CO 450/650 with Chaitanya Swamy

Eason Li

2025 F

Contents

1 Recap of Linear Programming (LP) Theory

Lecture 2 - Tuesday, September 09

Definition 1.1: Linear Program

A **linear peogram** (LP) is a problem of the following form:

$$\max c^T x \ \ s.t. \quad \begin{aligned} Ax &\leq b \\ x &\geq 0 \end{aligned} \tag{P}$$

where $A \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^n$, and $b \in \mathbb{R}^m$.

There are three things we care in a linear program:

- Linear objective;
- Finite number of linear constraints;
- All constraints have non-strict inequalities.

Any linear program has one of the three possibilities: infeasible, unbounded, attains an optimal solution.

1.1 (LP) Duality

Question 1.1. How can we give bounds on the optimal value of (P)?

Using the dual LP, which is an LP that expresses the problem of finding the *best* bound on optimal value of (P).

The dual to (P) is the following:

$$\min b^T y \ \ s.t. \quad \begin{aligned} A^T y &\geq c \\ y &\geq 0 \end{aligned} \tag{D}$$

Comment 1.1

The dual is essentially a linear combinations of the constraints to recreate the objective function, so we can attain a meaningul upper bound for the objective value of the primal by minimize the new objective function.

Exercise 1.1

Take the dual of the dual LP (D), verify that this is (P).

1.1.1 Weak Duality

Theorem 1.1: Weak Duality

If x is feasible to (P) and y is feasible to (D), then

$$c^T x \leq b^T y$$

Proof. We simply have

$$c^T x \leq y^T Ax \leq y^T(Ax) \leq y^T b$$

as desired. □

1.1.2 Strong Duality

Theorem 1.2: Strong Duality

(P) has optimal solution if and only if (D) has optimal solution. Moreover, x^* and y^* are optimal solutions to (P) and (D) if and only if x^* and y^* are feasible solutions to (P) and (D) and

$$c^T x = b^T y$$

1.1.3 Complementary Slackness

It is often convenient to rephrase the Strong Duality as follows:

Theorem 1.3: Complementary Slackness

x^* and y^* are optimal solutions to (P) and (D) if and only if x^* and y^* are feasible solutions to (P) and (D), and

- $x_j^* \neq 0$ implies the corresponding dual constraint is tight at y^*. i.e., $(A^T y^*)_j = c_j$;
- $y_i^* \neq 0$ implies the corresponding dual constraint is tight at x^*.

Comment 1.2

LP's can be solved efficiently, i.e., there is a polytime algorithm such that given an LP (P), it determines the outcome of (P):

- if (P) has an optimal solution, it returns the optimal solution;
- if (P) is infeasible or unbounded, it returns a certificate of that outcome.

In practice, we use the Simplex method, however, there exists LP such that the Simplex method runs in exponential time. The first polytime algorithm for LP is the ellipsoidal method, which is hopeless to implement in practice. Nowadays, there is another method knows is the interior point method, which pratically runs as fast as the Simplex method.

1.2 Geometry of LP

Feasible region of an LP is called a polyhedron. So, $P \subseteq \mathbb{R}^n$ is a polyhedron if it can be expressed as $\{x \in \mathbb{R}^n : Ax \leq b\}$ for some matrix A (which has a finite number of rows) and vector b.

Discovery 1.1

A polyhedron is a convex set.

Definition 1.2: Convex Combination

We say $x \in \mathbb{R}^n$ is a **convex combination** of points $p^{(1)}, \dots, p^{(k)} \in \mathbb{R}^n$ if there exists $\lambda_1, \dots, \lambda_k \in \mathbb{R}_+$ such that

$$\sum_{i=1}^{k} \lambda_i = 1 \qquad \text{and} \qquad x = \sum_{i=1}^{n} \lambda_i p^{(i)}$$

Definition 1.3: Extreme Point

An **extreme point** of a convex set $S \subseteq \mathbb{R}^n$ is a point $\hat{x} \in S$ such that we cannot write $\hat{x}$ as a convex combination of two distinct points in S.

Discovery 1.2

A polyhedron has a finite number (maybe 0) of extreme points.

Comment 1.3

For the extreme points in polyhedron P, we can also give an algebraic characterization of the extreme points, as well as an optimization characterization.

Theorem 1.4

Let $P \subseteq \mathbb{R}^n$ be a polyhedron, $\hat{x} \in P$ is an extreme point of P if and only if there exists $c \in \mathbb{R}^n$ such that $\hat{x}$ is the *unique optimal solution* to

$$\max c^T x \quad s.t. \quad x \in P$$

Theorem 1.5

Consider the LP

$$\max c^T x \quad s.t. \quad x \in P \tag{LP}$$

where $P \subseteq \mathbb{R}^n$ is a polyhedron. If (LP) has an optimal solution, and P has an extreme points, then there is always an optimal solution to (LP) that is an extreme point of P.

Comment 1.4

An optimal solution isn't necessarily an extreme point, consider:

$$\max 0^T x \ \ s.t. \quad 0^T x \leq 0$$

Definition 1.4: Convex Hull

Let $S \subseteq \mathbb{R}^n$. The **convex hull** of S is the smallest convex set containing S.

Exercise 1.2

Show that the following

$$\text{conv}(S) = \{x \in \mathbb{R}^n : x \text{ is the convex combination of finite number of points in } S\}$$

is equivalent to the convex hull of S.

Definition 1.5: Polytope

A **polytope** is a bounded polyhedron.

Comment 1.5

1. A polytope is the convex hull of its extreme points;
2. $P \subseteq \mathbb{R}^n$ is a polytope if and only if $P = \text{conv}(S)$ for some finite set S.

2 Minimum Spanning Tree (MST)

Lecture 3 - Thursday, September 11

Definition 2.1: Spanning Tree

T is a **spanning tree** of $G = (V, E)$ if T is a connected, acyclic subgraph of G with vertex set V.

Note 2.1

If T is a spanning tree, then

1. T is minimally connected. i.e., $T - e$ is not connected for any $e \in T$.
2. T is maximally acyclic. i.e., $T + f$ contains a cycle for any $f \notin T$.

Lemma 2.1

Let $|V| = n$, TFAE:

1. T is a spanning tree;
2. T is connected with $n - 1$ edges;
3. T is acyclic with $n - 1$ edges.

Definition 2.2: Minimum Spanning Tree

Minimum spanning tree (MST) problem: Given connected, undirected graph $G = (V, E)$ with edge costs $\{c_e\}_{e \in E}$, find a spanning tree T of minimum total cost $c(T) := \sum_{e \in T} c_e$.

2.1 Kruskal's Algorithm (Greedy Algorithm)

Algorithm 2.1: Kruskal's Algorithm

1. Enumerate the edges in increasing order of costs;
2. Initialize $H = (V, \emptyset)$;
3. For each edge $e = uv$ in sorted order, if $H \cup \{e\}$ is acyclic, update $H \leftarrow H \cup \{e\}$;
4. Return H.

Comment 2.1

Notations: For $A \subseteq V$, $\delta(A)$ = set of edges on boundary of A. i.e., the set $\{uv \in E : u \in A, v \notin A\}$. An edge set of the form $\delta(A)$ for some $\emptyset \neq A \subsetneq V$ is called a cutset of G.

Lemma 2.2

Let T be a spanning tree, then T is a MST if and only if for any $e = uv \in E - T$, all edges f on the unique u–v path in T have $c_f \leq c_e$.

Proof. [$\Longrightarrow$]: Denote the unique u–v path in T as T_{uv}. SFAC that there exists $f \in T$ such that $c_f > c_e$. Consider $T' := T - f + e$, which is a spanning tree (connected, $n-1$ edges) with lower cost $\lightning$.
[$\Longleftarrow$]: Among all MSTs, choose an MST T^* that minimizes. Define k to be

$$k := |E(T) \cap E(T^*)|$$

We wish to show that $k = n-1$. SFAC that $k < n-1$, i.e., $T \neq T^*$. Consider some $e = uv \in T^* - T$. Let T_1^*, T_2^* be the components of $(V, T^* - \{e\})$. We know that there exists a different u–v path in T, and so there exists $f \in T_{uv}$ such that $f \in \delta(V(T_1^*))$. We know that $c_f \leq c_e$. Consider $T' = T^* - e + f$, which is a spanning tree with

$$c(T') \leq c(T^*)$$

which implies that T' is also a MST. Notice, $|E(T') \cap E(T)| = n - 1$. $\square$

Theorem 2.1

Kruskal's Algorithm produces a MST.

Proof. Let $H \leftarrow$ output of Kruskal. It is easy to see that H is a spanning tree: H is acyclic by construction, and H is connected because otherwise there exists $e \in G - H$ that connects the two components of H. STP that H has the property mentioned in the above lemma. Consider $e = uv \in E - H$. Since Kruskal does not pick e, u and v are in the same component of H when e is considered by Kruskal, and this further implies that all edges on the u–v path in H have cost $\leq c_e$ due to Kruskal's order of considering edges. $\square$

2.1.1 Running Time of Kruskal's Algorithm

Denote $m =: |E|$ and $n =: |V|$.

1. Sorting m numbers: $O(m \log m) = O(m \log n)$;
2. Checking if $e = uv$ is in some component can be done efficiently, $\approx O(\log n)$

   So the running time of the algorithm is $O(m \log n)$.

2.2 MST Polytope

How can we write a linear program that represents the MST problem?

2.2.1 LP-relaxation for MST Problem

We introduce one variable for each $e \in E$ (representing whether $e \in MST$), think of this as a characteristic vector.

Comment 2.2

Notation: For $S \subseteq V$, denote $E(S) = \{e \subseteq S : e \in E\}$. For $F \subseteq E$, denote $x(F) = \sum_{e \in F} x_e$ where x^F is the characteristic vector of F, which is the vector in $\mathbb{R}^E$ whose e^{th} coordinate is 1 if $e \in F$ and 0 otherwise.

Since a spanning tree has exactly $n-1$ edges, we have

$$x(E) = n - 1$$

(Recall that x is a characteristic vector of a spanning tree of (V, E)). If T is a spanning tree, then $(S, T \cap E(S))$ is acyclic, so

$$|T \cap E(S)| \le |S| - 1 \quad \forall\, S \subseteq V \qquad \equiv \qquad x(E(S)) \le |S| - 1 \quad \forall\, \emptyset \ne S \subsetneq V$$

Definition 2.3: LP-relaxation of MST Problem

The LP-relaxation of the MST Problem is

$$\min \sum_{e \in E} c_e x_e \ \ s.t. \ \ \begin{aligned} x(E) &= n-1 \\ x(E(S)) &\le |S| - 1 \qquad \forall\, \emptyset \ne S \subsetneq V \\ x &\ge 0 \end{aligned} \tag{MST-P}$$

Discovery 2.1

Let $F \subseteq E$, then F is a spanning tree if and only if x^F is a feasible solution to (MST-P).

Theorem 2.2

The tree H output by Kruskal is such that x^H is an optimal solution to (MST-P).

Corollary 2.1

Let $P =$ feasible region of (MST-P), then

$$P = \text{conv}(\{x^T : T \text{ is a spanning tree of } G\})$$

which is called the *MST polytope*.

Proof. We first call

$$Q := \text{conv}(\{x^T : T \text{ is a spanning tree of } G\})$$

[$P \supseteq Q$]: $x^T \in P$ for every spanning tree T implies that $Q \subseteq P$.
[$P \subseteq Q$]: $P = \text{conv}(\text{ext pts of } P)$. Hence STP that every extreme points of P is in Q. Let $\hat{x}$ be an extreme point of P. By theorem 1.4 there exists an objective function $\{c_e\}_{e \in E}$ such that $\hat{x}$ is the unique optimal solution to (MST-P). But Kruskal runs on input $\{c_e\}_{e \in E}$ produces a spanning tree H such that x^H is the optimal solution to (MST-P) (theorem 2.2), so $\hat{x} = x^H$, which implies that every extreme point is of the form x^T for some spanning tree T as desired. □

2.2.2 Dual of (MST-P)

Recall the (MST-P):

$$\min \sum_{e\in E} c_e x_e \quad s.t. \quad \begin{array}{rlll} x(E) & = n-1 & &] \times y_V, y_V \text{ free} \\ x(E(S)) & \le |S|-1 & \forall\, \emptyset \neq S \subsetneq V &] \times (-y_S), y_S \ge 0 \\ x & \ge 0 & & \end{array} \tag{MST-P}$$

The sum of the two products yields us

$$\sum_{e\in E} x_e \left(y_V - \sum_{\emptyset\neq S\subsetneq V, e\in E(S)} y_S \right) \ge (n-1)y_V - \sum_{S:\emptyset\neq S\subsetneq V} (|S|-1)y_S$$

Hence the dual is given as

$$\max\ (n-1)y_V - \sum_{S:\emptyset\neq S\subsetneq V} (|S|-1)y_S \quad s.t. \quad \begin{array}{rll} y_S & \ge 0 & \forall\, \emptyset\neq S\subsetneq V \\ y_V - \sum_{S:\emptyset\neq S\subsetneq V, e\in E(S)} y_S & \le c_e & \forall\, e \end{array} \tag{MST-D}$$

Lecture 4 - Thursday, September 18

2.2.3 Why Kruskal's Algorithm Yields us Optimal Solution to (MST-P)

Proof of Theorem 2.2. We first define some notations to simplify our proof. Let $e_1, e_2, \ldots, e_{n-1}$ be the edges of T^* in increasing order of c_e (with ties broken as in Kruskal). Let $c_i := c_{e_i}$, and let $c_0 := \min_{e\in E} c_e$ and let $F_i := \{e_1, \ldots e_i\}$ (so $F_0 = \emptyset$). For $F \subseteq E$, let $\kappa(F) = \#$ of components of (V, F).

Note 2.2

- $\kappa(F_i) = n - i$ since $\kappa(K_0) = n$ and each added edge decrements the number of components by 1.
- If F is all the edges considered by Kruskal up to and including e_i, then components of $(V, F) =$ components of (V, F_i).

Claim: $c(T^*) = c_0(n-1) + \sum_{i=1}^{n-1}(c_i - c_{i-1}) \cdot \kappa(F_i)$.

To see the reason why the above equality holds, let's expand it out:

$$\begin{aligned} c_0(n-1) + \sum_{i=1}^{n-1}(c_i - c_{i-1}) \cdot \kappa(F_i) &= c_0(n-1) + (c_1 - c_0)(n-1) + \cdots + (c_{n-1} - c_{n-2}) \cdot 1 \\ &= c_1 + c_2 + \cdots + c_{n-1} \end{aligned}$$

The expression on the right can be written as

$$\begin{aligned}
c_0(n-1) + \sum_{i=1}^{n-1}(c_i - c_{i-1}) \cdot \kappa(F_i) &= c_0(n-1) + \sum_{i=1}^{n-1}(c_i - c_{i-1}) \cdot (\kappa(F_{i-1}) - 1) \\
&= c_0(n-1) + \sum_{i=1}^{n-1}(c_i - c_{i-1}) \left(n - 1 - \sum_{S : S \in \text{a comp. of } (V, F_{i-1})} (|S| - 1) \right) \\
&= (n-1) \left[c_0 + \sum_{i=1}^{n-1}(c_i - c_{i-1}) \right] \\
&\quad + \sum_{\emptyset \neq S \subsetneq V} (|S| - 1) \sum_{\substack{i \in \{1, \ldots, n-1\} \\ S \text{ in a comp. of } (V, F_{i-1})}} (c_i - c_{i-1})
\end{aligned}$$

Defining

$$y_V^* := c_0 + \sum_{i=1}^{n-1}(c_i - c_{i-1}) = c_{n-1}$$

and

$$y_S^* = \sum_{\substack{i \in \{1, \ldots, n-1\} \\ S \text{ in a comp. of } (V, F_{i-1})}} (c_i - c_{i-1}) \qquad \forall\, \emptyset \neq S \subsetneq V$$

we get $c(T^*) =$ dual objective value of y^*.

Note 2.3

If $\emptyset \neq S \subsetneq V$ is such that S is not a component of (V, F_{i-1}) for any $i \in \{1, \ldots, n-1\}$, then $y_S^* = 0$.

If we show y^* is dual feasible, then we have shown that x^{T^*}, y^* are optimal for (MST-P) and (MST-D) respectively. Hence we now want to show y^*'s feasibility.

`[Proof one, Primal-Dual interpretation of Kruskal, which will lead to` y^*`]:`

1. *Initialize*: $F_0 \leftarrow \emptyset, i \leftarrow 1$ AND $y_V^* = t := c_0 = \min_{e \in E} c_e$, where we call t as "time", $y_S^* \leftarrow 0$ for all other sets S.

2. *Maintain Invariant*: For all $e \in \delta(S)$, where S is a component of (V, F_{i-1}), LHS of the second constraint of (MST-D) $= t \leq c_e$.

3. *Update*: Let e_i be the minimum cost edge on boundary of some component of (V, E_{i-1}), we update the following:

   - $F_i \leftarrow F_{i-1} \cup \{e_i\}$;
   - $y_V^* \leftarrow y_V^* + (c_{e_i} - t)$;
   - $y_S^* \leftarrow y_S^* + (c_{e_i} - t)$ for all S : component of (V, F_{i-1});
   - $t \leftarrow t + (c_{e_i} - t) = c_{e_i}$.

4. $i \leftarrow i + 1$.

CONTINUE AS ABOVE until we have a spanning tree. i.e., (V, F_i) is connected.

Exercise 2.1

Verify that the y^* at the end of the above process is precisely the $y^* = (y^*_V, y^*_S)$ we defined earlier.

`[Proof two, explicitly verifying dual feasibility]`: Consider an edge e, let i be the first iteration j that e is internal to a component of (V, F_j). Consider the sets S with $y^*_S \neq 0$ that contain e:

- V, components of (V, F_i), $(V, F_{i+1}), \ldots, (V, F_{n-1})$ that contain e.

Hence the LHS of the second constraint of (MST-D) for e is

$$y^*_V - \sum_{j=i}^{n-2} (c_{j+1} - c_j) = c_{n-1} - \sum_{j=i}^{n-2} (c_{j+1} - c_j) = c_i = c_{e_i}$$

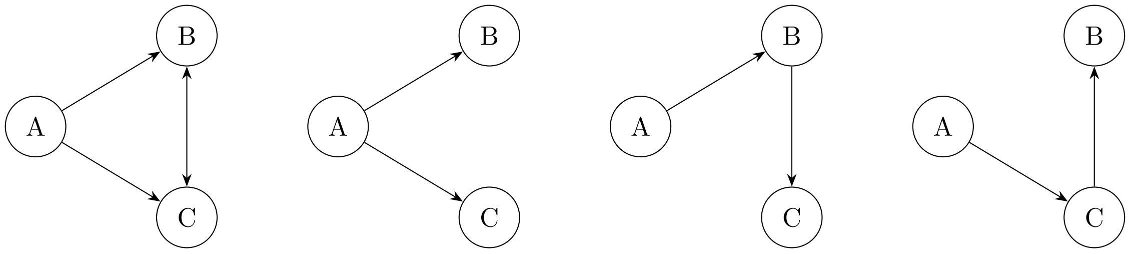
which meets the constraint $c_{e_i} \leq c_e$ since e and e_i are on the boundary of some components of (V, F_ℓ) precisely for $\ell = 1, \ldots, i-1$ and e_i was considered by Kruskal's algorithm prior ro e. This proves y^*'s dual feasibility. $\square$

3 Matroids and Greedy Algorithm

Question 3.1.

The question we want to ask is "When does a greedy algorithm like Kruskal's algorithm work"?

1. `Max-weight Matching`.
2. `Min-cost Arborescence`: Here is an example: For the following directed graph (leftmost), it has precisely three arborescences, drawn to the right of it:



Despite the fact that this problem is similar to the Minimum Spanning Tree problem, greedy algorithm does not work at all.

Lecture 5 - Tuesday, September 23

Definition 3.1: Independence System

Let U be a finite set and $\mathcal{I} \subseteq 2^U$ a collection of subsets of U. We say that (U, I) is an **independence system** if

(P0) $\emptyset \in I$;

(P1) If $A \in I$ and $B \subseteq A$, then $B \in I$.

Sets in I are called **independent sets**.

3.1 Max-weight Independent Set (MWIS) Problem

The problem asks that, given weights $\{w_e\}_{e \in U}$, find a set $S \in I$ that has the maximum total weights, where

$$w(S) = \sum_{s \in S} w_s$$

3.1.1 Greedy Algorithm for MWIS

```
1 Start with $J \leftarrow \emptyset$ ;
2 while $\exists e \notin J$ such that $w_e > 0$ and $J \cup \{e\} \in \mathcal{I}$ do
3     Choose such an $e$ with maximum $w_e$ ;
4     Set $J \leftarrow J \cup \{e\}$ ;
5 Return $J$ ;
```

Algorithm 1: Greedy Algorithm for MWIS

Comment 3.1

Greedy Algorithm does not always return a max-weight independent set for an independence system.

3.2 Matroid

Definition 3.2: Matroid

The tuple $(U, \mathcal{I})$ is a **matroid** if it is an independence system and (P2) for every $A \subseteq U$, every inclusion-wise *maximal independent set* contained in A has the same size.

Definition 3.3: Basis

For $A \subseteq U$, a maximal independent set contained in A is called a **basis of** A. If $A = U$, we will just call it **basis**.

Definition 3.4: Rank

We define the **rank** function $r : 2^U \to \mathbb{Z}_+$ as

$$r(A) := \max\{|B| : B \subseteq A, B \in \mathcal{I}\}$$

which is the max-size of independent set in A.

3.2.1 Matroids Examples

Example 3.1: Forests of a graph, aka Graphic Matroid or Cyclic Matroid

Given a graph $G = (V, E)$, we take $U = E$ and $\mathcal{I} = \{F \subseteq E : F \text{ is acyclic}\}$. We claim that $(U, \mathcal{I})$ is a matroid.

