graph has only one face and is still connected. So the resulting graph R' is a tree. Also note that we need some tricky replacement at the two lower corners of the outer face. See Fig. 4.

Starting at the upper left corner, we traverse the tree R' with depth first manner (with left priority). When we visit a vertex, we have only two choices for the direction of the next vertex, as shown in Fig. 5. Since each edge has traced exactly twice, we need two bits for each edge, except for the first edge, which needs only one bit because we always trace the first edge to bottom, and since we have two dummy edges at the two lower corners, we can code the tree R' with $2m+3$ bits. Given the $2m+3$ bits code we can easily reconstruct the original floorplan R with a simple algorithm with a stack.

4 The Removing Sequence

Let R_1 be the based floorplan having exactly one inner face. Assume that $R(\neq R_1)$ is a based floorplan having $f>1$ faces.

Let F be the inner face of R having the upper-left corner of the outer face of R. We call such a face *the first* face of the based floorplan R. The first faces of based floorplans are shaded in Figs. 7– 9. Let v be the lower-right corner of F. The first face F is *upward removable* if R has a vertical line segment with upper end v. See Fig. 7 (a). Otherwise, R has a horizontal line segment with left end v, and the first face F is *leftward removable*. See Fig. 7 (b). Note that we have assumed that R has no vertex with degree four.

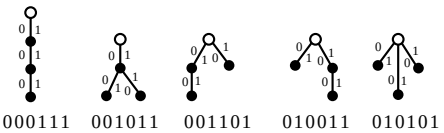


Figure 1: A code for ordered trees.

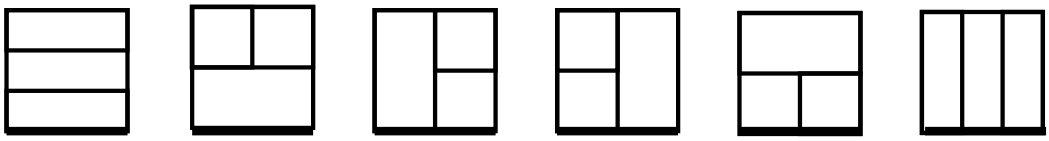


Figure 2: Floorplans with three faces.

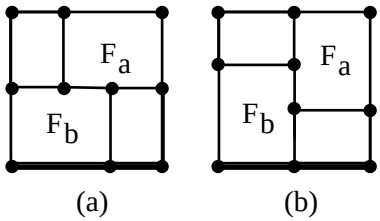


Figure 3: Two floorplans corresponding to the same plane graph.

Since $R \neq R_1$, the first face of R is either upward removable or leftward removable. If F is upward removable, then we can obtain a floorplan with one less faces by continually shrinking the first face F into the uppermost horizontal line of R, with preserving the width of F and enlarging the faces below F, as shown in Fig. 8. Similarly, if F is leftward removable, then we can obtain a floorplan with one less faces by continually shrinking F into the leftmost line of R with preserving the height of F. After we remove the first face from R, the resulting floorplan is again a based floorplan with one less faces. We denote such a floorplan as $P(R)$. Thus we can define the based floorplan $P(R)$ for each based floorplan $R \neq R_1$.

Given a based floorplan R having exactly f faces, by repeatedly removing the first face, we can have the unique sequence $R, P(R), P(P(R)), \cdots$ of based floorplans which eventually ends with R_1, which is the based floorplan having exactly one inner face. See an example in Fig. 9, in which the first faces are shaded. We call the sequence $R, P(R), P(P(R)), \cdots$ *the removing sequence* of R.

Let R_k be a floorplan with $k \leq f$ faces. Assume that the first face of R_k has $s(R_k)$ neighbor faces to the bottom and $e(R_k)$ neighbor faces to the right. Given $R_{k-1} = P(R_k)$, if we know (1) whether the first face of R_k is upward removable or leftward removable, and (2) the two values $s(R_k)$ and $e(R_k)$, then we can reconstruct R_k from those information. Thus, for each $k = 2, 3, \cdots, n$, if we store (1) whether the first face of R_k is upward removable or leftward removable, and (2) the two values $s(R_k)$ and $e(R_k)$, then we can reconstruct $R_2, R_3 \cdots, R = R_f$.

A simple coding needs $f-1$ bits for (1), and $2(f-1)\log f$ bits for (2). In the next section we give more efficient coding for floorplans. The coding needs only $5m/3$ bits for each floorplan.

5 Our Coding

In this section we give a simple coding for based floorplans. The coding needs only $5m/3$ bits for each floorplan.

Basically, given a floorplan R with f faces, we code R as the removing sequence of R.

