ISyE/CS/Math 728: Integer Optimization
Perfect Formulations

Prof. Luedtke

UW-Madison

Spring 2021

Outline

Examples and applications of integer programs that have perfect formulations

- IP's with totally unimodular constraint matrix
- Example: Shortest path
- Application: Max-flow Min-Cut theorem
- Matchings in graphs

Perfect Formulations

Let $P \subseteq \mathbb{R}^n \times \mathbb{R}^p$ be a rational polyhedron, and let $S := P \cap (\mathbb{Z}^n \times \mathbb{R}^p)$.

Recall:

- We say P is a *perfect formulation* for S if $P = \operatorname{conv}(S)$.
- In the pure integer case $p = 0$, if P is a perfect formulation for S we also say P is an *integral polyhedron*

We like perfect formulations!

- Imply we can solve the MIP as an LP
- Sometimes we obtain a perfect formulation for only *part* of our problem
  - Then we know we have a "best possible" formulation for that part of the problem

When can we obtain a perfect formulation?

- Can we identify a general condition on the matrix A that guarantees that

$$P = \{x \in \mathbb{R}^n_+ : Ax \leq b\}$$

  is an integral polyhedron for *any* integer right-hand side b?
- Yes! If the matrix A is *totally unimodular*

Total Unimodular? Totally Cool!

Submatrix of a matrix A: Any matrix obtained by deleting any rows or columns from A.

Total Unimodularity

A matrix A is totally unimodular (TU) if *every* square submatrix of A has determinant $+1, -1$, or 0.

TU matrix: $\begin{pmatrix} -1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$ non-TU matrix: $\begin{pmatrix} -1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$

- A matrix A is TU if and only if $A^\top$ is TU if and only if (AI) is TU.

What's so cool about TU matrices?

Theorem

Suppose A is TU. Then for any integral vector b, the polyhedron $P(b) = \{x \in \mathbb{R}^n : Ax \leq b,\ x \geq 0\}$ is integral.

- Theorem also holds for $Ax = b$.

Let's prove this.

What's so cool about TU matrices?

Theorem [Converse]

Suppose $P(b) = \{x \in \mathbb{R}^n_+ : Ax \leq b\}$ is integral for all integer vectors b. Then A is TU.

Question:

- Why does this not imply that the constraint matrix of every integral polyhedron is TU?

Do TU matrices exist?

- Ghouila-Houri gives a useful characterization of total unimodularity.
- An equitable bicoloring of a matrix A is a partition of its columns into two sets, red and blue, such that the sum of the red columns minus the sum of the blue columns is a vector whose entries are $0, \pm 1$.

$$\text{Example:} \quad \begin{array}{cc} & \text{red - blue} \\ \begin{pmatrix} -1 & 1 & 1 \\ 1 & 1 & 1 \\ -1 & -1 & 0 \end{pmatrix} & \begin{array}{c} -1 \\ 1 \\ 0 \end{array} \end{array}$$

Equitable bicoloring

Theorem 4.6 (Ghouila-Houri).

A matrix A is TU if and only if every column submatrix of A admits an equitable bicoloring.

We will not prove this.

TU matrix: $\begin{pmatrix} -1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$

$$\begin{pmatrix} -1 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} -1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix}$$

Equitable bicoloring

Theorem 4.6 (Ghouila-Houri).

A matrix A is TU if and only if every column submatrix of A admits an equitable bicoloring.

We will not prove this.

$$\text{non-TU matrix: } \begin{pmatrix} -1 & 1 & 1 \\ 1 & 1 & 1 \\ -1 & -1 & 0 \end{pmatrix}$$

$$\begin{pmatrix} -1 & 1 \\ 1 & 1 \\ -1 & -1 \end{pmatrix}, \quad \begin{pmatrix} -1 & 1 \\ 1 & 1 \\ -1 & 0 \end{pmatrix}, \quad \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ -1 & 0 \end{pmatrix}$$

Equitable bicoloring

Theorem 4.6 (Ghouila-Houri).

A matrix A is TU if and only if every column submatrix of A admits an equitable bicoloring.

We will not prove this.

Because A is TU iff A^T is TU, can alternatively formulate this condition using ever row submatrix admitting an equitable row bicoloring.

