Integer Programming, TU matrices

László Papp

BME

30th of May, 2023

Complexity of the decision version of Integer Programming

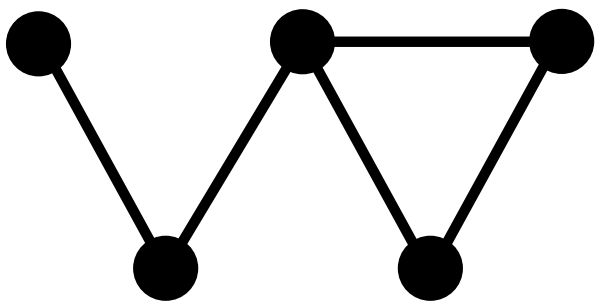
Decision version of INTEGER PROGRAMMING (IP for short):
Input: A matrix $A \in \mathbb{R}^{m \times n}$, vectors $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$ and a real number k.
Question: Is there an integer vector $x \in \mathbb{Z}^n$ which satisfies that $Ax \leq b$ and $c^T x \geq k$?

Theorem
INTEGER PROGRAMMING is NP-Complete.

Proof: INTEGER PROGRAMMING is in NP, because a witness is an integer vector x, which is a solution of $Ax \leq b,\ c^T x \geq k$. The verification algorithm checks this. It involves two matrix multiplication and comparing some elements. Matrix multiplication is a polynomial time algorithm, so the verification algorithm is also polynomial.
To show that INTEGER PROGRAMMING is NP-Hard we give a Karp reduction from CLIQUE.

CLIQUE$\prec$INTEGER PROGRAMMING

The input of the CLIQUE is a graph G and a number k. We create an integer program for it: We construct n variables, each of them corresponds to a vertex: $x_1, x_2 \ldots x_n$.



G, k

F

$$x_i \in \mathbb{Z}\ \forall i \in [1..n] \quad (1)$$

$$x_i \leq 1\ \forall i \in [1..n] \quad (2)$$

$$x_i \geq 0\ \forall i \in [1..n] \quad (3)$$

$$x_i + x_j \leq 1\ \forall \{i,j\} \notin E(G) \quad (4)$$

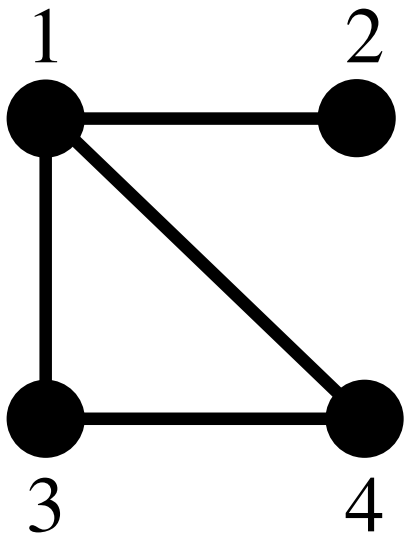
$$\max \sum_{i=1}^{n} x_i \geq k? \quad (5)$$

Note that (2) and (3) together with the fact that x is integer guarantees that x_i is either 1 (selected) or 0 (not selected).
(4) guarantees that we cannot select two not adjacent vertices, therefore the selected vertices form a clique.
The objective function is $\sum_{i=1}^{n} x_i$, which equals to the number of the selected vertices which form a clique.
This IP can be constructed from the graph in polynomial time.

IP/LP formalization

If we have a combinatorial optimization problem, it is a common technique, that we formalize it as an integer or linear program. Then we solve the obtained IP or LP by some algorithms (computer programs) designed for handling them.