Let $RS = (\, R_f(=R),\ R_{f-1}(=P(R)),\ R_{f-2}(=P(P(R))), \cdots,\ R_1\,)$ be the removing sequence of R. We construct two new sequences from RS as follows.

By choosing the based floorplans having upward removable first faces from RS, with preserving the order, we derive a new sequence $RS_U = (R_1^U, R_2^U, \cdots, R_{f_U}^U)$. Similarly, by choosing the based floorplans having leftward removable first faces

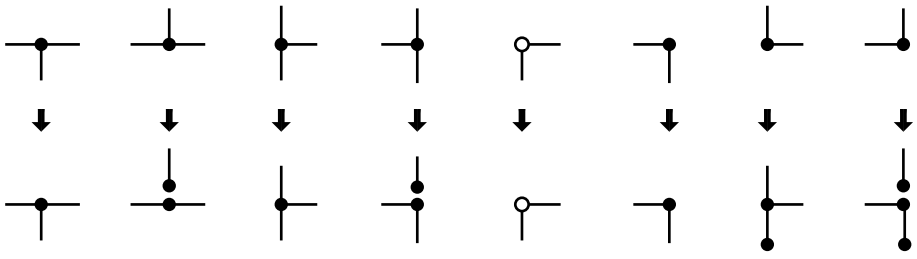


Figure 4: The replacement of vertices.

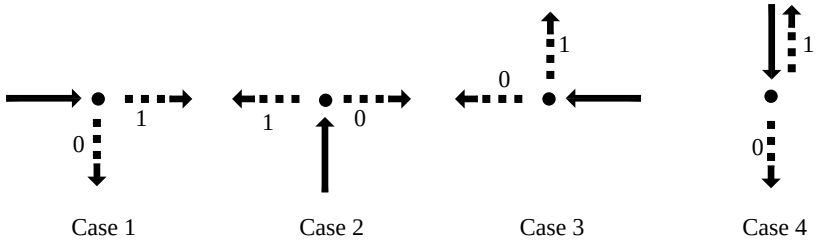


Figure 5: The code for the DFS.

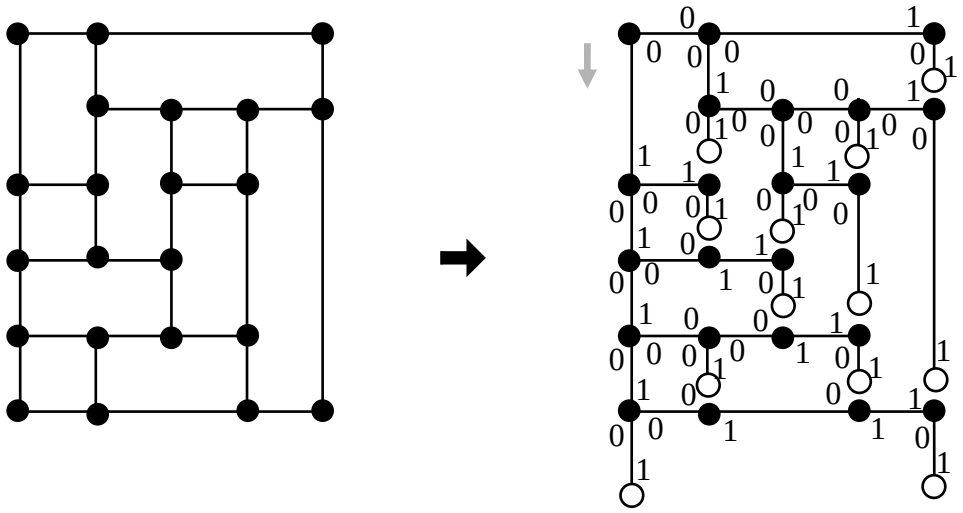


Figure 6: The $2m+3$ bit code for a floorplan based on the DFS.

from RS, we derive a new sequence $RS_L = (R_1^L, R_2^L, \cdots, R_{f_L}^L)$. Note that $f_U + f_L + 1 = f$ holds, since R_1 is contained in neither RS_U nor RS_L.

The coding consists of the following five sections.

Section 1: This section codes whether the first face of each $R_k, k = 2,3,\cdots,f$ is upward removable or leftward removable. For $k = 1,2,\cdots,f-1$, the k-th bit is 0 if the first face of R_{k+1} is upward removable, and 1 otherwise. Section 1 has lenght $f-1$ bits in total.

Section 2: Section 2 and 5 code each $s(R_k)$, $k = 2,3,\cdots,f$. If $R_k \in RS_U$, then we code $s(R_k)$ in Section 2, otherwise $R_k \in RS_L$, and we code $s(R_k)$ in Section 5. Section 2 has lenght $f-1$ bits in total.