A simpler one to test

Theorem [A simpler sufficient condition for TU]

A matrix A is TU if:

(i) $A_{ij} \in \{0, \pm 1\}, \ \forall i, j$

(ii) Every column has at most one $+1$ and at most one -1.

- Follows easily from row version of the equitable bicoloring condition (for any row partition, assign all rows the same color)
- But we will provide a direct proof (since we did not prove equitale bicoloring condition)

Do interesting TU matrices exist?

Minimum cost network flow (MCNF) problem

Given

- Directed graph $G = (V, A)$, Arc capacities: $h_a \; \forall a = ij \in A$, Arc flow costs: $c_a \; \forall a = ij \in A$, Node demands/supplies: $b_i \; \forall i \in V$.

Find a flow that meets all demand at minimum cost.

$$
\begin{aligned}
\min \quad & \sum_{a \in A} c_a x_a \\
\text{s.t.} \quad & \sum_{a \in \delta^+(i)} x_a - \sum_{a \in \delta^-(i)} x_a = b_i, \quad \forall i \in V \\
& 0 \le x_a \le h_a, \quad \forall a \in A.
\end{aligned}
$$

Incidence matrices of digraphs

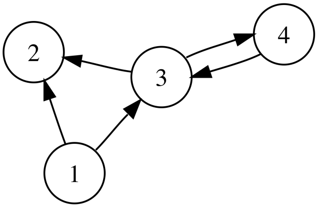
The incidence matrix A_D of a digraph $D = (V, A)$ is the $|V| \times |A|$ matrix with

- rows corresponding to the nodes of D,
- columns corresponding to the arcs of D,

For each arc $a = ij \in A$, the corresponding column has a $+1$ in row i, a -1 in row j, and zeros elsewhere

The incidence matrices of digraphs are those $0, \pm 1$ matrices with exactly one $+1$ and one -1 in each column.

Incidence matrices of digraphs



$$
\begin{array}{c c}
 & \begin{array}{ccccc} 12 & 13 & 32 & 34 & 43 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{pmatrix} -1 & -1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 0 & 1 & -1 \end{pmatrix}
\end{array}
$$

$V = \{1, 2, 3, 4\}$,
$A = \{12, 13, 32, 34, 43\}$.

The constraints of a MCNF problem can then be written in matrix form as:

$$A_D x = b$$

$$Ix \leq h$$

Node-arc incidence matrices

The constraints of a MCNF problem in matrix form:

$$\begin{aligned} A_D x &= b \\ Ix &\leq h \end{aligned}$$

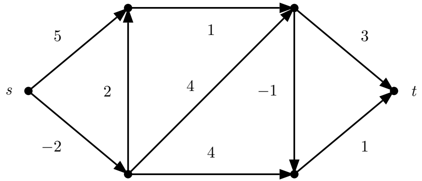
It is then clear that the constraint matrix is TU:

- A_D is TU by sufficient condition $\Rightarrow A_D^T$ is TU $\Rightarrow (A_D^T \; I)$ is TU $\Rightarrow$ Constraint matrix is TU

If b and h are integer, then MCNF has an optimal integer solution whenever it has an optimal solution.

Special case: Shortest Path Problem

- Consider a digraph $D = (V, A)$ with lengths (or costs) ℓ_a on its arcs $a \in A$, and two distinct nodes $s, t \in V$.



- An s, t-path is a sequence of arcs of the form

$$sv_1, v_1v_2, \dots, v_kt.$$

  that traverses each node at most once.
- The length of an s, t-path is $\ell(P) := \sum_{a \in P} \ell_a$.
- The shortest path problem consists in finding an s, t-path P of minimum length $\ell(P)$.

Shortest Path Problem

Let $x_a = 1$ if and only if arc $a \in A$ is in the path

$$\min \sum_{a \in A} \ell_a x_a$$

$$\text{s.t.} \quad \sum_{a \in \delta^+(i)} x_a - \sum_{j \in \delta^-(i)} x_a = \begin{cases} 1 & i = s \\ -1 & i = t \\ 0 & i \in V \setminus \{s, t\} \end{cases}$$

$$x_a \in \{0, 1\}$$

where $\delta^+(i) = \{(i, j) \in A\}$ and $\delta^- = \{(j, i) \in A\}$

This is a MCNF Problem:

- $b(s) = 1,\ b(t) = -1,\ b(i) = 0$ for all $i \in V \setminus \{s, t\}$
- Can take $h_{ij} = 1$, but this is not necessary

How to solve a shortest path problem

The optimal LP solution is integer – so we can solve by linear programming

- Significantly more efficient algorithms exist
- Dijkstra's algorithm (if $\ell_a \geq 0$)
- Bellman-Ford algorithm (general costs, identifies negative weight cycle if exists)

Puzzle

If we have $\ell_a \leq 0$, the shortest path problem becomes the longest path problem, which is "hard". The constraint matrix is still TU, why doesn't this provide a polynomial time algorithm for longest path?

