

C&O 355
Mathematical Programming
Fall 2010
Lecture 17

[N. Harvey](#)

Topics

- Integer Programs
- Computational Complexity Basics
- What is Combinatorial Optimization?
- Bipartite Matching
 - Combinatorial Analysis of Extreme Points
 - Total Unimodularity

Mathematical Programs We've Seen

- Linear Program (LP)

$$\begin{aligned} \min \quad & c^\top x \\ \text{s.t.} \quad & a_i^\top x \leq b_i \quad \forall i = 1, \dots, m \end{aligned}$$

- Convex Program

$$\begin{aligned} \min \quad & f(x) \quad (\text{where } f \text{ is convex}) \\ \text{s.t.} \quad & a_i^\top x \leq b_i \quad \forall i = 1, \dots, m \end{aligned}$$

- Semidefinite Program (SDP)

$$\begin{aligned} \min \quad & c^\top x \\ \text{s.t.} \quad & a_i^\top x \leq b_i \quad \forall i = 1, \dots, m \\ & y^\top X y \geq 0 \quad \forall y \in \mathbb{R}^n \end{aligned}$$

- Integer Program (IP)

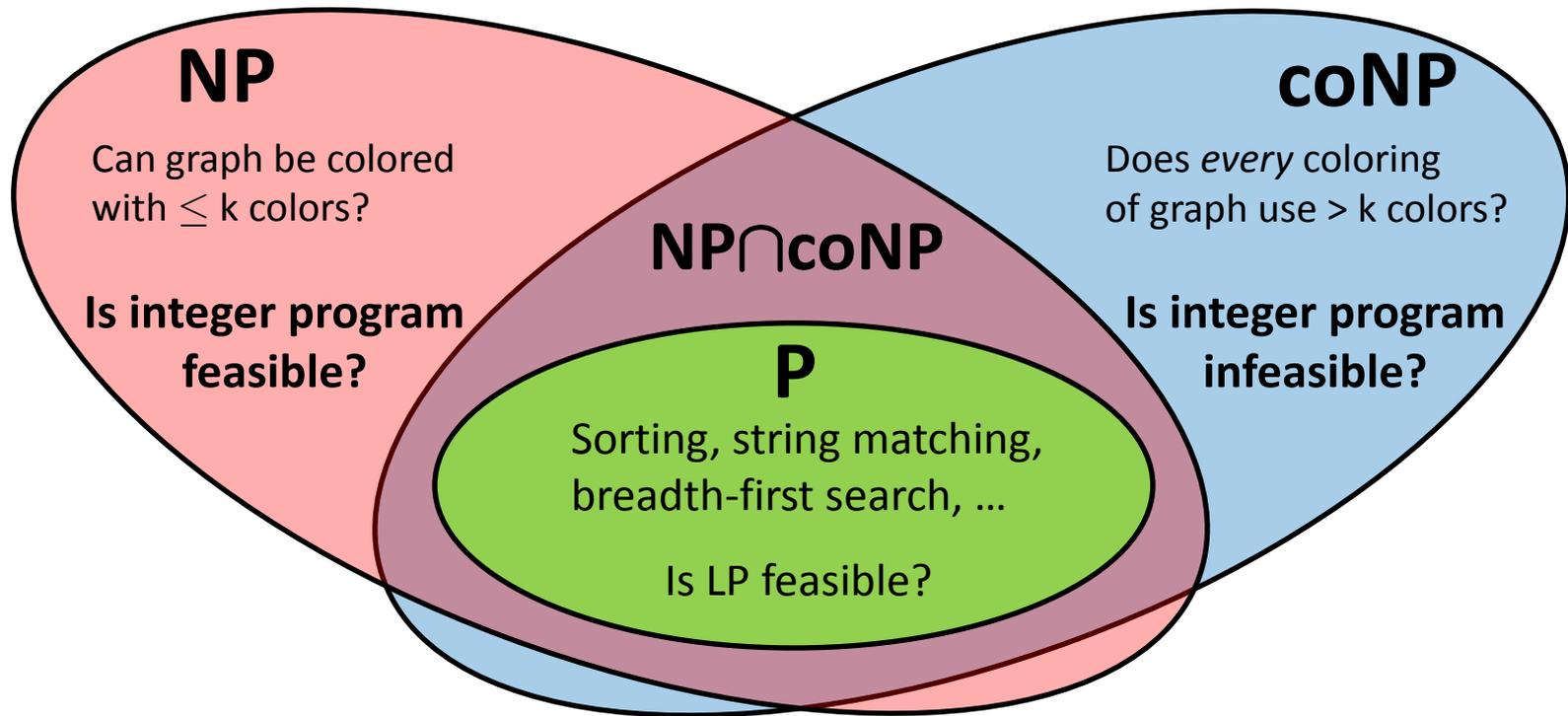
$$\begin{aligned} \min \quad & c^\top x \\ \text{s.t.} \quad & a_i^\top x \leq b_i \quad \forall i = 1, \dots, m \\ & x \in \mathbb{Z}^n \end{aligned}$$

