Important separators and parameterized algorithms

Dániel Marx[1]

[1]Institute for Computer Science and Control,
Hungarian Academy of Sciences (MTA SZTAKI)
Budapest, Hungary

ICERM
Providence, RI
April 27, 2014

Overview

Main message

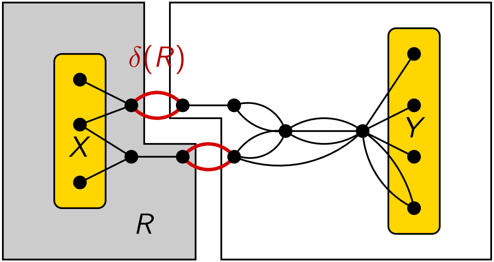
Small separators in graphs have interesting extremal properties that can be exploited in combinatorial and algorithmic results.

- Bounding the number of “important” cuts.
- Edge/vertex versions, directed/undirected versions.
- Algorithmic applications: FPT algorithm for
  - Multiway cut,
  - Directed Feedback Vertex Set, and
  - (p, q)-Clustering.

Important cuts

Definition: $\delta(R)$ is the set of edges with exactly one endpoint in R.
Definition: A set S of edges is a **minimal (X,Y)-cut** if there is no $X-Y$ path in $G \setminus S$ and no proper subset of S breaks every $X-Y$ path.

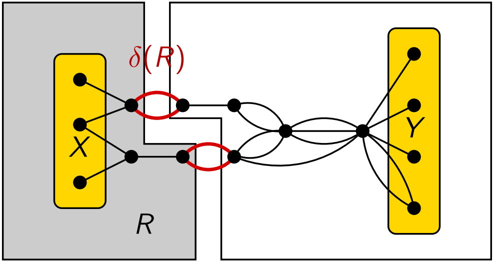
Observation: Every minimal (X,Y)-cut S can be expressed as $S = \delta(R)$ for some $X \subseteq R$ and $R \cap Y = \emptyset$.



Important cuts

Definition: A minimal (X, Y)-cut $\delta(R)$ is **important** if there is no (X, Y)-cut $\delta(R')$ with $R \subset R'$ and $|\delta(R')| \leq |\delta(R)|$.

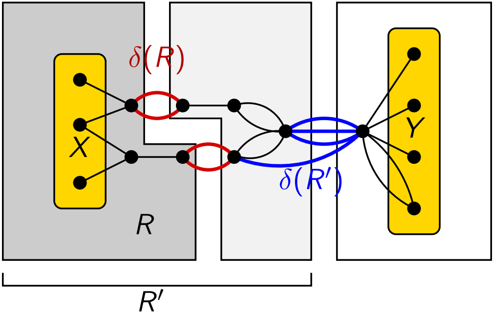
Note: Can be checked in polynomial time if a cut is important.



Important cuts

Definition: A minimal (X, Y)-cut $\delta(R)$ is **important** if there is no (X, Y)-cut $\delta(R')$ with $R \subset R'$ and $|\delta(R')| \leq |\delta(R)|$.

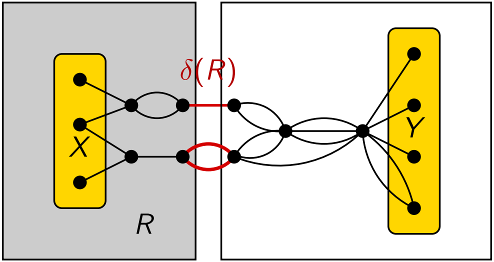
Note: Can be checked in polynomial time if a cut is important.



Important cuts

Definition: A minimal (X, Y)-cut $\delta(R)$ is **important** if there is no (X, Y)-cut $\delta(R')$ with $R \subset R'$ and $|\delta(R')| \le |\delta(R)|$.

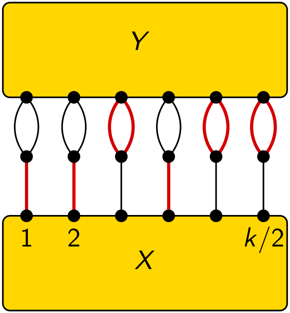
Note: Can be checked in polynomial time if a cut is important.



Important cuts

The number of important cuts can be exponentially large.

Example:

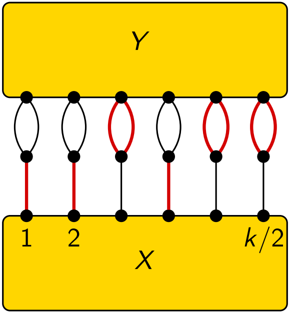


This graph has $2^{k/2}$ important (X, Y)-cuts of size at most k.

Important cuts

The number of important cuts can be exponentially large.

Example:



This graph has $2^{k/2}$ important (X, Y)-cuts of size at most k.

Theorem

There are at most 4^k important (X, Y)-cuts of size at most k.

(Proof is implicit in [Chen, Liu, Lu 2007], worse bound in [M. 2004].)

Submodularity

Fact: The function δ is **submodular:** for arbitrary sets A, B,

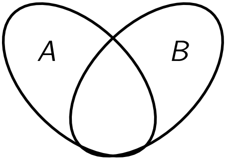
$$|\delta(A)| \quad + \quad |\delta(B)| \quad \geq \quad |\delta(A \cap B)| \quad + \quad |\delta(A \cup B)|$$

Submodularity

Fact: The function δ is **submodular:** for arbitrary sets A, B,

$$|\delta(A)| \quad + \quad |\delta(B)| \quad \geq \quad |\delta(A \cap B)| \quad + \quad |\delta(A \cup B)|$$

Proof: Determine separately the contribution of the different types of edges.

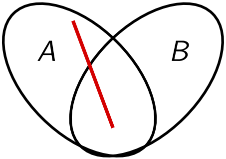


Submodularity

Fact: The function δ is **submodular:** for arbitrary sets A, B,

$$\underset{0}{|\delta(A)|} \quad + \quad \underset{1}{|\delta(B)|} \quad \geq \quad \underset{1}{|\delta(A \cap B)|} \quad + \quad \underset{0}{|\delta(A \cup B)|}$$

Proof: Determine separately the contribution of the different types of edges.

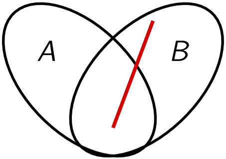


Submodularity

Fact: The function δ is **submodular:** for arbitrary sets A, B,

$$\underset{1}{|\delta(A)|} \quad + \quad \underset{0}{|\delta(B)|} \quad \geq \quad \underset{1}{|\delta(A \cap B)|} \quad + \quad \underset{0}{|\delta(A \cup B)|}$$

Proof: Determine separately the contribution of the different types of edges.

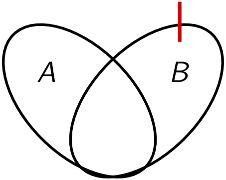


Submodularity

Fact: The function δ is **submodular:** for arbitrary sets A, B,

$$\underset{0}{|\delta(A)|} \quad + \quad \underset{1}{|\delta(B)|} \quad \geq \quad \underset{0}{|\delta(A \cap B)|} \quad + \quad \underset{1}{|\delta(A \cup B)|}$$

Proof: Determine separately the contribution of the different types of edges.

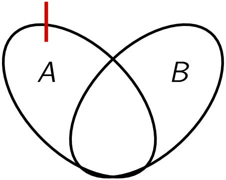


Submodularity

Fact: The function δ is **submodular:** for arbitrary sets A, B,

$$\underset{1}{|\delta(A)|} \quad + \quad \underset{0}{|\delta(B)|} \quad \geq \quad \underset{0}{|\delta(A \cap B)|} \quad + \quad \underset{1}{|\delta(A \cup B)|}$$

Proof: Determine separately the contribution of the different types of edges.