Diversion: Steiner Tree Problem

These ideas can help us find an IP formulation of the Steiner Tree problem:

- Given an undirected graph $G = (V, E)$, edge costs $c_{ij} \geq 0$ for $\{i, j\} \in E$, and a subset of nodes $T \subset V$ that are the *terminals*, find a minimium cost subset of edges such that every terminal is connected to each other.

The Steiner Tree problem is $\mathcal{NP}$-hard in general

- But it reduces to a shortest path problem when $|T| = 2$
- And it reduces to a minimum cost tree problem when $T = V$ (coming later)

Formulating the Steiner Tree Problem

Main decision is to choose which edges to include:

- $x_e = 1$ if $e = \{i, j\} \in E$ is selected and $x_e = 0$ otherwise

Objective is to minimize total cost:

$$\min \quad \sum_{e \in E} c_e x_e$$

How to enforce connectivity?

- Pick any node $r \in T$, and let it be the "source"
- Ensure there is a path from r to k for each $k \in T \setminus \{r\}$

Formulating the Steiner Tree Problem

Directed flow variables

- f_{ak} = flow through *directed* arc $a = ij$ that originated at node r and has destination node k

Send one unit of flow (on a path) from r to k for each $k \in T \setminus \{r\}$

$$\sum_{j \in V^+(i)} f_{ijk} - \sum_{j \in V^-(i)} f_{jik} = \begin{cases} 1 & i = r \\ -1 & i = k \\ 0 & i \in V \setminus \{r, k\} \end{cases}$$

where $V^+(j) = V^-(j) = \{i \in V : \{i, j\} \in E\}$

- This is *a lot* of variables and constraints! $2|E|(|T| - 1)$ vars and $(|T| - 1)|V|$ constraints

Formulating the Steiner Tree Problem

We can only send flow on an arc if the edge is selected:

$$f_{ijk} \leq x_e,\ f_{jik} \leq x_e, \quad \forall e = \{i,j\} \in E, k \in T \setminus \{r\}$$

This can be strengthened: Only send flow in one direction on a path!

$$f_{ijk} + f_{jik} \leq x_e, \quad \forall e = \{i,j\} \in E, k \in T \setminus \{r\}$$

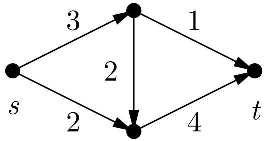
This can be strengthened much more: For *any pair* of destinations h and k in $T \setminus \{r\}$, only send flow in one direction:

$$f_{ijk} + f_{jih} \leq x_e, \quad \forall e = \{i,j\} \in E, k \in T \setminus \{r\}, h \in T \setminus \{r\}$$

Maximum flow problem

- Given a digraph $D = (V, A)$ and two distinct nodes $s, t \in V$, an s, t-flow is a nonnegative vector $x \in \mathbb{R}^A$ such that:
  - the amount of flow entering any node $v \neq s, t$ equals the amount of flow leaving v.
- The quantity x_a, $a \in A$, is called the flow on arc a.
- The amount of flow leaving node s (equals the amount entering node t) is called the value of the s, t-flow:

$$\nu(x) := \sum_{a \in \delta^+(s)} x_a - \sum_{a \in \delta^-(s)} x_a$$

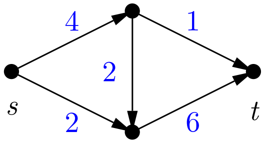


Maximum flow problem

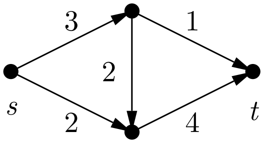
- Given a digraph $D = (V, A)$, capacities $h_a \geq 0$ for $a \in A$, and $s, t \in V$, a feasible s, t flow is an s, t-flow which does not exceed the arc capacities, that is

$$x_a \leq h_a \quad \forall a \in A$$

- The maximum flow problem consists in finding a feasible s, t-flow of maximum value.