Can be efficiently solved
e.g., by Ellipsoid Method

(where X is symmetric matrix
corresponding to x)

Cannot be efficiently solved
assuming $P \neq NP$

Computational Complexity



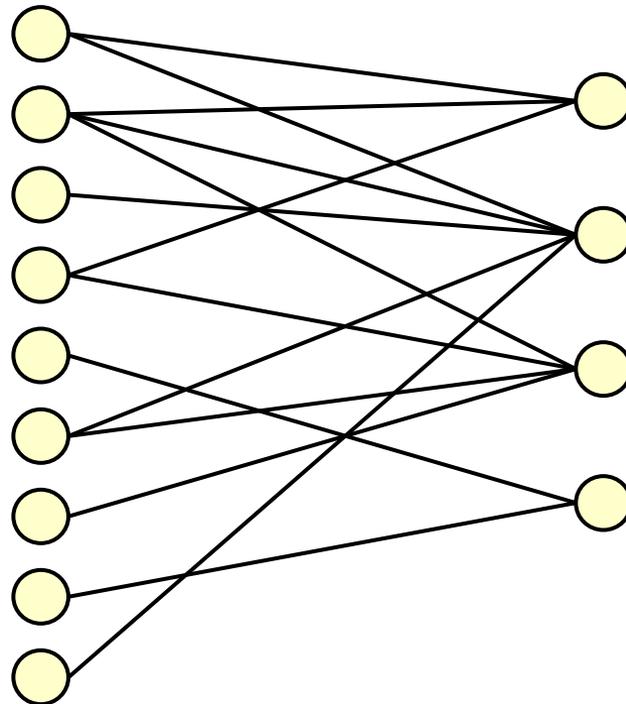
- If you could efficiently (i.e., in polynomial time) decide if every integer program is feasible, then $P = NP$
 - And all of modern cryptography is broken
 - And you win \$1,000,000
 - ...

Combinatorial Optimization

- Study of optimization problems that have **discrete solutions** and some **combinatorial flavor** (e.g., involving graphs)
- Why are we interested in this?
 - Applications: **OR** (planning, scheduling, supply chain), **Computer networks** (shortest paths, low-cost trees), **Compilers** (coloring), **Online advertising** (matching)...
 - Rich theory of what can be solved **efficiently** and what cannot
 - Underlying math can be very interesting: high-dimensional geometry, polyhedra, discrete probability, Banach space theory, Fourier analysis, ...