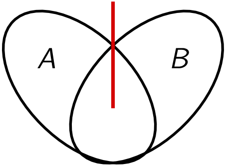


Submodularity

Fact: The function δ is **submodular:** for arbitrary sets A, B,

$$\underbrace{|\delta(A)|}_{1} \quad + \quad \underbrace{|\delta(B)|}_{1} \quad \geq \quad \underbrace{|\delta(A \cap B)|}_{1} \quad + \quad \underbrace{|\delta(A \cup B)|}_{1}$$

Proof: Determine separately the contribution of the different types of edges.

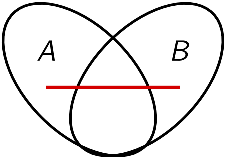


Submodularity

Fact: The function δ is **submodular:** for arbitrary sets A, B,

$$\underset{1}{|\delta(A)|} \quad + \quad \underset{1}{|\delta(B)|} \quad \geq \quad \underset{0}{|\delta(A \cap B)|} \quad + \quad \underset{0}{|\delta(A \cup B)|}$$

Proof: Determine separately the contribution of the different types of edges.



Submodularity

Lemma

Let λ be the minimum (X, Y)-cut size. There is a unique maximal $R_{\max} \supseteq X$ such that $\delta(R_{\max})$ is an (X, Y)-cut of size λ.

Submodularity

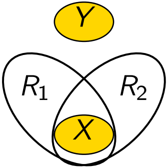
Lemma

Let λ be the minimum (X, Y)-cut size. There is a unique maximal $R_{\max} \supseteq X$ such that $\delta(R_{\max})$ is an (X, Y)-cut of size λ.

Proof: Let $R_1, R_2 \supseteq X$ be two sets such that $\delta(R_1), \delta(R_2)$ are (X, Y)-cuts of size λ.

$$\underbrace{|\delta(R_1)|}_{\lambda} + \underbrace{|\delta(R_2)|}_{\lambda} \geq \underbrace{|\delta(R_1 \cap R_2)|}_{\geq \lambda} + |\delta(R_1 \cup R_2)|$$

$$\Rightarrow |\delta(R_1 \cup R_2)| \leq \lambda$$



Note: Analogous result holds for a unique minimal $R_{\min}$.

Important cuts

Theorem

There are at most 4^k important (X, Y)-cuts of size at most k.

Proof: Let λ be the minimum (X, Y)-cut size and let $\delta(R_{\max})$ be the unique important cut of size λ such that $R_{\max}$ is maximal.

(1) We show that $R_{\max} \subseteq R$ for every important cut $\delta(R)$.

Important cuts

Theorem

There are at most 4^k important (X, Y)-cuts of size at most k.

Proof: Let λ be the minimum (X, Y)-cut size and let $\delta(R_{\max})$ be the unique important cut of size λ such that $R_{\max}$ is maximal.

(1) We show that $R_{\max} \subseteq R$ for every important cut $\delta(R)$.

By the submodularity of δ:

$$\underbrace{|\delta(R_{\max})|}_{\lambda} + |\delta(R)| \geq \underbrace{|\delta(R_{\max} \cap R)|}_{\geq \lambda} + |\delta(R_{\max} \cup R)|$$

$$\Downarrow$$

$$|\delta(R_{\max} \cup R)| \leq |\delta(R)|$$

$$\Downarrow$$

If $R \neq R_{\max} \cup R$, then $\delta(R)$ is not important.

Important cuts

Theorem

There are at most 4^k important (X, Y)-cuts of size at most k.

Proof: Let λ be the minimum (X, Y)-cut size and let $\delta(R_{\max})$ be the unique important cut of size λ such that $R_{\max}$ is maximal.

(1) We show that $R_{\max} \subseteq R$ for every important cut $\delta(R)$.

By the submodularity of δ:

$$\underbrace{|\delta(R_{\max})|}_{\lambda} + |\delta(R)| \ge \underbrace{|\delta(R_{\max} \cap R)|}_{\ge \lambda} + |\delta(R_{\max} \cup R)|$$

$$\Downarrow$$

$$|\delta(R_{\max} \cup R)| \le |\delta(R)|$$

$$\Downarrow$$

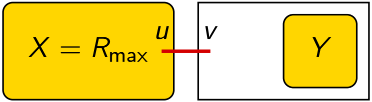
If $R \neq R_{\max} \cup R$, then $\delta(R)$ is not important.

Thus the important (X, Y)- and $(R_{\max}, Y)$-cuts are the same.
$\Rightarrow$ We can assume $X = R_{\max}$.

Important cuts

(2) Search tree algorithm for enumerating all these cuts:

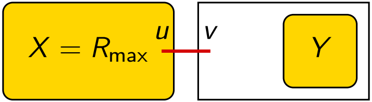
An (arbitrary) edge uv leaving $X = R_{\max}$ is either in the cut or not.



Important cuts

(2) Search tree algorithm for enumerating all these cuts:

An (arbitrary) edge uv leaving $X = R_{\max}$ is either in the cut or not.



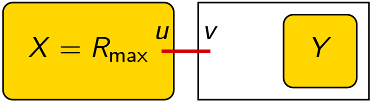
Branch 1: If $uv \in S$, then $S \setminus uv$ is an important (X, Y)-cut of size at most $k - 1$ in $G \setminus uv$.

Branch 2: If $uv \notin S$, then S is an important $(X \cup v, Y)$-cut of size at most k in G.

Important cuts

(2) Search tree algorithm for enumerating all these cuts:

An (arbitrary) edge uv leaving $X = R_{\max}$ is either in the cut or not.



Branch 1: If $uv \in S$, then $S \setminus uv$ is an important (X, Y)-cut of size at most $k - 1$ in $G \setminus uv$.

$\Rightarrow$ k decreases by one, λ decreases by at most 1.

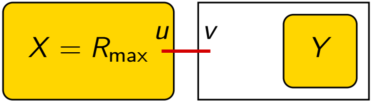
Branch 2: If $uv \notin S$, then S is an important $(X \cup v, Y)$-cut of size at most k in G.

$\Rightarrow$ k remains the same, λ increases by 1.

Important cuts

(2) Search tree algorithm for enumerating all these cuts:

An (arbitrary) edge uv leaving $X = R_{\max}$ is either in the cut or not.



Branch 1: If $uv \in S$, then $S \setminus uv$ is an important (X, Y)-cut of size at most $k-1$ in $G \setminus uv$.

$\Rightarrow$ k decreases by one, λ decreases by at most 1.

Branch 2: If $uv \notin S$, then S is an important $(X \cup v, Y)$-cut of size at most k in G.

$\Rightarrow$ k remains the same, λ increases by 1.

The measure $2k - \lambda$ decreases in each step.

$\Rightarrow$ Height of the search tree $\leq 2k$

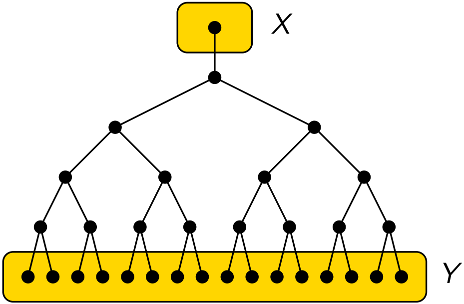
$\Rightarrow$ $\leq 2^{2k} = 4^k$ important cuts of size at most k.

