Integer Programming with a Totally Unimodular Coefficient Matrix

While linear programming is polynomially solvable, integer programming will probably never be. On the other hand, many applications can only be handled with integer programming. So what is the way out? One direction is to try and develop as efficient algorithms as possible with the hope of being able to solve large enough problem instances to handle real-life applications; the previously mentioned Branch and Bound method is just a small step towards this direction. However, there is another promising direction: to identify a special case of integer programming that is narrow enough to be efficiently solvable but still wide enough to contain interesting applications. In what follows, we take this latter direction.

Definition. *A matrix is* totally unimodular *(or* TU *for short) if each of its square submatrices has determinant 0 or 1 or* -1*.*

Recall that a *square submatrix* is obtained by selecting certain (not necessarily adjacent) rows and an equal number of columns from a bigger matrix. Obviously, all elements of a totally unimodular matrix have to be 0 or 1 or -1 as they are 1×1 submatrices. However, this is not sufficient as shown by the matrix A below: it is not totally unimodular since its determinant is 2. B is totally unimodular, which can be verified by checking all 3 of its 2×2 submatrices (and its 6 elements). C is again not totally unimodular: the 2×2 submatrix obtained by selecting its four corners has determinant -2 (while all its remaining square submatrices would fit the definition of a TU matrix).

$$A = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \qquad B = \begin{pmatrix} 1 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix} \qquad C = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix}$$

The importance of the notion of a TU matrix resides in the following theorem.

Theorem. *Assume that the linear program* $\max\{cx : Ax \leq b\}$ *is given such that* $Ax \leq b$ *is solvable and* $\{cx : Ax \leq b\}$ *is bounded from above. Assume further that* A *is totally unimodular and* b *is an integer vector. Then the maximum of* $\max\{cx : Ax \leq b\}$ *is attained at an integer vector.*

The above theorem implies that if A is TU and b is integer then the difference between the complexity of integer programming and linear programming vanishes: the integer program $\max\{cx : Ax \leq b, x \text{ integer}\}$ can simply be solved by solving its LP relaxation. The proof of the above theorem will be omitted here, we will rather concentrate on its applications.

In various applications of the above theorem it frequently happens that the problem to be solved is slightly modified which results in a small change in the coefficient matrix. For example, if a modification of the problem has the effect that an inequality turns into an equation, that adds an extra row to the coefficient matrix that is identical to the negated version of an already existing row. Another example is if an upper bound is imposed on one of the variables: that adds a row to the coefficient matrix that is comprised of all zero entries except for a single 1. The following simple, but very useful lemma claims that these small modifications preserve the TU property of a matrix.

Lemma. *A matrix remains to be totally unimodular if*

1. *the negative of one of its rows or columns is taken;*
2. *a new row or new column is added whose entries are all zero except for a single 1;*
3. *a copy of one of its rows or columns is added to the matrix as a new row or column, respectively;*
4. *its transpose is taken.*

Proof. Obviously, in all four cases we only need to consider those submatrices that are affected by the respective operation.

1. If a row or column is negated, the determinant of an affected submatrix is also negated, so it stays in $\{1, -1, 0\}$.

2. Expanding along the newly added row or column of an affected $k \times k$ submatrix M immediately gives that $\det M$ is either 0 (if the part of the new row or column belonging to M only contains zeros) or $\det M = (\pm 1) \cdot \det M'$, where M' is a $(k-1) \times (k-1)$ submatrix. Since $\det M' \in \{1, -1, 0\}$, $\det M \in \{1, -1, 0\}$ is also obvious.

3. If a submatrix contains (a part of) both the original and the new version of the copied row or column then it has two identical rows or columns, so its determinant is 0. If it only contains the new version then it is obtained from a square submatrix of the original matrix by permuting its rows or columns. Since any such permutation can be obtained as a series of exchanging rows or columns and each exchange negates the determinant, the final determinant is equal to (± 1) times the original determinant; hence it is in $\{1, -1, 0\}$.

4. Finally, transposing a matrix also transposes all of its submatrices. Since transposing does not modify the determinant, neither does it affect total unimodularity. □

Integer Flow Problems

In order to apply the above theorem on integer programs with totally unimodular coefficient matrices, we need to find totally unimodular matrices that arise in applications of integer programming. One class of these is given by the following definition.