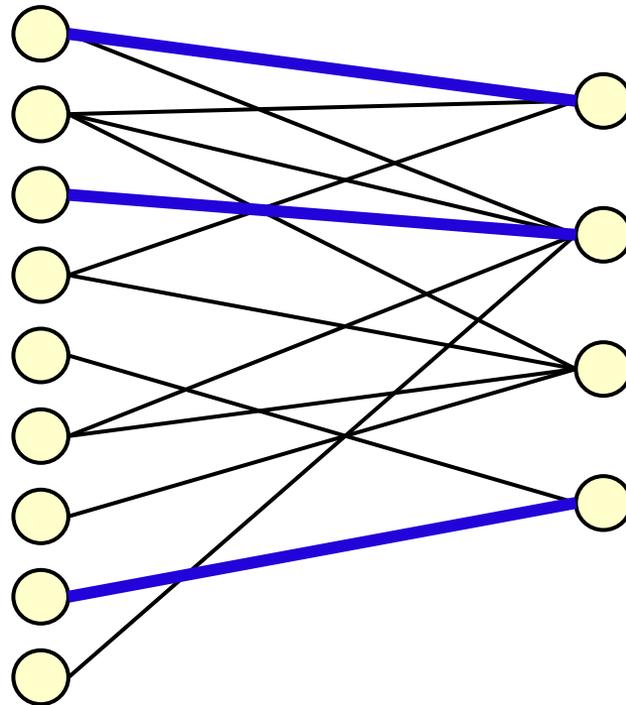
Combinatorial IPs are often nice

- **Maximum Bipartite Matching** (from Lecture 2)
- Given bipartite graph $G=(V, E)$
- Find a maximum size matching
 - A set $M \subseteq E$ s.t. every vertex has at most one incident edge in M



Combinatorial IPs are often nice

- **Maximum Bipartite Matching** (from Lecture 2)
- Given bipartite graph $G=(V, E)$
- Find a maximum size matching
 - A set $M \subseteq E$ s.t. every vertex has at most one incident edge in M



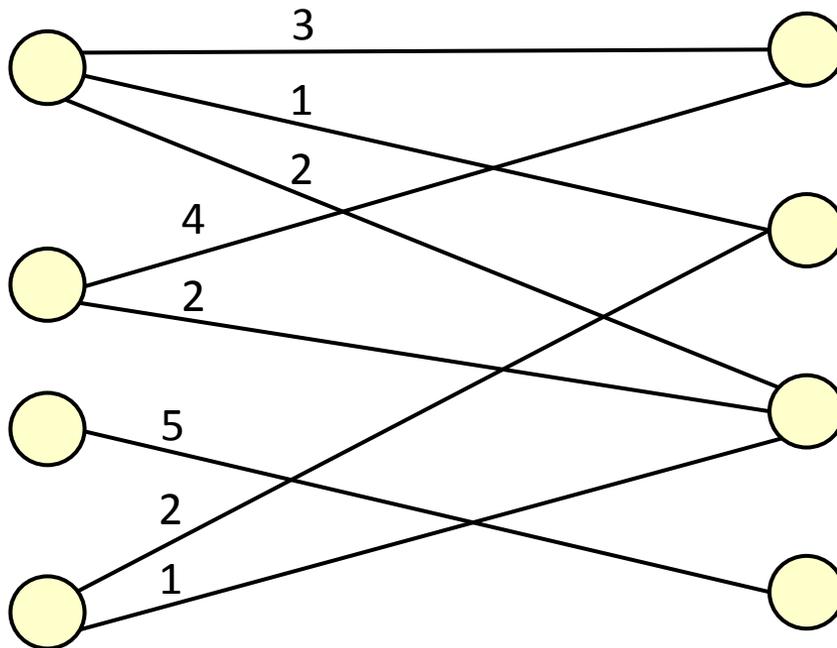
The blue edges are a matching M

Combinatorial IPs are often nice

- **Maximum Bipartite Matching** (from Lecture 2)
- Given bipartite graph $G=(V, E)$
- Find a maximum size matching
 - A set $M \subseteq E$ s.t. every vertex has at most one incident edge in M
- The natural integer program
$$\begin{array}{ll} \max & \sum_{e \in E} x_e \\ \text{s.t.} & \sum_{e \text{ incident to } v} x_e \leq 1 \quad \forall v \in V \\ & x_e \in \{0, 1\} \quad \forall e \in E \end{array}$$
- This IP **can** be efficiently solved, in many different ways