Important cuts

Theorem

There are at most 4^k important (X, Y)-cuts of size at most k.

Example: The bound 4^k is essentially tight.

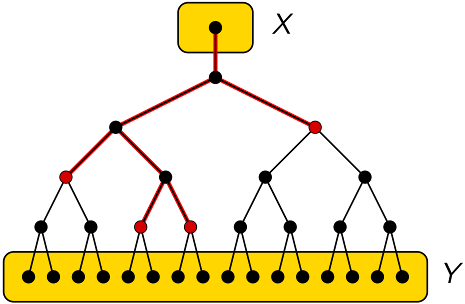


Important cuts

Theorem

There are at most 4^k important (X, Y)-cuts of size at most k.

Example: The bound 4^k is essentially tight.



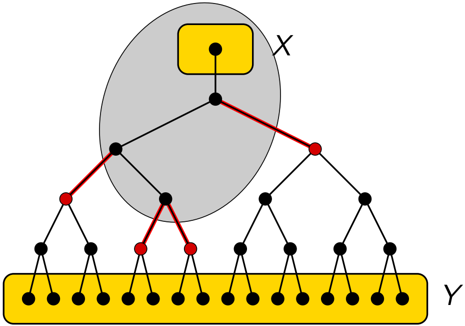
Any subtree with k leaves gives an important (X, Y)-cut of size k.

Important cuts

Theorem

There are at most 4^k important (X, Y)-cuts of size at most k.

Example: The bound 4^k is essentially tight.



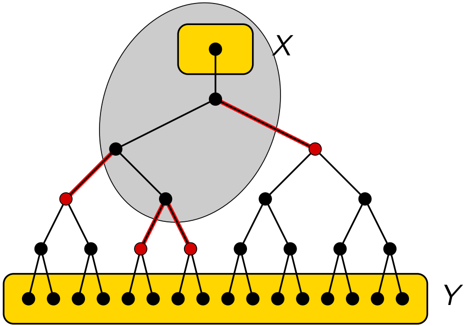
Any subtree with k leaves gives an important (X, Y)-cut of size k.

Important cuts

Theorem

There are at most 4^k important (X, Y)-cuts of size at most k.

Example: The bound 4^k is essentially tight.



Any subtree with k leaves gives an important (X, Y)-cut of size k.
The number of subtrees with k leaves is the Catalan number

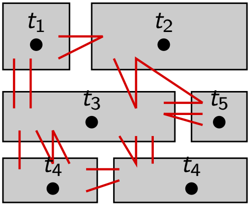
$$C_{k-1} = \frac{1}{k}\binom{2k-2}{k-1} \geq 4^k/\mathrm{poly}(k).$$

Multiway Cut

Definition: A **multiway cut** of a set of terminals T is a set S of edges such that each component of $G \setminus S$ contains at most one vertex of T.

Multiway Cut

Input:	Graph G, set T of vertices, integer k
Find:	A **multiway cut** S of at most k edges.

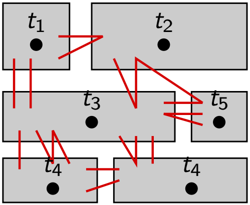


Polynomial for $|T| = 2$, but NP-hard for any fixed $|T| \geq 3$ [Dalhaus et al. 1994].

Multiway Cut

Definition: A **multiway cut** of a set of terminals T is a set S of edges such that each component of $G \setminus S$ contains at most one vertex of T.

Multiway Cut	
Input:	Graph G, set T of vertices, integer k
Find:	A **multiway cut** S of at most k edges.



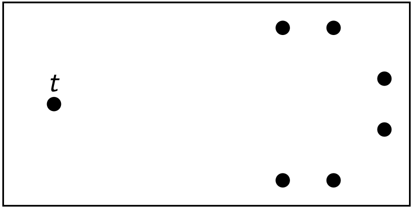
Trivial to solve in polynomial time for fixed k (in time $n^{O(k)}$).

Theorem

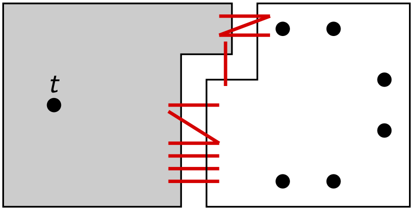
Multiway cut can be solved in time $4^k \cdot n^{O(1)}$, i.e., it is fixed-parameter tractable (FPT) parameterized by the size k of the solution.

Multiway Cut

Intuition: Consider a $t \in T$. A subset of the solution S is a $(t, T \setminus t)$-cut.



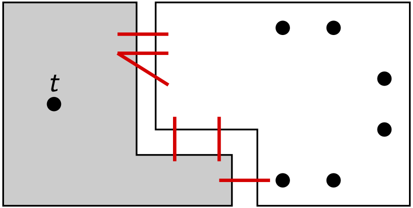
Intuition: Consider a $t \in T$. A subset of the solution S is a $(t, T \setminus t)$-cut.



There are many such cuts.

Multiway Cut

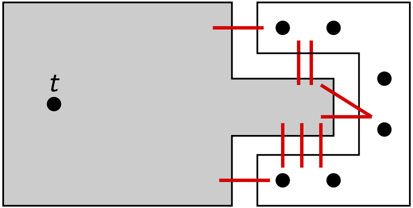
Intuition: Consider a $t \in T$. A subset of the solution S is a $(t, T \setminus t)$-cut.



There are many such cuts.

Multiway Cut

Intuition: Consider a $t \in T$. A subset of the solution S is a $(t, T \setminus t)$-cut.



There are many such cuts.

But a cut farther from t and closer to $T \setminus t$ seems to be more useful.

Multiway Cut and important cuts

Pushing Lemma

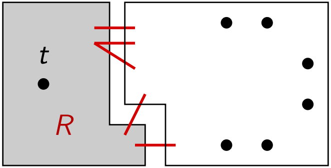
Let $t \in T$. The Multiway Cut problem has a solution S that contains an important $(t, T \setminus t)$-cut.

Multiway Cut and important cuts

Pushing Lemma

Let $t \in T$. The Multiway Cut problem has a solution S that contains an important $(t, T \setminus t)$-cut.

Proof: Let R be the vertices reachable from t in $G \setminus S$ for a solution S.

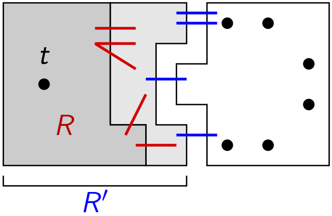


Multiway Cut and important cuts

Pushing Lemma

Let $t \in T$. The Multiway Cut problem has a solution S that contains an important $(t, T \setminus t)$-cut.

Proof: Let R be the vertices reachable from t in $G \setminus S$ for a solution S.



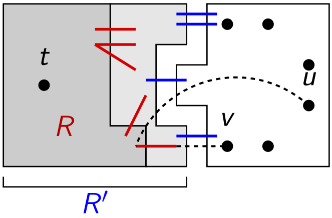
$\delta(R)$ is not important, then there is an important cut $\delta(R')$ with $R \subset R'$ and $|\delta(R')| \le |\delta(R)|$. Replace S with $S' := (S \setminus \delta(R)) \cup \delta(R') \Rightarrow |S'| \le |S|$

Multiway Cut and important cuts

Pushing Lemma

Let $t \in T$. The Multiway Cut problem has a solution S that contains an important $(t, T \setminus t)$-cut.

Proof: Let R be the vertices reachable from t in $G \setminus S$ for a solution S.



