C&O 355
Lecture 19

[N. Harvey](#)

Topics

- Solving Integer Programs
- Basic Combinatorial Optimization Problems
  – Bipartite Matching, Minimum s-t Cut, Shortest Paths, Minimum Spanning Trees
- Bipartite Matching
  – Combinatorial Analysis of Extreme Points
  – Total Unimodularity

Mathematical Programs We've Seen

- Linear Program (LP)

$$\begin{array}{lll} \min & c^{\mathsf{T}} x & \\ \text{s.t.} & a_i^{\mathsf{T}} x \le b_i & \forall i = 1, \ldots, m \end{array}$$

- Convex Program

$$\begin{array}{lll} \min & f(x) & \\ \text{s.t.} & a_i^{\mathsf{T}} x \le b_i & \forall i = 1, \ldots, m \end{array}$$

(where f is convex)

- Semidefinite Program (SDP)

$$\begin{array}{llll} \min & c^T x & & \\ \text{s.t.} & a_i^{\mathsf{T}} x & \le b_i & \forall i = 1, \ldots, m \\ & y^{\mathsf{T}} X y & \ge 0 & \forall y \in \mathbb{R}^n \end{array}$$

(where X is symmetric matrix corresponding to x)

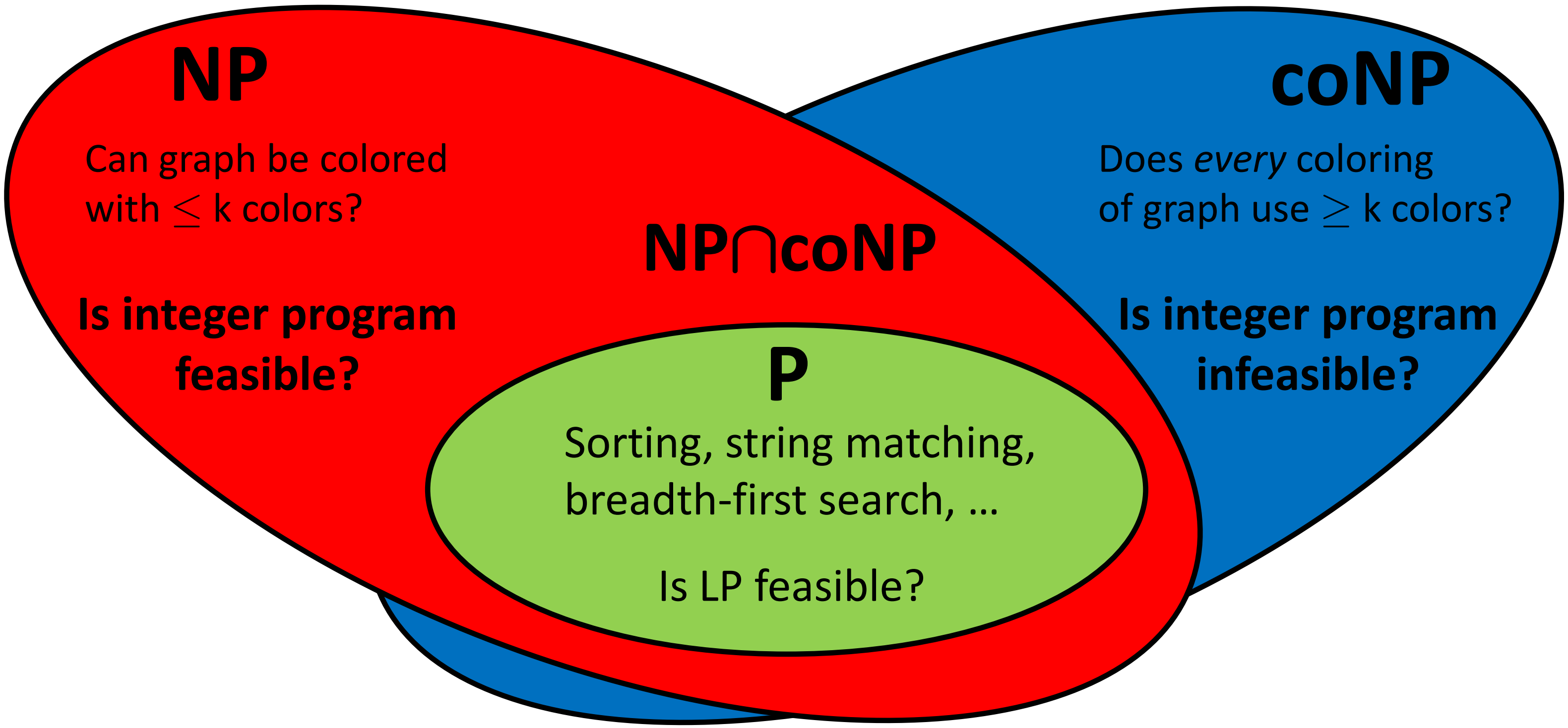
Can be efficiently solved e.g., by Ellipsoid Method