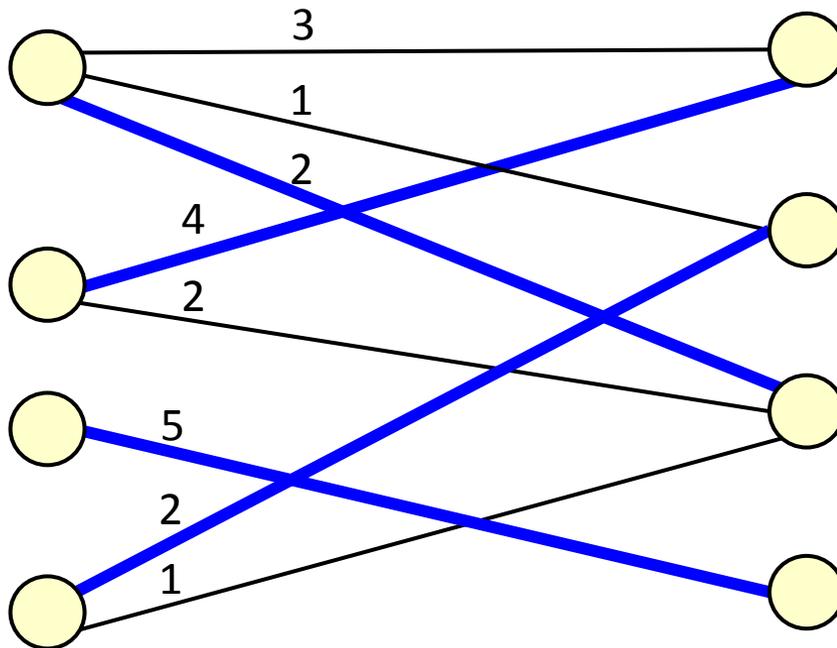
Combinatorial IPs are often nice

- **Max-Weight Perfect Matching**
- Given bipartite graph $G=(V, E)$. Every edge e has a weight w_e .
- Find a maximum-weight perfect matching
 - A set $M \subseteq E$ s.t. every vertex has **exactly** one incident edge in M



Combinatorial IPs are often nice

- **Max-Weight Perfect Matching**
- Given bipartite graph $G=(V, E)$. Every edge e has a weight w_e .
- Find a maximum-weight perfect matching
 - A set $M \subseteq E$ s.t. every vertex has **exactly** one incident edge in M



The **blue** edges are a max-weight perfect matching M

Combinatorial IPs are often nice

- **Max-Weight Perfect Matching**
- Given bipartite graph $G=(V, E)$. Every edge e has a weight w_e .
- Find a maximum-weight perfect matching
 - A set $M \subseteq E$ s.t. every vertex has **exactly** one incident edge in M
- The natural integer program

$$\begin{array}{ll} \max & \sum_{e \in E} w_e \cdot x_e \\ \text{s.t.} & \sum_{e \text{ incident to } v} x_e = 1 \quad \forall v \in V \\ & x_e \in \{0, 1\} \quad \forall e \in E \end{array}$$

- This IP **can** be efficiently solved, in many different ways

Birth of Computational Complexity

PATHS, TREES, AND FLOWERS

JACK EDMONDS

1965

2008, Aussois, France



Birth of Computational Complexity

PATHS, TREES, AND FLOWERS

JACK EDMONDS

1965

Richard Karp

2008, Aussois, France



Birth of Computational Complexity

PATHS, TREES, AND FLOWERS

JACK EDMONDS

2. Digression. An explanation is due on the use of the words “efficient algorithm.” First, what I present is a conceptual description of an algorithm and not a particular formalized algorithm or “code.”

For practical purposes computational details are vital. However, my purpose is only to show as attractively as I can that there is an efficient algorithm. According to the dictionary, “efficient” means “adequate in operation or performance.” This is roughly the meaning I want—in the sense that it is conceivable for maximum matching to have no efficient algorithm. Perhaps a better word is “good.”

I am claiming, as a mathematical result, the existence of a *good* algorithm for finding a maximum cardinality matching in a graph.

There is an obvious finite algorithm, but that algorithm increases in difficulty exponentially with the size of the graph. It is by no means obvious whether *or not* there exists an algorithm whose difficulty increases only algebraically with the size of the graph.

Efficient algorithm for matching,
even in non-bipartite graphs

This means “**polynomial time**”.
So Edmonds is defining the complexity class P.

How to solve combinatorial IPs?