An IP formalization of an instance of max clique search:

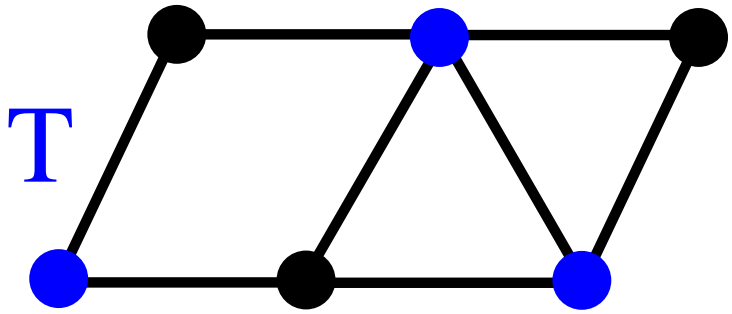


$$x_1, x_2, x_3, x_4 \in \mathbb{Z}$$
$$x_1, x_2, x_3, x_4 \leq 1$$
$$x_1, x_2, x_3, x_4 \geq 0$$
$$x_2 + x_3 \leq 1$$
$$x_2 + x_4 \leq 1$$
$$\max x_1 + x_2 + x_3 + x_4$$

The optimal solution is $x_1 = x_3 = x_4 = 1$, $x_2 = 0$ which correspond to the clique spanned by vertices $1, 2, 3$.

Recall: Minimum vertex cover

Remainder: In a graph G, $T \subseteq V(G)$ is a vertex cover if for every edge u, v, either $u \in T$ or $v \in T$ or both. The size of the minimum vertex cover is denoted by $\tau(G)$.



Optimization version of VERTEX COVER
Input: Graph G.
Task: Find a minimum vertex cover.

We can give an IP formalization for this. Remainder: Its decision version is NP-complete.

IP formalization of vertex cover problem:

We create a variable for each vertex, and each solution of the following linear system of inequalities is a characteristic vector of a vertex cover.

$$x_i \in \mathbb{Z}\ \forall i \in [1..|V(G)|] \quad (1)$$

$$x_i \leq 1\ \forall i \in [1..|V(G)|] \quad (2)$$

$$x_i \geq 0\ \forall i \in [1..|V(G)|] \quad (3)$$

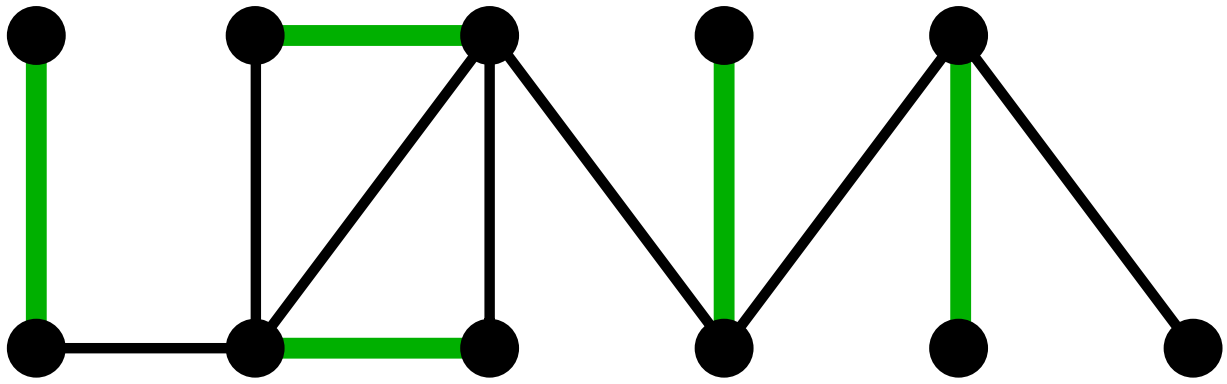
$$x_i + x_j \geq 1 \forall i, j \text{ where \{i,j\} is an edge} \quad (4)$$

$$\min \sum_{i=1}^{|V(G)|} x_i \quad (5)$$

(4) means that at least one endpoint of each edge must be selected. Therefore the selected vertices form a vertex cover. The size of the selected vertex cover is determined by the objective function.

Recall: Maximum matching problem

Input: An undirected simple graph G.
Task: Find a maximum matching in G.



We have seen that the augmenting path algorithm solve this problem in polinomial time if the graph is bipartite. There is also a polinomial time algorithm for not bipartite graphs which runs in polynomial time. On the other hand we can formalize this problem as an integer program.