Proof. Clearly $\emptyset \in \mathcal{I}$, and if $A \in \mathcal{I}$, then for $B \subseteq A$, $B \in \mathcal{I}$. Hence it suffices to prove (P2). Consider $A \subseteq E$, recall that

$$\kappa(A) = \# \text{ of components of } (V, A)$$

To obtain a maximal acyclic set of edges contained in A, we must choose a spanning tree from each component of A, so the size of such a set is

$$\sum_{i=1}^{\kappa(A)} = (|S_i| - 1) = |V| - \kappa(A) = n - \kappa(A)$$

where $S_1, \ldots, S_{\kappa(A)} \subseteq V$ are the components of (V, A). Notice how the size of the maximal elements in A does not depend on the specific element itself, we have verified (P2). $\square$

Example 3.2: A non-example instead

Given a grah $G = (V, E)$, take $U = E$ and $\mathcal{I} = \{M \subseteq E : M \text{ is a matching}\}$. Then $(U, \mathcal{I})$ is an independent system, but not a matroid.

Example 3.3: Uniform Matroid

Let U be any finite set, and $k \geq 0$ an integer. Then $M = (U, \mathcal{I} = \{A \subseteq U : |A| \leq k\})$ is a matroid. This is called a uniform matroid.

Proof. (P0) and (P1) clearly holds. For (P2), a basis of $A \subseteq U$ has size $\min\{k, |A|\} := r(A)$. ☐

Example 3.4: Linear Matroid

Given a $m \times n$ matrix M over a field $\mathbb{F}$ (i.e., n vectors in $\mathbb{F}^m$). Take $U = [n]$ and $\mathcal{I} = \{A \subseteq U :$ columns of M corres. to A are lin. ind.$\}$. Then $(U, \mathcal{I})$ is a matroid known as the linear matroid.

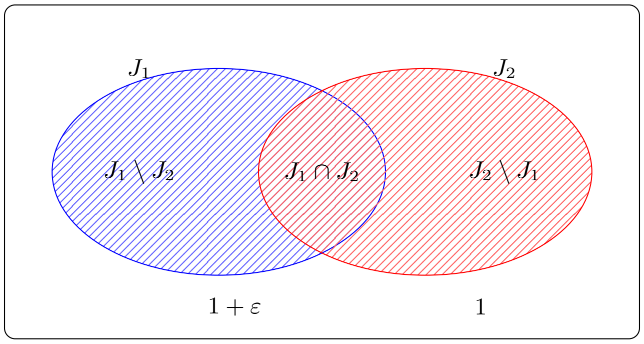
Proof. (P0) and (P1) clearly holds. (P2) follows by linear algebra. ☐

3.2.2 Rado '57, Edmonds '71

Theorem 3.1: Rado '57, Edmonds '71

Let $M = (U, \mathcal{I})$ be an independence system. Then the greedy algorithm returns a MWIS for all $w \in \mathbb{R}^{|U|}$ if and only if M is a matroid.

Proof. [$\Longrightarrow$]: We need to show that (P2) holds. Suppose for a contradiction that (P2) does not hold. Then there exists $A \subseteq U$ for which J_1, J_2 two bases of A having different sizes $|J_1| < |J_2|$.



As shown in the picture above, consider weight function w_e:

$$w_e = \begin{cases} 1 + \varepsilon & e \in J_1 \\ 1 & e \in J_2 - J_1 \\ 0 & \text{otherwise} \end{cases}$$

for $\varepsilon > 0$. Then, (verify) the greedy algorithm will return J_1, but for sufficiently small $\varepsilon > 0$, we have

$$|J_1|(1+\varepsilon) = w(J_1) < w(J_2) = (1+\varepsilon)|J_1 \cap J_2| + |J_2 - J_1|$$

a contradiction.
[$\Longleftarrow$]: We will prove something stronger. Let's look at the LP-relaxation for MWIS. The variable of our problem will be

$$x_e \qquad \forall e \in U$$

i.e., $x_e = 1$ if $e \in$ the set or 0 otherwise.

$$\begin{aligned} \max \quad & \sum_{e \in U} w_e x_e \\ \text{subject to} \quad & x(A) \leq r(A) \quad \forall A \subseteq U \\ & x \geq 0 \end{aligned} \tag{P}$$

Note 3.1

Note that the constraint $x(A) \leq r(A) \quad \forall A \subseteq U$ implies $x_e \leq 1$ for all $e \in U$.

Exercise 3.1

Let $J \subseteq U$, then $x = \chi^J$ is feasible to (P) if and only if $J \in \mathcal{I}$.

Solution. [$\Longrightarrow$]: If $J \notin \mathcal{I}$, so we know that

$$r(J) := \max\{|A| : A \subseteq J, A \in \mathcal{I}\} < x(J)$$

which shows that J is in fact not feasible, a contradiction.
[$\Longleftarrow$]: Since $J \in \mathcal{I}$, so for all $J' \subseteq J$, $J' \in \mathcal{I}$. Therefore, for all $A \subseteq U$, if $A \subseteq J$, then $x(A) = |A| = r(A)$; else if $J \subseteq A$, then $x(J) \leq r(A)$ indeed holds. ☐

Now we use the next theorem. ☐

Theorem 3.2

Let J^* be the output of greedy algorithm for $(U, \mathcal{I})$ with weights $\{w_e\}_{e \in U}$, then χ^{J^*} is the optimal solution to (P).

Proof. Take the dual of the LP:

$$\begin{aligned} \min \quad & \sum_{A \subseteq U} y_A \cdot r(A) \\ \text{subject to} \quad & \sum_{A \subseteq U, e \in A} y_A \geq w_e \quad \forall e \in U \\ & y_A \geq 0 \quad \forall A \subseteq U \end{aligned} \tag{1}$$

Let $|U| = m$. Order the elements so that

$$w_{e_1} \geq w_{e_2} \geq \cdots \geq w_{e_p} \geq 0 \geq w_{e_{p+1}} \geq \cdots \geq w_{e_m}$$

Let $A_i = \{e_1, \ldots, e_i\}$. Define

$$y^*_{A_i} = \begin{cases} w_{e_i} - w_{e_{i+1}} & i = 1, \ldots, p \\ w_{e_p} & \text{otherwise} \end{cases}$$

and $y^*_A = 0$ for all other $A \subseteq U$. We now show its dual feasibility. It is clear that $y^* \geq 0$. It is also clear that (1) holds if $e \in \{e_{p+1}, \ldots, e_m\}$ since $w_e \leq 0$ for all such e's. Suppose $e = e_i$ for $i \in [p]$. Then

$$\sum_{A \subseteq U, e \in A} y^*_A = \sum_{j=i}^{p} y^*_{A_j} = w_{e_i} = w_e \tag{*}$$

Lecture 6 - Thursday, September 25

Now we define $x^* = \chi^{J^*}$, and we wish to show that x^* and y^* satisfy the Complementary Slackness conditions.

[Suppose $x^*_e > 0$] This means that $e = e_i$ for some $i \in [p]$, and so (*) shows dual constraint for e is tight.

[Suppose $y^*_A > 0$] This means $A = A_i$ for some $i \in [p]$, $w_{e_i} > w_{e_{i+1}}$ (note that this tells us that $A_i = \{e \in U : w_e \geq w_{e_i}\}$), and $w_{e_i} > 0$. Our claim is that $J^* \cap A_i$ is a basis of A_i.
SFAC that there exists $f \in A_i - J^*$ such that $(J^* \cap A_i) \cup \{f\} \in \mathcal{I}$. Consider the point when Greedy considered f, at that point we have some $S \in \mathcal{I}$ such that $S \subseteq J^* \cap A_i$. Hence if $(J^* \cap A_i) \cup \{f\} \in \mathcal{I}$, then $S \cup \{f\} \subseteq (J^* \cap A_i) \cup \{f\} \in \mathcal{I}$, and so Greedy should have added f, which yields us a contradiction.
Now we use the fact that $(U, \mathcal{I})$ is a matroid to show that the corresponding constraint in the primal is tight. □

Exercise 3.2

Adapt Greedy to find the maximum weight basis. Show that if $(U, \mathcal{I})$ is a matroid, then the adapted Greedy returns an optimal solution to the LP:

$$\begin{aligned} \max \quad & \sum_{e \in U} w_e x_e \\ \text{subject to} \quad & x(A) \leq r(A) \quad \forall A \subseteq U \\ & x \geq 0 \\ & \boxed{x(U) = r(U)} \end{aligned} \tag{P'}$$

So the feasible region of (P') is $\text{conv}(\{x^B : B \text{ a basis of } (U, \mathcal{I})\})$.

Corollary 3.1

The feasible region for (P) is

$$\text{conv}(\{x^J : J \in \mathcal{I}\})$$

when $(U, \mathcal{I})$ is a matroid. This polytope is called the **matroid polytope**.

Proof. Left as an exercise. □

3.2.3 Greedy on Independence Systems

For an arbitrary independence system $(U, \mathcal{I})$, does Greedy have any performance guarantee? The answer is yes. Define $\rho : 2^U \to \mathbb{R}$ as

$$\rho(A) := \min\{|B| : B \text{ is a basis of } A\}$$

and define rank-quotient q as

$$q(U, \mathcal{I}) := \min_{A \subseteq U} \frac{\rho(A)}{r(A)}$$

Note 3.2

Note that $q(U, \mathcal{I}) \leq 1$, where equality holds if and only if $(U, \mathcal{I})$ is a matroid.

See detail in A1: Greedy returns an independent set J such that

$$w(J) \geq OPT_{MWIS} \cdot q(U, \mathcal{I})$$

3.2.4 Applications

One simple application would be matching: Given a graph $G = (V, E)$ and let $U = E$, $\mathcal{I} = \{M \subseteq E : M \text{ is a matching}\}$. Our claim is that

$$q(U, \mathcal{I}) \geq \frac{1}{2}$$

Proof. Consider $A \subseteq U$ with $\rho(A) < r(A)$. Let M_1, M_2 be two maximal matchings of A with $|M_1| < |M_2|$. Let

$$V_1 = \{v : v \text{ is some end-point of some edge in } M_1\}$$

Since M_1 is maximal, for all $e \in M_2 - M_1$, we know that e has an end-point in V_1, and so all egdes $e \in M_2$ have an end-point in V_1. Since M_2 is a matching, no two egdes in M_2 can share an end-point, so

$$|M_2| \leq |V_1| = 2|M_1|$$

□

3.2.5 Running Time of Greedy, and Representation of Matroids

Matroid $(U, \mathcal{I})$ specified via an *independence oracle*: takes a set $A \subseteq U$ and answers

$$YES \text{ if } A \in \mathcal{I} \quad \text{and} \quad NO \text{ if } A \notin \mathcal{I}$$

Each call to independence oracle is treated as an elementary operation, and is counted running time. As a result, let $m = |U|$, Greedy Algorithm has $O(m \log m)$ (used for sorting) running time, and the number of calls to independence oracle is in $O(m)$.

Another oracle could be the *rank oracle*: given $A \subseteq U$, it outputs $r(A)$.

Note 3.3

For a matroid $(U, \mathcal{I})$, independence oracle and rank oracle are **polynomially equivalent**, meaning we can get one oracle using polynomially many calls to another oracle.

3.3 Polymatroids

Definition 3.5: Submodular

Let U be a finite set, a function $f : 2^U \to \mathbb{R}$ is **submodular** if

$$f(A) + f(B) \geq f(A \cap B) + f(A \cup B)$$

for all $A, B \subseteq U$.

Exercise 3.3

Another equivalent definition for the above is: For all $A \subseteq B \subseteq U$, for all $e \notin B$

$$f(B \cup \{e\}) - f(B) \leq f(A \cup \{e\}) - f(A)$$

Proof. [$\Longrightarrow$] Assume f is submodular. Take any $A \subseteq B \subseteq U$ and $e \notin B$. Apply submodularity to the pair of sets $A_1 = A \cup \{e\}$ and $B_1 = B$, we obtain

$$f(A \cup \{e\}) + f(B) \ \geq \ f(\underbrace{A_1 \cap B_1}_{=A}) + f(\underbrace{A_1 \cup B_1}_{=B \cup \{e\}}).$$

since $A \subseteq B$ and $e \notin B$, Therefore

$$f(A \cup \{e\}) + f(B) \ \geq \ f(A) + f(B \cup \{e\}).$$

Rearranging gives

$$f(B \cup \{e\}) - f(B) \ \leq \ f(A \cup \{e\}) - f(A),$$

[$\Longleftarrow$] To be finished. $\square$

Example 3.5

Given a graph $G = (V, E)$. Define $f : 2^V \to \mathbb{R}$ by $f(S) = |\delta(S)|$. Then f is submodular.

Example 3.6

If $(U, \mathcal{I})$ is a matroid, then its rank function is submodular.

Definition 3.6: Polymatroid

Let $f : 2^U \to \mathbb{R}$ be submodular with $f(A) \geq 0$ for all $A \subseteq U$. Then the polytope

$$P_f : \{x \in \mathbb{R}^U : x(A) \leq f(A) \;\; \forall A \subseteq U, \; x \geq 0\}$$

is called a **polymatroid** defined by f. When f is the rank function of a matroid, this is the matroid polytope.

Note 3.4

We may assume that f is monotone, i.e., if $A \subseteq B$, then $f(A) \leq f(B)$.

Proof. For any $A \subseteq B$, then

$$x(A) \leq x(B) \leq f(B)$$

for any $x \in P_f$. Thus we can define $g(A) := \min_{B \supseteq A} f(B)$, and then we have

$$x(A) \leq g(A) \qquad \forall\; A \subseteq U$$

hence $P_f = P_g$ and (exercise) g is submodular and monotone. □

3.3.1 Greedy Algorithm on Polymatroids

Consider the problem

$$\max w^T x \quad \text{s.t.} \quad x \in P_f \tag{MW-P_f}$$

where f is non-negative, submodular, and monotone. Our claim is that Greedy algorithm can be extended to solve (MW-P_f).

```
1 Sort elements in decreasing order ;
2 Start with $x^* \leftarrow 0$ ;
3 Considering elements in sorted order, increment $x^*_{e_i}$ as much as possible while maintaining feasibility
    $x^*_{e_i} \leftarrow f(\{e_1, \ldots, e_i\}) - f(\{e_1, \ldots, e_{i-1}\}) \quad \forall\, i = 1, \ldots, m := |U|$
```

Algorithm 2: Greedy Algorithm for (MW-P_f)

Comment 3.2

We can show that

- x^* is the optimal solution to (MW-P_f);
- If $f : 2^U \to \mathbb{Z}_+$, then all extreme points of P_f are integral.

3.4 Matroid Construction

Lecture 7 - Tuesday, September 30

Let $M = (U, \mathcal{I})$ be a matroid. We have the following construction tools:

1. `Deletion`: Let $B \subseteq U$, we define

$$M' = (U' = U - B, \mathcal{I}' = \{J \subseteq U' : J \in \mathcal{I}\} = \{J - B : J \in \mathcal{I}\})$$

Exercise 3.4

Show that M' is a matroid (obtained by deleting B).

Proof. (P0) and (P1) are pretty clear. To show that the independence system is a matroid, we can show the exchange property. □

2. `Truncation`: Given integer $k \geq 0$, define

$$M' = (U, \mathcal{I}' = \{J \in \mathcal{I} : |J| \leq k\})$$

Exercise 3.5

Show that M' is still a matroid.

Proof. Again, (P0) and (P1) clearly holds. To show it is a matroid, we can then show exchange property as well. □

3. `Disjoint Union`: Let $M_j = (U_j, \mathcal{I}_j)$, $j = 1, \dots, k$ be matroids with rank functions r_j. Define the disjoint union of M_j's $M' = M_1 \oplus \cdots \oplus M_k$ as

$$M' = \left(U' = \bigcup_j U_j, \mathcal{I}' = \{A \subseteq U' : A \cap U_j \in \mathcal{I}_j \quad \forall j\}\right)$$

Note that each U_j is necessarily defined over distinct elements.

Exercise 3.6

We can verify that M' satisfies (P0) and (P1). For (P2), we can infer that M' has rank function r' defined as

$$r'(A) := \sum_{j=1}^{k} r_j(A \cap U_j)$$

Comment 3.3

Notation: We write

$$M\text{–}\{\text{independent, basis}\} \equiv \{\text{independent, basis}\} \text{ in matroid } M$$

Example 3.7: (Generalized) Partition Matroid

Let U be a finite set, and $S_1, \dots, S_k$ be a partition of U. Let $b_1, \dots, b_k$ be non-negative integers. Consider

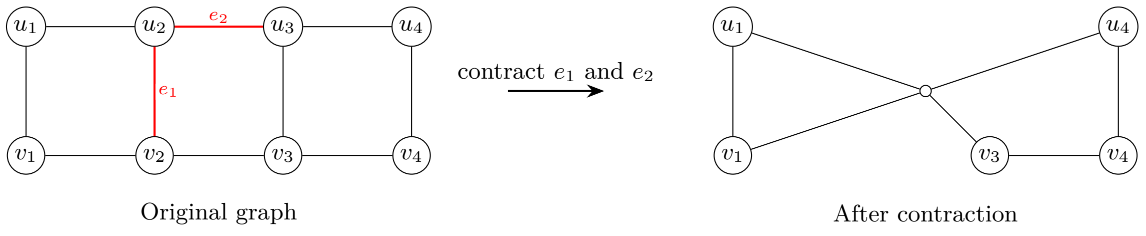
$$M = (U,\ \mathcal{I} = \{A \subseteq U : |A \cap S_j| \leq b_j \quad \forall j\})$$

We can verify that M is a matroid, which is known as the generalized partition matroid.

Proof of M defined as above is a matroid. M is the disjoint union of $M_j = (S_j, \mathcal{I}_j)$, where $\mathcal{I}_j = \{X \subseteq S_j : |X| \leq b_j\}$. Each M_j is a uniform matriod, so M is a matroid as a result of the matroid construction. □

4. `Contraction`: Let $B \subseteq U$, and let J be an M-basis of B. Define

$$M' = (U' = U - B,\ \ \mathcal{I}' = \{J' \subseteq U' : J' \cup J \in \mathcal{I}\})$$



Theorem 3.3

M', also denoted as M/B, is a matroid (matroid obtained by contracting B), and it does not depend on choice of J. Moreover, the rank function of M' is given by r' defined as

$$r'(A) := r(A \cup B) - r(B)$$

where $A \subseteq U' = U - B$.

Proof. Recall that the rank function of an independence system completely defines the independence system. Now we verify that M' is a matroid:

(P0) We have $\emptyset \in \mathcal{I}'$ since $J \in \mathcal{I}$;

(P1) $A \in \mathcal{I}'$, $X \subseteq A$, then $X \cup J \subseteq A \cup J$ and $A \cup J \in \mathcal{I}$, so $X \cup J \in \mathcal{I}$, and hence $X \in \mathcal{I}'$.

(P2) Let $A \subseteq U - B$, and J' be a M'-basis of A, we wish to show that $|J'| = r'(A)$ (this implies (P2) and shows that M' does not depend on choice of J).
Claim: $J' \cup J$ is an M-basis of $A \cup B$.
Proof of claim: Consider $e \in A \cup B - (J \cup J')$. Suppose $J \cup J' \cup \{e\} \in \mathcal{I}$,

- If $e \in A$, then $(J' \cup \{e\}) \cup J \in \mathcal{I}$, which implies that $J' \cup \{e\} \in \mathcal{I}'$, contradicting the fact that J' is an M' basis of A.
- If $e \in B$, then $J \cup \{e\} \subseteq J \cup J' \cup \{e\}$, so $J \cup \{e\} \in \mathcal{I}$, contradicting the fact that J is an M-basis of B.

Hence $|J'| = |J \cup J'| - |J'|$, which by claim $= r(A \cup B) - r(B) = r'(A)$.

$\square$

5. `Duality`: Let $M = (U,\ \mathcal{I})$ be a matroid with rank function r, we define

$$M^* = (U,\ \ \mathcal{I}^* = \{J \subseteq U : U - J \text{ contains an } M\text{-basis}\})$$

Theorem 3.4

M^* is a matroid, called the *dual matroid*.

Proof. (P0) $\emptyset \in \mathcal{I}^*$ since U contains an M-basis;

(P1) $A \in \mathcal{I}^*$ and $B \subseteq A$, then $U - A \subseteq U - B$, and if $U - A$ contains a M-basis then so does $U - B$.

(P2) Let $A \subseteq U$, and let J^* be an M^*-basis of A. Hence $J^* \in \mathcal{I}^*$ implies that $U - J^*$ contains an M-basis ($r(U - J^*) = r(U)$). Moreover, because J^* is a M^*-basis (i.e, $J^* \cup \{e\} \notin \mathcal{I}^*$ for all $e \in A - J^*$), we have that every M-basis in $U - J^*$ must contain all of $A - J^*$. We can construct a M-basis in $U - J^*$ as follows: Start with a basis of $U - A$, and extend it to a basis B of $U - J^*$, which is a basis of U, and so

$$r(U) = |B| = \underbrace{|B \cap A|}_{|A| - |J^*|} + \underbrace{|B - A|}_{r(U-A)}$$
$$\Rightarrow \quad |J^*| = |A| + r(U - A) - r(U)$$

which is the rank function of M^*, proving (P2). $\square$

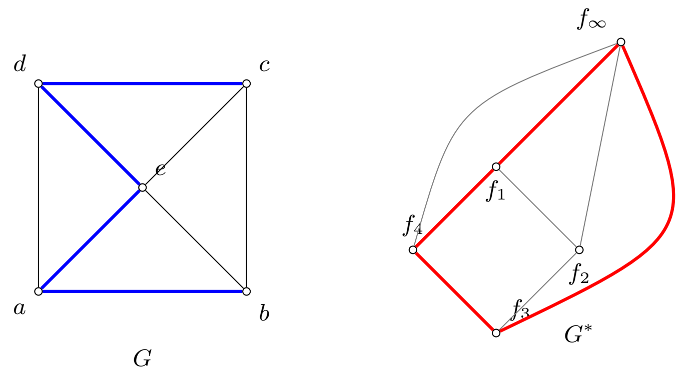
Example 3.8: Cool Application of Duality Matroid

Let $G = (V, E)$ be a planar graph. We can associate another graph $G^* = (V^*, E^*)$, called the planar dual of G, as follows: There is a vertex in V^* for every face of G, and an edge $e = u^*v^* \in E^*$ if the two faces corresponding to u^* and v^* share an edge. We now have the following theorem:

Theorem 3.5

Graphic matroid M_{G^*} of G^* is the dual of graphic matrioid M_G of G. i.e., $M_{G^*} = (M_G)^*$.