Network with capacities



Maximum flow

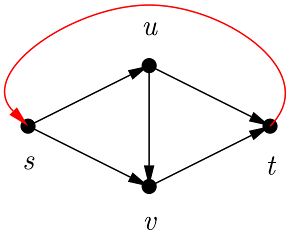
Maximum Flow Formulation

- The maximum flow problem can be formulated as the following linear program

$$
\begin{aligned}
\max \quad & \nu \\
\text{s.t.} \quad & \sum_{a \in \delta^-(v)} x_a - \sum_{a \in \delta^+(v)} x_a = 0 && \forall v \in V \setminus \{s, t\} \\
& \nu + \sum_{a \in \delta^-(s)} x_a - \sum_{a \in \delta^+(s)} x_a = 0 \\
& -\nu + \sum_{a \in \delta^-(t)} x_a - \sum_{a \in \delta^+(t)} x_a = 0 \\
& 0 \le x_a \le h_a && \forall a \in A.
\end{aligned}
\tag{MF}
$$

Maximum Integral Flow

The constraint matrix of the linear program (MF) is the incidence matrix of the digraph D' obtained by adding to D a new arc from t to s, corresponding to the variable ν.



$$A_{D'} = \begin{array}{c} \\ s \\ u \\ v \\ t \end{array} \begin{array}{c} \begin{array}{cccccc} ts & su & sv & uv & ut & vt \end{array} \\ \begin{pmatrix} 1 & -1 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & -1 & 0 \\ 0 & 0 & 1 & 1 & 0 & -1 \\ -1 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \end{array}$$

Maximum Integral Flow

- The LP formulation of the maximum flow problem then can be written as

$$\begin{aligned} \max \quad & \nu \\ \text{s.t.} \quad & A_{D'}(\nu, x) = 0 \qquad\qquad\qquad\qquad\qquad\qquad \text{(MF)} \\ & 0 \leq x_a \leq h_a \qquad\qquad\qquad\qquad \forall a \in A. \end{aligned}$$

- Thus the constraint matrix of (MF) is TU.
- Whenever the capacities h_a, $a \in A$, are all integer numbers, TU implies that the feasible region of (MF) is an integral polyhedron.
- An s,t-flow x is integral if it is an integral vector.
- In particular, finding a maximum integral s,t-flow amounts to solving a linear program.
  Even if the capacities are not integral! Why?

Minimum cut

Definition: s-t cut

Let $D = (V, A)$ be a directed graph, with capacities $h_a \geq 0$ for $a \in A$.

- $C = \delta^+(S)$ is an s-t cut in D if $s \in S$ and $t \in V \setminus S$, where $\delta^+(S) = \{ij \in A : i \in S, j \in V \setminus S\}$
- The capacity of an s-t cut C is $h(C) = \sum_{ij \in C} h_{ij}$
- The min-cut problem is to find a minimum capacity s-t cut

Weak duality

Theorem

Let $D = (V, A)$ have arc capacities h_a for $a \in A$. Let x be a feasible s, t-flow and let $C = \delta^+(S)$ be an s-t cut in D. Then

$$\nu(x) \leq h(C).$$

Why?

- By definition, the arcs in $\delta^+(S)$ disconnect s from t
- So any unit of a feasible flow most go through some arc in $\delta^+(S)$

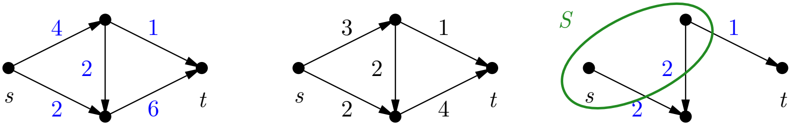
Conclusion: Maximum s-t flow value $\leq$ Minimum s-t cut capacity

The FAMOUS Max flow/Min cut theorem

Theorem [Strong Duality]

Let $D = (V, A)$ have arc capacities h_a for $a \in A$. Suppose the maximum s-t flow is bounded with optimal value z_f^*. Then

$$\max\{\nu(x) : x \text{ is a feasible } s,t\text{-flow}\} = \min\{h(C) : C \text{ is an } s,t\text{-cut}\}.$$