IP formalization of maximum matching problem:

Variable x_i correspond to the ith edge of the graph.

$$x_i \in \mathbb{Z} \; \forall i \in [1..|E(G)|] \quad (1)$$

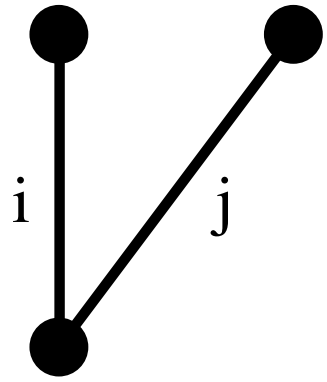
$$x_i \leq 1 \; \forall i \in [1..|E(G)|] \quad (2)$$

$$x_i \geq 0 \; \forall i \in [1..|E(G)|] \quad (3)$$

$$x_i + x_j \leq 1 \; \forall i, j \text{ where the ith and jth} \quad (4)$$

$$\text{edges share an endpoint} \quad (5)$$

$$\max \sum_{i=1}^{|E(G)|} x_i \quad (6)$$



If $x_i = 1$, then the ith edge is selected. (4) guarantees that two selected edges do not have a common endpoint, therefore the selected edges form a matching. The size of a matching equals to $\sum_{i=1}^{|E(G)|} x_i$.

IP formalization of maximum matching problem 2nd version:

Variable x_i correspond to the ith edge of the graph.

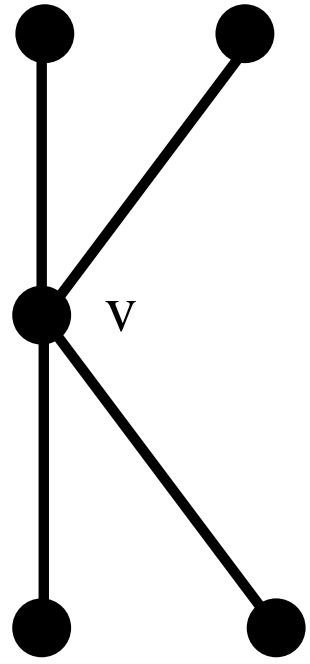
$$x_i \in \mathbb{Z} \; \forall i \in [1..|E(G)|] \tag{1}$$

$$x_i \leq 1 \; \forall i \in [1..|E(G)|] \tag{2}$$

$$x_i \geq 0 \; \forall i \in [1..|E(G)|] \tag{3}$$

$$\sum_{i|v \text{ is an endpoint of } i} x_i \leq 1 \; \forall v \in V(G) \tag{4}$$

$$\max \sum_{i=1}^{|E(G)|} x_i \tag{5}$$



(4) means that at most one edge can be selected from a set of edges which are incident to the same vertex. Thus the selected edges form a matching. This system contains less inequalities than the previous, therefore it is easier to solve.

How to solve Integer Programming problems?

In theory:

- No polynomial time algorithm is known for Integer Programming and it is unlikely that such an algorithm will be found.
- There are exponential-time algorithms which works well for medium sized inputs. For example: Dual simplex method with branch and bound. This is out of the scope of this class.

How to solve Integer Programming problems?

In theory:

- No polynomial time algorithm is known for Integer Programming and it is unlikely that such an algorithm will be found.
- There are exponential-time algorithms which works well for medium sized inputs. For example: Dual simplex method with branch and bound. This is out of the scope of this class.

In practice:

- IP solvers (computer programs developed for several years) works well for medium sized real world problems. For example: CPLEX, GUROBI, etc.
- Modelling tools for IP problems: LPSOLVE, AIMMS, etc

Note: These tools can be used to handle LP problems as well.

LP and IP formalization

LP and IP formalization is a well known technique to solve combinatorial optimization problems.

Why are they good:

- We do not need to invent a specific algorithm, which can consume a lot of time and maybe the obtained algorithm is not efficient.
- We do not have to write computer program code, it is enough to write a set of inequalities.
- The state of the art LP and IP solvers (i.e. CPLEX) work really well, they are optimized for recent architectures. They have been developed by many experts for decades.