$\delta(R)$ is not important, then there is an important cut $\delta(R')$ with $R \subset R'$ and $|\delta(R')| \leq |\delta(R)|$. Replace S with $S' := (S \setminus \delta(R)) \cup \delta(R') \Rightarrow |S'| \leq |S|$

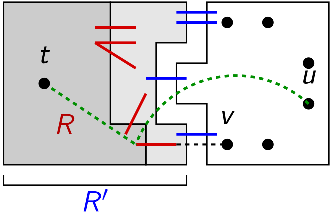
S' is a multiway cut: (1) There is no t-u path in $G \setminus S'$ and (2) a u-v path in $G \setminus S'$ implies a t-u path, a contradiction.

MULTIWAY CUT and important cuts

Pushing Lemma

Let $t \in T$. The MULTIWAY CUT problem has a solution S that contains an important $(t, T \setminus t)$-cut.

Proof: Let R be the vertices reachable from t in $G \setminus S$ for a solution S.



$\delta(R)$ is not important, then there is an important cut $\delta(R')$ with $R \subset R'$ and $|\delta(R')| \leq |\delta(R)|$. Replace S with $S' := (S \setminus \delta(R)) \cup \delta(R') \Rightarrow |S'| \leq |S|$

S' is a multiway cut: (1) There is no t-u path in $G \setminus S'$ and (2) a u-v path in $G \setminus S'$ implies a t-u path, a contradiction.

Algorithm for MULTIWAY CUT

1. If every vertex of T is in a different component, then we are done.
2. Let $t \in T$ be a vertex that is not separated from every $T \setminus t$.
3. Branch on a choice of an important $(t, T \setminus t)$ cut S of size at most k.
4. Set $G := G \setminus S$ and $k := k - |S|$.
5. Go to step 1.

We branch into at most 4^k directions at most k times.

(Better analysis gives 4^k bound on the size of the search tree.)

Multicut

Multicut

Input: Graph G, pairs (s_1, t_1), ..., (s_ℓ, t_ℓ), integer k

Find: A set S of edges such that $G \setminus S$ has no s_i-t_i path for any i.

Theorem

Multicut can be solved in time $f(k, \ell) \cdot n^{O(1)}$ (FPT parameterized by combined parameters k and ℓ).

Multicut

Multicut

Input: Graph G, pairs (s_1, t_1), ..., (s_ℓ, t_ℓ), integer k

Find: A set S of edges such that $G \setminus S$ has no s_i-t_i path for any i.

Theorem

Multicut can be solved in time $f(k, \ell) \cdot n^{O(1)}$ (FPT parameterized by combined parameters k and ℓ).

Proof: The solution partitions $\{s_1, t_1, \dots, s_\ell, t_\ell\}$ into components. Guess this partition, contract the vertices in a class, and solve Multiway Cut.

Theorem [Bousquet, Daligault, Thomassé 2011] [M., Razgon 2011]

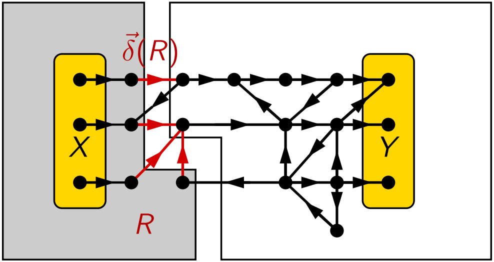
Multicut is FPT parameterized by the size k of the solution.

Directed graphs

Definition: $\vec{\delta}(R)$ is the set of edges leaving R.

Observation: Every inclusionwise-minimal directed (X, Y)-cut S can be expressed as $S = \vec{\delta}(R)$ for some $X \subseteq R$ and $R \cap Y = \emptyset$.

Definition: A minimal (X, Y)-cut $\vec{\delta}(R)$ is **important** if there is no (X, Y)-cut $\vec{\delta}(R')$ with $R \subset R'$ and $|\vec{\delta}(R')| \leq |\vec{\delta}(R)|$.

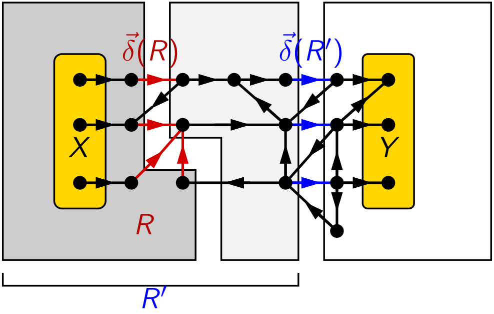


Directed graphs

Definition: $\vec{\delta}(R)$ is the set of edges leaving R.

Observation: Every inclusionwise-minimal directed (X, Y)-cut S can be expressed as $S = \vec{\delta}(R)$ for some $X \subseteq R$ and $R \cap Y = \emptyset$.

Definition: A minimal (X, Y)-cut $\vec{\delta}(R)$ is **important** if there is no (X, Y)-cut $\vec{\delta}(R')$ with $R \subset R'$ and $|\vec{\delta}(R')| \leq |\vec{\delta}(R)|$.



Directed graphs

Definition: $\vec{\delta}(R)$ is the set of edges leaving R.

Observation: Every inclusionwise-minimal directed (X,Y)-cut S can be expressed as $S=\vec{\delta}(R)$ for some $X \subseteq R$ and $R \cap Y = \emptyset$.

Definition: A minimal (X,Y)-cut $\vec{\delta}(R)$ is **important** if there is no (X,Y)-cut $\vec{\delta}(R')$ with $R \subset R'$ and $|\vec{\delta}(R')| \le |\vec{\delta}(R)|$.

The proof for the undirected case goes through for the directed case:

Theorem

There are at most 4^k important directed (X,Y)-cuts of size at most k.

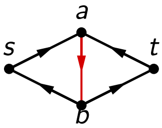
Directed Multiway Cut

The undirected approach does not work: the pushing lemma is not true.

Pushing Lemma (for undirected graphs)

Let $t \in T$. The Multiway Cut problem has a solution S that contains an important $(t, T \setminus t)$-cut.

Directed counterexample:



Unique solution with $k = 1$ edges, but it is not an important cut (boundary of $\{s, a\}$, but the boundary of $\{s, a, b\}$ has same size).

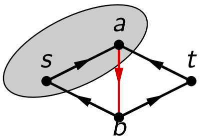
Directed Multiway Cut

The undirected approach does not work: the pushing lemma is not true.

Pushing Lemma (for undirected graphs)

Let $t \in T$. The Multiway Cut problem has a solution S that contains an important $(t, T \setminus t)$-cut.

Directed counterexample:



Unique solution with $k = 1$ edges, but it is not an important cut (boundary of $\{s, a\}$, but the boundary of $\{s, a, b\}$ has same size).

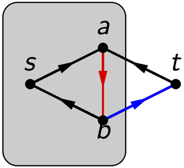
Directed Multiway Cut

The undirected approach does not work: the pushing lemma is not true.

Pushing Lemma (for undirected graphs)

Let $t \in T$. The Multiway Cut problem has a solution S that contains an important $(t, T \setminus t)$-cut.

Directed counterexample:



Unique solution with $k = 1$ edges, but it is not an important cut (boundary of $\{s, a\}$, but the boundary of $\{s, a, b\}$ has same size).

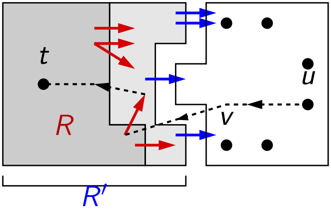
Directed Multiway Cut

The undirected approach does not work: the pushing lemma is not true.