- Integer Program (IP)

$$\begin{array}{llll} \min & c^{\mathsf{T}} x & & \\ \text{s.t.} & a_i^{\mathsf{T}} x & \le b_i & \forall i = 1, \ldots, m \\ & x & \in \mathbb{Z}^n & \end{array}$$

Cannot be efficiently solved assuming P $\neq$ NP

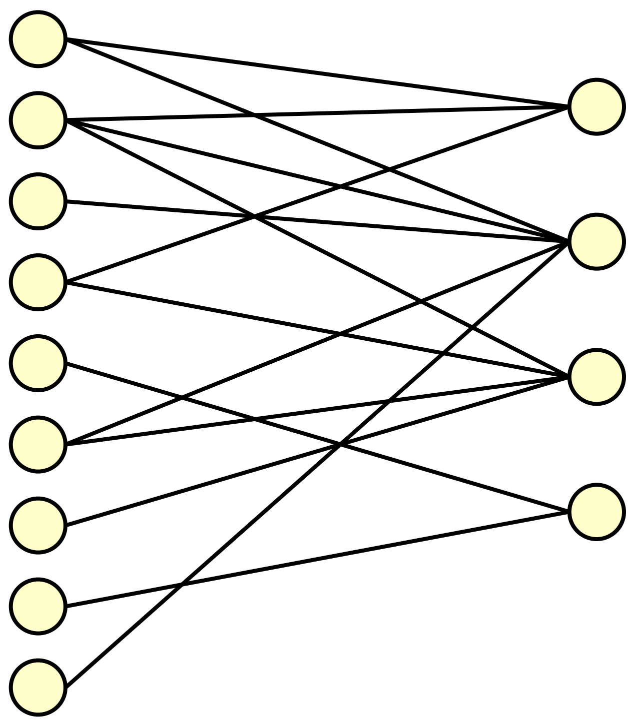
Computational Complexity



- 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
  - …

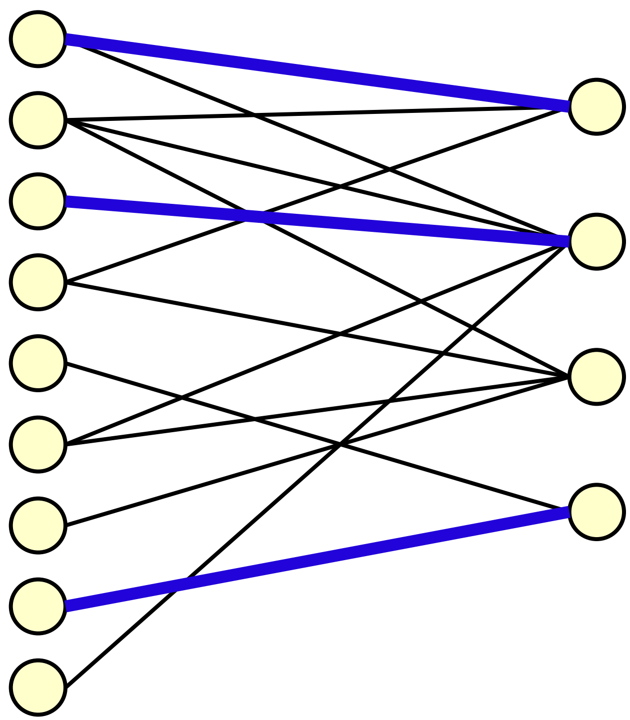
Combinatorial IPs are often nice

- **Maximum Bipartite Matching** (from Lecture 2)
- Given bipartite graph G=(V, E)
- Find a maximum size matching
  - A set M ⊆ 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 ⊆ 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