Do not use always IP formalization!

Sometimes we have pretty good faster algorithms for a problem. For example for maximum matching searching. So first check the literature for such an algorithm and if you do not find anything then you shall start an IP approach!

For example we have shown an IP formalization of the maximum matching problem. Solving this IP may require so much time. Remember that we have learnt a polynomial time algorithm for finding maximum matching in a bipartite graph. The formalization works for all kind of graphs so it looks like a stronger result. However, there is a polynomial time algorithm for finding a maximum matching in a general graph.

Sometimes we can solve an IP problem in polynomial time

It is a general technique, that we forget the integrality conditions and solve the obtained linear programming problem. This LP is called as the fractional relaxation of the IP. If we are lucky and the obtained optimal solution is an integer vector, then we are done.

Question: What can guarantee that this happen?

There are some conditions which implies that the fractional relaxation has an integer optimal solution. Today we are going to learn one.

TU matrices

Definition: Let A be an $m \times n$ matrix and we select k rows and l columns ($k \leq m$, $l \leq n$) of A. The elements which are contained in the intersection of the selected rows and columns forms a $k \times l$ matrix B. We say that B is a $k \times l$ submatrix of A.

Example:

$$A = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 0 & 1 \\ 2 & 3 & 4 & 5 \end{bmatrix} \qquad B = \begin{bmatrix} 5 & 6 \\ 3 & 4 \end{bmatrix}$$

Definition: A matrix is **totally unimodular**, or **TU** for short, if all of its square ($k \times k$) submatrices have determinant 0, 1 or -1.

Example:

This is a TU matrix:

$$\begin{bmatrix} 0 & 1 & 0 & -1 \\ 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \end{bmatrix}$$

The reason why we like TU matrices

Theorem

Let A be a totally unimodular matrix, let b be an integer vector, and let c be a real vector. If the LP problem $\max\{c^T x | Ax \leq b\}$ has an optimal solution, then the IP problem $\max\{c^T x | Ax \leq b, x \in \mathbb{Z}^n\}$ is also have an optimal solution and its objective value equals to the objective value of the LP problem's optimal solution.

Note that this implies that some optimal solutions of the LP are optimal solutions of the IP, but it is not necessary that all optimal solutions of the *LP* are integer vectors.

How to construct TU matrices?

Claim

If A is a TU matrix, then these operations create a new TU matrix:

1. Multiplying a row or a column of A by -1.
2. Adding a new row or column to A which contains exactly one 1 and the other elements are zeros.
3. We add an existing column or an existing row of A again to A.
4. Transposing A.

The fact that first three operations creates a TU matrix can be proven by using the Laplace expansion ▸ Link to wikipedia and the fourth creates a TU matrix because $det(M) = det(M^T)$ for any square matrix M.

Incidence matrix of graph

Incidence matrix is an encoding of a graph. Each edge and each vertex has a corresponding column and a corresponding row, respectively. If G is undirected, then:

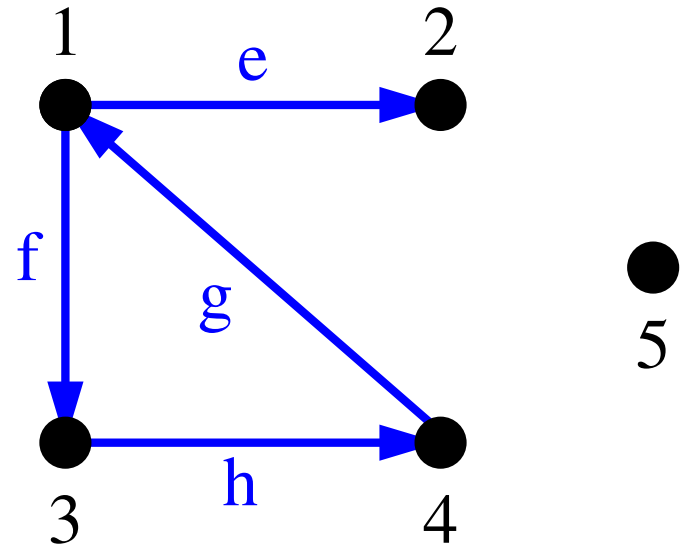
$$A_{i,j} = \begin{cases} 1 & \text{if edge } j \text{ is incident to vertex } i \\ 0 & \text{otherwise.} \end{cases}$$