Proof. We observe that every cut in G corresponds to a cycle in G^*, and every cycle in G yields a cut in G^*. Here is a simple example for the visual:



Therefore, we have the following equivalent statements (this is a sketch of the proof, we will assume that the original graph G is connected)

$$\begin{aligned} A \in \mathcal{I}_{M_{G^*}} &\iff A \text{ does not contain a cycle in } G^* \\ &\iff A \text{ does not contain a cut of } G \\ &\iff \forall S \subseteq V,\ \ \delta(S) - A \neq \emptyset \\ &\iff E - A \text{ is connected} \iff A \in \mathcal{I}_{(M_G)^*} \end{aligned}$$

□

Lecture 8 - Thursday, October 02

Definition 3.7: Circuit and Cut

Let $M = (U, \mathcal{I})$ be a matroid.

- A set $C \subseteq U$ is called a **circuit** of M if C is a minimal dependent set;
- A set $C \subseteq U$ is called a **cut** of M if C is a minimal set that intersets every basis of M.

Example 3.9

In a graphic matroid, a circuit is a cycle, and a cut is just a cut.

Theorem 3.6

Let $M = (U, \mathcal{I})$ be a matroid and $B \subseteq U$.

1. $(M^*)^* = M$;
2. $(M \setminus B)^* = M^*/B$ (deletion in M followed by dual $\equiv$ dual followed by contraction);
3. B is a circuit of M if and only if B is a cut of M^*.
4. B is a cut of M if and only if B is a circuit of M^*.

Proof. Notice that part (4) follows directly from part (1) and (3). Here we provide the proof for part (1):
Let r be the rank function of M. Let $A \subseteq U$, then

$$\begin{aligned} r_{(M^*)^*}(A) &= |A| + r_{M^*}(U - A) - r_{M^*}(U) \\ &= |A| + (|U - A| + r(U - (U - A)) - r(U)) - (|U| + r(U - U) - r(U)) \\ &= r(A) \end{aligned}$$

and so $(M^*)^* = M$. □

3.5 Matroid Intersection

Given two matroids $M_1 = (U, \mathcal{I}_1)$ and $M_2 = (U, \mathcal{I}_2)$, we wish to find a max-size set that is independent in both M_1 and M_2. i.e.,

$$\max_{J \in \mathcal{I}_1 \cap \mathcal{I}_2} |J|$$

More generally, given $\{w_e\}_{e \in U}$, we wish to find a max-weight common independent set:

$$\max_{J \in \mathcal{I}_1 \cap \mathcal{I}_2} w(J)$$

We first start with two applications:

3.5.1 Two Applications

Biaprtite Matching Given a bipartite graph $G = (L \cup R, E)$, we wish to find a maxmimum-size matching.

Note 3.5

Note that $F \subseteq E$ is a matching in G if and only if F is independent in $M_1 = (E, \mathcal{I}_1 = \{J \subseteq E : |J \cap \delta(v)| \leq 1 \ \forall v \in L\})$ and $M_2 = (E, \mathcal{I}_2 = \{J \subseteq E : |J \cap \delta(v)| \leq 1 \ \forall v \in R\})$.

Comment 3.4

M_1 and M_2 are partition matroids since G is bipartite.

Branchings and Arborescences Given a directed graph $D = (V, A)$, a set $F \subseteq A$ is a branching if

1. undirected version (F but we forget about the directions) of F is a forest;
2. Every node has indegree at most 1.

In other words, this is just a directed analogue of forest.

Note 3.6

A branching with $n - 1$ edges is the same as a spanning tree (viewed as a undirected graph). Moreover, every node except the special "root" node has in-degree $= 1$, while the "root" has in-degree $= 0$.

Defining $M_1 = (A, \mathcal{I}_1)$ as the graphic matroid of the undirected version of D, and $M_2 = (A, \mathcal{I}_2 = \{J \subseteq A : |J \cap \delta^{in}(v)| \leq 1 \ \forall\, v \in V\})$. Now, F is a branching if and only if $F \in \mathcal{I}_1 \cap \mathcal{I}_2$.

Discovery 3.1

We have

$$\underbrace{\text{Branching with } n-1 \text{ edges}}_{\text{Arborescence}} \equiv \text{basis of } M_1 \text{ that is independent in } M_2$$

assuming undirected version of D is connected.

3.5.2 Min-max Formula for Matroid Intersection (Matroid Intersection Theorem)

Let $M_1 = (U, \mathcal{I}_2)$ and $M_2 = (U, \mathcal{I}_2)$ be two matroids with rank functions r_1 and r_2 respectively.

Let A and $U - A$ be a partition of U, and let $J \in \mathcal{I}_1 \cap \mathcal{I}_2$. Now we have

$$\begin{aligned} |J| &= \underbrace{|J \cap A|}_{\in \mathcal{I}_1} + \underbrace{|J \cap (U - A)|}_{\in \mathcal{I}_2} \\ &\leq r_1(A) + r_2(U - A) \end{aligned}$$

This inequality holds for all partitions A, $U - A$ of U. Hence

$$\boxed{\max\{|J| : J \in \mathcal{I}_1 \cap \mathcal{I}_2\} \quad \leq \quad \min_{A \subseteq U}\{r_1(A) + r_2(U - A)\}}$$

Theorem 3.7: Edmonds 1971, Matroid Intersection Theorem

Let $M_1 = (U, \mathcal{I}_2)$ and $M_2 = (U, \mathcal{I}_2)$ be two matroids with rank functions r_1 and r_2 respectively,

$$\max\{|J| : J \in \mathcal{I}_1 \cap \mathcal{I}_2\} \quad = \quad \min_{A \subseteq U}\{r_1(A) + r_2(U - A)\} \tag{MIT}$$

Comment 3.5

We will have two proofs for the above theorem.

Proof 1 of Theorem 3.7. We have shown that

$$\max\{|J| : J \in \mathcal{I}_1 \cap \mathcal{I}_2\} \quad \leq \quad \min_{A \subseteq U}\{r_1(A) + r_2(U - A)\}$$

Hence we focus on showing

$$\min_{A \subseteq U}\{r_1(A) + r_2(U - A)\} \quad \leq \quad \max\{|J| : J \in \mathcal{I}_1 \cap \mathcal{I}_2\}$$

We prove by induction on the size of U. For the base case, we have $|U| = 0$, which is a trivial case. Suppose the statement holds for $|U| \geq 1$ and let $k := \min_{A \subseteq U}\{r_1(A) + r_2(U - A)\}$.
`[Case 1]`: Suppose for all $e \in U$, $\{e\} \notin \mathcal{I}_1$ or $\{e\} \notin \mathcal{I}_2$, then $k = 0$, since we can take

$$A = \{e \in U : r_1(e) = 0\}$$

`[Case 2]`: There exists some $e \in U$ such that $\{e\} \in \mathcal{I}_1 \cap \mathcal{I}_2$. Now we have two subcases:

1. Let $M_1^{del} = M_1 - \{e\}$ and $M_2^{del} = M_2 - \{e\}$. Let $U' = U - \{e\}$, so $|U'| < |U|$. Define k^{del} to be the RHS of (MIT) for M_1^{del} and M_2^{del}. If $k^{del} = k$, then by induction hypothesis, we have some J who is both M_1^{del}-independent and M_2^{del}-independent such that

$$|J| \geq k^{del} = k$$

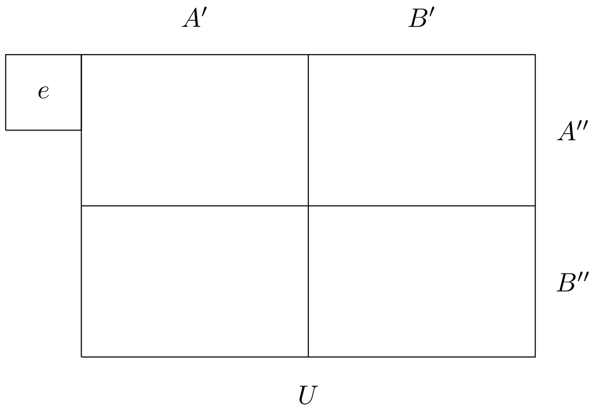
2. Let $M_1^{cont} = M_1/\{e\}$ and $M_2^{cont} = M_2/\{e\}$, and let k^{cont} be the RHS of (MIT) for M_1^{cont} and M_2^{cont}.

If $k^{cont} \geq k-1$, then by the induction hypothesis, there exists $J \subseteq U' = U - \{e\}$ such that

$$J \in \mathcal{I}(M_1^{cont}) \cap \mathcal{I}(M_2^{cont}) \implies J \cup \{e\} \in \mathcal{I}_1 \cap \mathcal{I}_2$$

and so $|J| \geq k^{cont} = k-1$, which further implies $|J \cap \{e\}| \geq k$

Now, suppose $k^{del} \leq k-1$ and $k^{cont} \leq k-2$.



The first inequality implies that there exists a partition A', B' of $U' = U - \{e\}$ such that

$$r_1(A') + r_2(B') = k^{del} \leq k-1$$

and the second inequality implies that there exists a partition A'', B'' of U' such that

$$\underbrace{r_1(A'' \cup \{e\}) - r_1(\{e\})}_{r(M_1^{cont})} + \underbrace{r_2(B'' \cup \{e\}) - r_2(\{e\})}_{r(M_2^{cont})} \leq k-2$$

Adding the two inequalities we obtain

$$\big[r_1(A') + r_1(A'' \cup \{e\})\big] + \big[r_2(B') + r_2(B'' \cup \{e\})\big] \leq 2k-1 \tag{1}$$

Recall that r_1 and r_2 are submodular. By submodularity,

$$r_1(A') + r_1(A'' \cup \{e\}) \geq r_1(A' \cap (A'' \cup \{e\})) + r_1(A' \cup (A'' \cup \{e\})) \tag{2}$$

$$r_2(B') + r_2(B'' \cup \{e\}) \geq r_2(B' \cap (B'' \cup \{e\})) + r_2(B' \cup (B'' \cup \{e\})) \tag{3}$$

Now, (1), (2), and (3) yields us

$$\Big[r_1(A' \cap (A'' \cup \{e\})) + r_2(B' \cup (B'' \cup \{e\}))\Big] + \Big[r_1(A' \cup (A'' \cup \{e\})) + r_2(B' \cap (B'' \cup \{e\}))\Big] \leq 2k-1$$

Each of which is a partition of U (see the above picture for a visualization), so there exists some partition $S, T \subseteq U$ such that $r_1(S) + r_2(T) < k$, which is a contradiction. □

Lecture 9 - Tuesday, October 07

3.5.3 Three Applications of Matroid Intersection Theorem

Here we introduce some of the applications of the Matroid Intersection Theorem.

Application 1. König's Theorem

Definition 3.8: Cover

Given a graph $G = (V, E)$. A set $C \subseteq V$ is called a (vertex) **cover** of G if for all $e = uv \in E$, $u \in C$ (inclusive) or $v \in C$.

Theorem 3.8: König's Theorem

Let $G = (L \cup R, E)$ be a bipartite graph. Then

$$\max\{|F| : F \subseteq E \text{ is a matching}\} = \min\{|C| : C \subseteq V \text{ is a vertex cover}\}$$

Proof. Let $M_1 = (E,\ \mathcal{I}_1 = \{J \subseteq E : |\delta(v) \cap J| \leq 1 \ \ \forall v \in L\})$ and $M_2 = (E,\ \mathcal{I}_2 = \{J \subseteq E : |\delta(v) \cap J| \leq 1 \ \ \forall v \in R\})$, then

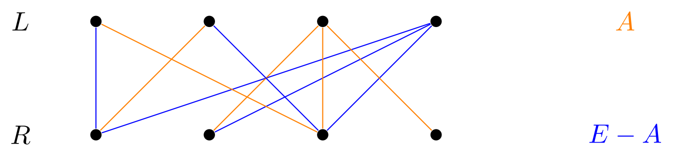
$$F \text{ is a matching} \iff F \in \mathcal{I}_1 \cap \mathcal{I}_2$$

Hence by the Matroid Intersection Theorem 3.7,

$$\max\{|F| : F \text{ is a matching}\} = \min\{r_1(A) + r_2(E - A) : A \subseteq E\}$$

where $r_1(A) = |\underbrace{\{v \in L : \delta(v) \cap A \neq \emptyset\}}_{=: S_1}|$, and $r_2(E - A) = |\underbrace{\{v \in R : \delta(v) \cap (E - A) \neq \emptyset\}}_{=: S_2}|$.

Here is an picture for the sake of illustration:

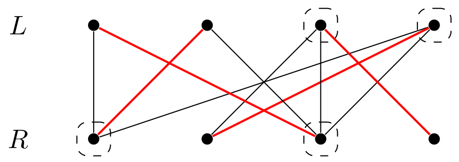


Note 3.7

Note that the union of the two sets, call them $S_1 \cup S_2$, is a vertex cover of G. Since for all $uv \in E$, $u \in L$ and $v \in R$, and if $uv \in A$, then $u \in S_1$, otherwise $v \in S_2$.

Hence there exists a minimizer $A \subseteq E$ such that $r_1(A) + r_2(E - A) = |C|$ for some vertex cover C, so the equality in the theorem holds.

For the above example, we have the following matching and vertex cover (both of size 4):



□

Application 2. Orientations and Hakimi's Theorem Let $G = (V, E)$ be an undirected graph and let $b_v \geq 0$ integers associated to all $v \in V$.

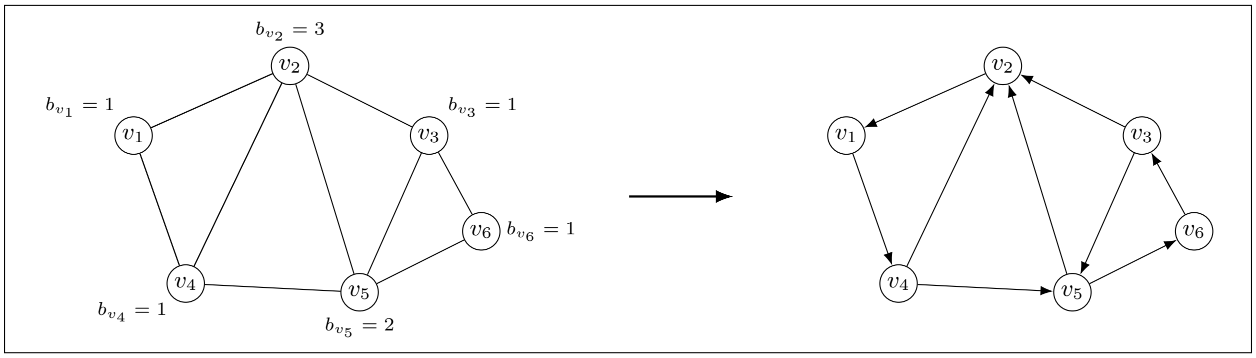
Definition 3.9: Orientation

An **orientation** of G means to direct every edge of G to obtain a digraph.

Theorem 3.9: Hakimi's Theorem

G has an orientation where every $v \in V$ has $\leq b_v$ incoming edges, we call this graph G has b-orientation, if and only if

$$|E(S)| \leq b(S) := \sum_{v \in S} b_v \qquad \forall\, S \subseteq V$$



Comment 3.6

Easy to notice that the inequality condition is necessary. This is because the sum of b_vs is the total budget we have, which shouldn't be less than the edges we have within those vertices.

Proof. Define Z to be

$$Z := \{(u, v), (v, u) : uv \in E\}$$

Hence, an orientation means that we have to choose exactly one of (u, v) and (v, u) for all $uv \in E$. Define partition matroid

$$M_1 = (Z,\ \mathcal{I}_1 = \{J \subseteq Z : |J \cap \{(u, v), (v, u)\}| \leq 1\ \ \forall uv \in E\})$$

Encode in-degree constraint via another partition matroid:

$$M_2 = (Z,\ \mathcal{I}_2 = \{J \subseteq Z : |\delta^{in}(v) \cap J| \leq b_v\ \ \forall v \in V\})$$

Hence

$$J \subseteq Z \text{ is a } b\text{-orientation of } G \iff J \in \mathcal{I}_1 \cap \mathcal{I}_2 \text{ and } |J| = |E|$$

Such a J must be a basis of M_1, hence a max-size common independent set. So G has a b-orientation if and only if $\max\{|J| : J \in \mathcal{I}_1 \cap \mathcal{I}_2\} = |E|$. By Matroid Intersection Theorem 3.7, G has a b-orientation if and only if $\underbrace{\min\{r_1(A) + r_2(Z - A) : A \subseteq Z\}}_{=:\,\alpha} = |E|$.

We claim that there exists a minimizer A such that for all $v \in V$,

$$\left[\delta^{in}(v) \cap (Z - A) = \emptyset\right] \text{ OR } \left[\delta^{in}(v) \subseteq (Z - A) \text{ and } |\delta^{in}(v) \cap (Z - A)| \geq b_v\right]$$

Let $x = |\delta^{in}(v) \cap (Z - A)|$. If $0 < x < b_v$, then taking $A' = A \cup \delta^{in}(v)$, we get

$$r_1(A') \leq r_1(A) + x$$

$$r_2(Z - A') = r_2(Z - A) - x$$

If $x \geq b_v$, but $x < |\delta^{in}(v)|$, then taking $A'' = A - \delta^{in}(v)$ gives

$$r_1(A'') \leq r_1(A)$$

$$r_2(Z - A'') \leq r_2(Z - A)$$

Hence our claim holds. By the claim, there exists a minimizer A such that

$$Z - A = \bigcup_{v \in S} \delta^{in}(v)$$

for some $S \subseteq V$, and $|\delta^{in}(v)| \geq b_v$ for all $v \in S$, so

$$r_2(Z - A) = b(s), \quad \text{and} \quad r_1(A) = r_1\left(\bigcup_{v \notin S} \delta^{in}(v)\right) = |E - E(S)|$$

so $$\alpha \geq \min\{b(S) + |E| - |E(S)| : S \subseteq V\}$$

In fact, equality holds for the above inequality. This means that G has a b-orientation if and only if for all $S \subseteq V$,

$$b(S) + |E| - |E(S)| \geq |E|$$

which is equivalent to saying $b(S) \geq |E(S)|$ for all $S \subseteq V$. $\square$

Application 3. Matroid Partition Let $M_j = (U, \mathcal{I}_j)$ for all $j = 1, \ldots, k$ be k matroids.

Definition 3.10: Partitionable

We say that $J \subseteq U$ is **partitionable** with respect to $(M_1, \ldots, M_k)$ if there exists a partition $J_1 \cup \cdots \cup J_k$ of J such that $J_r \in \mathcal{I}_r$ for all r.

Definition 3.11: Matroid Partition Problem

The **Matroid Partition Problem** asks us to find a max-size partitionable set.

Comment 3.7

The union being disjoint is not the crucial part but covering the entire J is. As we can always use heredity to shrink each subset down to make them disjoint.

Theorem 3.10: Edmonds & Fulkerson, 1965; Rado, 1942

We have

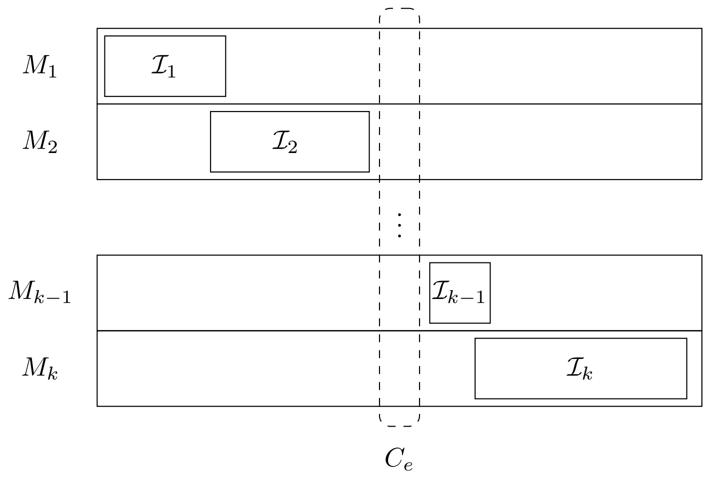
$$\max\{|J| : J \text{ is partitionable}\} = \min\left\{|U - A| + \sum_{j=1}^{k} r_j(A) : A \subseteq U\right\}$$

where $r_1, \ldots, r_k$ are rank functions of $M_1, \ldots, M_k$ respectively.

Proof.

Comment 3.8

Here we give an idea of the proof. We create k disjoint copies $U_1, \ldots, U_k$ of U and view M_j as a matroid over U_j. We denote C_e to be the set of all copies of e for every $e \in U$.