- Two common approaches
 1. Design combinatorial algorithm that directly solves IP
 - Often such algorithms have a nice LP interpretation
 2. Relax IP to an LP; prove that they give same solution; solve LP by the ellipsoid method
 - Need to show special structure of the LP's extreme points
 - Sometimes we can analyze the extreme points **combinatorially**
 - Sometimes we can use **algebraic** structure of the constraints. For example, if constraint matrix is **Totally Unimodular** then IP and LP are equivalent
- We'll see examples of these approaches

Perfect Matching Problem

- Let $G=(V, E)$ be a bipartite graph. Every edge e has a weight w_e .
- Find a maximum-weight, **perfect** matching
 - A set $M \subseteq E$ s.t. every vertex has **exactly** one incident edge in M
- Write an integer program

$$\begin{array}{ll} \text{(IP)} & \max \quad \sum_{e \in E} w_e \cdot x_e \\ & \text{s.t.} \quad \sum_{e \text{ incident to } v} x_e = 1 \quad \forall v \in V \\ & \quad \quad x_e \in \{0, 1\} \quad \forall e \in E \end{array}$$

- Relax integrality constraints, obtain an LP

$$\begin{array}{ll} \text{(LP)} & \max \quad \sum_{e \in E} w_e \cdot x_e \\ & \text{s.t.} \quad \sum_{e \text{ incident to } v} x_e = 1 \quad \forall v \in V \\ & \quad \quad x_e \geq 0 \quad \forall e \in E \quad (x_e \leq 1 \text{ is implicit}) \end{array}$$

- **Theorem:** Every BFS of (LP) is actually an (IP) solution!

Combinatorial Analysis of BFSs

- **Lemma:**

Every BFS of perfect matching (LP) is an (IP) solution.

- **Proof:** Let x be BFS, suppose x not integral.

- Pick any edge $e_1 = \{v_0, v_1\}$ with $0 < x_{e_1} < 1$.

- The LP requires $\sum_{e \text{ incident on } v_1} x_e = 1$

\Rightarrow there is **another** edge $e_2 = \{v_1, v_2\}$ with $0 < x_{e_2} < 1$.

- The LP requires $\sum_{e \text{ incident on } v_2} x_e = 1$

\Rightarrow there is **another** edge $e_3 = \{v_2, v_3\}$ with $0 < x_{e_3} < 1$.

- Continue finding distinct edges until eventually $v_i = v_k, i < k$

- We have $e_{i+1} = \{v_i, v_{i+1}\}, e_{i+2} = \{v_{i+1}, v_{i+2}\}, \dots, e_k = \{v_{k-1}, v_k\}$.

(all edges and vertices distinct, except $v_i = v_k$)

Combinatorial Analysis of BFSs

- Let x be BFS of matching (LP). Suppose x not integral.
- WLOG, $e_1=\{v_0,v_1\}$, $e_2=\{v_1,v_2\}$, ..., $e_k=\{v_{k-1},v_k\}$ and $v_0=v_k$.

$$0 < x_{e_i} < 1 \quad \forall i = 1, \dots, k$$

$$\sum_{e \text{ incident on } v_i} x_e = 1 \quad \forall i = 1, \dots, k$$

- These edges form a simple cycle, of **even length**.

(Even length since G is bipartite.)

- Define the vector:
$$d_e = \begin{cases} 0 & \text{if } e \neq e_j \text{ for any } j \\ 1 & \text{if } e = e_j \text{ and } j \text{ odd} \\ -1 & \text{if } e = e_j \text{ and } j \text{ even} \end{cases}$$

- **Claim:** If $|\epsilon|$ is sufficiently small, then $x+\epsilon d$ is feasible
- So x is convex combination of $x+\epsilon d$ and $x-\epsilon d$, both feasible
- This contradicts x being a BFS. ■

How to solve combinatorial IPs?

- Two common approaches
 1. Design combinatorial algorithm that directly solves IP
 - Often such algorithms have a nice LP interpretation
 2. Relax IP to an LP; prove that they give same solution; solve LP by the ellipsoid method
 - Need to show special structure of the LP's extreme points
 - Sometimes we can analyze the extreme points **combinatorially**
 - Sometimes we can use **algebraic** structure of the constraints. For example, if constraint matrix is **Totally Unimodular** then IP and LP are equivalent



LP Approach for Bipartite Matching

- Let $G=(V, E)$ be a bipartite graph. Every edge e has a weight w_e .
- Find a maximum weight matching
 - A set $M \subseteq E$ s.t. every vertex has at most one incident edge in M