Pushing Lemma (for undirected graphs)

Let $t \in T$. The Multiway Cut problem has a solution S that contains an important $(t, T \setminus t)$-cut.

Problem in the undirected proof:



Replacing R by R' cannot create a $t \to u$ path, but can create a $u \to t$ path.

Directed Multiway Cut

The undirected approach does not work: the pushing lemma is not true.

Pushing Lemma (for undirected graphs)

Let $t \in T$. The Multiway Cut problem has a solution S that contains an important $(t, T \setminus t)$-cut.

Using additional techniques, one can show:

Theorem [Chitnis, Hajiaghayi, M. 2011]

Directed Multiway Cut is FPT parameterized by the size k of the solution.

Directed Multicut

Directed Multicut

Input: Graph G, pairs (s_1, t_1), ..., (s_ℓ, t_ℓ), integer k

Find: A set S of edges such that $G \setminus S$ has no $s_i \to t_i$ path for any i.

Theorem [M. and Razgon 2011]

Directed Multicut is W[1]-hard parameterized by k.

Directed Multicut

Directed Multicut

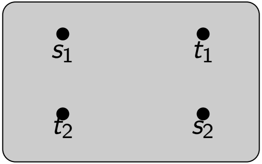
Input: Graph G, pairs (s_1, t_1), ..., (s_ℓ, t_ℓ), integer k

Find: A set S of edges such that $G \setminus S$ has no $s_i \to t_i$ path for any i.

Theorem [M. and Razgon 2011]

Directed Multicut is W[1]-hard parameterized by k.

But the case $\ell = 2$ can be reduced to Directed Multiway Cut:



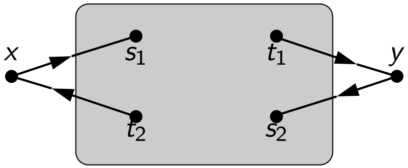
Directed Multicut

Directed Multicut	
Input:	Graph G, pairs $(s_1, t_1), \ldots, (s_\ell, t_\ell)$, integer k
Find:	A set S of edges such that $G \setminus S$ has no $s_i \to t_i$ path for any i.

Theorem [M. and Razgon 2011]

Directed Multicut is W[1]-hard parameterized by k.

But the case $\ell = 2$ can be reduced to Directed Multiway Cut:



Directed Multicut

Directed Multicut

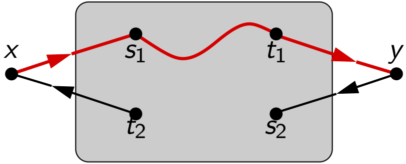
Input: Graph G, pairs (s_1, t_1), $\ldots$, (s_ℓ, t_ℓ), integer k

Find: A set S of edges such that $G \setminus S$ has no $s_i \to t_i$ path for any i.

Theorem [M. and Razgon 2011]

Directed Multicut is W[1]-hard parameterized by k.

But the case $\ell = 2$ can be reduced to Directed Multiway Cut:



Directed Multicut

Directed Multicut

Input: Graph G, pairs $(s_1, t_1), \ldots, (s_\ell, t_\ell)$, integer k

Find: A set S of edges such that $G \setminus S$ has no $s_i \to t_i$ path for any i.

Theorem [M. and Razgon 2011]

Directed Multicut is W[1]-hard parameterized by k.

Corollary

Directed Multicut with $\ell = 2$ is FPT parameterized by the size k of the solution.

?

Open: Is Directed Multicut with $\ell = 3$ FPT?

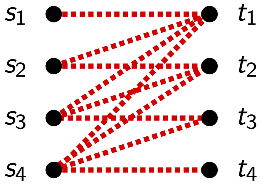
Open: Is there an $f(k, \ell) \cdot n^{O(1)}$ algorithm for Directed Multicut?

Skew Multicut

Skew Multicut

Input: Graph G, pairs $(s_1, t_1), \ldots, (s_\ell, t_\ell)$, integer k

Find: A set S of k directed edges such that $G \setminus S$ contains no $s_i \to t_j$ path for any $i \geq j$.

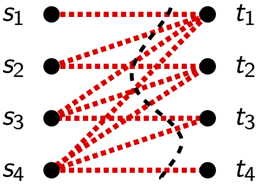


Skew Multicut

Skew Multicut

Input: Graph G, pairs $(s_1, t_1), \ldots, (s_\ell, t_\ell)$, integer k

Find: A set S of k directed edges such that $G \setminus S$ contains no $s_i \to t_j$ path for any $i \geq j$.



Pushing Lemma

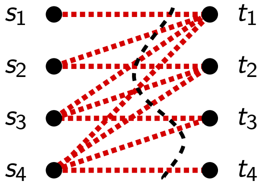
Skew Multcut problem has a solution S that contains an important $(s_\ell, \{t_1, \ldots, t_\ell\})$-cut.

Skew Multicut

Skew Multicut

Input: Graph G, pairs $(s_1, t_1), \ldots, (s_\ell, t_\ell)$, integer k

Find: A set S of k directed edges such that $G \setminus S$ contains no $s_i \to t_j$ path for any $i \geq j$.



Pushing Lemma

Skew Multcut problem has a solution S that contains an important $(s_\ell, \{t_1, \ldots, t_\ell\})$-cut.

Theorem [Chen, Liu, Lu, O'Sullivan, Razgon 2008]

Skew Multicut can be solved in time $4^k \cdot n^{O(1)}$.

Directed Feedback Vertex Set

Directed Feedback Vertex/Edge Set	
Input:	Directed graph G, integer k
Find:	A set S of k vertices/edges such that $G \setminus S$ is acyclic.

Note: Edge and vertex versions are equivalent, we will consider the edge version here.

Theorem [Chen, Liu, Lu, O'Sullivan, Razgon 2008]

Directed Feedback Edge Set is FPT parameterized by the size k of the solution.

Solution uses the technique of **iterative compression** introduced by [Reed, Smith, Vetta 2004].

The compression problem

Directed Feedback Edge Set Compression	
Input:	Directed graph G, integer k, a set W of $k+1$ edges such that $G \setminus W$ is acyclic
Find:	A set S of k edges such that $G \setminus S$ is acyclic.

Easier than the original problem, as the extra input W gives us useful structural information about G.

Lemma

The compression problem is FPT parameterized by k.

The compression problem

Directed Feedback Edge Set Compression	
Input:	Directed graph G, integer k, a set W of $k+1$ **vertices** such that $G \setminus W$ is acyclic
Find:	A set S of k edges such that $G \setminus S$ is acyclic.

Easier than the original problem, as the extra input W gives us useful structural information about G.

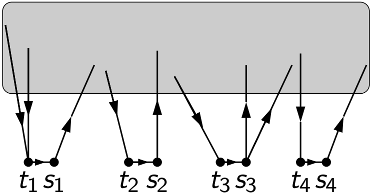
Lemma

The compression problem is FPT parameterized by k.

A useful trick for edge deletion problems: we define the compression problem in a way that a solution of $k+1$ **vertices** are given and we have to find a solution of k **edges**.

The compression problem

Proof: Let $W = \{w_1, \ldots, w_{k+1}\}$
Let us split each w_i into an edge $\overrightarrow{t_i s_i}$.

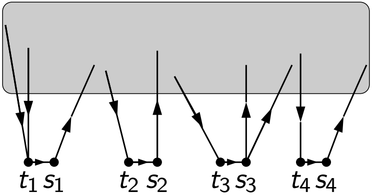


- By guessing the order of $\{w_1, \ldots, w_{k+1}\}$ in the acyclic ordering of $G \setminus S$, we can assume that $w_1 < w_2 < \cdots < w_{k+1}$ in $G \setminus S$ [$(k+1)!$ possibilities].