Let M_1' be the disjoint union of $M_1, \ldots, M_k$, so

$$M_1' = (U' = U_1 \cup \cdots \cup U_k,\ \mathcal{I}_1')$$

We also define

$$M_2' = (U',\ \mathcal{I}_2' = \{J' \subseteq U' : |J' \cap C_e| \le 1\ \ \forall e \in U\})$$

Therefore, J is partitionable with partition $(J_1, \dots, J_k)$ if and only if disjoint union of J_r's for all r lies in $\mathcal{I}_1' \cap \mathcal{I}_2'$. Let r_1' and r_2' be the rank functions of M_1' and M_2' respectively. Hence

$$\max\{|J| : J \text{ partitionable}\} = \underbrace{\min\{r_1'(B') + r_2'(U' - B') : B' \subseteq U'\}}_{=:\,\alpha}$$

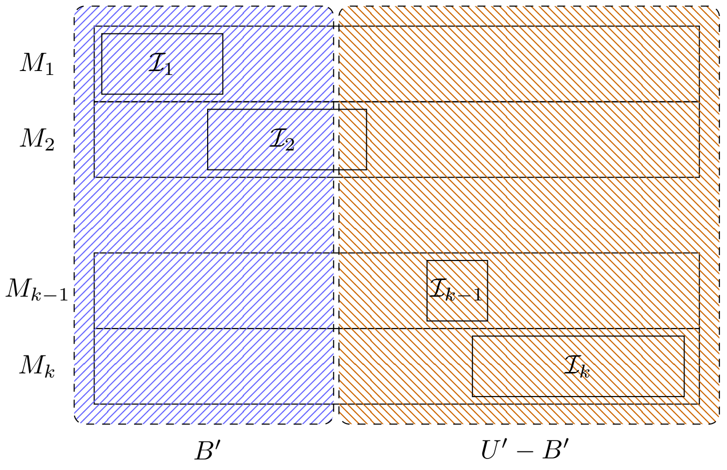
where we know that

$$r_1'(B') = \sum_{j=1}^{k} r_j(B' \cap U_j)$$

$$r_2'(U' - B') = |\{e \in U : (U' - B') \cap C_e \neq \emptyset\}|$$

From the same kind of argument as in the Hakimi's Theorem 3.9, there exists a minimizer B' such that

$$\Big[(U' - B') \cap C_e = \emptyset\Big] \text{ OR } \Big[U' - B' \supseteq C_e\Big]\ \ \forall e \in U$$



Equivalently, $B' = \bigcup_{e \in A} C_e$ for some $A \subseteq U$. Then $r_1'(B') = \sum_{j=1}^{k} r_j(A)$ and $r_2'(U' - B') = |U - A|$. Hence α is exactly the expression we wanted. □

3.5.4 Algorithm for (unweighted) Matroid Intersection and its Runtime

Lecture 10 - Tuesday, October 9

We first take a look at a special case.

Bipartite Matchings

Definition 3.12: Some Terminologies

Let $M \subseteq E$ be a matching.

- A node $v \in V$ is M**-exposed** if $\delta(v) \cap M = \emptyset$, otherwise it is M**-covered**.
- A path P in G is called M**-alternating** if its edges alternate between being in M and not being in M.
- An M-alternating path whose start and end nodes are M-exposed is called an M**-augmenting** path.

Here we state some facts without proving them. Feel free to try to prove. We will prove the result in a more general context, aka matroids.

Note 3.8

Any augmenting path can be used to increase the size of matching by

$$M \leftarrow M \Delta P := (M - P) \cup (P - M)$$

Note 3.9

If we direct edges in M with $L \to R$ and edges not in M with $R \to L$, then a M-augmenting path is exactly a $R \to L$ path in this digraph whose end parts are M-exposed.

Note 3.10

If there exists no M-augmenting path, then we can use König's theorem and find a vertex cover equal to the size of matching, which is a certificate for the optimality of our matching.

This gives us some intuition of how the algorithm would work.

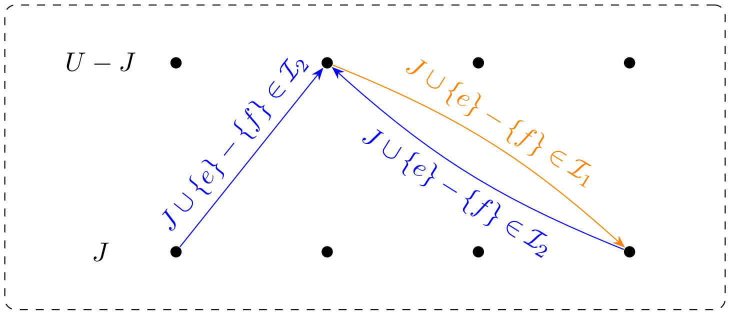
General Case, Matroid Intersection Let $M_1 = (U, \mathcal{I}_1)$ be a matroid with rank function r_1 and $M_2 = (U, \mathcal{I}_2)$ be a matroid with rank function r_2. Let $J \in \mathcal{I}_1 \cap \mathcal{I}_2$. Construct the following directed bipartite graph

$$D = D(M_1, M_2, J)$$

whom has bipartition $J \cup (U - J)$. For every $e \in U - J$ and $f \in J$, we establish

- an edge (e, f) if $J \cup \{e\} - \{f\} \in \mathcal{I}_1$;
- an edge (f, e) if $J \cup \{e\} - \{f\} \in \mathcal{I}_2$.

Now we define $X_1 = \{e \in U - J : J \cup \{e\} \in \mathcal{I}_1\}$ and $X_2 = \{e \in U - J : J \cup \{e\} \in \mathcal{I}_2\}$.


$U - J$
J
$J \cup \{e\} - \{f\} \in \mathcal{I}_2$
$J \cup \{e\} - \{f\} \in \mathcal{I}_1$
$J \cup \{e\} - \{f\} \in \mathcal{I}_2$

Definition 3.13: Chordless

We say a path $P = u_1 u_2 \dots u_k$ is **chordless** if for any $1 \le i, j \le k$, $i < j + 1$, (u_i, u_j) is not an edge.

Comment 3.9

Easy to see that the shortest u-v path in D is always chordless.

We use e to denote the nodes in $U - J$ and f the nodes in J.

Theorem 3.11: Augmenting Path Theorem for Matroid Intersection

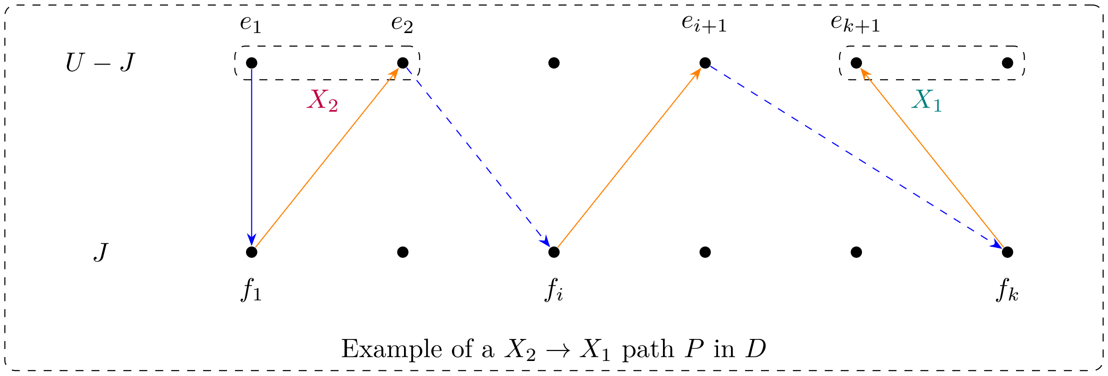
1. Let $P = e_1 f_1 e_2 f_2 \dots e_k f_k e_{k+1}$ be a shortest $X_2 \to X_1$ path in D, then

$$J' = J - \{f_1, \dots, f_k\} + \{e_1, \dots, e_{k+1}\} \in \mathcal{I}_1 \cap \mathcal{I}_2$$

   and of course $|J'| = |J| + 1$. We also have several observations:

   - $e_1 \in X_2$, $e_{k+1} \in X_1$;
   - Shortest implies that the path is chordless, $e_i \notin X_1 \cup X_2$ for all $i = 2, \dots k$;
   - We could have $k = 0$, and this happens if and only if there exists $e \in X_1 \cap X_2$.

2. If there is no chordless $X_2 \to X_1$ path in D, then J is a max-size common independent set.



The above theorem yields us a polytime algorithm for find the max-size common independent set:

```
1 $J \leftarrow \emptyset$ ;
2 Construct $D = D(M_1, M_2, J)$ ;
3 **while** *exists* $X_2 \to X_1$ *path in* $D$ **do**
4     Let $P$ be the shortest $X_2 \to X_1$ path ;
5     Update $J \leftarrow J\Delta P$ ;
6     Update $D$
7 **return** $J$
```

Algorithm 3: Polytime algorithm for max-size common independent set

Lemma 3.1

Let $M = (U, \mathcal{I})$ be a matroid, $J \in \mathcal{I}$. For any $e \in U$,

1. $J \cup \{e\}$ contains at most one circuit;
2. If $J \cup \{e\}$ has a circuit in C, then there exists $f \in C$ such that $J + \{e\} - \{f\} \in \mathcal{I}$.

Note 3.11

If $e \notin X_2$, then $J \cup \{e\}$ contains an M_2-circuit C' and have edges (f, e) precisely for all $f \in C' - \{e\}$. Similarly, if $e \notin X_1$, then there exists an edge (e, f) in D.

Lemma 3.2

Let $M = (U, \mathcal{I})$ be a matroid. $A \subseteq U$ be such that A has a circuit C. Then for any $e \in C$, $r(A - \{e\}) = r(A)$.

Proof of Theorem 3.11. `[Part 1]`: For all $i = 0, \ldots, k$, define

$$A_i = \{f_1, \ldots, f_i\} \cup J'$$

which is the same as $J \cup \{e_1, \ldots, e_{k+1}\} - \{f_{i+1}, \ldots, f_k\}$.

We will show that $r_1(A_i) \geq |J|+1$ for all $i = 0, \ldots, k$ by induction on $k-i$.
This implies that for $i = 0$, we get $r_1(A_0) = r_1(J') = |J'|$, which shows that $J' \in \mathcal{I}_1$.

`[Base case]`: The base case is when $i = k$, so $A_k \supseteq J \cup \{e_{k+1}\}$. By definition of $e_{k+1} \in X_1$, we know that $J \cup \{e_{k+1}\} \in \mathcal{I}_1$, so $r_1(A_k) \geq r_1(J \cup \{e_{k+1}\}) = |J|+1$.

`[Inductive step]`: Suppose inductively $r_1(A_i) \geq |J|+1$, we consider A_{i-1}. Recall $A_{i-1} = A_i - \{f_i\}$. By lemma 3.2, if we show that f_i lies in some M_1-circuit $C \subseteq A_i$, then

$$r_1(A_{i-1}) = r_1(A_i - \{f_i\}) \geq |J|+1$$

Note that $J \cup \{e_i\}$ contains a M_1-circuit C since $e_i \notin X_1$, and C does not contain $\{f_{i+1}, \ldots, f_k\}$ because otherwise (e_i, f_j) would be a chord for some $j > i$. Also, $f_i \in C$ since (e_i, f_i) is an edge, so $C \subseteq A_i$ and $f_i \in C$.

To show that $J' \in \mathcal{I}_2$, we define $B_i = \{f_i, \ldots, f_k\} \cup J' = J \cup \{e_1, \ldots, e_k\} - \{f_1, \ldots, f_{i-1}\}$. We will show by induction on i that $r_2(B_i) \geq |J|+1$.

Here we give a sketch as the proof is analogous to above.

`[Base case]`: $i = 1$. $B_1 \supseteq J \cup \{e_1\}$, and $e_1 \in X_2$, so $J \cup \{e_1\} \in \mathcal{I}_2$

`[Inductive step]`: Suppose inductively $r_2(B_i) \geq |J|+1$, we consider $B_{i+1} = B_i - \{f_i\}$. We can show that B_i contains a circuit C who contains f_i, so $r_2(B_{i+1}) = r_2(B_i)$. Thus,

$$r_2(B_{k+1}) = r_2(J') \geq |J'| \quad \Longrightarrow \quad J' \in \mathcal{I}_2$$

`[Part 2]`: Now we prove the second part of the theorem.

Since there exists no $X_2 \to X_1$ path in D, let A be the set of nodes reachable from X_2 in D. Then

$$X_2 \subseteq A, \qquad X_1 \subseteq U - A, \qquad X_1 \cap X_2 = \emptyset, \qquad \delta_D^{out}(A) = \emptyset$$

Let $J_1 = J \cap A$ and $J_2 = J - A$. We show that

$$\begin{Bmatrix} J_1 : \ M_1\text{-bases of } A \\ J_2 : \ M_2\text{-bases of } U - A \end{Bmatrix}$$

This shows that $|J_1| = r_1(A)$, $|J_2| = r_2(U-A)$, so $|J| = |J_1| + |J_2| = r_1(A) + r_2(U-A)$, which implies that J is the max-size common independent set.

`[`J_1 `is a` M_1`-basis of` A`]`: Suppose not, and there exists $e \in A - J$ such that $J_1 \cup \{e\} \in \mathcal{I}_1$. Since $e \in A$, $e \notin X_1$, so $J \cup \{e\}$ has a M_1-circuit C, and we cannot have $C \subseteq J_1 \cup \{e\}$ since $J_1 \cup \{e\} \in \mathcal{I}_1$, which tells us that there exists $f \in C - J_1$. But then $f \in J_2$ and we would have an edge $(e, f) \in \delta_D^{out}(A)$, a contradiction.
`[`J_2 `is a` M_2`-basis of` $U - A$`]`: Symmetric argument, left as an exercise. □

Lecture 11 - Tuesday, October 21

Proof of Lemma 3.1. `[Statement 1]` Suppose $J \cup \{e\}$ contains two circuits, call them C_1 and C_2. It must be the case that $e \in C_1 \cap C_2$, let $f \in C_1 - C_2$. We have $C_1 - \{f\} \in \mathcal{I}$, so we can extend $C_1 - \{f\}$ to an independent set J' of size $|J'| = |J|$ by adding elements in $J - (\{C_1 - \{f\}\})$ (by exchange property). Then $J' \subseteq J \cup \{e\}$, and $f \notin J'$. This implies that $J' = J - \{f\} + \{e\}$, who contains C_2, a contradiction.
`[Statement 2]` Follows from statement 1, if $(J \cup \{e\}) - \{f\} \notin \mathcal{I}$, then it contains some circuit C' such that $C \neq C'$ since $f \notin C'$. But then $J \cup \{e\} \supseteq C, C'$, a contradiction. □

Proof of Lemma 3.2. Consider $C - \{e\} \in \mathcal{I}$, and extend this to a basis J of A. We essentially have $J \subseteq A - \{e\}$ because otherwise $J \supseteq C$. Hence

$$r(A - \{e\}) \geq |J| = r(A) \quad \Longrightarrow \quad r(A - \{e\}) = r(A)$$

□

Running Time of Augmenting Path Algorithm for Matroid Intersection There are at most $n = |V|$ iterations, and in each iteration,

- Constructing D, X_2, X_1 takes $O(n^2)$ independence oracle calls;
- Finding the shortest $X_2 \to X_1$ path needs $O(n^2)$ operations, using BFS.

Hence there are $O(n^3)$ independence oracle calls and $O(n^3)$ operations.

3.5.5 Weighted Matroid Intersection

Given $M_1 = (U, \mathcal{I}_1)$ and $M_2 = (U, \mathcal{I}_2)$, and weights $\{w_e\}_{e \in U}$, we wish to find

$$\max_{J \in \mathcal{I}_1 \cap \mathcal{I}_2} w(J)$$

The LP-relaxation for weighted matroid intersection is given as:

$$\max \sum_{e \in E} w_e x_e \;\; s.t. \;\; \begin{array}{rll} x(S) & \leq r_1(S) & \forall S \subseteq U \\ x(S) & \leq r_2(S) & \forall S \subseteq U \\ x & \geq 0 & \end{array} \tag{LP}$$

Note 3.12

(LP) has integral extreme points, so the feasible reagion for it is

$$\text{conv}(\{X^J : J \in \mathcal{I}_1 \cap \mathcal{I}_2\})$$

Here is a cool application: Minimum Degree-bounded Spanning Tree (MDST)
Given graph $G = (V, E)$, edge costs $\{c_e\}_{e \in E}$, and (node) degree bounds $\{b_v\}_{v \in V}$, we wish to find

$$\min c(T) \;\; s.t. \;\; \begin{array}{c} T \text{ is a spanning tree} \\ |\delta_T(v)| \leq b_v \quad \forall v \subseteq V \end{array} \tag{MDST}$$

This problem is NP-hard, but we have the following really nice result.

Theorem 3.12

In polytime, we can determine if the above problem is infeasible, or find a spanning tree T such that $c(T) \leq OPT$ and

(a) $|\delta_T(v)| \leq b_v + 1$ (best possible due to Singh-Lau);

(b) $|\delta_T(v)| \leq b_v + 2$ (due to Goemans '06)

The LP-relaxation for MDST is

$$\min \sum_{e \in E} c_e x_e \quad s.t. \quad \begin{aligned} x(E(S)) &\leq |S| - 1 \\ x(E) &= n - 1 \\ x(\delta(v)) &\leq b_v \quad \forall v \in V \\ x &\geq 0 \end{aligned} \tag{MDST-LP}$$

Let x^* be the optimal solution to (MDST-LP), and define

$$F = \{e \in E : x_e^* > 0\} = \operatorname{supp}(x^*)$$

Here is the idea: orient F to get a digraph D. Suppose we could do this so that $|\delta_D^{in}(v)| \leq k$ for some constant k, then we can define M_1 to be the graphic matroid on F and M_2 the partition matroid that encodes $\leq b_v$ outgoing edges for all $v \in V$. Then x^* is the feasible solution to the resulting weighted matroid intersection problem, so by note 3.12, we can find T such that T is a spanning tree and $|\delta_T^{out}(v)| \leq b_v$ for all v and $c(T) \leq c^{\mathsf{T}} x^* = OPT_{\text{MDST-LP}}$. Note that

$$|\delta_T^{out}(v)| \leq b_v \implies |\delta_T(v)| \leq b_v + k$$

However, here comes the question:

Question 3.2.

Does an orientation D with $|\delta_D^{in}(v)| \leq 2$ exist?

By Hakimi's Theorem 3.9, we can get such a D if and only if

$$|F(S)| \leq 2|S| \qquad \forall S \subseteq V$$

which is done in the form of another matroid intersection problem. Using LP-theory about extreme points, and the fact that tight spanning tree constraints can be assumed to come from a laminar family of tight sets, we are actually able to show that

$$|F(S)| \leq 2|S| \qquad \forall S \subseteq V$$

4 Flows and Cuts

The problems are all about directed graphs unless specified otherwise

Lecture 12 - Thursday, October 23

Let $D = (V, E)$ be a digraph, and $\{u_e\}_{e\in E}$ be some *edge capacities*.

Definition 4.1: Feasible s-t Flow

Let $s, t \in V$. A **feasible s-t flow** is a vector $x \in \mathbb{R}^E$ satisfying

(i) $x(\delta^{in}(v)) - x(\delta^{out}(v)) = 0$ for all $v \in V - \{s, t\}$. We denote the expression as $f_x(v)$, which can be understood as the *net inflow* of node v;

(ii) $0 \le x_e \le u_e$ for all $e \in E$.

The **value** of an s-t flow is $f_x(t)$.

Note 4.1

Easy to observe that $f_x(t) = -f_x(s)$.

Definition 4.2: Feasible Flow

More generally, given node demands $\{b_v\}_{v\in V}$ where $b_v \in \mathbb{R}$, lower bounds $\{\ell_e\}_{e\in E}$, and capacities $\{u_e\}_{e\in E}$, a **feasible flow** is a vector $x \in \mathbb{R}^E$ satisfying $f_x(v) = b_v$ for all v and $\ell_e \le x_e \le u_e$ for all e.

Note 4.2

For feasibility, we must have $\ell_e \le u_e$, and we will allow $u_e = +\infty$ and $\ell_e = -\infty$ as possible values.

4.1 Flow Problems

4.1.1 Maximum s-t Flow

The problem asks to find an s-t flow of maximum value, i.e.,

$$\max\ f_x(t)\ \ s.t.\ \begin{array}{rll} f_x(v) & = 0 & \forall v \in V - \{s,t\} \\ 0 \le x_e & \le u_e & \forall e \in E \end{array} \tag{MF}$$

This is always feasible (as we can just pick the 0 flow). We will often seek a maximum integeral flow.

Comment 4.1

Notice that this is essentially LP, but we want to study if there is a more efficient algorithm solving it instead of treating it as an ordinary LP problem.

4.1.2 (General) Feasible Flows

Given node demands $\{b_v\}_{v\in V}$ where $b_v \in \mathbb{R}$, lower bounds $\{\ell_e\}_{e\in E}$, and capacities $\{u_e\}_{e\in E}$, does there exist a feasible (integral) flow?

Discovery 4.1

We have two necessary conditions: $\ell_e \leq u_e$ for all e and $\sum_{v\in V} b_v = 0$.

4.1.3 Minimum Cost Feasible Flow

Given node demands $\{b_v\}_{v\in V}$ where $b_v \in \mathbb{R}$, lower bounds $\{\ell_e\}_{e\in E}$, capacities $\{u_e\}_{e\in E}$, and some costs $\{c_e\}_{e\in E}$, we want to find a feasible (integral) flow who minimizes the costs.

4.1.4 Useful Special Case of Minimum Cost Feasible Flow, Circulations

1. Given $\{\ell_e, u_e\}_{e\in E}$, does there exist a (integral) circulation x such that $\ell_e \leq x_e \leq u_e$ for all $e \in E$?
2. Given $\{c_e, \ell_e, u_e\}_{e\in E}$, does there exist a min-cost (integral) circulation x such that $\ell_e \leq x_e \leq u_e$ for all $e \in E$?

Definition 4.3: Circulation

A vector $x \in \mathbb{R}^E$ satisfying $f_x(v) = 0$ for all $v \in V$ is called a **circulation**.