Network with capacities　　Maximum flow　　Minimum cut

Solving Max Flow Problems

- Again, using LP is not the best way!
- Basic algorithms aren't too bad, but we won't cover them here (see ISyE/CS/Math 425)
- Complexity of some algrithms ($n = |V|$, $m = |A|$): $O(n^3)$, $O(nm \log n)$, $O(mn^{3/4} \log(n^2/m) \log h_{\max})$
- Very impressive algorithms exist for graphs with special structure (can solve instances with *millions* of edges)

A different kind of graph cut

Global cuts: (No specific s and t)

- A cut in a *undirected* graph $G = (V, E)$ is a set of edges $\delta(S) = \{e = \{i, j\} : \text{exactly one of } \ i, j \in S\}$, where $S \subseteq V$.
- The capacity of a cut $C = \delta(S)$ is

$$h(C) = \sum_{e \in C} h_e$$

Global minimum cut problem

Given an undirected graph $G = (V, E)$, find the global minimum cut $C = \delta(S)$ with minimum capacity.

- How can we find a global minimum cut using an algorithm for finding minimum $s - t$ cuts?

Application: Edge connectivity of a graph

- Consider an undirected graph $G = (V, E)$ and set $h_e = 1$ for all $e \in E$
- Min capacity cut: minimum number of edges that would need to be removed to disconnect the graph
  - This number is called the *edge connectivity*
  - Measures robustness of the network
  - Tree: Edge connectivity $= 1$
  - Cycle: Edge connectivity $= 2$

Application: Separation of TSP subtour inequalities

Inequalities for TSP on directed graph $G = (V, A)$:

$$\sum_{a \in \delta^+(i)} x_a = 1, \quad i \in V \tag{1}$$

$$\sum_{a \in \delta^-(i)} x_a = 1, \quad i \in V \tag{2}$$

$$\sum_{a \in \delta^+(S)} x_a \geq 1, \quad \forall \emptyset \subset S \subset V \tag{3}$$

$$x_a \geq 0, \quad \forall a \in A \tag{4}$$

- Let P be the polyhedron defined by (1) - (4).
- Given an x^*, can we solve the separation problem over P?
  - Checking (1), (2), and (4) is no problem
  - How to check (3)?

Separation of TSP subtour inequalities, cont'd

We may assume x^* satisfies $x^* \geq 0$ and:

$$\sum_{a \in \delta^+(i)} x_a^* = 1, \quad i \in V$$
$$\sum_{a \in \delta^-(i)} x_a^* = 1, \quad i \in V$$

We want to check:

$$\sum_{a \in \delta^+(S)} x_a^* \geq 1 \quad \forall \emptyset \subset S \subset V$$

True if and only if: $\min\{\sum_{a \in \delta^+(S)} x_a^* : S \subset V, S \neq \emptyset\} \geq 1$
Take $h_a = x_a^*$, this is a (directed) global minimum cut problem.

Graph search as an efficient min-cut heuristic

Let $A^* = \{a : x^*_a > \epsilon\}$, where $\epsilon \in (0,1)$

For any $r \in V$, conduct a graph search from r, using only arcs in A^*:

$\text{visited}[r] \leftarrow 1$, $\text{visited}[i] \leftarrow 0$ for all $i \in V \setminus \{r\}$, $\text{OPEN} \leftarrow \{r\}$;
while $\text{OPEN} \neq \emptyset$ **do**
  Choose $i \in \text{OPEN}$;
  $\text{OPEN} \leftarrow \text{OPEN} \setminus \{i\}$;
  for $a = ij \in \delta^+(i)$ **do**
    if $\text{visited}[j] == 0$ **then**
      $\text{OPEN} \leftarrow \text{OPEN} \cup \{j\}$, $\text{visited}[j] \leftarrow 1$;
    end
  end
end

Result: $\text{visited}[i] = 1$ if and only if i is reachable from r using arcs in A^*

Graph search as an efficient min-cut heuristic

Let $A^* = \{a : x^*_a > \epsilon\}$

Conduct a graph search from some node r, using only arcs in A^*:

- $\text{visited}[i] = 1 \Leftrightarrow i$ is reachable from r using arcs in A^*