- Write an integer program

$$\begin{array}{ll} \text{(IP)} & \max \quad \sum_{e \in E} w_e \cdot x_e \\ & \text{s.t.} \quad \sum_{e \text{ incident to } v} x_e \leq 1 \quad \forall v \in V \\ & \quad \quad x_e \in \{0, 1\} \quad \forall e \in E \end{array}$$

- Relax integrality constraints, obtain an LP

$$\begin{array}{ll} \text{(LP)} & \max \quad \sum_{e \in E} w_e \cdot x_e \\ & \text{s.t.} \quad \sum_{e \text{ incident to } v} x_e \leq 1 \quad \forall v \in V \\ & \quad \quad x_e \geq 0 \quad \forall e \in E \quad (x_e \leq 1 \text{ is implicit}) \end{array}$$

- **Theorem:** Every BFS of (LP) is actually an (IP) solution!

Total Unimodularity

- Let A be a real $m \times n$ matrix

• **Definition:** Suppose that every square submatrix of A has determinant in $\{0, +1, -1\}$. Then A is **totally unimodular (TUM)**.

– In particular, every entry of A must be in $\{0, +1, -1\}$

- **Lemma:** Suppose A is TUM. Let b be any integer vector. Then every basic feasible solution of $P = \{x : Ax \leq b\}$ is integral.

- **Proof:** Let x be a basic feasible solution.

Then the constraints that are tight at x have rank n .

Let A' be a submatrix of A and b' a subvector of b corresponding to n linearly independent constraints that are tight at x .

Then x is the unique solution to $A'x = b'$, i.e., $x = (A')^{-1}b'$.

Cramer's Rule: If M is a square, non-singular matrix then

$$(M^{-1})_{i,j} = (-1)^{i+j} \det \underbrace{M_{\text{del}(j,i)}} / \det M.$$

Submatrix of M obtained by deleting row j and column i

Total Unimodularity

- Let A be a real $m \times n$ matrix
- **Definition:** Suppose that every square submatrix of A has determinant in $\{0, +1, -1\}$. Then A is **totally unimodular (TUM)**.
- **Lemma:** Suppose A is TUM. Let b be any integer vector. Then every basic feasible solution of $P = \{x : Ax \leq b\}$ is integral.
- **Proof:** Let x be a basic feasible solution.

Then the constraints that are tight at x have rank n .

Let A' be the submatrix of A and b' the subvector of b containing n linearly independent constraints that are tight at x .

Then x is the unique solution to $A'x = b'$, i.e., $x = (A')^{-1}b'$.

Cramer's Rule: If M is a square, non-singular matrix then $(M^{-1})_{i,j} = (-1)^{i+j} \det M_{\text{del}(j,i)} / \det M$.

Thus all entries of $(A')^{-1}$ are in $\{0, +1, -1\}$.

Since b' is integral, x is also integral. ■

Operations Preserving Total Unimodularity

- Let A be a real $m \times n$ matrix
- **Definition:** Suppose that every square submatrix of A has determinant in $\{0, +1, -1\}$. Then A is **totally unimodular (TUM)**.
- **Lemma:** Suppose A is TUM. Let b be any integer vector. Then every basic feasible solution of $P = \{x : Ax \leq b\}$ is integral.
- **Claim:** Suppose A is TUM. Then $\begin{pmatrix} A \\ -I \end{pmatrix}$ is also TUM.
- **Proof:** Exercise? □

• **Corollary:** Suppose A is TUM. Let b be any integer vector. Then every basic feasible solution of $P = \{x : Ax \leq b, x \geq 0\}$ is integral.

- **Proof:** By the Claim, $\begin{pmatrix} A \\ -I \end{pmatrix}$ is TUM. So apply the Lemma to

$$P = \left\{ x : \begin{pmatrix} A \\ -I \end{pmatrix} x \leq \begin{pmatrix} b \\ 0 \end{pmatrix} \right\}$$



Bipartite Matching & Total Unimodularity

- Let $G=(U \cup V, E)$ be a bipartite graph.
 - So all edges have one endpoint in U and the other in V .
- Let A be the “incidence matrix” of G .
 A has a row for every vertex and a column for every edge.

$$A_{w,e} = \begin{cases} 1 & \text{if vertex } w \text{ is an endpoint of edge } e \\ 0 & \text{otherwise} \end{cases}$$

Note: Every column of A has exactly two non-zero entries.