For flow feasibility or min-cost flow, we can always assume that all lower boudns are 0. Why? Given $\{b_v\}_{v\in V}$ and $\{\ell_e, u_e\}_{e\in E}$, we can always transfer this problem to another problem with node demands

$$b'_v = b_v - \Big(\ell(\delta^{in}(v)) - \ell(\delta^{out}(v))\Big) = b_v - f_\ell(v)$$

and capacities

$$(\ell'_e, u'_e) = (0, u_e - \ell_e)$$

and we have the following proposition:

Proposition 4.1

x is a feasible solution for $(\{b_v\}_{v\in V}, \{\ell_e, u_e\}_{e\in E})$ if and only if $x' = (x_e - \ell_e)_{e\in E}$ is a feasible solution for $(\{b'_v\}_{v\in V}, \{\ell'_e, u'_e\}_{e\in E})$.

Comment 4.2

Important to note that we cannot assume that the lower bounds for a circulation problem are all zero, because after the transformation shown above, we may not get a circulation problem.

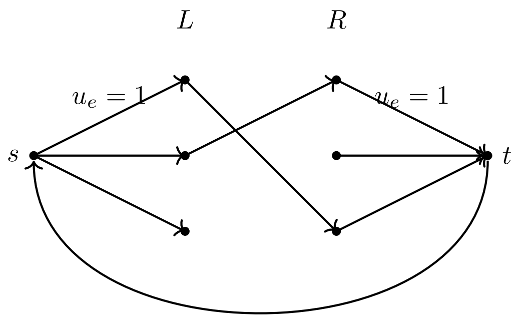
4.1.5 Some Application

Example 4.1: Max-weight bipartite matching

Let $G = (L \cup R, E)$ be a bipartite graph and weights $\{w_e\}_{e \in E}$, we can construct digraph $D = (\{s,t\} \cup L \cup R, E')$ where E' consists of

1. Arcs (s,v) for all $v \in L$ with $u_e = 1$, $c_e = 0$;
2. Arcs (u,t) for all $u \in R$ with $u_e = 1$, $c_e = 0$;
3. Arcs (v,u) for all $uv \in E$, $u \in L, v \in R$ with $u_e = \infty$, $c_e = -w_e$;
4. Arc (t,s) with $u_e = \infty$, $c_e = 0$.

and $\ell_e = 0$ for all $e \in E'$.



Note that now we have

$$\text{matching in } G \;\equiv\; \text{integral circulation in } (D, \{\ell_e, u_e\})$$

$$\text{max weight matching in } G \;\equiv\; \text{min-cost integral circulation in } (D, \{\ell_e, u_e\})$$

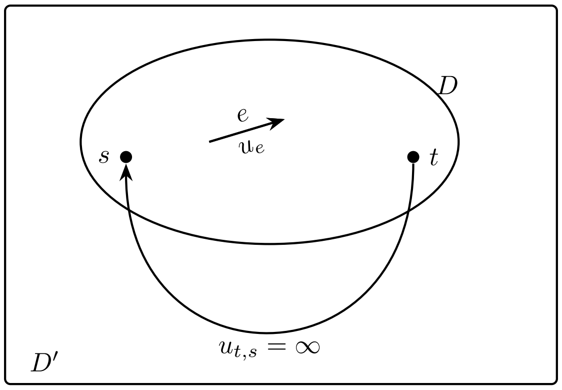
Exercise 4.1

Model min-cost matching of size k as suitable flow problem (assume integrality property of flows).

Exercise 4.2

Model min-cost max-size matching as suitable flow problem (assume integrality property of flows).

Example 4.2: Max s-t flow: special case of min-cost circulation



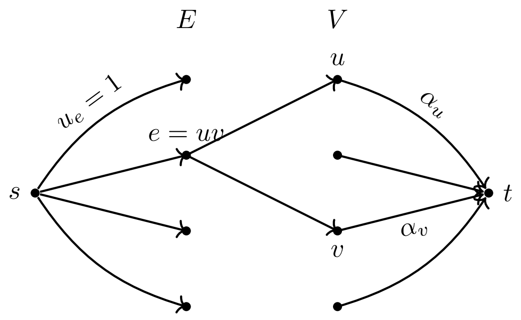
We can simply add the (t,s) arc with $u_{t,s} = +\infty$, all other edges have the same u_e as in max-flow instnace, and all $\ell_e = 0$. Now,

$$x \text{ is a feasible } s\text{-}t \text{ flow in } D \text{ if and only if } (x, x_{t,s} = f_x(t)) \text{ is a circulation in } D'$$

Setting $c_e = 0$ for all $e \in E$, $c_{t,s} = -1$, then the min-cost circulation is equivalent to the max s-t flow.

Example 4.3: Orientations

Recall that given an undirected graph $G = (V, E)$ and integers $\{\alpha_v \geq 0\}_{v \in V}$, an α-orientation of G is the process of directing edges in G such that the in-degree for all v is bounded above by α_v.

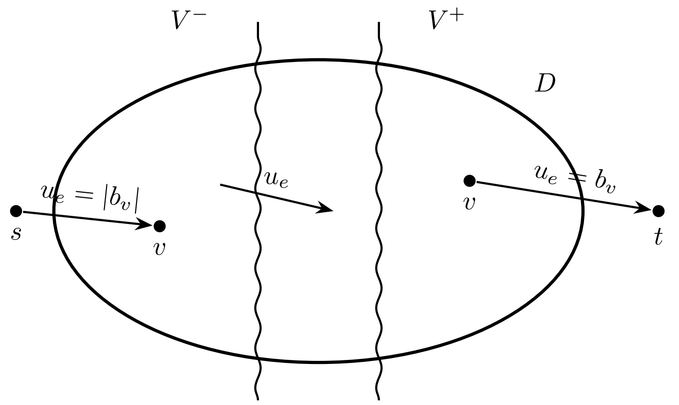


We add a node for each edge $e = uv$ and a node for every $v \in V$. Given the above max s-t flow instance, a orientation of G corresponds to an integer s-t flow such that flow on every (s,e) arc is equal to 1, which corresponds to a max s-t flow value equals to $|E|$ (and integrality property of flows).

Example 4.4: Flow feasibility can be readuced to max s-t flows

Given instance $(D = (V,E), \{b_v\}_{v \in V}, \{\ell_e\}_{e \in E}, \{u_e\}_{e \in E})$ of flow feasibility, we have basic necessary conditions for flow feasibility:

$$u_e \geq \ell_e \quad \forall e \in E \qquad \text{and} \qquad \sum_{v \in V} b_v = 0$$



$$V^- := \{\, v \in V : \; b_v < 0 \,\}$$

$$V^+ := \{\, v \in V : \; b_v > 0 \,\}$$

$$\sum_{v \in V^+} b_v \; = \; - \sum_{v \in V^-} b_v$$

We construct $D' = (\{s,t\} \cup V, E')$, where E' consists of arcs:

1. $e = (u,v)$ for all $e \in E$ with the same capacity $u'_e = u_e$;
2. (s,v) for all $v \in V^-$ with capacity $|b_v|$;
3. (v,t) for all $v \in V^+$ with capacities b_v.

Now we have

$$x \text{ is a feasible flow for } (D,b,u) \iff x' = \Big(x, \{x'_{sv} = |b_v|\}_{v\in V^-}, \{x'_{vt} = b_v\}_{v \in V^+}\Big) \text{ is a feasible } s\text{-}t \text{ flow in } (D',u')$$

Such a feasible s-t flow in (D',u') must be a max s-t flow and have value $= \sum_{v\in V^+} b_v + \sum_{v \in V^-} |b_v|$, so (D,b,u) has a feasible flow if and only if max s-t flow value in (D',u') is equal to $\sum_{v\in V^+} b_v$.

4.2 Max s-t Flow and Minimum s-t Cut

Lecture 13 - Tuesday, October 28

Let $\Big(D = (V,E),\ s,t \in V,\ \{u_e \geq 0\}_{e\in E}\Big)$ be an instance of max s-t flow.

Definition 4.4: s-t Cut

An **s-t cut** is an edge-set of the form $\delta^{out}(Z)$ where $s \in Z$ and $t \notin Z$.

Definition 4.5: Capacity

The **capacity of an s-t cut** $\delta^{out}(Z)$ is defined as

$$u\Big(\delta^{out}(Z)\Big) := \sum_{e \in \delta^{out}(Z)} u_e$$

Lemma 4.1

Let x be an s-t flow and $\delta^{out}(Z)$ be an s-t cut, then

$$f_x(t) = x\Big(\delta^{out}(Z)\Big) - x\Big(\delta^{in}(Z)\Big) \leq u\Big(\delta^{out}(Z)\Big)$$

Proof. We have

$$\begin{aligned} f_x(t) = \sum_{v \in V-Z} f_x(v) &= x\Big(\delta^{in}(V-Z)\Big) - x\Big(\delta^{out}(V-Z)\Big) \\ &= x\Big(\delta^{out}(Z)\Big) - x\Big(\delta^{in}(Z)\Big) \end{aligned}$$

as desired. □

4.2.1 Min s-t Cut Problem

Find an s-t cut $\delta^{out}(Z)$ that minimizes $u\Big(\delta^{out}(Z)\Big)$. By Lemma 4.1, we have

$$\text{max } s\text{-}t \text{ flow value } \leq \text{ capacity of min } s\text{-}t \text{ cut value}$$

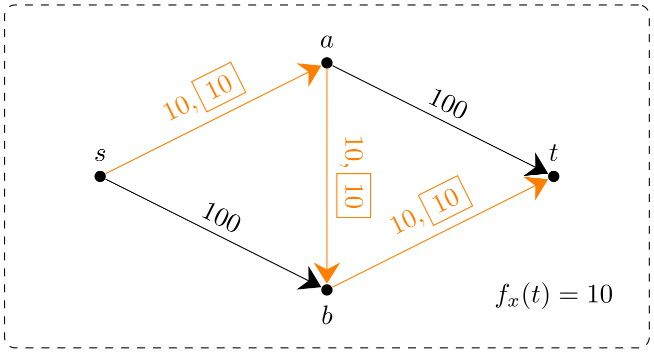
Theorem 4.1: Max-flow Min-cut Theorem

We in fact have

$$\text{max } s\text{-}t \text{ flow value } = \text{ capacity of min } s\text{-}t \text{ cut value}$$

4.2.2 Algorithm for Max s-t Flows

The essence of the algorithm is *Residual/ Auxilliary Diagraph and Augmenting Paths.*

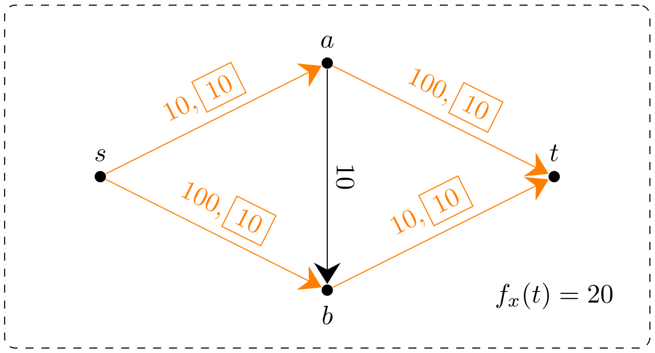


(∗)

See the above picture as an example. To get a better flow, we can:

1. Push "forward" (i.e., $\uparrow$) 10 units on s, b;
2. Push "backward" 10 units on b, a;
3. Push "forward" 10 units on a, t.

to obtain



Definition 4.6: Residual/ Auxilliary Digraph

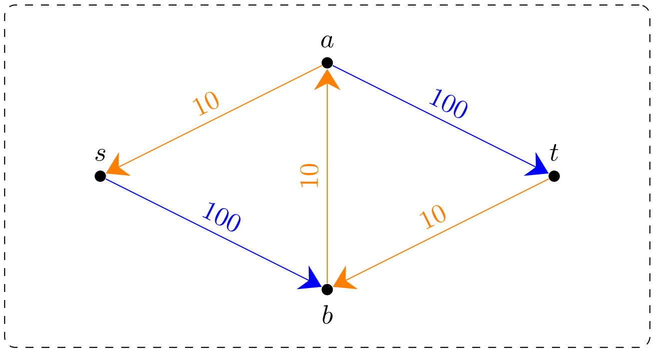
Given an s-t flow instance $\Big(D = (V, E),\ s, t,\ \{u_e\}\Big)$ and a s-t flow x, the **residual/ auxilliary digraph w.r.t.** x, denoted as $D(x)$, is defined as follows: $D(x) = (V, E(x))$, where

1. for every $(a, b) \in E$ with $x_{a,b} < u_{a,b}$, we include $(a, b) \in E(x)$ and give it capacity $= u_{a,b} - x_{a,b}$. We call $(a, b) \in E(x)$ a *forward arc.*

2. for every $(a, b) \in E$ with $x_{a,b} > 0$, we include $(b, a) \in E(x)$ and give it capacity $= x_{a,b}$. We call $(b, a) \in E(x)$ a *backward arc.*

Example 4.5

Consider the example above (example $(*)$) again, we have



where the orange arcs are backward arcs and blue arcs are backward arcs. We call the capacity of an arc in $D(x)$ its **residual capacity**.

Comment 4.3

Residual digraph and augmenting paths generalizes what we saw for bipartite matchings.

Lemma 4.2

Let x be an s-t flow in $\Big(D = (V, E),\ s, t,\ \{u_e\}\Big)$. Let P be a s-t path in $D(x)$, which is called an x-augmenting path. Let $\gamma(P)$ be the minimum residual capacity of an edge in P,

$$\gamma(P) = \min\Big\{\min\{u_e - x_e : e \text{ an forward arc in } P\},\ \min\{x_{a,b} : (b, a) \text{ an backward arc in } P\}\Big\}$$

define $x' \in \mathbb{R}^E$ as follows:

$$x'_{a,b} = \begin{cases} x_{a,b} + \gamma(P) & \text{if } (a, b) \in P \text{ and is a forward arc} \\ x_{a,b} - \gamma(P) & \text{if } (b, a) \in P \text{ and is a backward arc} \\ x_{a,b} & \text{otherwise} \end{cases}$$

Then x' is an s-t flow, and

$$f_{x'}(t) = f_x(t) + \gamma(P)$$

Proof.

▌ This proof is left as an exercise, the argument below is from CS341.

We first have several things to check, in order to show that x' is still an s-t flow:

1. `Are capacity constraints satisfied?` We add/ subtract $\gamma(P)$ to/ from each edge, where $\gamma(P)$ is the minimum of the smallest remaining capacity, and the current flow.

2. `What about conservation of flow?` Consider how the flow into and out of each vertex $u \in V - \{s, t\}$ changes as a result of running augmenting, we show change in $f^{in}(u) =$ change in $f^{out}(u)$. There are four cases, depending on whether the edges of entering/ leaving u are forward/backward:

Case 1: [Forward and forward] Both $x'\Big(\delta^{in}(u)\Big)$ and $x'\Big(\delta^{out}(u)\Big)$ are increased by $\gamma(P)$;

Case 2: [Backward and backward] This case is similar to the case above;

Case 3: [Forward and backward] Both $x'\Big(\delta^{in}(u)\Big)$ and $x'\Big(\delta^{out}(u)\Big)$ are increased by $\gamma(P)$;

Case 4: [Forward and forward] This case is similar to the case above.

3. `Does source` s `still have no flow into it?` Since x is a flow, $x(\delta^{in}(s)) = 0$. To get $x'(\delta^{in}(s))$, an augmenting path must include an edge into s, but now an augmenting path starts at s, then returns to s, forming a cycle, which is a contradiction.

4. `Does sink` t `still have no flow out of it?` Similar argument as above.

Now it suffices to show that

$$f_{x'}(t) = f_x(t) + \gamma(P)$$

Notice that this is equivalent to saying that

$$x'(\delta^{out}(s)) = x(\delta^{out}(s)) + \gamma(P)$$

Think about what really is an augmenting path? It is a path that comes out of the source and ends at the sink, and it is necessarily starting on a forward edge, hence the extra flow that goes through the augmenting path P is added to the original flow x, which implies what we want. □

Theorem 4.2

x is a maximum s-t flow if and only if there exists no augmenting path in $D(x)$.

Proof. Forward direction follows from Lemma 4.2. Conversely, suppose there is no augmenting path, so if we take

$$Z = \{v \in V : \exists s \rightsquigarrow v \text{ path in } D(x)\}$$

then $s \in Z$, $t \notin Z$, and $\delta^{out}_{D(x)}(Z) = \emptyset$. Consider an edge $(a, b) \in \delta^{in}_D(Z) \cup \delta^{out}_D(Z)$:

1. If $(a, b) \in \delta^{out}_D(Z)$. Because (a, b) does not appear in the residual digraph, we must have that $x_{a,b} = u_{a,b}$.

2. If $(a, b) \in \delta^{in}_D(Z)$. Since (b, a) is not a backward arc in $D(x)$, we must have $x_{a,b} = 0$.

Upshot: Arcs in $\delta_D^{out}(Z)$ are saturateed ($x_e = u_e$), and arcs in $\delta_D^{in}(Z)$ carry 0 flow ($x_e = 0$). Hence

$$\begin{aligned} f_x(t) &= x\Big(\delta^{out}(Z)\Big) - \underbrace{x\Big(\delta^{in}(Z)\Big)}_{=0} \\ &= u\Big(\delta^{out}(Z)\Big) \end{aligned}$$

So x is the max s-t flow, and $\delta_D^{out}(Z)$ is the min s-t cut. □

Proof of Theorem 4.1. Max s-t flow always exists since the max s-t flow problem is an LP that is feasible, not unbounded. Therefore, by theorem 4.2, we get

$$\text{max } s\text{-}t \text{ flow value} \;=\; \text{capacity of min } s\text{-}t \text{ cut value}$$

done. □

Exercise 4.3

Use Max-flow Min-cut Theorem to prove

(a) König's Theorem, and

(b) Hakimi's Theorem

4.2.3 Application of MFMC Theorem

Theorem 4.3

Let $\mathcal{I} = \Big(D = (V, E),\ \{b_v\}_{v\in V},\ \{\ell_e, u_e\}_{e\in E}\Big)$ be a flow-feasibility instance. A feasible flow exists for the above instance if and only if

(1) $b(V) = 0$, and $\ell_e \le u_e$ for all e;

(2) $b(X) \le u\Big(\delta^{in}(X)\Big) - \ell\Big(\delta^{out}(X)\Big)$ for all $X \subseteq V$.

Proof. Assuming that (1) holds, we will that that a feasible flow exists if and only if (2) holds.

A feasible flow exists for $\mathcal{I}$ if and only if a feasible flow exists for $\mathcal{I}'$ defined as

$$\mathcal{I}' = \Big(D = (V, E),\ \{b'_v := b_v - f_\ell(v)\}_{v\in V},\ \{\ell'_e = 0, u'_e := u_e - \ell_e\}_{e\in E}\Big)$$

Recall, we showed that for instance $\mathcal{I}'$ shown above, we have that

$$\begin{aligned} \mathcal{I}' \text{ has a feasible flow} \quad &\iff \quad \text{Max } s\text{-}t \text{ flow value for } \mathcal{I}' \;= \sum_{v\in V^+} b'_v \\ &\iff \quad \text{Min } s\text{-}t \text{ cut capacity in } \mathcal{I}' \;= b'(V^+) \\ &\iff \quad \text{Min } s\text{-}t \text{ cut capacity in } \mathcal{I}' \;\ge b'(V^+) \end{aligned}$$

(See the definition of V^+ on page 42). Consider an s-t cut $\delta_{D'}^{out}(Z)$, where $s \in Z, t \notin Z$. Let $X = V - Z$, so

$$u'\Big(\delta_{D'}^{out}(Z)\Big) = u'\Big(\delta^{in}(Z)\Big) + b'(V^+ - C) + \Big(-b'(V^- \cap X)\Big)$$

so we have

$$\begin{aligned} & u'\Big(\delta_{D'}^{out}(Z)\Big) \geq b(V^+) \\ \equiv\ & u\Big(\delta^{in}(X)\Big) - \ell\Big(\delta^{in}(X)\Big) \geq b'(V^+ \cap X) + b'(V^- \cap X) \\ \equiv\ & u\Big(\delta^{in}(X)\Big) - \ell\Big(\delta^{in}(X)\Big) \geq b'(X) \\ \equiv\ & u\Big(\delta^{in}(X)\Big) - \ell\Big(\delta^{in}(X)\Big) \geq b(X) - \Big[\ell(\delta^{in}(X)) - \ell(\delta^{out}(X))\Big] \end{aligned}$$

In other words, $u(\delta^{in}(X)) - \ell(\delta^{out}(X)) \geq b(X)$. Therefore, we conclude that $\mathcal{I}$ has a feasible flow if and only if $u(\delta^{in}(X)) - \ell(\delta^{out}(X)) \geq b(X)$ for any $X \subseteq V$. □

Lecture 14 - Thursday, October 30

4.2.4 Ford-Fulkerson (FF) Algorithm (1956) for Max s-t Flow

We establish the following algorithm:

```
1 Start with $x \leftarrow 0$;
2 while $\exists$ augmenting path in $D(x)$ do
3     Pick an augmenting path $P$;
4     Update $x$ by augmenting along $P$;
5     Update $D(x)$;
6 return $x$;
```

Algorithm 4: Ford-Fulkerson Algorithm (1956)

We care about the runtime of the above algorithm. We note that termination always occurs with:

- integer capacities, in which case we will maintain an integral flow, which implies that we have an integral max s-t flow;
- rational capacities;

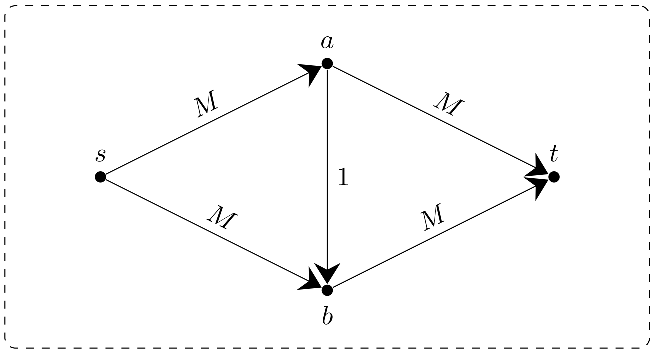
However, with irrational capacities, termintation need not occur, and limit of s-t flow need not be max s-t flow.