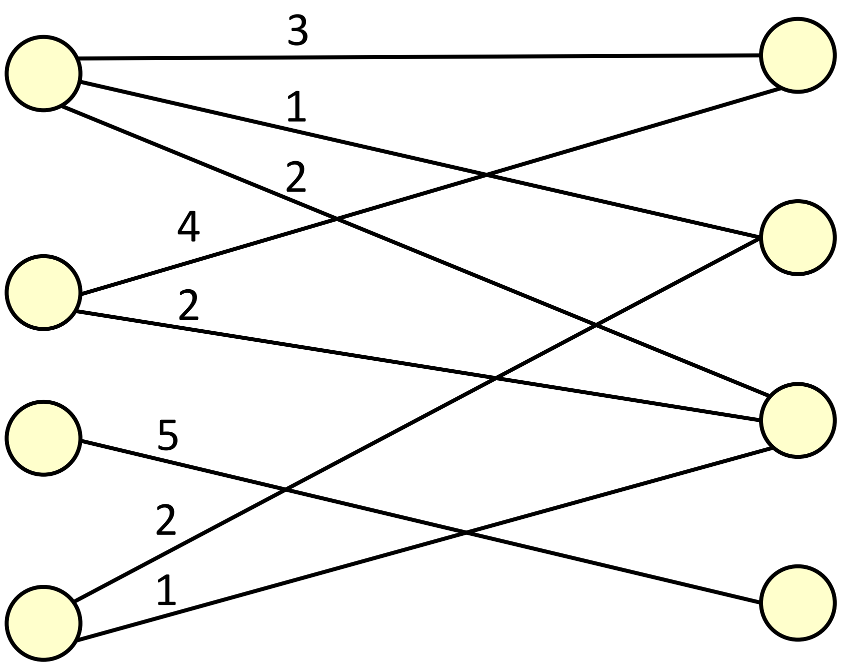
- **Maximum Bipartite Matching** (from Lecture 2)
- Given bipartite graph G=(V, E)
- Find a maximum size matching
  - A set M ⊆ E s.t. every vertex has at most one incident edge in M
- The natural integer program

$$\begin{array}{llll} \max & \sum_{e \in E} x_e & & \\ \text{s.t.} & \sum_{e \text{ incident to } v} x_e & \leq 1 & \forall v \in V \\ & x_e & \in \{0,1\} & \forall e \in E \end{array}$$