If $\text{visited}[k] = 0$ for some $k \in V \setminus r$:

- Let $S = \{i \in V : \text{visited}[i] = 1\}$
- No arcs in A^* are in $\delta^+(S)$:

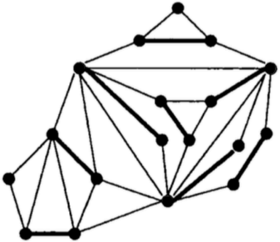
$$\sum_{a \in \delta^+(S)} x^*_a \leq |\delta^+(S)|\epsilon$$

If $x^* \in \{0,1\}^{|A|}$, this algorithm *always* finds a violated inequality if one exists

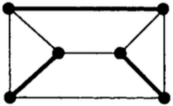
- Otherwise, it is a heuristic: different values of ϵ yield different "candidate cuts"

Matchings in Graphs

- A matching in an undirected graph $G = (V, E)$ is a set $M \subseteq E$ of pairwise disjoint edges.
- A matching is perfect if it covers every node of the graph (that is, every node is contained in exactly one edge of the matching).



A matching



A perfect matching

Matchings in Graphs

Some basic matching problems:

- Maximum cardinality matching problem:
  Finding a matching of maximum cardinality.
- Perfect matching problem:
  Understanding if there exists a perfect matching.
- Minimum weight perfect matching problem:
  Given also weights w_e, $\forall e \in E$, determining a perfect matching M of minimum total weight

$$w(M) := \sum_{e \in M} w_e.$$

Reasons the matching problem is interesting

It has many applications

It can be solved efficiently, but is close to the border of hard problems

- The natural IP formulation is not integral in general, but is on bipartite graphs
- But, an exponential class of valid inequalities is known which defines the convex hull
- There can be *no* compact formulation (even using additional variables) for the convex hull

Matching applications

NASA: Assignment of projects to space shuttle simulators

- Nodes: each project
- Edges: between projects that can go in parallel
- Maximum matching yields minimum wasted simulator time

Drug testing:

- Half of test population receives drug, half receives placebo
- Want populations receiving drug/placebo to be similar
- Nodes: People
- Edge weights: Measure of difference between people
- Find minimum weight perfect matching
- Give drug to one person in each matching

Personnel assignment:

- Bipartite: assign people to jobs
- Nonbipartite: pair compatible people up

Matchings in Graphs Formulation

Variables:

- Binary variables $x_e,\ e \in E$, where

$$x_e = 1 \qquad \Leftrightarrow \qquad e \in M.$$

Constraints:

- A binary vector x is the incidence vector of a matching if and only if it satisfies the degree constraints

$$\sum_{e \in \delta(v)} x_e \leq 1 \qquad \forall v \in V.$$

Matchings in Graphs Formulation

Variables:

- Binary variables x_e, $e \in E$, where

$$x_e = 1 \qquad \Leftrightarrow \qquad e \in M.$$

Constraints:

- A binary vector x is the incidence vector of a matching if and only if it satisfies the degree constraints

$$\sum_{e \in \delta(v)} x_e \leq 1 \qquad \forall v \in V.$$

- A binary vector x is the incidence vector of a perfect matching if and only if it satisfies the degree constraints at equality

$$\sum_{e \in \delta(v)} x_e = 1 \qquad \forall v \in V.$$

Incidence matrices of graphs

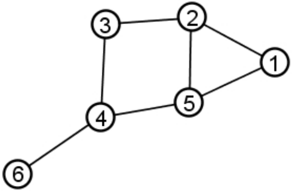
These systems of constraints can be written in terms of the incidence matrix A_G.

The incidence matrix A_G of a graph $G = (V, E)$ is the $|V| \times |E|$ matrix with

- rows corresponding to the nodes of G,
- columns corresponding to the edges of G,
- entries for every node w and edge $e = \{u, v\}$ equal to 1 if $w \in \{u, v\}$ and equal to 0 otherwise

The incidence matrices of graphs are those $0, 1$ matrices with exactly two 1 in each column.

Incidence matrices of graphs



$V = \{1, 2, 3, 4, 5, 6\}$,
$E = \{12, 15, 23, 25, 34, 45, 46\}$.