The compression problem

Proof: Let $W = \{w_1, \ldots, w_{k+1}\}$

Let us split each w_i into an edge $\overrightarrow{t_i s_i}$.



Claim:

$G \setminus S$ is acyclic and has an ordering with $w_1 < w_2 < \cdots < w_{k+1}$

$\Downarrow$

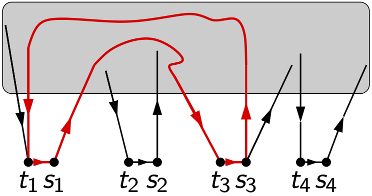
S covers every $s_i \to t_j$ path for every $i \geq j$

$\Downarrow$

$G \setminus S$ is acyclic

The compression problem

Proof: Let $W = \{w_1, \ldots, w_{k+1}\}$
Let us split each w_i into an edge $\overrightarrow{t_i s_i}$.



Claim:

$G \setminus S$ is acyclic and has an ordering with $w_1 < w_2 < \cdots < w_{k+1}$

$\Downarrow$

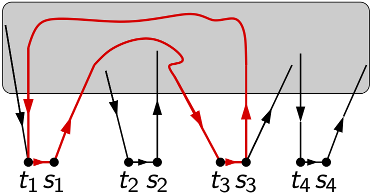
S covers every $s_i \to t_j$ path for every $i \geq j$

$\Downarrow$

$G \setminus S$ is acyclic

The compression problem

Proof: Let $W = \{w_1, \ldots, w_{k+1}\}$
Let us split each w_i into an edge $\overrightarrow{t_i s_i}$.



Claim:

$$G \setminus S \text{ is acyclic and has an ordering with } w_1 < w_2 < \cdots < w_{k+1}$$
$$\Downarrow$$
$$S \text{ covers every } s_i \to t_j \text{ path for every } i \geq j$$
$$\Downarrow$$
$$G \setminus S \text{ is acyclic}$$

$\Rightarrow$ We can solve the compression problem by $(k+1)!$ applications of Skew Multicut.

Iterative compression

We have given a $f(k)n^{O(1)}$ algorithm for the following problem:

Directed Feedback Edge Set Compression	
Input:	Directed graph G, integer k, a set W of $k+1$ vertices such that $G \setminus W$ is acyclic
Find:	A set S of k edges such that $G \setminus S$ is acyclic.

Nice, but how do we get a solution W of size $k+1$?

Iterative compression

We have given a $f(k)n^{O(1)}$ algorithm for the following problem:

DIRECTED FEEDBACK EDGE SET COMPRESSION	
Input:	Directed graph G, integer k, a set W of $k+1$ vertices such that $G \setminus W$ is acyclic
Find:	A set S of k edges such that $G \setminus S$ is acyclic.

Nice, but how do we get a solution W of size $k+1$?

We get it for free!

Powerful technique: **iterative compression** (introduced by [Reed, Smith, Vetta 2004] for BIPARTITE DELETION).

Iterative compression

Let v_1, ..., v_n be the edges of G and let G_i be the subgraph induced by $\{v_1, \ldots, v_i\}$.

For every $i = 1, \ldots, n$, we find a set S_i of at most k edges such that $G_i \setminus S_i$ is acyclic.

Iterative compression

Let $v_1, \ldots, v_n$ be the edges of G and let G_i be the subgraph induced by $\{v_1, \ldots, v_i\}$.

For every $i = 1, \ldots, n$, we find a set S_i of at most k edges such that $G_i \setminus S_i$ is acyclic.

- For $i = 1$, we have the trivial solution $S_i = \emptyset$.
- Suppose we have a solution S_i for G_i. Let W_i contain the head of each edge in S_i. Then $W_i \cup \{v_{i+1}\}$ is a set of at most $k + 1$ vertices whose removal makes G_{i+1} acyclic.
- Use the compression algorithm for G_{i+1} with the set $W_i \cup \{v_{i+1}\}$.
    - If there is no solution of size k for G_{i+1}, then we can stop.
    - Otherwise the compression algorithm gives a solution S_{i+1} of size k for G_{i+1}.

We call the compression algorithm n times, everything else is polynomial.

$\Rightarrow$ Directed Feedback Edge Set is FPT.

Outline

So far we have seen:

- Definition of important cuts.
- Combinatorial bound on the number of important cuts.
- Pushing argument: we can assume that the solution contains an important cut. Solves Multiway Cut, Skew Multiway Cut.
- Iterative compression reduces Directed Feedback Vertex Set to Skew Multiway Cut.

Next:

- Randomized sampling of important separators.

Randomized sampling of important cuts

A new technique used by several results:

- Multicut [M. and Razgon STOC 2011]
- Clustering problems [Lokshtanov and M. ICALP 2011]
- Directed Multiway Cut [Chitnis, Hajiaghayi, M. SODA 2012]
- Directed Multicut in DAGs [Kratsch, Pilipczuk, Pilipczuk, Wahlström ICALP 2012]
- Directed Subset Feedback Vertex Set [Chitnis, Cygan, Hajiaghayi, M. ICALP 2012]
- Parity Multiway Cut [Lokshtanov, Ramanujan ICALP 2012]
- List homomorphism removal problems [Chitnis, Egri, and M. ESA 2013]
- . . . more work in progress.

Clustering

We want to partition objects into clusters subject to certain requirements (typically: related objects are clustered together, bounds on the number or size of the clusters etc.)

(p, q)-CLUSTERING

Input: A graph G, integers p, q.

Find: A partition $(V_1, \dots, V_m)$ of $V(G)$ such that for every i

- $|V_i| \leq p$ and
- $\delta(V_i) \leq q$.

$\delta(V_i)$: number of edges leaving V_i.

Theorem [Lokshtanov and M. 2011]

(p, q)-CLUSTERING can be solved in time $2^{O(q)} \cdot n^{O(1)}$.

A sufficient and necessary condition

Good cluster: size at most p and at most q edges leaving it.

Necessary condition:

Every vertex is contained in a good cluster.

A sufficient and necessary condition

Good cluster: size at most p and at most q edges leaving it.

Necessary condition:

Every vertex is contained in a good cluster.

But surprisingly, this is also a **sufficient condition!**

Lemma

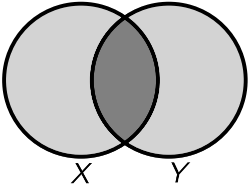
Graph G has a (p, q)-clustering if and only if every vertex is in a good cluster.

A sufficient and necessary condition

Lemma

Graph G has a (p, q)-clustering if and only if every vertex is in a good cluster.

Proof: Find a collection of good clusters covering every vertex and having minimum total size. Suppose two clusters intersect.

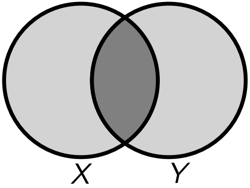


A sufficient and necessary condition

Lemma

Graph G has a (p,q)-clustering if and only if every vertex is in a good cluster.

Proof: Find a collection of good clusters covering every vertex and having minimum total size. Suppose two clusters intersect.



$$\delta(X) + \delta(Y) \geq \delta(X \setminus Y) + \delta(Y \setminus X)$$

(posimodularity)

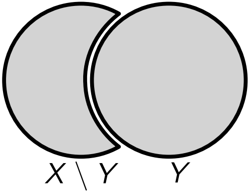
$\Rightarrow$ either $\delta(X) \geq \delta(X \setminus Y)$ or $\delta(Y) \geq \delta(Y \setminus X)$ holds.