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

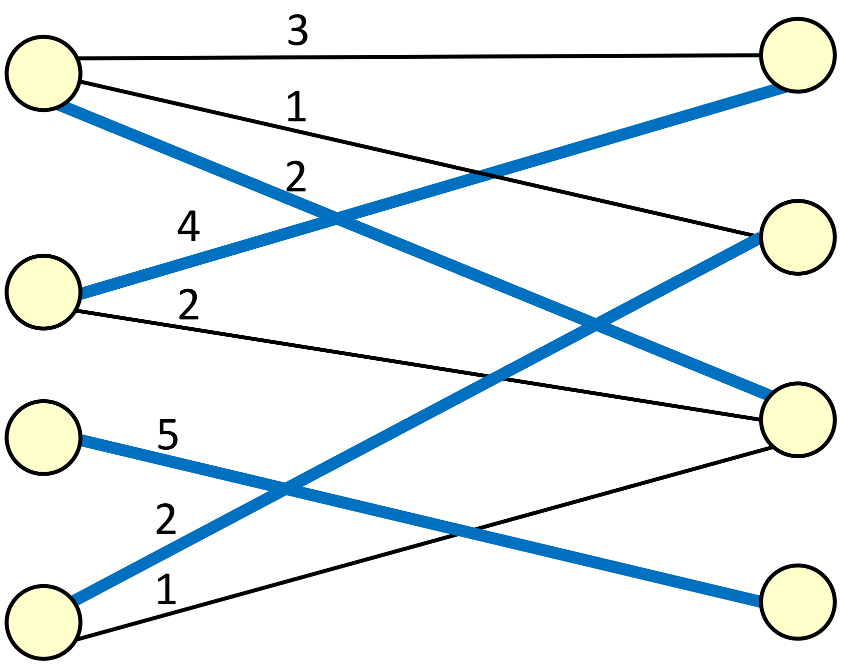
Combinatorial IPs are often nice

- **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

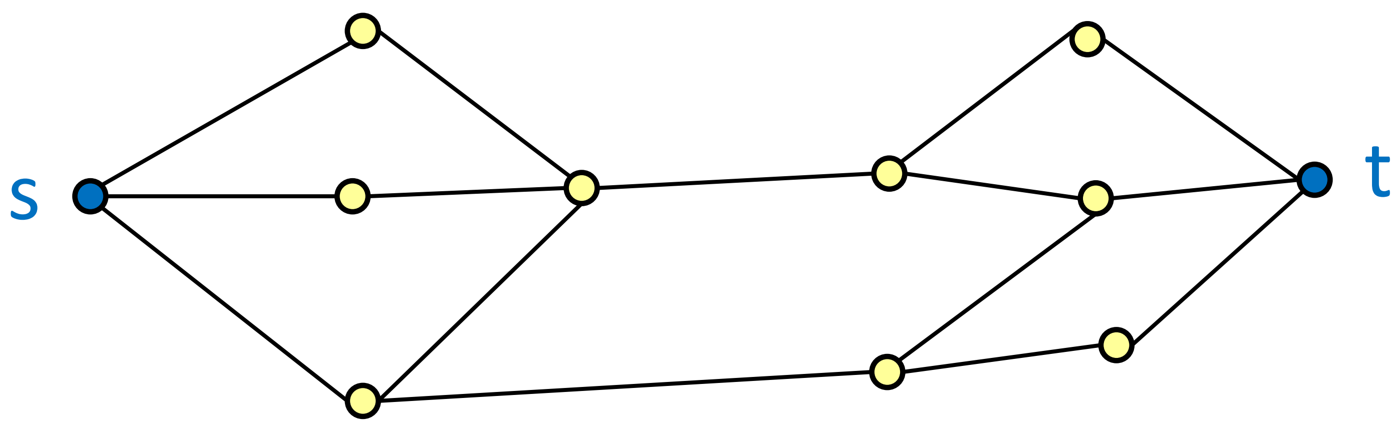
- **Max-Weight Perfect Matching**
- Given bipartite graph G=(V, E). Every edge e has a weight w_e.
- Find a maximum-weight perfect matching
  - A set M $\subseteq$ E s.t. every vertex has **exactly** one incident edge in M
- The natural integer program

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

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

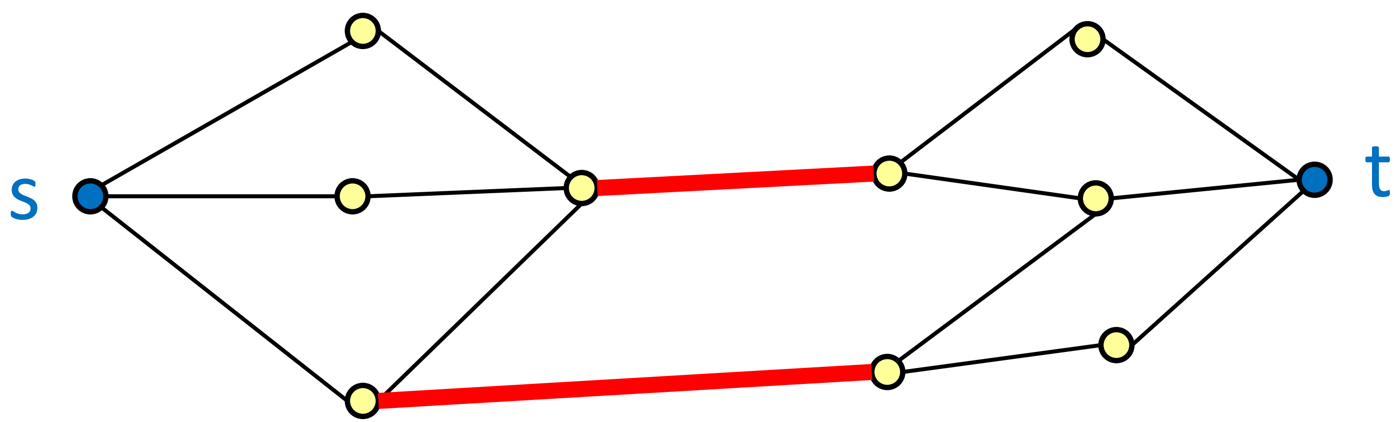
Combinatorial IPs are often nice

- **Minimum s-t Cut in a Graph** (from Lecture 12)
- Let G=(V,E) be a graph. Fix two vertices s,t∈V.
- An **s-t cut** is a set F⊆E such that, if you delete F, then s and t are disconnected
  i.e., there is no s-t path in G\F = (V,E\F).