Note 4.3

It is important to point out that even when termination happens, but it needs not happen in poly-time. Here is an example:

Example 4.6

Suppose we have the capacities shown below:



where $M \gg 1$ is an integer. Choosing P containing (a, b) or (b, a) in every iteration will requite $2M$ iterations, which is not poly-time.

Hence, we need to be careful in choosing augmenting paths:

- Choose P with maximum $\gamma(P)$;
- Shortest-Path Rule: Choose P with fewest number of arcs[1];

Theorem 4.4: Edmonds-Karp 1972, Dinitz 1970

Ford-Fulkerson with the Shortest-Path rule terminates in $\leq m \cdot n$ iterations.

Note 4.4

Note that a shortest augmenting path can be found using BFS in $O(m + n)$ time, so the running time of Ford-Fulkerson with the Shortest Path rule is $O(mn(m + n))$.

Proof of Theorem 4.4. Let $D = (V, E, s, t, \{u_e\})$ be a max s-t flow instance and let x be the current s-t flow. Let $D(x)$ be the residual digraph w.r.t. x. Suppose we choose augmenting path P in $D(x)$. Let x' be the new s-t flow after augmenting along P. How does look $D(x') = (V, E(x'))$ compare with $D(x)$?

`[Observation]:`

1. $E(x') = \underbrace{\{(a, b) \in E(x) : \text{ neither } (a, b) \text{ nor } (b, a) \text{ is in } P\ \}}_{E_1} \cup \underbrace{\{(b, a) : (a, b) \in P\}}_{E_2} \cup$ strict subset of P.

   We note that

   (a) Flow values on arcs of D corresponding to E_1 does not change;

   (b) In the E_2 perspective, it means that change in flow on arcs of D due to P can always be reversed in the next iteration;

   (c) The reason why there is a strict subset is because for every P, there is some edges of P, corresponding to arc that determines $\gamma(P)$, that does not appear in $D(x')$. We call thsoe arcs **critical arcs of the current interation**.

[1] Ford-Fulkerson Algorithm with the Shortest-Path-Rule is known as the Edmonds-Karp Algorithm

We define some notations: let

$$\ell_x(u,v) := \text{ the least number of arcs on a } u \rightsquigarrow v \text{ path in } D(x)$$

this value is ∞ if there exists no $u \rightsquigarrow v$ path. $\boxed{\text{Claim 1}}$: we will show later that in the new flow x', the shortest path between s-t does not increase:

$$\ell_x(s,t) \geq \ell_{x'}(s,t)$$

We call it a phase, a maximal sequence of iterations where $\ell_x(s,t)$ does not change. $\boxed{\text{Claim 2}}$: we will show that a phase lasts for $\leq m$ iterations. Merging the above two results, we will have that: the number of phases is $\leq n$, and so the number of iterations is $\leq nm$.

Definition 4.7: Feasible Node Potential

Let $G = (V, E)$ be a digraph, $\{c_e\}_{e\in E}$ be edge costs. We say $\{y_v\}_{v\in V}$ are **feasible node potentials** if

$$y_v - y_u \leq c_{u,v} \qquad \forall (u,v) \in E$$

Fact 1: Consider any $a \rightsquigarrow b$ path P:

$$a\bullet \longrightarrow \bullet \longrightarrow \cdots \longrightarrow \bullet b$$

Adding all the potential differences of edges in P gives us $c(P) \geq y_b - y_a$. Therefore, the shortest a-b path length under $\{c_e\}_{e\in E} \geq y_b - y_a$.

Fact 2: Let $G = (V, E)$ be a digraph, $\{c_e\}_{e\in E}$ be edge costs. Suppose G has no negative-cost cycles:

(a) Then setting $y_v = \min\{c(P) : P \text{ is an } s \to v \text{ path for all } v\}$ gives feasible node potentials.

Analogously, if we define $\overline{V} := \{v \in V : \exists s \to v \text{ path in } G\}$, then setting $y_v = \min\{c(P) : P \text{ is an } s \to v \text{ path for all } v \in \overline{V}\}$ gives a feasible node potentials for $(\overline{V}, \{(u,v) : u, v \in \overline{V}\})$.

(b) Setting $y_v = -\min\{c(P) : P \text{ is an } v \to t \text{ path}\}$ for all $v \in V'' = \{v \in V : \exists\, v \to t \text{ path}\}$ gives feasible node potentials for $(V'', \{(u,v) \in E : u, v \in V''\})$.

`[Observation]`: If v does not lie on a s-t path in $D(x)$, then it does not lie on a s-t path in $D(x')$. Hence we will only consider $\overline{D}(x)$ = portion of $D(x)$ consisting of nodes that lie on some s-t path in $D(x)$.

Now we have everything we need. Let x be the current s-t flow. Let P be the shortest s-t path in $D(x)$, and let x' be the new s-t flow after augmenting along P. We have the following lemma:

Lemma 4.3

We have

1. $\ell_{x'}(s,v) \geq \ell_x(s,v)$ for all $v \in V$;
2. $\ell_{x'}(v,t) \geq \ell_x(v,t)$ for all $v \in V$;

Proof. For (1), consider nodes reachable from s in $D(x')$. By Fact 2(a) above, setting $y_v = \ell_x(s,v)$ gives feasible node potentials for $\overline{D}(x)$ with edge costs $= 1$. We claim y remains feasible node potential for $\overline{D}(x')$ (with edge costs $= 1$). This suggests that

$$\ell_{x'}(s,v) \geq y_v - y_s = \ell_x(s,v)$$

To show the claim, we only need to consider an edge $e = (a,b) \in E(x') - E(x)$ (which is essentially a backward egde). But then $(b,a) \in P$, so $y_a - y_b \leq 1$. Also,

$$y_a = \ell_x(s,a) = 1 + \ell_x(s,b) = y_b$$

Since P is the shortest $s \to t$ path in $D(x)$, so $y_b - y_a = -1 \leq 1$, so for $e = (a,b)$s feasibility condition of no potentials to holds.

For (2), similar argument using node potentials from fact 2(b). This is left as an exercise. □

The above Lemma proves the first claim, so it now suffices to prove the second claim. We have another Lemma:

Lemma 4.4

Consider any arc $(a,b) \in E$, then in any phase, (a,b) or (b,a) together can be critical for at most 1 iteration of the phase.

Proof. Consider the first iteration, call it i, of the phase, where one of (a,b) or (b,a) is a critical arc of that iteration. Let $\overline{x} = x^{(i)}$ be the flow at the start of iteration, and $\overline{P}$ be the augmenting path in $D(\overline{x})$. Let $\overline{\overline{x}} = x^{(i+1)}$ be the flow at the end of the iteration. Say (a,b) is the critical arc in iteration i. For (a,b) or (b,a) to be critical in some later iteration of the phase, we must have that in some later iteration, the augmenting path Q chosen contains the arc (b,a). Let x'' be the flow at the start of the iteration when Q is chosen, we know

$$\begin{aligned} \ell_{x''}(s,t) &= \ell_{x''}(s,b) + 1 + \ell_{x''}(a,t) \\ &\geq \ell_{\overline{x}}(s,b) + 1 + \ell_{\overline{x}}(a,t) \end{aligned}$$

Also $\ell_{\overline{x}}(s,b) = 1 + \ell_{\overline{x}}(s,a)$, so

$$\begin{aligned} \ell_{x''}(s,t) &\geq 2 + \ell_{\overline{x}}(s,a) + \ell_{\overline{x}}(a,t) \\ &> \ell_{\overline{x}}(s,t) \end{aligned}$$

which contradicts that these two iterations are part of the same phase. □

The above Lemma implies that a phase lasts for at most m iterations. Hence we are done proving the Theorem. □

4.3 Min-Cut Problem

Lecture 15 - Tuesday, November 11

Given an undirected graph $G = (V, E)$, non-negative, integer costs $\{c_e \geq 0\}_{e \in E}$, the min-cut problem asks to find a set $S \neq \varnothing$, $S \subsetneq V$ that minimizes $c(\delta(S))$.

Comment 4.4

The instance (G, c) is equivalent to a unweighted multigraph such that edge $(u, v) \in G$ with cost c_e can be viewed a set of c_e edges connecting u and v.

4.3.1 Randomized Algorithm CONTRACT due to Karger 1993'

The algorithm is based on contracting an edge.

Note 4.5

Recall that contracting an edge $e = ab$ means: (1) identify a and b into a single node, (2) throw out self loops, and (3) keep parallel edges.

We first establish the algorithm:

```
Input: Unweighted multigraph $G = (V, E)$
1 $H \leftarrow G$;
2 while $H$ has more than two nodes do
3     Choose an edge $e = xy$ uniformly at random from $H$;
4     $H \leftarrow H/xy$        // $H$ after contracting $xy$
5 Let $(S, V - S)$ be the node-sets corresponding to the two nodes left;
6 return $\delta(S)$.
```

Algorithm 5: CONTRACT

Proposition 4.2

We have the following observations:

1. A cut $\delta(S)$ is returned if and only if none of the edges in $\delta(S)$ is contracted by CONTRACT;
2. If the min-cut value is k, then $|\delta(v)| \geq k$ for all v, and this implies that the number of edges $\geq nk/2$, where $n = |V|$;
3. Min-cut value does not decrease after we contract an edge.

Theorem 4.5

Let $\delta(S)$ be a specific min-cut, then

$$\Pr\Big[\text{CONTRACT returns } \delta(S)\Big] \geq \frac{2}{n^2}$$

Proof. Consider the i^{th} iteration of CONTRACT, when we have $n_i = n - i + 1$ nodes in H. We have

$$\Pr\Big[\text{CONTRACT contracts an edge from } \delta(S) \Big| \substack{\text{CONTRACT did not pick an edge} \\ \text{to contract in any of the previous iterations}}\Big] = \frac{|\delta(S)|}{|E(H)|}$$

Let $k = |\delta_G(S)|$. Then, given that we did not contract an edge from $\delta(S)$ in iterations $1, \ldots, i-1$, min-cut value in H is also k (by 4.2(c)). Moreover, we know that $|E(H)| \geq n_i k/2$ by 4.2(b), so

$$\Pr\Big[\text{CONTRACT contracts an edge from } \delta(S) \Big| \substack{\text{CONTRACT did not pick an edge} \\ \text{to contract in any of the previous iterations}}\Big] \leq \frac{2}{n_i}$$

Therefore,

$$\begin{aligned}\Pr\Big[\text{CONTRACT returns } \delta(S)\Big] &\geq \left(1 - \frac{2}{n}\right)\left(1 - \frac{2}{n-1}\right) \cdots \left(1 - \frac{2}{3}\right) \\ &= \frac{2}{n(n-1)} \geq \frac{2}{n^2}\end{aligned}$$

as desired. □

Corollary 4.1

There are at most $\binom{n}{2}$ min-cuts in any graph G, and if we run CONTRACT $\binom{n}{2} \log n$ times, we will see every min-cut of G with high probability.

Proof. The probability of CONTRACT return a specific min cut is over $2/n^2$, and the probability of CONTRACT yields an arbitrary min-cut is $[2/n^2 \cdot \text{number of min-cuts}]$, which cannot exceed 1. □

4.3.2 Picking an Edge (of Integer Cost $c_e \geq 0$) Arbitrarily at Random

1. Pick a node x with probability
$$\frac{c(\delta(x))}{\sum_{v \in V} c(\delta(v))}$$
2. Pick $e \in \delta(x)$ with probability
$$\frac{c_e}{c(\delta(x))}$$

The above choosing process uses only polynomial number random bits.

4.3.3 More Applications

Application 1: Optimal Closure Problem Given a digraph $D = (V, E)$ and (could be negative) rewards $\{\pi_v\}_{v \in V}$, we wish to find a set $A \subseteq V$ with $\delta_D^{out}(A) = \varnothing$ (what we call a **closure**) that maximizes $\pi(A)$.

Define

$$V^+ = \{v \in V : \pi_v > 0\}$$
$$V^- = \{v \in V : \pi_v < 0\}$$

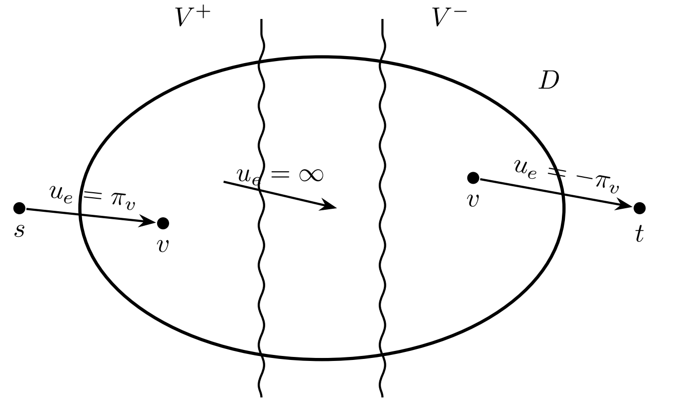
and now

$$\max \pi(A) \equiv \min\Big(\pi(V^+) - \pi(A)\Big) = \underbrace{\sum_{v \in V^+ - A} \pi_v + \sum_{v \in V^- \cap A} (-\pi_v)}_{:= \pi'(A)}$$

We will create min s-t cut instance $(D', s, t, \{u_e\})$ such that

$$A \subseteq V \text{ is a closure} \iff \begin{array}{l} (1)\ \ \delta_{D'}^{out}(A \cup \{s\}) \text{ is an } s\text{-}t \text{ cut with finite capacity} \\ (2) \quad \text{and } u\left(\delta_{D'}^{out}(A \cup \{s\}) = \pi'(A)\right) \end{array}$$

Therefore, solving the min s-t cut on $(D', s, t, \{u_e\})$ solves the optimal closure problem. Here is how we construct the min s-t cut instance:



(s, v) with $u_{(s,v)} = \pi_v$ for all $v \in V^+$

(v, t) with $u_{(v,t)} = -\pi_v$ for all $v \in V^-$

$u_e = \infty$ for all $e \in E$

Note 4.6

In the constructions of the directed egdes, the first two constructions take care of the second requirement, while the third takes care of the first requirement.

Matrix Rounding Given a matrix A, we wish to round entries of A to obtain $\widetilde{A} = \{\widetilde{a}_{ij}\}_{i,j}$ such that

1. $\widetilde{a}_{ij} \in \{\lfloor a_{ij} \rfloor, \lceil a_{ij} \rceil\}$;
2. $\widetilde{R}_i = \sum_j \widetilde{a}_{ij} \in \{\lfloor R_j \rfloor, \lceil R_j \rceil\}$;
3. $\widetilde{C}_i = \sum_i \widetilde{a}_{ij} \in \{\lfloor C_i \rfloor, \lceil C_i \rceil\}$.
4. $\widetilde{T} = \sum_{i,j} \widetilde{a}_{ij} \in \{\lfloor T \rfloor, \lceil T \rceil\}$.

Comment 4.5

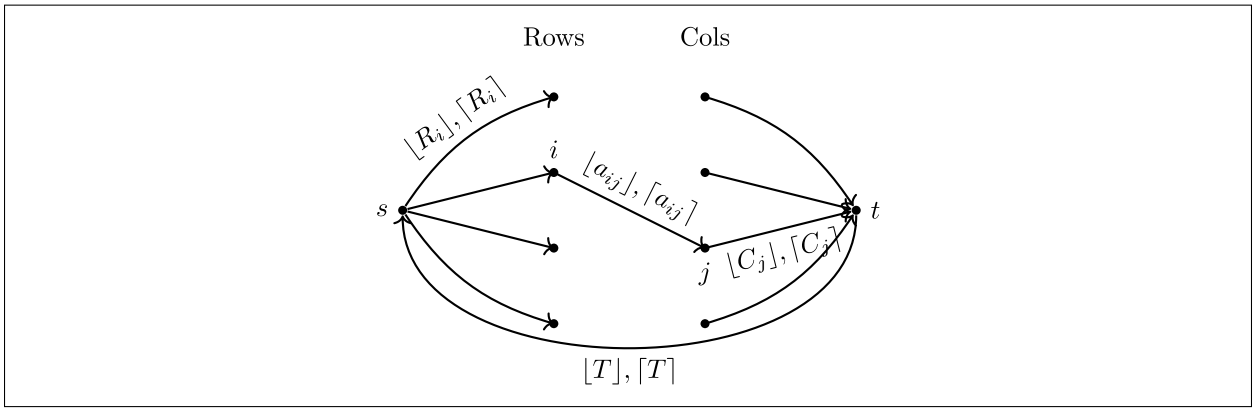
R_i and $\widetilde{R}_i$ denotes the sum of the i^{th} row of matrix A and $\widetilde{A}$ respectively. Similar, C denotes the column sum and T denotes the total sum (i.e., the sum of all the entries).

Lecture 16 - Thursday, November 13

Theorem 4.6

A rounded matrix always exists and can be found by solving a flow feasible problem.

Proof. Consider the following flow network:


Rows
Cols
$\lfloor R_i \rfloor, \lceil R_i \rceil$
i
$\lfloor a_{ij} \rfloor, \lceil a_{ij} \rceil$
s
t
j
$\lfloor C_j \rfloor, \lceil C_j \rceil$
$\lfloor T \rfloor, \lceil T \rceil$

An integer feasible flow is equivalent to a rounded matrix $\widetilde{A}$, and

$$\text{Feasible flow} \iff \text{Integral feasible flow}$$

since all ℓ_e and u_e and b_v are integers. Moreover, we know that a feasible flow exists because we can use the input matrix A to produce a feasible flow. $\square$

4.4 Min-Cost Flows

The question is worded as follows: Given a digraph $D = (V, E)$, node demands $\{b_v\}_{v\in V}$, $\{\ell_e \leq u_e\}_{e\in E}$, costs $\{c_e\}_{e\in E}$, we wish to find a min-cost feasible flow.

Comment 4.6

We assume that $\ell_e = 0$ (we have shown that we can assume this).

To find the min-cost feasible flow is essentially to solve the following LP:

$$\min \sum_{e\in E} c_e x_e \quad s.t. \quad \begin{array}{rll} \underbrace{x(\delta^{in}(v)) - x(\delta^{out}(v))}_{=:f_x(v)} & = b_v & \forall v \in V \\ 0 \leq x_e & \leq 1 & \forall e \in E \end{array} \tag{P}$$

Comment 4.7

We also assume that all u_es are finite and > 0.

Optimality Conditions: Finding the dual of (P), we obtain

$$\max \sum_{v\in V} b_v y_v - \sum_{e\in E} u_e z_e \quad s.t. \quad \begin{array}{rll} y_v - y_u - z_{u,v} & \leq c_{u,v} & \forall (u,v) \in E \\ z & \geq 0 & \end{array} \tag{D}$$

Note 4.7

Given any $y \in \mathbb{R}^V$, we can optimally complete it to (y, z) that is feasible to (D) by setting $z_{u,v} = \max\{0, y_v - y_u - c_{u,v}\}$ for all $(u, v) \in E$.

Question 4.1.

We want to ask when is x^* (feasible to (P)) optimal?

By LP theory, we know that x^* is optimal to (P) if and only if there exists (y^*, z^*) feasibile to (D) such that $x^*, (y^*, z^*)$ satisfy Complementary Slackness conditions. But here comes another question:

Question 4.2.

When does there exist y^* and z^* defined as established in note 4.7 such that Complementary Slackness conditions are met?

From how we established z^*, we can infer that if

$$y_v^* - y_u^* \geq c_{u,v}$$

then the first constraint will be tight for (u, v). Therefore, it is tight for (u, v) if and only if $y_v^* - y_u^* \geq c_{u,v}$.

Recall Complementary Slackness conditions:

- $x^*_{u,v} > 0$, then first constraint tight for (u, v), which happens if and only if $y^*_v - y^*_u \geq c_{u,v}$.
- $z^*_{u,v} > 0$, then $x^*_{u,v} = u_{u,v}$, i.e., $y^*_v - y^*_u \geq c_{u,v}$, and so $x^*_{u,v} = u_{u,v}$.

Interpret these two conditions:

- $x^*_{u,v} > 0$ implies that $y^*_u - y^*_v \leq -c_{u,v}$;
- If node-potential feasibility condition is violated for $(u, v) \in E$, then $(u, v) \notin D(x^*)$ (which is equivalent to $x^*_{u,v} = u_{u,v}$).

Consider the following edge costs for $D(x)$: Give edge (a, b) in $D(x)$ a cost of:

1. $c_{a,b}$, if (a, b) is a foward edge;
2. $-c_{b,a}$, if (a, b) is a backward edge.

(denote these edge costs as c'_e.) Then y^* is a feasible node potential for $(D(x^*), c')$.

Theorem 4.7

TFAE:

1. A feasible flows x^* is optimal to (P);
2. $(D(x^*), c')$ has feasible node potentials;
3. $(D(x^*), c')$ has no negative cycles;

Proof. The equivalence of statements 2 and 3 is omitted. □

4.4.1 Cycle Cancelling Algorithm

The above theorem 4.7 leads to the following generic cycle cancelling algorithm:

```
1 Start with a feasible flow $x$ (we know how to compute this by solving max $s$-$t$ flow problem);
2 while $(D(x), c')$ has a negative cycle do
3     Find such a negative cycle $Z$;
4     Update $x$ by pushing flow along $Z$;
5 return $x$;
```

Algorithm 6: Cycle Cancelling Algorithm (CCA)

Comment 4.8

For line 4 in the above algorithm:

1. $\uparrow$ flow on edges of $D \equiv$ forward edges on Z;
2. $\downarrow$ flow on edges of $D \equiv$ backward edges on Z;

The amount by which the flow changed is $\gamma(Z)$; Hence the change in cost is $\gamma(Z) \cdot c'(Z)$.

How to identify a negative cycle to turn CCA into an efficient algorithm?