A sufficient and necessary condition

Lemma

Graph G has a (p, q)-clustering if and only if every vertex is in a good cluster.

Proof: Find a collection of good clusters covering every vertex and having minimum total size. Suppose two clusters intersect.



$$\delta(X) + \delta(Y) \geq \delta(X \setminus Y) + \delta(Y \setminus X)$$

(posimodularity)

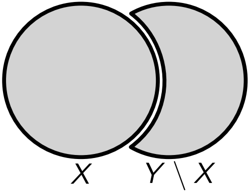
If $\delta(X) \geq \delta(X \setminus Y)$, replace X with $X \setminus Y$, strictly decreasing the total size of the clusters.

A sufficient and necessary condition

Lemma

Graph G has a (p, q)-clustering if and only if every vertex is in a good cluster.

Proof: Find a collection of good clusters covering every vertex and having minimum total size. Suppose two clusters intersect.



$$\delta(X) + \delta(Y) \geq \delta(X \setminus Y) + \delta(Y \setminus X)$$

(posimodularity)

If $\delta(Y) \geq \delta(Y \setminus X)$, replace Y with $Y \setminus X$, strictly decreasing the total size of the clusters. QED ■

Finding a good cluster

We have seen:

Lemma

Graph G has a (p, q)-clustering if and only if every vertex is in a good cluster.

All we have to do is to check if a given vertex v is in a good cluster. Trivial to do in time $n^{O(q)}$.

Finding a good cluster

We have seen:

Lemma

Graph G has a (p, q)-clustering if and only if every vertex is in a good cluster.

All we have to do is to check if a given vertex v is in a good cluster. Trivial to do in time $n^{O(q)}$.

We prove next:

Lemma

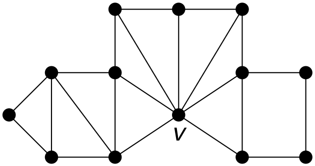
We can check in time $2^{O(q)} \cdot n^{O(1)}$ if v is in a good cluster.

Important sets

Definition

Fix a distinguished vertex v in a graph G. A set $X \subseteq V(G)$ is an **important set** if

- $v \notin X$,
- there is no set $X \subset X'$ with $v \notin X$ and $\delta(X') \le \delta(X)$.

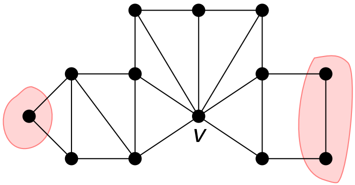


Important sets

Definition

Fix a distinguished vertex v in a graph G. A set $X \subseteq V(G)$ is an **important set** if

- $v \notin X$,
- there is no set $X \subset X'$ with $v \notin X$ and $\delta(X') \leq \delta(X)$.

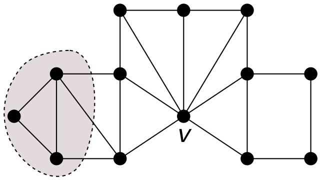


Important sets

Definition

Fix a distinguished vertex v in a graph G. A set $X \subseteq V(G)$ is an **important set** if

- $v \notin X$,
- there is no set $X \subset X'$ with $v \notin X$ and $\delta(X') \leq \delta(X)$.

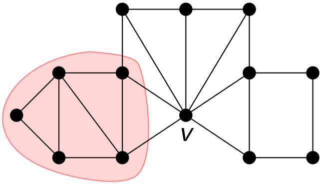


Important sets

Definition

Fix a distinguished vertex v in a graph G. A set $X \subseteq V(G)$ is an **important set** if

- $v \notin X$,
- there is no set $X \subset X'$ with $v \notin X$ and $\delta(X') \leq \delta(X)$.

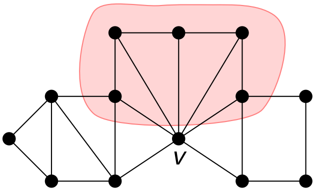


Important sets

Definition

Fix a distinguished vertex v in a graph G. A set $X \subseteq V(G)$ is an **important set** if

- $v \notin X$,
- there is no set $X \subset X'$ with $v \notin X$ and $\delta(X') \leq \delta(X)$.

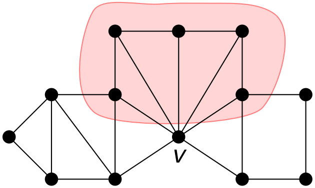


Important sets

Definition

Fix a distinguished vertex v in a graph G. A set $X \subseteq V(G)$ is an **important set** if

- $v \notin X$,
- there is no set $X \subset X'$ with $v \notin X$ and $\delta(X') \leq \delta(X)$.



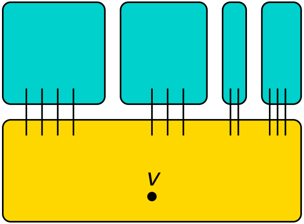
Observation: X is an important set if and only if $\delta(X)$ is an important (x, v)-cut for every $x \in X$.

Consequence: Every vertex is contained in at most 4^k important sets.

Pushing argument

Lemma

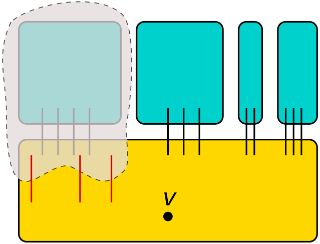
If C is a good cluster of minimum size containing v, then every component of $G \setminus C$ is an important set.



Pushing argument

Lemma

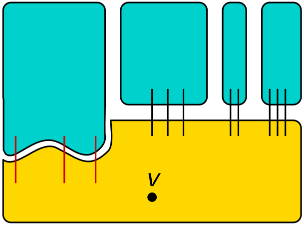
If C is a good cluster of minimum size containing v, then every component of $G \setminus C$ is an important set.



Pushing argument

Lemma

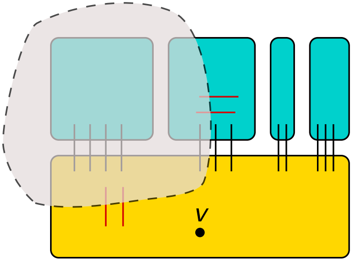
If C is a good cluster of minimum size containing v, then every component of $G \setminus C$ is an important set.



Pushing argument

Lemma

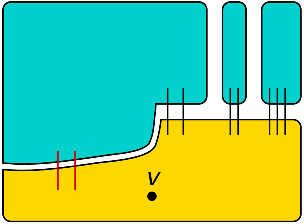
If C is a good cluster of minimum size containing v, then every component of $G \setminus C$ is an important set.



Pushing argument

Lemma

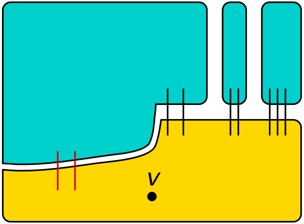
If C is a good cluster of minimum size containing v, then every component of $G \setminus C$ is an important set.



Pushing argument

Lemma

If C is a good cluster of minimum size containing v, then every component of $G \setminus C$ is an important set.



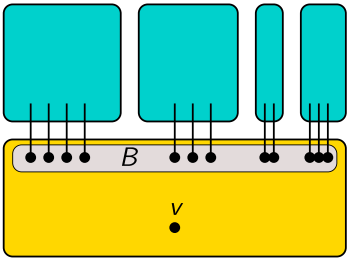
Thus C can be obtained by removing at most q important sets from $V(G)$ (but there are $n^{O(q)}$ possibilities, we cannot try all of them).