Assume that $R_k = R_j^U \in RS_U$. Now the first face F of R_k is upward removable.

Let $s(R_k)$ be the number of faces which are U-active in R_{k-1}, but not U-active in R_k. That is the number of faces which are located to the bottom of F, and become U-active when we remove F. Note that $s(R_k) \geq 1$, since the first face F always has some face (possibly the outer face) to the bottom.

Also note that $s(R_1^U) + s(R_2^U) + \cdots + s(R_{f_U}^U) \leq f-1$, since each face becomes U-active exactly once, and R_f has at least one face which is already U-active. We can observe that when we remove a leftward removable face, no face newly becomes

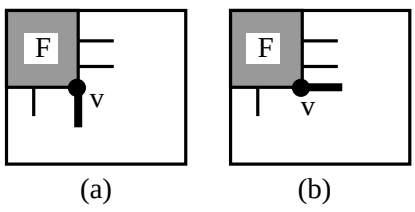


Figure 7: (a) An upward removable face and (b) a leftward removable face.

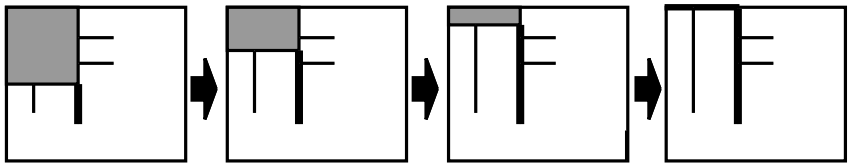


Figure 8: Removing the first face.

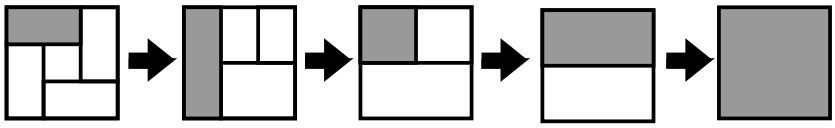


Figure 9: The removing sequence.

U-active.

We code each $s(R_j^U)$, $1 \leq j \leq f_U$, as the consecutive $s(R_j^U)-1$ copies of "0"s followed by a "1". (Note that, as we mentioned above, $s(R_j^U) \geq 1$ holds for each k.) For example, we code $s(R_j^U)=5$ as "00001". After we concatenate those codes, we finally append zeroes so that the length of Section 2 becomes $f-1$ bits in total.

We can easily decode each $s(R_k)$ from the code.

Section 3: Section 3 and 4 code each $e(R_k), k=2,3,\cdots,f$. If $R_k \in RS_U$, then we code $e(R_k)$ in Section 3, otherwise $R_k \in RS_L$, and we code $e(R_k)$ in Section 4.

We don't directly code each $e(R_k)$, since the sum of them may be large. Our idea is as follows. See Fig. 10(a). The first face of R_k has three $= s(R_k)$ neighbors to the bottom, and three $= e(R_k)$ neighbors to the right. Let $F_1, F_2, \cdots, F_{s(R_k)}$ be the faces located to the bottom of F. We are going to code the number $e(R_k)=3$.

Given R_{k-1}, we know the number e' of faces located to the right of $F_{s(R_k)}$. In the example of Fig. 10(b), $e'=6$. Let e_k be the number of faces which are located to the right of $F_{s(R_k)}$, and Uw-active in R_{k-1}, but not Uw-active in R_k. Those faces are shaded in Fig. 10(b). We can observe that each of those faces has a w-missing vertex at the upper left corner in R. (This is our idea to bound the length of Section 3 by n_W.) Those vertices are depicted as white circle in Fig. 10(b). If we have R_{k-1} and $s(R_k)$, then we can count the value of e'. Now we can compute $e(R_k)$ by $e'-e_k$, if we have e_k. Thus we can store e_k instead of $e(R_k)$.

Now we formally explain the code.

Assume that $R_k = R_j^U \in RS_U$. Now the first face F of R_k is upward removable. See Fig. 10(a).

We code each e_k as the consecutive e_k copies of "0"s followed by a "1". By concatenating those codes, we have the code for section 3. We can easily decode each e_k, and then compute $e(R_k)$.

Note that $e_2+e_3+\cdots+e_f \leq n_W$, since each face becomes Uw-active exactly once, and R_f has at least one face which is already Uw-active. Similar to Section 2, we finally append zeroes so that the length of Section 3 becomes n_W+f_U bits in total

Section 4: This section codes each $e(R_k), k=2,3,\cdots,f$ only for each $R_k \in RS_L$. (We code each $R_k \in RS_U$ in Section 3.)