Combinatorial IPs are often nice

- **Minimum s-t Cut in a Graph** (from Lecture 12)
- Let G=(V,E) be a graph. Fix two vertices s,t∈V.
- An **s-t cut** is a set F⊆E such that, if you delete F, then s and t are disconnected
  i.e., there is no s-t path in G\F = (V,E\F).



These edges are a **minimum** s-t cut

Combinatorial IPs are often nice

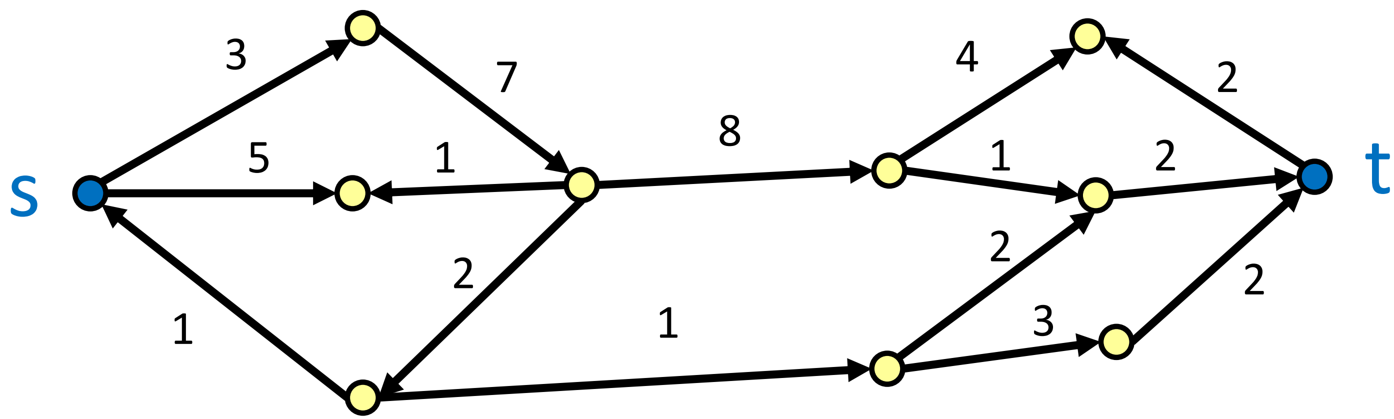
- **Minimum s-t Cut in a Graph** (from Lecture 12)
- Let G=(V,E) be a graph. Fix two vertices s,t$\in$V.
- An **s-t cut** is a set F$\subseteq$E such that, if you delete F, then s and t are disconnected.
- Want to find an s-t cut of minimum cardinality
- Write a (very big!) integer program. Make variable x_e for every e $\in$ E. Let $\mathcal{P}$ be set of all s-t paths.

$$\begin{array}{llll} \min & \sum_{e \in E} x_e & & \\ \text{s.t.} & \sum_{e \in p} x_e & \geq 1 & \forall p \in \mathcal{P} \\ & x_e & \in \{0,1\} & \forall e \in E \end{array}$$