If G is a directed graph, then:

$$A_{i,j} = \begin{cases} 1 & \text{if vertex } i \text{ is the head of edge } j \\ -1 & \text{if vertex } i \text{ is the tail of edge } j \\ 0 & \text{otherwise.} \end{cases}$$

Example

	e	f	g	h
1	-1	-1	1	0
2	1	0	0	0
3	0	1	0	-1
4	0	0	-1	1
5	0	0	0	0



Some incidence matrices are TU

Theorem

1. The incidence matrix of a directed graph is TU.
2. The incidence matrix of a bipartite graph is TU.

Proof (1): : Let A be the incidence matrix of a directed graph and let M be a $k \times k$ submatrix of A. We show that $det(M) \in \{-1, 0, 1\}$. We use induction on k.

Some incidence matrices are TU

Theorem

1. The incidence matrix of a directed graph is TU.
2. The incidence matrix of a bipartite graph is TU.

Proof (1): : Let A be the incidence matrix of a directed graph and let M be a $k \times k$ submatrix of A. We show that $det(M) \in \{-1, 0, 1\}$. We use induction on k.
If $k = 1$ then the statement holds, because each element of M is 0 or 1 or -1.

Some incidence matrices are TU

Theorem

1. The incidence matrix of a directed graph is TU.
2. The incidence matrix of a bipartite graph is TU.

Proof (1): : Let A be the incidence matrix of a directed graph and let M be a $k \times k$ submatrix of A. We show that $det(M) \in \{-1, 0, 1\}$. We use induction on k.
If $k = 1$ then the statement holds, because each element of M is 0 or 1 or -1.
If $k \geq 2$ and M has a column which contains at most one non-zero, then if we apply Laplace expression for this column and use the induction hypothesis for a $k-1 \times k-1$ submatrix we obtain that $det(M) \in \{-1, 0, 1\}$.

Some incidence matrices are TU

Theorem

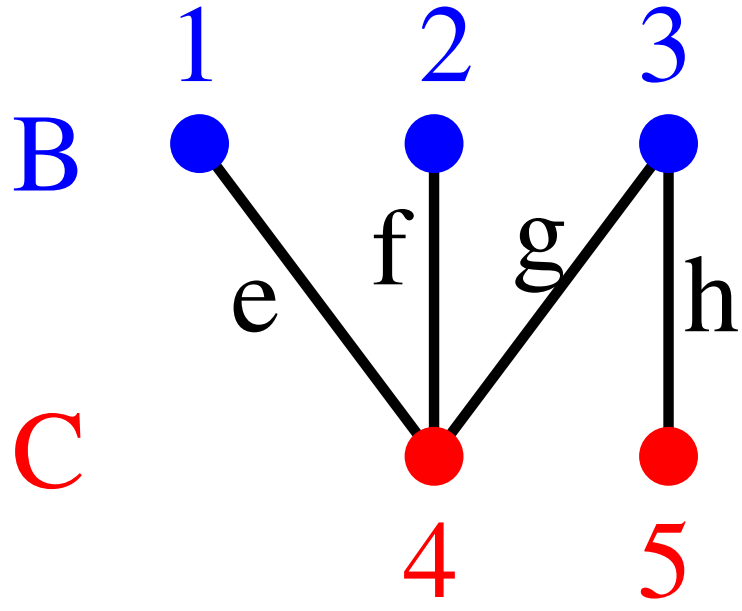
1. The incidence matrix of a directed graph is TU.
2. The incidence matrix of a bipartite graph is TU.