Definition. *Assume that G is a loopless directed graph with vertex set $V(G) = \{v_1, v_2, \ldots, v_n\}$ and arc set $E(G) = \{e_1, e_2, \ldots, e_m\}$. The* incidence matrix *$B(G)$ of G is an $n \times m$ matrix such that for every $1 \le i \le n$ and $1 \le j \le m$*

$$[B(G)]_{i,j} = \begin{cases} 1, & \text{if } e_j \text{ leaves } v_i; \\ -1, & \text{if } e_j \text{ enters } v_i; \\ 0, & \text{if } e_j \text{ is not incident to } v_i. \end{cases}$$

Theorem. *The incidence matrix of every directed graph is totally unimodular.*

Proof. Choose a $k \times k$ submatrix M from the incidence matrix $B(G)$ of a directed graph G. We prove by induction on k that $\det M \in \{1, -1, 0\}$.

Obviously, this is true for the $k = 1$ case: elements of $B(G)$ all belong to $\{1, -1, 0\}$ by definition. So assume $k \ge 2$ and that the statement is proved for all $(k-1) \times (k-1)$ submatrices. We distinguish between two cases.

Case 1: if M has a column that contains at most one nonzero entry. If this column is an all-zero column then $\det M = 0$. If not, then expand the determinant of M along that column. It follows that $\det M = (\pm 1) \cdot \det M'$, where M' is a $(k-1) \times (k-1)$ submatrix of $B(G)$. Hence we are done by induction.

Case 2: if each column of M contains exactly two nonzero elements (a 1 and a -1). In this case, adding all other rows of M to the first one will result in an all zero row; since such steps do not modify the value of a determinant and the determinant of the resulting matrix is 0 (due to the all zero row), the same holds for M. □

The integer maximum flow problem

Recall that, given a directed graph G with source and target nodes s and t, respectively and a capacity function $c : E(G) \to \mathbb{R}^+$, we previously formulated the maximum flow problem as a linear program in the following form:

$$\begin{array}{lll} & \max : \sum\{x(e) : e \text{ leaves } s\} - \sum\{x(e) : e \text{ enters } s\} & \\ & \text{subject to} & \\ (1) & \forall v \in V(G) \setminus \{s,t\} : \quad \sum\{x(e) : e \text{ leaves } v\} - \sum\{x(e) : e \text{ enters } v\} & = \quad 0 \\ (2) & \forall e \in E(G) : \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad x(e) & \leq \quad c(e) \\ (3) & \forall e \in E(G) : \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad x(e) & \geq \quad 0 \end{array}$$

Let $V(G) = \{v_1, v_2, \ldots, v_{n-2}, v_{n-1} = s, v_n = t\}$ and $E(G) = \{e_1, e_2, \ldots, e_m\}$. To rewrite the above linear program in matrix form, collect all variables in the column vector x such that $x_j = x(e_j)$ holds for all $1 \leq j \leq m$. Then the system of linear equations corresponding to (1) can be written in the form $B' \cdot x = 0$, where B' is obtained from the incidence matrix $B(G)$ of G by deleting its last two rows (the ones corresponding to s and t).

Then the above linear program can be written in the following form:

$$\max\{\overline{s} \cdot x : B' \cdot x = 0, x \leq c, x \geq 0\},$$

where $\overline{s}$ denotes the second to last row of the incidence matrix $B(G)$ of G, the one that corresponds to s. In other words, the above linear program is of the form $\max\{\overline{s} \cdot x : Ax \leq b\}$, where

$$A = \begin{pmatrix} B' \\ \hline -B' \\ \hline I \\ \hline -I \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} 0 \\ \hline 0 \\ \hline c \\ \hline 0 \end{pmatrix}.$$

Obviously, the matrix B' is totally unimodular (since every square submatrix of B' is also a submatrix of the incidence matrix $B(G)$ which was proved to be TU above). Furthermore, since the complete coefficient matrix A of the above linear program can be obtained from B' by the operations listed in the previously proved simple lemma, we can conclude that A is TU.