First attempt: Find the most negative cycle (i.e., cycle Z with smallest $c'(Z)$).

We have two issues:

1. For special case of max s-t flow, this transfers to choosing *any* augmenting path, which we do not get termination in polytime;
2. Find the most negative cycle is NP-hard.

Second attempt: Find cycle with $c'(Z) < 0$ that maximizes $|\gamma(Z) \cdot c'(Z)|$.

Although we can get termination in polytime, but finding such a cycle is NP-hard.

Third attempt: Minimum mean-cost cycle rule. Find cycle with the smallest mean cost, i.e., cycle Z such that minimizes $\frac{c'(Z)}{|Z|}$.

Note 4.8

For max s-t flow, this precisely corresponds to the shortest augmenting path.

4.4.2 Minimum Mean-Cost Cycle Rule

We will prove that the min-cycle rule works.

Definition 4.8:

We define

$$\mu(x) = \min_{\text{cycle } Z \text{ in } D(x)} \frac{c'(Z)}{|Z|}$$

We call x is ε-optimal for some $\varepsilon > 0$ if

$$(D(x, \{c'_e + \varepsilon\}_{e \in E(x)}))$$

has no negative cycles. We write $\varepsilon(x)$ to be the smallest $\varepsilon > 0$ such that x is ε-optimal.

Lemma 4.5

If $\mu(x) < 0$, then $\mu(x) = -\varepsilon(x)$.

Proof. We have

$$\begin{aligned}
\varepsilon \geq \varepsilon(x) &\iff \varepsilon \geq 0, x \text{ is } \varepsilon\text{-optimal} \\
&\iff \varepsilon \geq 0, c'(Z) + \varepsilon|Z| \geq 0 \quad \forall \text{ cycle Z in D(x)} \\
&\iff \varepsilon \geq 0, \varepsilon \geq \frac{-c'(Z)}{|Z|} \quad \forall \text{ cycle Z in D(x)} \\
&\iff \varepsilon \geq -\mu(x)
\end{aligned}$$

and so $\varepsilon(x) = -\mu(x)$. □

Lecture 17 - Monday, November 17

Fact (we take the following result as a fact): Given $(G, \{w_e\}_{e \in E(G)})$, we can efficiently find a negative cycle, if one exists, or find feasible potentials.

Note 4.9

We can use binary search to find $\varepsilon(x)$. We need upper bound on $\varepsilon(x)$ (note that we already have lower bound):

$$\varepsilon(x) \le C := \max_{e \in E(x)} |c'_e|$$

since $c'_e + C \ge 0$ for all $e \in E(x)$.

▌ We also assume that all edge costs are integers.

Note 4.10

Note that if Z_1 and Z_2 are such that $\frac{c'(Z_1)}{|Z_1|} \neq \frac{c'(Z_2)}{|Z_2|}$, then $\left|\frac{c'(Z_1)}{|Z_1|} - \frac{c'(Z_2)}{|Z_2|}\right| > \frac{1}{n^2}$. Hence, if we find an interval $(\varepsilon_1, \varepsilon_2)$ such that $\varepsilon_1 < \varepsilon(x) \le \varepsilon_2$ and $\varepsilon_2 - \varepsilon_1 \le 1/n^2$, then we must have $\varepsilon(x) = \varepsilon_2$. This implies that we can find $\varepsilon(x)$ using binary search in $O(\log(n^2 C))$ iterations.

Definition 4.9:

Given $y \in \mathbb{R}^V$, define **reduced cost**

$$\overline{c'_{a,b}}^{(y)} := c'_{a,b} - (y_b - y_a) \qquad \forall (a,b) \in E(x)$$

Now, we have the following equivalence,

$$\Big[\ y \text{ is a feasible node potential for } (D(x), \{c'_e + \varepsilon\}_{e \in E(x)})\ \Big] \equiv \Big[\ \overline{c'_e}^{(y)} \ge -\varepsilon \text{ for all } e \in E(x)\ \Big]$$

Let y be feasible node potentials for the residual digraph $(D(x), \{c'_e + \varepsilon(x)\}_{e \in E(x)})$, we claim that

$$Z \text{ is a minimum mean-cost cycle in } D(x) \iff \overline{c'_e}^{(y)} = -\varepsilon(x) \qquad \forall e \in Z$$

Proof of the above claim. The proof is left as an exercise. □

We will show that

1. $\varepsilon(x)$ does not increase;
2. After $O(m)$ iterations, $\varepsilon(x)$ decreases by a multiplication factor of $(1 - 1/n)$.

Proof of Claim 1. Let $\varepsilon_1 = \varepsilon(x)$ and y be feasible node potentials for $\Big(D(x), \{c'_e + \varepsilon_1\}_{e\in E(x)}\Big)$. Let x' be the flow after augmenting along a min mean-cycle Z in $\Big(D(x), \{c'_e\}\Big)$. We will show that y is a feasibile node potential for $\Big(D(x'), \{c'_e + \varepsilon_1\}_{e\in E(x')}\Big)$, which implies that $\varepsilon(x') \le \varepsilon_1$.

The only edges $(a, b) \in E(x') - E(x)$ are those for which $(b, a) \in Z$. This implies that

$$\begin{aligned} \overline{c'_{b,a}}^{(y)} &= -\varepsilon_1 \quad \equiv \quad c'_{b,a} - (y_a - y_b) = -\varepsilon_1 \\ \implies \quad & y_a - y_b = c'_{b,a} + \varepsilon_1 = -c'_{a,b} + \varepsilon_1 \\ \implies \quad & y_b - y_a = c'_{a,b} - \varepsilon_1 \le c'_{a,b} + \varepsilon_1 \end{aligned}$$

as desired. □

Proof of Claim 2. Consider flow $x^{(0)}$ at the start of some iteration i. Let $\varepsilon_0 = \varepsilon(x^{(0)})$ and $y^{(0)}$ be feasible node potentials for $\Big(D(x^{(0)}), \{c'_e + \varepsilon_0\}_{e\in E(x^{(0)})}\Big)$. Say that an edge (a, b) is $y^{(0)}$-negative if

$$\overline{c'_{a,b}}^{y^{(0)}} := c'_{a,b} - (y^{(0)}_b - y^{(0)}_a) < 0$$

Let $E^-(x)$ be the $y^{(0)}$-negative edges in $E(x)$, we observe that

If (a, b) is $y^{(0)}$-negative, then (b, a) is NOT $y^{(0)}$-negative edges.

We claim that: suppose we have a flow x, and the cycle Z picked in $D(x)$ is such that all edges of Z are $y^{(0)}$-negative. Let x' be the flow after augmenting along Z, then

1. $|E^-(x')| < |E^-(x)|$, and
2. If $y^{(0)}$ is a feasible potential for $\Big(D(x), \{c'_e + \varepsilon_0\}_{e\in E(x)}\Big)$, then it is a feasibile potential for $\Big(D(x'), \{c'_e + \varepsilon_0\}_{e\in E(x')}\Big)$.

Why does this claim proves what we want?

Note 4.11

We note that for $x^{(0)}$, all edges of cycle in $D(x^{(0)})$ are $y^{(0)}$-negative.

After $\le m$ iterations, starting from $x^{(0)}$, we have a flow x such that there exists no $y^{(0)}$-negative edges (i.e., $E^-(x) = \varnothing$), or the cycle Z picked from $D(x)$ is such that at least one of the egdes (call it e^*) of Z is not $y^{(0)}$-negative.

The former implies that x is our min-cost flow. If it's the latter, it means that

$$c'(Z) + \varepsilon(x)|Z| = 0$$

(from the fact that $\mu(x) = -\varepsilon(x)$). Also,

$$c'(Z) + \varepsilon_0|Z| = \overline{c'}^{(y^0)}(Z) + \varepsilon_0|Z| = \sum_{e\in Z: e\neq e^*} \left(\overline{c'_e}^{y^0} + \varepsilon_0\right) + \overline{e'_{e^*}}^{(y^0)} + \varepsilon_0$$

We know that the big sum has value ≥ 0 becase $y^{(0)}$ is a feasible node potential for $\Big(D(x), \{c'_e + \varepsilon_0\}\Big)$. We also know that $\overline{c'_{e^*}}^{(y^0)} \geq 0$ since e^* is not $y^{(0)}$-negative. Therefore,

$$c'(Z) + \varepsilon_0|Z| \geq \varepsilon_0$$

Hence

$$\varepsilon_0(|Z| - 1) \geq -c'(Z) = \varepsilon(x)|Z|$$

which further implies that

$$\varepsilon(x) \leq \varepsilon_0\left(1 - \frac{1}{|Z|}\right) \leq \varepsilon_0\left(1 - \frac{1}{n}\right)$$

as desired.

Therefore, it suffices for us to prove the subclaim now :). When we push flow along Z, for every $(a, b) \in Z$, we create $(b, a) \in E(x')$, but (a, b) is $y^{(0)}$-negative, (b, a) is NOT $y^{(0)}$-negative. Also, at least one edge in Z will not appear in $E(x')$, so we are indeed strictly reducing the number of $y^{(0)}$-negative edges (i.e., $|E^-(x')| < |E^-(x)|$).

For the second part. The only new edges in $E(x') - E(x)$ are (b, a) where $(a, b) \in Z$, this suggests that (a, b) is NOT $y^{(0)}$-negative. Henceforth, $y^{(0)}$ is a feasible node potential for $\Big(D(x'), \{c'_e + \varepsilon_0\}\Big)$.

To finish up, initial flow has

$$\varepsilon(x) \leq C := \max_{e \in E} |c'_e|$$

and if $\varepsilon(x) < 1/n$, then $\varepsilon(x) = 0$, we will get termination in $O(mn \log(nc))$ iterations. $\square$

5 Matchings

Recall that given an undirected graph $G = (V, E)$, a set $M \subseteq E$ is a **matching** if $|\delta(v) \cap M| \leq 1$ for all $v \in V$. Given a matching M, we say that a vertex v is **M-exposed** if $M \cap \delta(v) = \varnothing$, otherwise we call it **M-covered**.

A path P in G is **M-alternating** if its edges alternate between being in M and not in M. Such a path is **M-augmenting** if both of its endpoints are M-exposed. Here is a fact, if P is augmenting, then $M' = M\Delta P$ $(:= (M - P) \cup (P - M))$ is a matching with size $|M'| = |M| + 1$.

Theorem 5.1

M is a maximum matching if and only if there exists no M-augmenting path.

Proof. Forward direction directly follows from the above fact. Conversely, suppose for a contradiction that M is not a maximum matching, let M' be a matching with $|M'| > |M|$. Consider $F = M\Delta M'$. Each node v has degree at most 2 in F, so F consists M-alternating paths and M-alternating cycles. Also,

$$\begin{aligned} |M'| - |M| &= \sum_{\text{components } Z \text{ of } F} \Big(|Z \cap M'| - |Z \cap M|\Big) \\ &= \sum_{\text{components } Z \text{ of } F} \Big(|Z - M| - |Z \cap M|\Big) \end{aligned}$$

Since $|M'| > |M|$, there exists some component Z of F such that $|Z - M| > |Z \cap M|$. This implies that Z is an M-alternating path and its end points are M-exposed. $\square$

Lecture 18 - Tuesday, November 18

5.1 Min-max Formula for Matchings

Theorem 5.2: König's Theorem

If graph $G = (V, E)$ is bipartite, then

$$\max\{|M| : M \text{ is a matching}\} = \min\{|S| : S \subseteq V \text{ is a vertex cover}\}$$

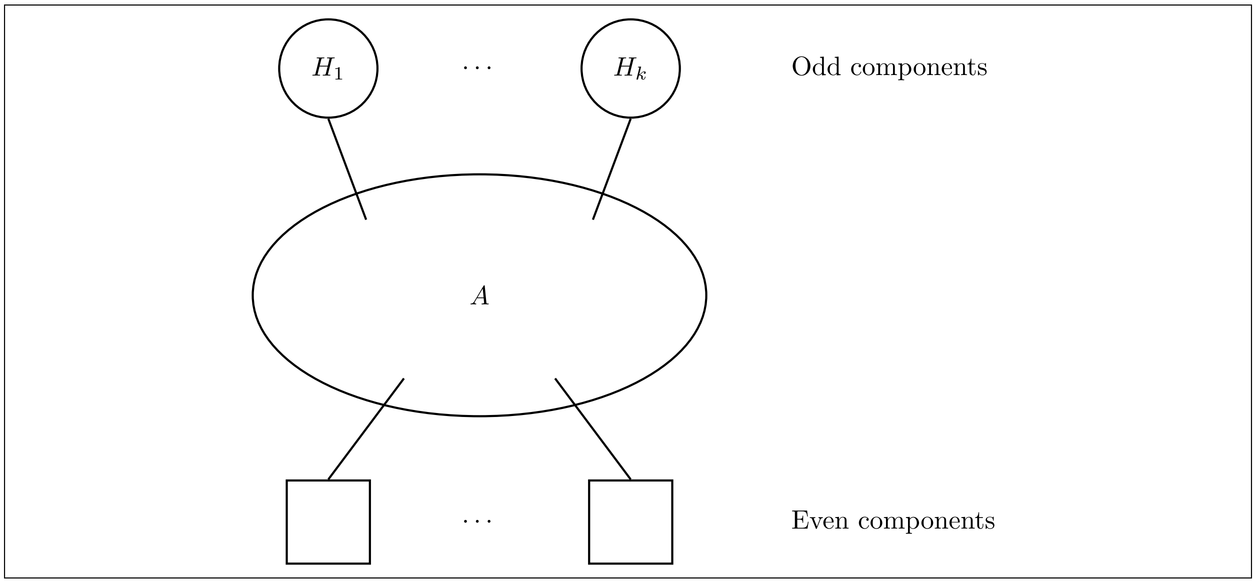
Comment 5.1

The above theorem does not hold for a general graph.

For a general graph $G = (V, E)$, let $A \subseteq V$, and let

$$oc(G - A) = \# \text{ of odd components of } G - A$$

where odd components are components with odd number of nodes. Let $H_1, \dots, H_k$ be the odd components of $G - A$, where $k = oc(G - A)$.



Discovery 5.1

We observe that for any matching M and any H_i,

$$|M \cap E(H_i)| \leq \frac{|V(H_i) - 1|}{2}$$

This implies that in any H_i, there is at least one node that is either M-exposed, or matched by M to a node in A, and thus the number of such nodes $\leq |A|$. Therefore, the number of M-exposed nodes is $\geq k - |A|$. As a result,

$$|M| \leq \frac{n - (k - |A|)}{2} = \frac{1}{2}(n - oc(G - A) + |A|)$$

5.1.1 Tutte-Berge Formula

Theorem 5.3: Tutte-Berge Formula

We have

$$\nu(G) = \min_{A \subseteq V} \frac{1}{2}(n - oc(G - A) + |A|)$$

We call a set $A \subseteq V$ attaining the minimum on the RHS a **Tutte-Berge set**.

Note 5.1

We note that

- If G is a cycle of $2k+1$ nodes, then $\nu(G)=k$, and $A=\varnothing$ is a Tutte-Berge (TB) set proving TB formula in this case;
- If $S \subseteq V$ is a vertex cover of G, then taking $A=S$. $G-S$ consists of singleton components, so $oc(G-S)=|V|-|S|$, so TB formula gives

$$\frac{1}{2}(n-oc(G-S)+|S|)=|S|$$

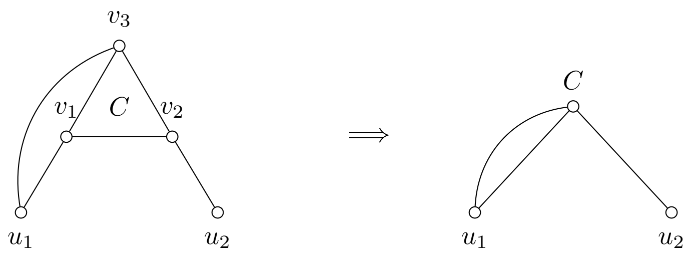
so TB formula gives $|S|$ as an upper bound on $\nu(G)$.

Corollary 5.1

G has a perfect matching if and only if

$$oc(G-A)\leq |A| \qquad \forall A\subseteq V$$

Basic Operations: Shrinking an odd cycle $C=(V_C,E_C)$. In other words, identify nodes of C or contracting C.



The resulting graph after shrinking C in G we denote it as $G'=G\times C$, which has nodes $(V-V_C)\cup\{C\}$ and edges $E-E_c$, with the convention that an edge $ab\in E$, where $a\notin V_c$ and $b\in V_c$ is now there as egde $aC\in E(G\times C)$.

Lemma 5.1

Given a graph G, let C be an odd cycle in G and let M' be a matching in $G\times C$. Then, M' can be extended to a matching M in G such that the number of M-exposed nodes in G is at most the number of M'-exposed nodes in $G\times C$.

Proof. If C is M'-exposed in $G\times C$, pick a matching of size $\frac{|C|-1}{2}$ from E_C and add it to M'. Else if C is M'-covered in $G\times C$, say $(u,c)\in M'$, which is equivalent to saying that edge uv $(v\in C)$ of G is a path. Then $C-\{v\}$ has $|C|-2$ edges, and contains no node incident to M', thus we can take a matching of size $\frac{|C|-1}{2}$ in $C-\{v\}$ and add this to M'. $\square$

Lemma 5.2

From the proof of the above lemma, we can see that

$$\nu(G) \geq \nu(G \times C) + \frac{|C| - 1}{2}$$

for any odd cycle C in G.

5.1.2 M-alternating Trees

Let M be our current matching. We will use M-alternating tree to efficiently find an M-augmenting path: Pick an M-exposed node r, and build a tree T rooted at r. Note that if there exists no M-exposed node, then we have already found a perfect matching, so we are done. Hence we may assume that such a node r exists.

Definition 5.1: Level

The **level** of a node $v \in T$ is the number of edges on $r \rightsquigarrow v$ path in T.

Let $B(T)$ be the nodes at even level and let $A(T)$ be the nodes at odd level. At each point v, we consider some edge vw where $v \in B(T)$. Different things can happen:

1. If w is M-exposed, then the r-v path in T union with $\{vw\}$ is an M-augmenting path, so we can use this to augment M and restart the process. We call this

   $$\texttt{Use } vw \texttt{ to augment } M$$

2. If w is M-covered and $w \notin V(T)$, then we

   $$\texttt{Use } vw \texttt{ to extend } T$$

   Namely, we add edge vw and $wz \in M$ to tree T.

   Note that $z \notin V(T)$ because otherwise z would have two edges of M incident to it.

3. If w is M-covered and $w \in A(T)$. We will talk about this later.

4. If w is M-covered and $w \in B(T)$. Now v-w in T union with $\{vw\}$ creates an odd cycle. Also more on this later.

Comment 5.2

Suppose for now that case 4 never occurs (as is in the case for bipartite graphs, who does not have odd cycles).

Note 5.2

We note that

1. Every $r \rightsquigarrow v$ path in T, where $v \in T$, is M-alternating.

2. Every $v \in T$, $v \neq r$, is M-covered.

3. $|B(T)| = |A(T)| + 1$.

Definition 5.2: Frustrated

Given a matching M and an M-alternating tree T, we say that T is **frustrated** if for all $v \in B(T)$ and all edges $vw \in E$, we have $w \in A(T)$.

Lemma 5.3

Let G has a mataching M and an M-alternatong tree T that is frustrated, then G does not have a perfect matching.

Proof. Every node $v \in B(T)$ is a singleton component in $G - A(T)$, so

$$oc(G - A) \geq |B(T)| > |A|$$

and thus G cannot have a perfect matching. □

Lemma 5.4

Let M be a matching in G. Suppose $T_1, \dots, T_k$ are M-alternating trees that are all frustrated, node-disjoint, and together cover V, then, because only the roots of the trees are M-exposed, we have

$$|M| = \frac{n-k}{2}$$

Moreover, taking $A = \bigcup_{i=1}^{k} A(T_i)$, we will get

$$|M| = \frac{1}{2}(n - oc(G - A) + |A|)$$

which implies $|M| = \nu(A)$, and so A is a TB set.

Proof. Exercise. □

5.1.3 Sketch of Algorithm for Finding Perfect Matching in a Bipartite Graph

We have the following algorithm for findint the perfect matching in a bipartite graph as a result of method of constructing alternating trees.

```
1  $M \leftarrow \varnothing$;
2  while exists $M$-exposed node do
3      Pick an $M$-exposed node $r$;
4      Set $T = (\{r\}, \varnothing)$;
5      while $\exists vw \in E$ such that $v \in B(T)$ and $w \notin V(T)$ do
6          if $w$ is $M$-exposed then
7              we "use $vw$ to augment $M$";
8              go to the next iterate of the outer while loop;
9          if $w$ is $M$-covered and $w \notin V(T)$ then
10             We "use $vw$ to extend $T$";
11     [If we are ever here] Terminate and return $G$ does not have a perfect matching.
12 return $M$
```

Algorithm 7: (Sketch) Finding Perfect Matching in a Bipartite Graph

Lecture 19 - Thursday, November 20

Recall the construction of our M-alternating tree: at each point v, we consider some edge vw where $v \in B(T)$. The fourth case that can happen is:

w is M-covered and $w \in B(T)$. Now v-w in T union with $\{vw\}$ creates an odd cycle

In this case, we can shrink the odd cycle present in the M-alternating tree and obtain a valid M-alternating tree for the contracted graph. We call this process

`Use` vw `to shrink, update` M `and tree` T

Definition 5.3: Blossom

The odd cycle described above in a M-alternating tree is called a **blossom**.

Here is a formalized statement of what we will do when we find a blossom:

```
1 Set $G' \leftarrow G \times C$;
2 Set $M' \leftarrow M - E_C$;
3 Set $T' \leftarrow$ tree in $G'$ with edge set $E_T - E_C$;
```

We claim that after the above operation, M' is a matching in G', T' is an M'-alternating tree in G', and $C \in B(T')$.