Proof (1): : Let A be the incidence matrix of a directed graph and let M be a $k \times k$ submatrix of A. We show that $det(M) \in \{-1, 0, 1\}$. We use induction on k.
If $k = 1$ then the statement holds, because each element of M is 0 or 1 or -1.
If $k \geq 2$ and M has a column which contains at most one non-zero, then if we apply Laplace expression for this column and use the induction hypothesis for a $k-1 \times k-1$ submatrix we obtain that $det(M) \in \{-1, 0, 1\}$.
Otherwise each column of M contains exactly one -1 and exactly one 1. Therefore the sum of the rows of M gives the zero vector. So the rows of M are linearly dependent, therefore $det(M) = 0$. $\square$

Proof of (2):

Let A be the incidence matrix of a bipartite graph and let B and C be the two subsets of the vertex set such that $B \cup C = V(G)$, $B \cap C = \emptyset$ such that each edge has an endpoint in both sets.

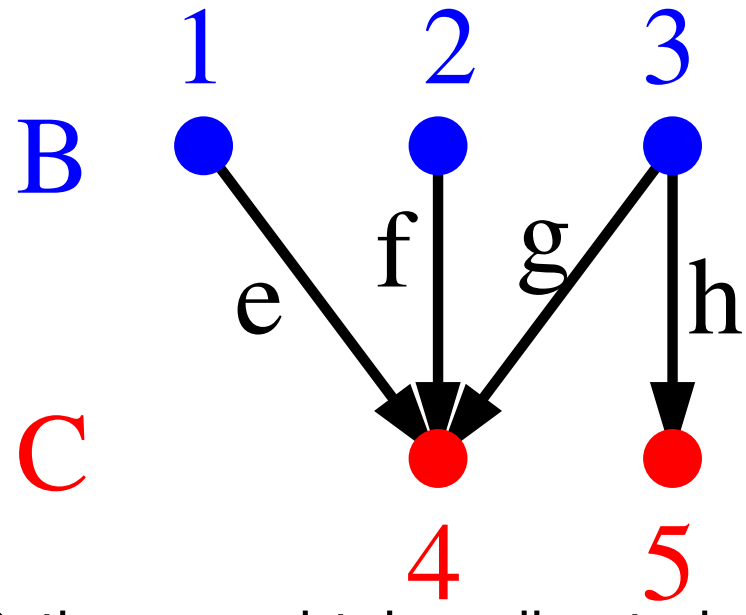
	e	f	g	h
1	1	0	0	0
2	0	1	0	0
3	0	0	1	1
4	1	1	1	0
5	0	0	0	1



Proof of (2):

Let A be the incidence matrix of a bipartite graph and let B and C be the two subsets of the vertex set such that $B \cup C = V(G)$, $B \cap C = \emptyset$ such that each edge has an endpoint in both sets.

	e	f	g	h
1	-1	0	0	0
2	0	-1	0	0
3	0	0	-1	-1
4	1	1	1	0
5	0	0	0	1



If we orient each edge from B to C, then we obtain a directed graph whose incidence matrix is TU. We can obtain A by multiplying the rows correspond to the vertices contained in B by -1. This operation keeps the TU property, therefore A is TU.

□

Application of TU matrices: maximum matching in bipartite graphs

Remember that we end up with this IP:

$$x_i \in \mathbb{Z} \; \forall i \in [1..|E(G)|] \tag{1}$$

$$x_i \leq 1 \; \forall i \in [1..|E(G)|] \tag{2}$$

$$x_i \geq 0 \; \forall i \in [1..|E(G)|] \tag{3}$$

$$\sum_{i|v \text{ is an endpoint of } i} x_i \leq 1 \; \forall v \in V(G) \tag{4}$$

$$\max \sum_{i=1}^{|E(G)|} x_i \tag{5}$$

Equivalently: $\{\max \underline{1}^T x | Ax \leq \underline{1}, Ix \leq \underline{1}, -Ix \leq \underline{0}, x \in \mathbb{Z}^{|E(G)|}\}$. Here A is the incidency matrix of the graph and I is an $|E(G)| \times |E(G)|$ identity matrix. If the graph is bipartite, then A is TU, therefore the matrix $\begin{bmatrix} A \\ I \\ -I \end{bmatrix}$ is also TU and this integer program can be solved in polynomial time.