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

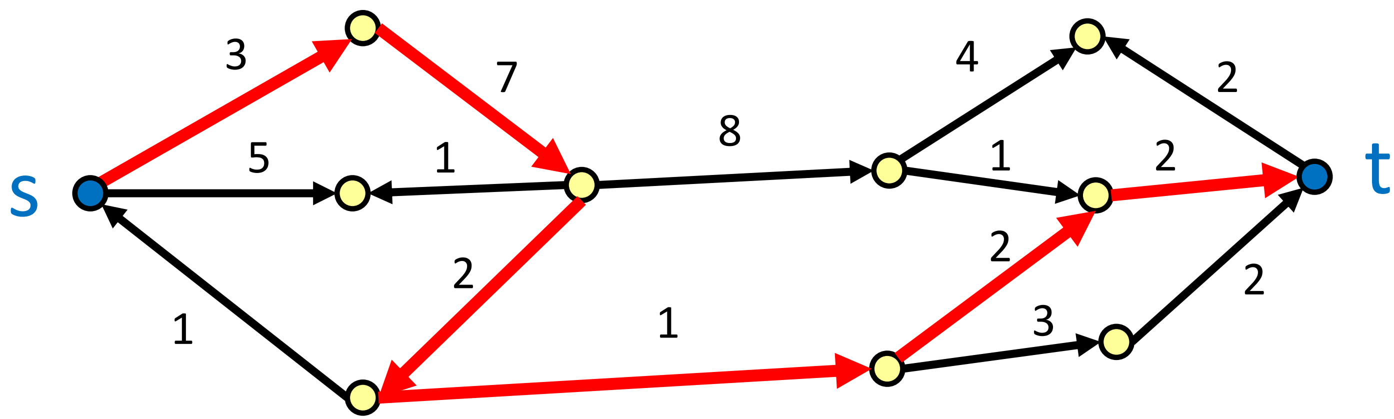
Combinatorial IPs are often nice

- **Shortest Paths in a Digraph**
- Let G=(V,A) be a directed graph. Every arc a has a "length" $w_a \geq 0$.
- Given two vertices s and t, find a path from s to t of minimum total length.



Combinatorial IPs are often nice

- **Shortest Paths in a Digraph**
- Let G=(V,A) be a directed graph. Every arc a has a "length" $w_a \geq 0$.
- Given two vertices s and t, find a path from s to t of minimum total length.



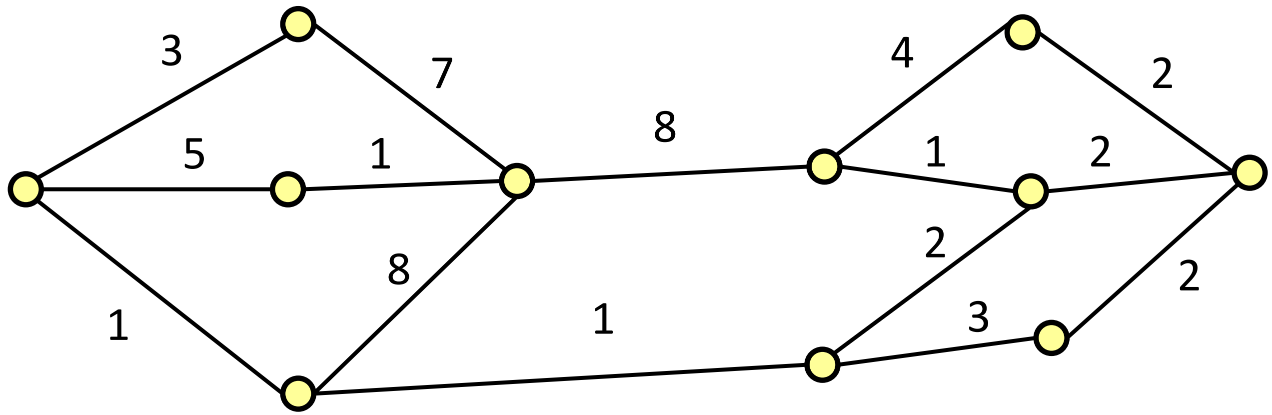
These edges form a **shortest s-t path**

Combinatorial IPs are often nice

- **Shortest Paths in a Digraph**
- Let G=(V,A) be a directed graph. Every arc a has a "length" $w_a \geq 0$.
- Given two vertices s and t, find a path from s to t of minimum total length.
- There is a natural IP for this problem that can be efficiently solved, in many different ways.