Omitted. Similar to Section 2. Section 4 has length $f-1$ bits in total.

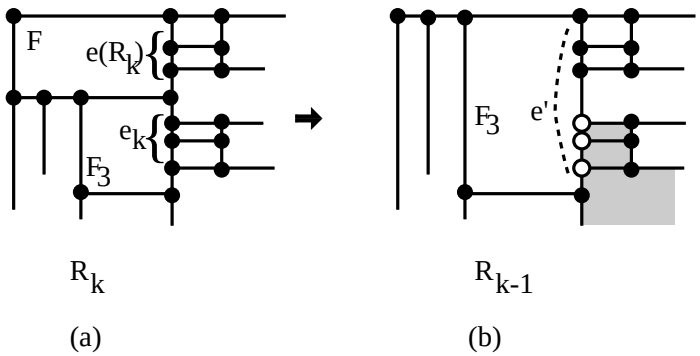


Figure 10: An illustration for Section 3.

Section 5: This section codes each $s(R_k), k = 2, 3, \cdots, n$ only for each $R_k \in RS_L$. (For $R_k \in RS_U$ we code each $s(R_k)$ in Section 2.)

Omitted. Similar to Section 3.

Section 5 has length $n_N + f_L$ bits in total.

Note that $n_W + n_N = (n-4)/2 = (f-2)$ holds. Thus the total length of the code consisting of the five sections above is

$$\begin{aligned}&(f-1)+(f-1)+(n_W+f_U)+(f-1)+(n_N+f_L)\\ &=(4f-4)+(f-2)=5f-6\\ &=\frac{5(m+2)}{3}-6=\frac{5m-8}{3}\end{aligned}$$

We have the following theorem.

Theorem 1 One can encode a floorplan with $5m/3$ bits.

6 Conclusion

In this paper we gave a simple and short coding for floorplans. The coding needs only $5m/3$ bits for each floorplan.

An efficient enumeration algorithm for based floorplans is known[6, 7]. Let N_k be the number of based floorplans with k faces. By implementing the algorithm we have counted $N_{11} = 10948768$. Thus we need at least $24 > \log N_{11} = 23.5$ bits to code based floorplans with 11 faces, while the total length of our coding is $5 \cdot 11 - 6 = 49$ bits. Thus there are still many chances to reduce the length of the code.

For $N_{12} = 89346128$ we need at least $27 > \log N_{12} = 26.4$ bits to code based floorplans with 12 faces, while our coding needs $5 \cdot 12 - 6 = 54$ bits.

References

[1] R. L. GRAHAM, D. E. KNUTH AND O. PATASHNIK, Concrete Mathematics, 2nd ed., (1994)

[2] Y.-T. CHIANG, C.-C. LIN AND H.-I LU, Orderly Spanning Trees with Applications to Graph Encoding and Graph Drawing, *Proc. of 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, (2001), pp.506–515.

[3] R. C.-N CHUANG, A. GARG, X. HE, M.-Y. KAO, H.-I LU, Compact Encodings of Planar Graphs via Canonical Orderings and Multiple Parentheses, *Proc. of 25th International Colloquium on Automata, Languages, and Programming, LNCS 1443*, (1998), pp.118–129.

[4] K. KEELER AND J. WESTBROOK, Short Encodings of Planar Graphs and Maps, *Discrete Applied Mathematics*, 58, No3, (1995), pp.239–252.

[5] J. IAN MUNRO AND VENKATESH RAMAN, Succinct Representation of Balanced Parentheses, Static Trees and Planar Graphs, *Proc. of 38th IEEE Symposium on Foundations of Computer Science*, (1997), pp.118–126.

[6] S. NAKANO, Enumerating Floorplans with n Rooms, *IEICE TRANS. FUNDAMENTALS*, Vol.E85-A, No. 7, (2002), pp.1746–1750.

[7] S. NAKANO, Enumerating Floorplans with Some Properties, *Interdisciplinary Information Sciences*, Vol. 8, No. 2, (2002), pp.199-206.

[8] C. PAPADIMITRIOU AND M. YANNAKAKIS, A Note on Succinct Representations of Graphs, *Information and Computation*, Vol. 71, (1985), pp.181–185.

[9] K. H. ROSEN (EDS.), Handbook of Discrete and Combinatorial Mathematics, *CRC Press, Boca Raton*, (2000).

[10] G. TURAN, Succinct Representations of Graphs, *Discrete Applied Math*, Vol. 8, (1984), pp.289–294.

Index