Random sampling

- Let $\mathcal{X}$ be the set of all important sets of boundary size at most q in G.
- Let $\mathcal{X}' \subseteq \mathcal{X}$ contain each set with probability $\frac{1}{2}$ independently.
- Let $Z = \bigcup_{X \in \mathcal{X}'} X$.
- Let B be the set of vertices in C with neighbors outside C.

Lemma

Let C be a good cluster of minimum size containing v. With probability $2^{-2^{O(q)}}$, Z covers $G \setminus C$ and is disjoint from B.

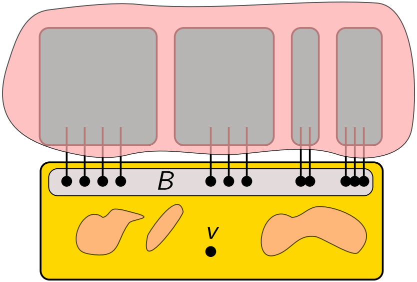


Random sampling

- Let $\mathcal{X}$ be the set of all important sets of boundary size at most q in G.
- Let $\mathcal{X}' \subseteq \mathcal{X}$ contain each set with probability $\frac{1}{2}$ independently.
- Let $Z = \bigcup_{X \in \mathcal{X}'} X$.
- Let B be the set of vertices in C with neighbors outside C.

Lemma

Let C be a good cluster of minimum size containing v. With probability $2^{-2^{O(q)}}$, Z covers $G \setminus C$ and is disjoint from B.



Random sampling

Lemma

Let C be a good cluster of minimum size containing v. With probability $2^{-2^{O(q)}}$, Z covers $G \setminus C$ and is disjoint from B.

Two events:

(E1) Z covers $G \setminus C$.
Each of the at most q components is an important set
$\Rightarrow$ all of them are selected by probability at least 2^{-q}.

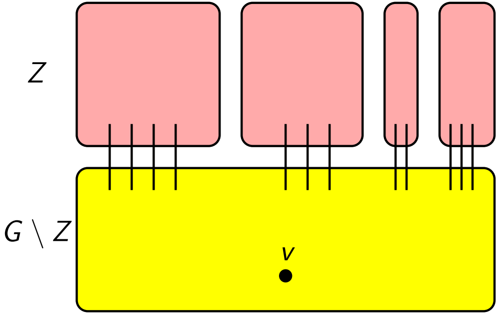
(E2) Z is disjoint from B.
Each vertex of B is in at most 4^q members of $\mathcal{X}$
$\Rightarrow$ all of them are selected by probability at least 2^{-q4^q}.

The two events are independent (involve different sets of $\mathcal{X}$), thus the claimed probability follows.

Finding good clusters

Let C be a good cluster of minimum size containing v and assume

- $G \setminus C$ is covered by Z, and
- Z is disjoint from B (hence no edge going out of C is contained in Z).

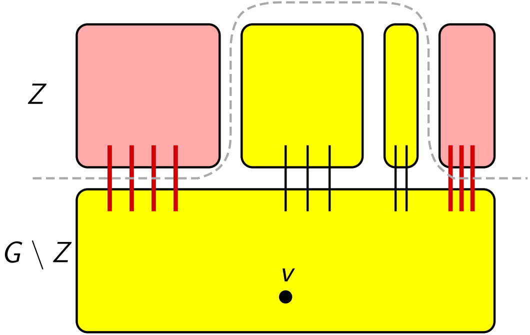


Where is the good cluster C in the figure?

Finding good clusters

Let C be a good cluster of minimum size containing v and assume

- $G \setminus C$ is covered by Z, and
- Z is disjoint from B (hence no edge going out of C is contained in Z).



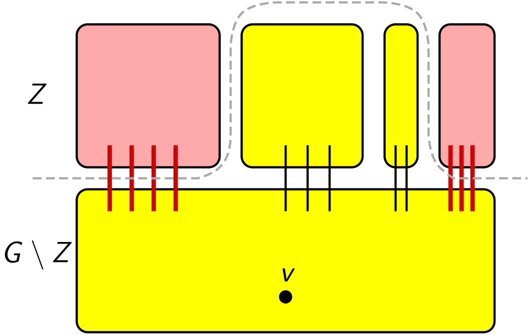
Where is the good cluster C in the figure?

Observe: Components of Z are either fully in the cluster or fully outside the cluster. What is this problem?

Finding good clusters

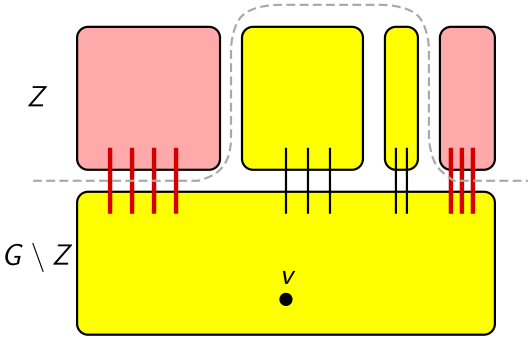
Let C be a good cluster of minimum size containing v and assume

- $G \setminus C$ is covered by Z, and
- Z is disjoint from B (hence no edge going out of C is contained in Z).



KNAPSACK!

Finding good clusters by KNAPSACK

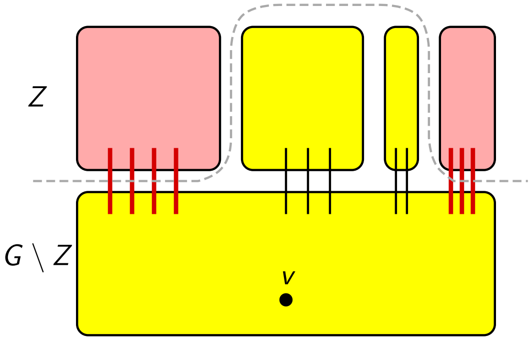


We interpret the componenents $V_1, \ldots, V_t$ of $G[Z]$ as items:

- V_i has value $\delta(V_i)$ and
- V_i has weight $|V_i|$.

The goal is to select items with total value at least $\delta(Z) - q$ and total weight at most $p - |V(G) \setminus Z|$.

Finding good clusters by KNAPSACK



Standard DP solves it in polynomial time: let $T[i,j]$ be the maximum value of a subset of the first i items having total weight at most j.

Recurrence:

$$T[i,j] = \max\{T[i-1,j], T[i-1,j-|V_i|] + \delta(V_i)\}$$

Summary of algorithm

(p, q)-CLUSTERING

Input: A graph G, integers p, q.

Find: A partition $(V_1, \ldots, V_m)$ of $V(G)$ such that for every i

- $|V_i| \leq p$ and
- $\delta(V_i) \leq q$.

- It is sufficient to check for each vertex v if it is in a good cluster.
- Enumerate all the important sets.
- Let Z be the union of random important sets.
- The solution is obtained by extending $G \setminus Z$ with some of the components of $G[Z]$.
- Knapsack.

(p, q)-CLUSTERING

- With a slightly different probability distribution, one can reduce the error probability to $2^{-O(q)}$.
- Derandomization is possible using standard techniques, but nontrivial to obtain $2^{O(q)}$ running time.
- Other variants: maximum degree in the cluster is at most p, etc.

Summary

- A simple (but essentially tight) bound on the number of important cuts.
- Algorithmic results: FPT algorithms for
  - Multiway Cut in undirected graphs,
  - Skew Multicut in directed graphs,
  - Directed Feedback Vertex/Edge Set, and
  - (p, q)-Clustering.