- **Lemma:** A is TUM.
- **Proof:** Let Q be a $k \times k$ submatrix of A . Argue by induction on k .
If $k=1$ then Q is a single entry of A , so $\det(Q)$ is either 0 or 1.
Suppose $k>1$.
If some column of Q has **no** non-zero entries, then $\det(Q)=0$.

- Let $G=(U \cup V, E)$ be a bipartite graph. Define A by

$$A_{v,e} = \begin{cases} 1 & \text{if vertex } v \text{ is an endpoint of edge } e \\ 0 & \text{otherwise} \end{cases}$$

- **Lemma:** A is TUM.
- **Proof:** Let Q be a $k \times k$ submatrix of A . Assume $k > 1$.

If some column of Q has **no** non-zero entries, then $\det(Q)=0$.

Suppose j^{th} column of Q has **exactly one** non-zero entry, say $Q_{t,j} \neq 0$

Use “Column Expansion” of determinant:

$$\det Q = \sum_i (-1)^{i+j} Q_{i,j} \cdot \det Q_{\text{del}(i,j)} = (-1)^{t+j} Q_{t,j} \cdot \det Q_{\text{del}(t,j)},$$

where t is the unique non-zero entry in column j .

By induction, $\det Q_{\text{del}(t,j)} \in \{0,+1,-1\} \Rightarrow \det Q \in \{0,+1,-1\}$.

- Let $G=(U \cup V, E)$ be a bipartite graph. Define A by

$$A_{v,e} = \begin{cases} 1 & \text{if vertex } v \text{ is an endpoint of edge } e \\ 0 & \text{otherwise} \end{cases}$$

- **Lemma:** A is TUM.

- **Proof:** Let Q be a $k \times k$ submatrix of A . Assume $k > 1$.

If some column of Q has **no** non-zero entries, then $\det(Q)=0$.

If j^{th} column of Q has **exactly one** non-zero entry, use induction.

Suppose **every** column of Q has **exactly two** non-zero entries.

- For each column, one non-zero is in a U -row and the other is in a V -row.

So summing all U -rows in Q gives the vector $[1,1,\dots,1]$.

Also summing all V -rows in Q gives the vector $[1,1,\dots,1]$.

So (sum of U -rows) – (sum of V -rows) = $[0,0,\dots,0]$.

Thus Q is singular, and $\det Q = 0$. ■

- Let $G=(U \cup V, E)$ be a bipartite graph. Define A by

$$A_{v,e} = \begin{cases} 1 & \text{if vertex } v \text{ is an endpoint of edge } e \\ 0 & \text{otherwise} \end{cases}$$

- **Lemma:** A is TUM.
- So every **BFS** of $P = \{ x : Ax \leq \mathbf{1}, x \geq 0 \}$ is **integral**.
- We can rewrite the LP $\max \{ w^T x : x \in P \}$ as

$$\begin{aligned} \max \quad & \sum_{e \in E} w_e \cdot x_e \\ \text{s.t.} \quad & \sum_{e \text{ incident to } v} x_e \leq 1 \quad \forall v \in V \\ & x_e \geq 0 \quad \forall e \in E \quad (x_e \leq 1 \text{ is implicit}) \end{aligned}$$

- For every objective function w , this LP has an **optimal solution at a BFS**. (Since P is bounded)
- So for every vector w , the LP has an **integral optimal solution** x .
 - Since $0 \leq x_e \leq 1$, and x is **integral**, we actually have $x_e \in \{0,1\}$.
- So every **optimal LP solution** is actually an **(optimal) IP solution**.
 \Rightarrow So we can solve the IP by solving the LP and returning a BFS.

How to solve combinatorial IPs?

- Two common approaches
 1. Design combinatorial algorithm that directly solves IP
 - Often such algorithms have a nice LP interpretation
 2. Relax IP to an LP; prove that they give same solution; solve LP by the ellipsoid method
 - Need to show special structure of the LP's extreme points
-  Sometimes we can analyze the extreme points **combinatorially**
-  Sometimes we can use **algebraic** structure of the constraints.
For example, if constraint matrix is **Totally Unimodular** then IP and LP are equivalent