Consequently, the above mentioned fundamental theorem on integer programs with TU coefficient matrices implies that if the capacities of all arcs in a network flow problem are integers (which makes the complete right hand side vector in the above LP formulation integer) then a maximum flow with integer flow values on all arcs exists. Furthermore, this also implies that the maximum flow problem can be solved efficiently simply by neglecting integrality constraints and solving the obtained linear program.

Actually, neither of these statements is new: the augmenting path algorithm (covered previously in the BSz2 course) provides a polynomial time algorithm for solving the maximum flow problem (if always one of the shortest augmenting paths is chosen) and this algorithm also terminates with an integer valued maximum flow if all capacities are integer. However, besides providing the theoretical background of this phenomenon, the above argument is still useful: with minor changes, it can be applied on the much more complicated minimum cost flow problem.

The integer minimum cost flow problem

Given a directed graph G, source and sink $s, t \in V(G)$, arc capacities $c : E \to \mathbb{R}^+$, arc costs $k : E \to \mathbb{R}^+$ and required flow value M, the linear programming formulation of the minimum cost flow problem was the following:

$\min : \sum_{e \in E(G)} k(e)x(e)$

subject to

(1)	$\forall v \in V(G) \setminus \{s,t\}:$	$\sum\{x(e) : e \text{ leaves } v\} - \sum\{x(e) : e \text{ enters } v\}$	$=$	0
(2)		$\sum\{x(e) : e \text{ leaves } s\} - \sum\{x(e) : e \text{ enters } s\}$	$\geq$	M
(3)	$\forall e \in E(G):$	$x(e)$	$\leq$	$c(e)$
(4)	$\forall e \in E(G):$	$x(e)$	$\geq$	0

Obviously, the *integer minimum cost flow problem* is defined analogously to the minimum cost flow problem with the only difference being that all flow values $x(e)$ are required to be integer. In this case, both M and all arc capacities $c(e)$ can also be assumed to be integer (otherwise they can be rounded up or down, respectively).

Comparing this with the LP formulation of the maximum flow problem we see that the set of constraints is identical except for (2) above. Therefore the matrix form of the above LP formulation is also very similar to that of the maximum flow problem: it can be written in the form $\min\{\overline{k} \cdot x : A'x \leq b'\}$, where $\overline{k}$ is a row vector that consists of all the arc costs $k(e)$ (such that $\overline{k}_j = k(e_j)$ holds for every $1 \leq j \leq m$),

$$A' = \begin{pmatrix} B' \\ \hline -B' \\ \hline -\overline{s} \\ \hline I \\ \hline -I \end{pmatrix} \quad \text{and} \quad b' = \begin{pmatrix} 0 \\ \hline 0 \\ \hline -M \\ \hline c \\ \hline 0 \end{pmatrix}.$$

(Here $\overline{s}$, as above, denotes the row of the incidence matrix $B(G)$ that corresponds to s.)

We can again observe that the coefficient matrix A' is totally unimodular: it can be obtained from the incidence matrix $B(G)$ (that was proved to be TU) by deleting one of its rows (the one that corresponds to the target node t) and then applying the operations listed in the above proved simple lemma. Besides that, the right hand side vector is also integer since it contains c, M (which were assumed to be integer) and zeros.

Therefore, similarly to the case of the maximum flow problem, the fundamental theorem on IP problems with TU coefficient matrices implies two important facts: firstly, if M and all arc capacities $c(e)$ are integer then a minimum cost flow with integer flow values on all arcs exists. Secondly, the integer minimum cost flow problem can be solved efficiently by running an LP solver on its LP relaxation. (In particular, the integer minimum cost flow problem can be solved in polynomial time.)

We remark that, similarly to the case of the original (fractional) minimum cost flow problem, there exist more efficient (polynomial time) algorithms for the integer minimum cost flow problem that do not directly rely on linear programming.

The integer multicommodity flow problem

Unfortunately, the coefficient matrix of the (LP formulation of the) k-commodity flow problem is *not* totally unimodular if $k \geq 2$. This is due to the capacity constraints: since the sum of all flow values on an arc is constrained to be at most the capacity, this adds a row vector with k pieces of 1's (and a bunch of zeros) to the coefficient matrix. This ruins the TU property if $k \geq 2$. In actual fact, the integer version of the multicommodity flow problem is known to be NP-hard, so it is not expected to be solvable by any polynomial time algorithm.