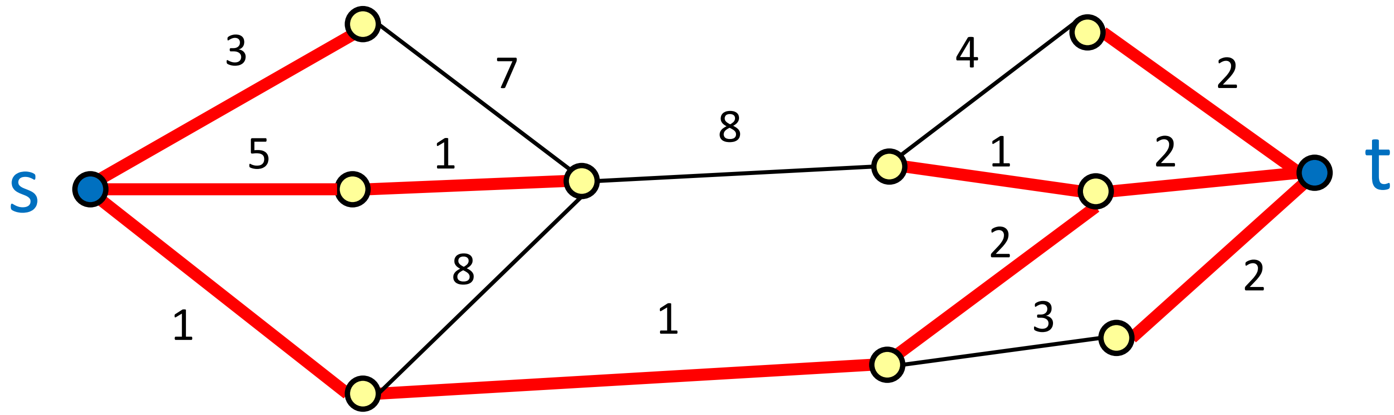
Combinatorial IPs are often nice

- **Minimum Spanning Tree in a Graph**
- Let G=(V,E) be a graph. Every edge e has a weight w_e.
- An **spanning tree** is a set $F \subseteq E$ with no cycles, such that F contains a path between every pair of vertices.



Combinatorial IPs are often nice

- **Minimum Spanning Tree in a Graph**
- Let G=(V,E) be a graph. Every edge e has a weight w_e.
- An **spanning tree** is a set $F \subseteq E$ with no cycles, such that F contains a path between every pair of vertices.



These edges are a **minimum spanning tree**

There is an s-t path in the tree

Combinatorial IPs are often nice

- **Minimum Spanning Tree in a Graph**
- Let $G=(V,E)$ be a graph. Every edge e has a weight w_e.
- An **spanning tree** is a set $F \subseteq E$ with no cycles, such that F contains a path between every pair of vertices.
- There is a natural IP for this problem that can be efficiently solved, in many different ways.

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⊆E s.t. every vertex has **exactly** one incident edge in M
- Write an integer program

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

- Relax integrality constraints, obtain an LP

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

($x_e \leq 1$ is implicit)

- **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 $0 < x_{e1} < 1$.
- The LP requires $\Sigma_{e \text{ incident on } v_1} x_e = 1$
  $\Rightarrow$ there is **another** edge $e_2=\{v_1,v_2\}$ with $0 < x_{e2} < 1$.
- The LP requires $\Sigma_{e \text{ incident on } v_2} x_e = 1$
  $\Rightarrow$ there is **another** edge $e_3=\{v_2,v_3\}$ with $0 < x_{e3} < 1$.
- Continue finding distinct edges until eventually $v_i=v_k$, $i<k$
- We have $e_{i+1}=\{v_i,v_{i+1}\}$, $e_{i+2}=\{v_{i+1},v_{i+2}\}$, ..., $e_k=\{v_{k-1},v_k\}$.
  (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 $v_0=v_k$.

$$0 < x_{e_i} < 1 \;\; \forall i = 1, ..., k$$
$$\Sigma_{e \text{ incident on } v_i}\, x_e = 1 \;\; \forall i = 1, ..., k$$

- 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

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

- Relax integrality constraints, obtain an LP

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

($x_e \leq 1$ is implicit)

- **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 $(M^{-1})_{i,j} = (-1)^{i+j} \det \underbrace{M_{del(j,i)}}_{\text{Submatrix of M obtained by deleting row j and column i}} / \det M$.

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

  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×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 on Assignment 5. □
- **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

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

■

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

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

- **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:

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

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

  By induction, $\det Q_{\mathrm{del}(t,j)}$ 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,…,1].

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

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

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

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

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

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

$$\begin{array}{llll} \max & \sum_{e \in E} w_e \cdot x_e & & \\ \text{s.t.} & \sum_{e \text{ incident to } v} x_e & \leq 1 & \forall v \in V \\ & x_e & \geq 0 & \forall e \in E \end{array}$$

($x_e \leq 1$ is implicit)

- 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
     - ometimes we can analyze the extreme points **combinatorially**
     - ometimes we can use **algebraic** structure of the constraints. or example, if constraint matrix is **Totally Unimodular** then IP and LP are equivalent