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array}
\begin{array}{c}
\begin{array}{ccccccc} 12 & 15 & 23 & 25 & 34 & 45 & 46 \end{array} \\
\begin{pmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}
\end{array}$$

- In the row corresponding to node w, there is a 1 for each edge containing w.

Matchings in Graphs Formulation

- A vector x is the incidence vector of a matching if and only if it satisfies

$$\begin{aligned} A_G x &\leq 1 \\ x &\in \{0,1\}^E. \end{aligned}$$

- A vector x is the incidence vector of a perfect matching if and only if it satisfies

$$\begin{aligned} A_G x &= 1 \\ x &\in \{0,1\}^E. \end{aligned}$$

Matchings in Graphs Formulation

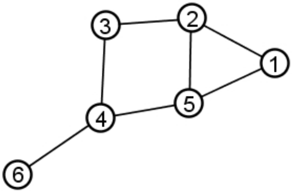
- A vector x is the incidence vector of a matching if and only if it satisfies
$$A_G x \leq 1$$
$$x \in \{0,1\}^E.$$
- A vector x is the incidence vector of a perfect matching if and only if it satisfies
$$A_G x = 1$$
$$x \in \{0,1\}^E.$$
- The matching polytope of G is the convex hull of all incidence vectors of matchings of G.
- The perfect matching polytope of G is the convex hull of incidence vectors of perfect matchings in G.

Maximum Cardinality Matching

- Thus the maximum cardinality matching problem can be formulated as the integer program

$$\begin{aligned} \max \quad & \sum_{e \in E} x_e \\ \text{s.t.} \quad & \sum_{e \in \delta(v)} x_e \le 1, \quad \forall v \in V \\ & x \in \{0,1\}^E. \end{aligned} \tag{MCM}$$

Incidence matrices of graphs



$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{ccccccc} 12 & 15 & 23 & 25 & 34 & 45 & 46 \end{array} \\ \left(\begin{array}{ccccccc} 1 & 1 & 0 & 0 & 0 & 0 & 01 \\ 1 & 0 & 1 & 1 & 0 & 0 & 01 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 00 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array} \right. \end{array} \begin{array}{c} \\ 1 \\ 0 \\ \\ \\ 1 \\ \\ \end{array}$$

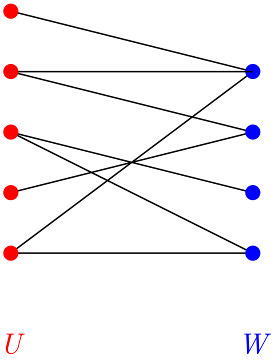
$V = \{1, 2, 3, 4, 5, 6\}$,
$E = \{12, 15, 23, 25, 34, 45, 46\}$.

- Question: Is this matrix TU?

Bipartite Graphs

A graph $G = (V, E)$ is bipartite if there exists a bipartition U, W of V such that each edge $e \in E$ has one end in U and one end in W.

Example:



Bipartite Graphs

Theorem 4.18.

Let A_G be the incidence matrix of a graph G. Then A_G is TU if and only if G is bipartite.

- Since A_G has two nonzero entries in each column, A_G is TU if and only if it has an equitable row-bicoloring.
- An equitable row-bicoloring of A_G corresponds to a bipartition of the nodes of G such that each edge has an endnode in each side of the bipartition.
- Such a bicoloring exists if and only if G is bipartite.

Bipartite Matching

- If G is bipartite, the polytopes

$$\{x \in \mathbb{R}^E : A_G x \leq 1,\ x \geq 0\},$$
$$\{x \in \mathbb{R}^E : A_G x = 1,\ x \geq 0\}$$

are integral. Thus we have the following.

Corollary 4.19.

If G is a bipartite graph, then the matching polytope of G is the set

$$\{x \in \mathbb{R}^E : A_G x \leq 1,\ x \geq 0\},$$

and the perfect matching polytope of G is the set

$$\{x \in \mathbb{R}^E : A_G x = 1,\ x \geq 0\}.$$

Question: What about $x \leq 1$?

Bipartite Matching

Maximum cardinality matching problem:

$$\begin{aligned} \max \quad & \sum_{e \in E} x_e \\ \text{s.t.} \quad & A_G x \leq 1 \\ & x \in \{0,1\}^E. \end{aligned}$$

- If G is bipartite, it can be solved by the linear program

$$\begin{aligned} \max \quad & \sum_{e \in E} x_e \\ \text{s.t.} \quad & A_G x \leq 1 \\ & x \geq 0. \end{aligned}$$

Bipartite Matching

Minimum weight perfect matching problem:

$$\begin{aligned} \min \quad & \sum_{e \in E} w_e x_e \\ \text{s.t.} \quad & A_G x = 1 \\ & x \in \{0,1\}^E. \end{aligned}$$

- If G is bipartite it can be solved by the linear program

$$\begin{aligned} \min \quad & \sum_{e \in E} w_e x_e \\ \text{s.t.} \quad & A_G x = 1 \\ & x \geq 0. \end{aligned}$$

- The minimum weight perfect matching problem in a bipartite graph is also known as the assignment problem.

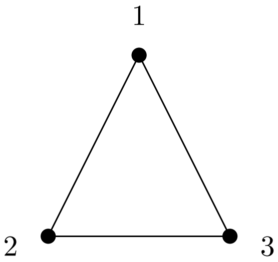
The Matching Polytope

- Corollary 4.19 states that, whenever G is bipartite, its matching polytope is the set

$$\{x \in \mathbb{R}^E : A_G x \leq 1,\ x \geq 0\}.$$

- This statement does not carry through in the general case.

The Matching Polytope



- Suppose G is a triangle.
- The system formed by nonnegativity and degree constraints is

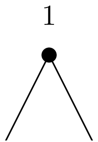
$$
\begin{aligned}
x_{12} + x_{13} \qquad\quad &\leq 1 \\
x_{12} \qquad\quad + x_{23} &\leq 1 \\
+\, x_{13} + x_{23} &\leq 1 \\
x &\geq 0.
\end{aligned}
$$

- Point $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ is a vertex of this polytope.
- Thus these are not sufficient to

The Matching Polytope

- Suppose G is a triangle.
- The system formed by nonnegativity and degree constraints is

$$
\begin{aligned}
x_{12} + x_{13} \qquad\quad &\leq 1\\
x_{12} \qquad\quad + x_{23} &\leq 1\\
+ x_{13} + x_{23} &\leq 1\\
x &\geq 0.
\end{aligned}
$$



- Point $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ is a vertex of this polytope.
- Thus these are not sufficient to

The Matching Polytope

Blossom inequalities

$$\sum_{e \in E(U)} x_e \leq \frac{|U| - 1}{2} \qquad \forall U \subseteq V, \ |U| \text{ odd.} \qquad \text{(BI)}$$

where $E(U) := \{uv \in E : u, v \in U\}$.

To see that they are valid for the matching polytope:

- Every edge in $M \cap E(U)$ covers two distinct nodes in $E(U)$.
$$\Rightarrow \quad 2|M \cap E(U)| \leq |U| \quad \Rightarrow \quad |M \cap E(U)| \leq \left\lfloor \frac{|U|}{2} \right\rfloor.$$
- Thus the incidence vector x of any matching satisfies
$$\sum_{e \in E(U)} x_e \leq \left\lfloor \frac{|U|}{2} \right\rfloor.$$

The Matching Polytope

Blossom inequalities

$$\sum_{e \in E(U)} x_e \leq \frac{|U| - 1}{2} \qquad \forall U \subseteq V,\ |U| \text{ odd.} \tag{BI}$$

where $E(U) := \{uv \in E : u, v \in U\}$.

To see that they are valid for the matching polytope:

- Every edge in $M \cap E(U)$ covers two distinct nodes in $E(U)$.
$$\Rightarrow \quad 2|M \cap E(U)| \leq |U| \quad \Rightarrow \quad |M \cap E(U)| \leq \left\lfloor \frac{|U|}{2} \right\rfloor.$$
- Thus the incidence vector x of any matching satisfies
$$\sum_{e \in E(U)} x_e \leq \left\lfloor \frac{|U|}{2} \right\rfloor.$$

The Matching Polytope

- Edmonds showed that, in fact, adding the blossom inequalities to the nonnegativity and degree constraints is always sufficient to describe the matching polytope.

Matching Polytope Theorem

The matching polytope of a graph $G = (V, E)$ is the set

$$\{x \in \mathbb{R}^E : x \geq 0,\ A_G x \leq 1,\ x \text{ satisfies (BI)}\}.$$

We will not prove this.