Definition 5.4: Derived Graph

Let G' be obtained from $G = (V, E)$ by a sequence of odd-cycle shrinkages. We call G' a **derived graph of** G.

- Each node v of G' corresponds to a set $S(v)$ of nodes of G (all nodes that has been contracted to form v). If $v \in V$, then $S(v) = \{v\}$, otherwise we call v a **pseudo-node**.

  Note 5.3

  If $v = C$, and $G' = G'' \times C$, then $S_{G'}(v) = \bigcup_{w \in C} S_{G''}(w)$.

- The $S(v)$-sets partition V;
- $|S(v)|$ is odd for all $v \in V_{G'}$.

Lemma 5.5: Generalization of Lemma 5.3

Let G' be a derived graph of G, M' be a matching in G', and T' be an M'-alternating tree in G' such that no node of $A(T')$ is a pseudo-node. If T' is frustrated in G', then G does not have a perfect matching.

Proof. Take $A = A(T')$. For each $v \in B(T')$ we get $S(v)$ as an odd component of $G - A$. Also, $A \subseteq V$ (since every $v \in A(T')$ is a node of G), so since $oc(G - A) \geq |B(T')| > |A(T')| = |A|$, G does not have a perfect matching. □

5.1.4 Edmonds' Blossom Algorithm for Perfect Matching

```
1  Given a graph $G$ and a matching $M$ of $G$;
2  Initialize $G' \leftarrow G$, and $M' \leftarrow M$;
3  if exists no $M'$-exposed node in $G'$ then
4      return perfect matching $M'$
5  Let $r$ be an $M'$ exposed node, set $T = (\{r\}, \varnothing)$;
6  while exists $vw \in E_{G'}$, $v \in B(T)$, $w \notin A(T)$ do
7      if $w$ is $M'$-exposed then
8          we "use $vw$ to augment $M$";
9          extend $M'$ to a matching $M$ of $G$;
10         Back to line 2;
11     else if $w$ is $M'$-covered, $w \notin V(T)$ then
12         we "use $vw$ to extend $T$", back to line 6;
13     else if $w$ is $M'$-covered, $w \in B(T)$ then
14         we "use $vw$ to shrink, update $M'$ and tree $T$", back to line 6;
15     else
16         We satisfies conditions of Lemma 5.5, so we terminate and report $G$ does not have a perfect
             matching.
```

Algorithm 8: Edmonds' Blossom Algorithm for Perfect Matching

Comment 5.3

On line 9, why do we always go back to the original graph? Because otherwise, we cannot guarantee that there exists no pseudo-node in the odd level of the tree we build.

Theorem 5.4

The Blossom Algorithm terminates after $O(n)$ augmentations, $O(n^2)$ shrinking steps, and $O(n^2)$ tree-extension steps. It also correctly determines if G has a perfect matching.

Proof. Proof of correctness is already covered in Lemma 5.5. We will only consider runtime here.

Each augmentation increases the size of our matching, and thus there are $O(n)$ augmentations, this is pretty clear. Between two consecutive augmentations,

- Each shrinkage decrement the size of $V(G')$, leaves $|V(T)^c|$ untouched,
- Each tree extension decrement $|V(T)^c|$, leaves $|V(G')|$ unchanged,

each of which can happen at most n times, and each operation does not "undo" what the other one has done. Therefore, there are at most $O(n)$ shrinkages and tree extensions between two consecutive augmentings, giving us $O(n^2)$ shrinkages and tree extensions steps overall. □

Lemma 5.6

Let G' be a derived graph G, M' be a matching in G' and $T'_1, \dots, T'_k$ be M'-alternating trees in G' such that

1. each T'_i is frustrated in G', and every node in $A(T'_i)$ is a node of G;
2. $T'_1, \dots, T'_k$ are vertex-disjoint;
3. M' contains a perfect matching of $G' - (V(T'_1) \cup \dots \cup V(T'_k))$.

Let M be a matching of G obtained by extending M', then

$$|M| = \frac{n-k}{2}$$

and $A = \bigcup_{i=1}^{k} A(T'_i)$ is such that

$$|M| = \frac{1}{2}(n - oc(G-A) + |A|)$$

Therefore, M is a max-size matching, and A is a TB set (and TB formula holds).

Proof. By definition, only roots of $T'_1, \dots, T'_k$ are M'-exposed, so M has $\leq k$ exposed nodes and $|M| \geq (n-k)/2$. Every $v \in \bigcup_{i=1}^{k} B(T'_i)$ gives rise to an odd component in $G - A$, so

$$oc(G-A) = \sum_{i=1}^{k} B(T'_i) = \sum_{i=1}^{k} (1 + |A(T'_i)|) = |A| + k$$

this implies that

$$\frac{1}{2}(n - oc(G - A) + |A|) = \frac{n-k}{2}$$

Hence we must have $|M| = \frac{1}{2}(n - oc(G - A) + |A|)$. □

5.1.5 Consequence of Tutte-Berge Formula

Definition 5.5: Inessential

We say that a node v is **inessential** if there is a maximal matching M such that v is M-exposed.

Definition 5.6: Essential

If v is not inessential, then v is **essential**.

Lemma 5.7

Let A be a Tutte-Berge set, then every node in A is essential.

Proof. We argue that after deleting any vertex in A, the maximal-size matching is going to decrease. Let $v \in A$, consider $G' = G - \{v\}$ and $A' = A - \{v\}$, then

$$oc(G' - A') = oc(G - A)$$

As a result,

$$\begin{aligned}
\nu(G') &\leq \frac{1}{2}(|V(G')| - oc(G' - A') + |A'|) \\
&= \frac{1}{2}(|V(G)| - 1 - oc(G - A) + |A| - 1) \\
&= \frac{1}{2}(|V(G)| - oc(G - A) + |A|) - 1 \\
&= \nu(G) - 1
\end{aligned}$$

by Tutte-Berge Formula, so v is essential. □

5.2 Min-Cost Perfect Matching

Lecture 20 - Tuesday, November 25

Given undirected graph $G = (V, E)$ and edge costs $\{c_e\}_{e \in E}$. We want to find a perfect matching with the minimum costs $c(M) = \sum_{e \in M} c_e$, if a perfect matching exists.

Here is the LP-relaxation:

$$\min \sum_{e \in E} c_e x_e \quad s.t. \quad \begin{aligned} x(\delta(v)) &= 1 \quad \forall v \in V \\ x &\geq 0 \end{aligned} \tag{P}$$

which has the dual:

$$\max \sum_{v \in V} y_v \quad s.t. \quad y_u + y_v \leq c_e \quad \forall e \in E \tag{D}$$

5.2.1 Bipartite Graphs

Using network flows theory and LP theory, we know that

- (P) is feasible if and only if G has a perfect matching.
- (P) is feasible implies that (P) has an optimal solution, and the optimal value of (P) is equal to the cost of the min-cost perfect matching (i.e., LP (P) is tight).

Note 5.4

Observe by LP theory:

- It is easy to find a dual feasible solution y;
- A perfect matching M is a min-cost perfect matching if and only if there exists a dual feasible y such that M is a perfect matching using only edges in

$$E^{=}(y) := \{uv \in E : y_u + y_v = c_{uv}\}$$

Therefore, here is the idea:

1. Start with a dual feasible solution y
2. Find a perfect matching in $E^{=}(y)$. If we do not find a perfect matching, then we have an M-alternating tree T that is frustrated in $E^{=}(y)$.

See below for an example, we have two cases:

Case 1, T is frustrated in G In this scenario, G does not have a perfect matching.

Case 2, T is not frustrated in G This means that there exists $vw \in E - E^{=}$ where $v \in B(T)$ and $w \notin V(T)$. Here, we cannot conclude that G does not attain a perfect matching, but rather we know that the dual solution we obtained is not the right one. Hence we wish to change the dual solution y such that

1. make "progress" by increasing dual objective value, while
2. maintaining the dual feasibility of y and
3. keeping all edges of M and T in $E^{=}(y)$.

Algorithm 5.1

This is what we are going to do: for suitable $\varepsilon > 0$, we increase y_v by ε for all $v \in B(T)$ and decrease y_v by ε for all $v \in A(T)$. This way,

1. All edges in T remain tight under the new dual solution y^{new};
2. For edges $uv \in E$, where $u, v \notin B(T)$, the constraint $y_u + y_v \leq c_{uv}$ still holds.
3. For edges $vw \in E$, where $v \in B(T)$ and $w \notin V(T)$, the value $y_v + y_w$ increases, this suggests that we need to take

$$\varepsilon := \min\Big\{c_{vw} - y_v - y_w : vw \in E, v \in B(T), w \notin V(T)\Big\} \qquad \text{(Change } y\text{)}$$

Since T is not frustrated in G, there exists at least one of those edges, so $\varepsilon > 0$.

Comment 5.4

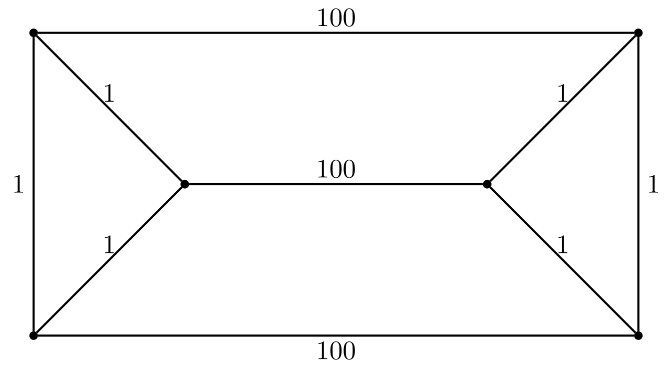
T is frustrated implies that if $vw \in E$ with $v \in B(T)$ and $w \notin V(T)$, then $vw \notin E^{=}(y)$.

Example 5.1

Flesh out this min-cost perfect matching algorithm for bipartite graphs.

5.2.2 General Graphs

The earlier LP is no longer a good LP, consider



We can set $x_e = 1/2$ for all $e \in \triangle$, which is feasible for (P), so $OPT_{(P)} \le 3$. However, any perfect matching must use one of the edges with $c_e = 100$, and in fact $OPT_{(P)} = 102$.

We strengthen the LP as follows:

$$\min \sum_{e\in E} c_e x_e \;\; s.t. \quad \begin{array}{rll} x(\delta(v)) & = 1 & \forall v \in V \\ x & \ge 0 & \\ x(\delta(S)) & \ge 1 & \forall S \subseteq V : |S| \text{ odd} \end{array} \tag{P'}$$

We denote the set $\mathcal{O} := \{S \subseteq V : |S| \ge 3 \text{ odd}\}$. This LP has the dual

$$\max \sum_{v\in V} y_v + \sum_{S\in\mathcal{O}} y_S \;\; s.t. \quad \begin{array}{rll} y_u + y_v + \sum_{S\in\mathcal{O}:uv\in\delta(S)} y_S & \le c_e & \forall e \in E \\ y_S & \ge 0 & \forall S \in \mathcal{O} \end{array} \tag{D'}$$

Again follow the similar template, we know that a perfect matching M and dual solution y satisfy Complementary Slackness if

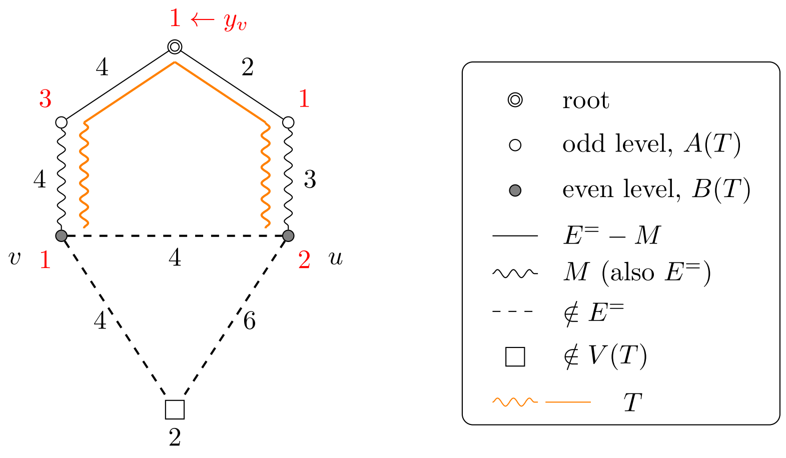
- $e = uv \in M$ implies that $y_u + y_v + \sum_{S\in\mathcal{O}:uv\in\delta(S)} y_S = c_{uv}$;
- $y_S > 0$ for some $S \in \mathcal{O}$ implies that $|M \cap \delta(S)| = 1$.

Again, define $E^=(y)$ analogously:

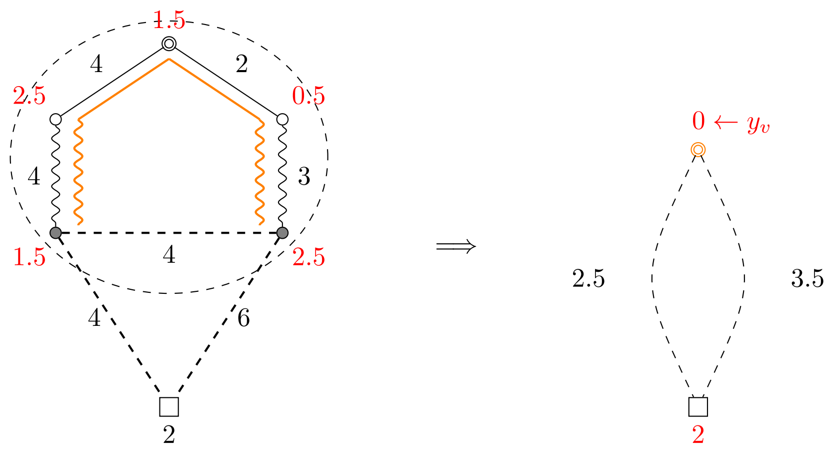
$$E^=(y) = \{uv \in E : y_u + y_v + \sum_{S\in\mathcal{O}:uv\in\delta(S)} y_S = c_{uv}\}$$

Example 5.2

Consider the following example (encounter while trying to find a perfect matching in $E^=(y)$).



Notice that T is frustrated in $E^=$, but not frustrated in G. We can increase $y_v, \forall v \in B(T)$ and decrease $y_v, \forall v \in A(T)$ by ε. Observe that edge vu will become tight first, and it limits $\varepsilon = 1/2$:



after obtaining an odd cycle, we shrink. However, the shrink operation is now as follows:

``‘‘Use $vw$ to shrink, update $M$ and tree $T$, and edge costs’’``

1. Let C be the odd cycle, formed by vw path in T union with $\{vw\}$;
2. Update $G' \leftarrow G \times C$, $M' \leftarrow M - E(C)$, $T' \leftarrow T - E(C)$;
3. For all $uv \in \delta(C)$ with $v \in C$, update cost $c'_{uv} \leftarrow c'_{uv} - y_v$ (edges $e \in E - E_C$ remain unchanged). This ensures that tight edges in $E^= - E_C$ remain tight in G'.
4. Set $y_C \leftarrow 0$, which we have to maintain $y_C \geq 0$ because of the ‘non-negativity of odd-sized sets’ constraint.

How do we get back to the original graph from the derived graph?

Proposition 5.1

Let (G', C') be obtained from (G, C) by shrinking an odd cycle C of tight edges with respect to dual solution y that is feasibile for (G, C). Let M' be a perfect matching for G' and y' be a dual feasibile solution for (G', C') such that (a) M' and y' satisfy Complementary Slackness (for (G', C')) and (b) $y'_C \geq 0$, then

1. M' can be expanded to a perfect matching M in G;
2. y' can be extended to a dual feasibile solution $\overline{y}$ for (G, C) by defining

   (a) $\overline{y}_v = \begin{cases} y'_v & \text{for all } v \in V_{G'}, v \notin C \\ y_v & \text{for all } v \in C \end{cases}$

   (b) $\overline{y}_C = y'_C$;

   (c) For all odd sets $S \subseteq V_{G'}$, let $\overline{S}$ be the corresponding odd set in G, set $\overline{y}_{\overline{S}} = y'_S$ (All other sets $S \in \mathcal{O}$ get $\overline{y}_S = 0$).

3. M, $\overline{y}$ defined above satisfy Complementary Slackness conditions (so M is a min-cost perfect matching for G and $\overline{y}$ is an optimal dual solution for (G, C)).

Lecture 21 - Thursday, November 27

Note 5.5

Important invariant (I) that we will maintain:

In a derived graph G', the only y-variables that will be positive are the y_v's corresponding to nodes of G' (which could be pseudo-nodes, i.e., non-singleton odd sets of V).

In other words, we will never set $y_S > 0$ for a non-singleton set S of $V(G')$.

Issues and Solutions

Issue 1: When we augment, we may not be able to expand all the way back to the original graph G. (Because there could be a pseudo-node v with $y_v > 0$, which gives an odd set $S \subseteq V(G)$ with $y_S > 0$ and $|S| > 0$, violating our invariant (I)).

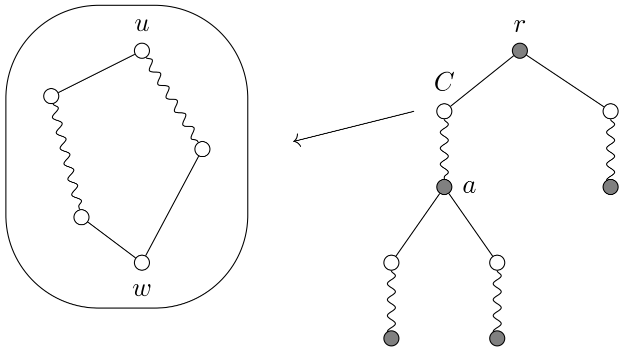
Solution. We only expand a pseudo-node v where $y_v = 0$, otherwise we keep working in the derived graph. □

Issue 2: Augmentation in derived graph could create pseudo-nodes of odd level (recall that we increase the values for nodes in B and decrease the values for nodes in A, see algorithm 5.1).

Solution. This suggests that when we "`change` y", the bounds on ε are impsoed by

1. $B(T)$–to–$C(T)^c$ edges;
2. $B(T)$–to–$B(T)$ edges;
3. $y_V \geq 0$ for all pseudo-nodes $v \in A(T)$.

Hence we need one additional operation: `Expand odd pseudo-node`. See below for an example:



Suppose odd pseudo-node C has

- incoming edge (from parent) incident to $u \in C$;
- outgoinh edge (to child) incident to $w \in C$.

Note 5.6

We know that in the cycle, all the edges are tight (this is why we contract the cycle). Hence when we expand the pseudo-node, we can pick which $(|C|-1)/2$ edges of C to add to our matching so that the new tree remains an alternating tree with respect to the new matching (and we do not increase the number of exposed nodes).

We add even-length u-w path in C to T, extend matching M' in G' to a matching after opening up C. For every edge $st \in \delta(C)$, $s \in C$, we update $c'_{st} \leftarrow c'_{st} + y_s$ (for edges of C, they appear with their costs c'_e when C was shrinked). We call the above process

`‘‘ Expand odd pseudo-node, update` G', M'`, tree` T' `and edge costs` c' `’’` □

5.2.3 Blossom Algorithm for min-cost Perfect Matching

```
1  Start with a feasible dual solution $y$ such that $y_S = 0$ for all non-singleton $S \in \mathcal{O}$. Let $G' \leftarrow G$,
     $c' \leftarrow c$. Let $M'$ be some matching in $E^=(y)$. Maintain that $y$ is dual feasible, $M$ and $y$ satisfy CS
     conditions;
2  if $M'$ is a perfect matching then
3      Extend $M'$ to a perfect matching of $G$, $y$ to a dual solution for $(G, c)$ such that $M$ and $y$ satisfy
         CS conditions;
4      Terminate because $M$ is a min-cost perfect matching;
5  Let $r$ be an $M'$-exposed node, set $T \leftarrow (\{r\}, \varnothing)$;
6  while true do
7      if exists $vw \in E^=$, $v \in B(T)$, $w \notin V(T)$ then
8          use $vw$ to augment;
9          Goto line 2;
10     if exists $vw \in E^=$, $v \in B(T)$, $w$ is $M'$-covered, $w \notin V(T)$ then
11         use $vw$ to extend $T$;
12     if exists $vw \in E^=$, $v, w \in B(T)$ then
13         use $vw$ to shrink, update $M'$, $T$, $c'$, and dual solution $y$;
14     if exists pseudo-node $v \in A(T)$ with $y_v = 0$ then
15         expand $v$, update $G', M'$, tree $T$, costs $c'$, dual and solution $y$;
16     if $T$ is frustrated in $G'$, and $A(T)$ has no pseudo-node then
17         We stop, return $G$ has no perfect matching and (P) is infeasible.
18     break;
19 If we are here, so every pseudo-node $v \in A(T)$ has $y_v > 0$, $T$ is not frustrated in $G'$ or $T$ has an odd
     pseudo-node, then we change $y$.
```

Algorithm 9: Blossom Algorithm for min-cost Perfect Matching

Theorem 5.5

The Blossom Algorithm for min-cost perfect matching terminates after $O(n)$ augmentations, and $O(n^2)$ tree-extending, shrinking, expanding, and dual change operations. At termination,

1. it returns that G has no perfect matching if and only if (P) is infeasible;
2. it returns a perfect matching M and dual feasible y such that M and y satisfy Complementary Slackness conditions , so M is a min-cost perfect matching.

Corollary 5.2

Feasible region of (P') is

$$\operatorname{conv}\Big(\Big\{x^M : M \text{ is a perfect matching of } G\Big\}\Big)$$

and this is known as the **perfect-matching polytope**.

Proof of Theorem 5.5. It is clear that there are $O(n)$ augmentations, as $|M|$ only goes up after each augmentation, and this can only happen at most n times. We consider iteration between two consecutive augmentations. We will show that there are $\leq n$ operations of each type between them.

Every dual change leads to a shrink(S), extend(E), augment(A), or expand(Exp) operation in the next iteration. Hence it suffices to bound (S), (E), and (Exp) steps.

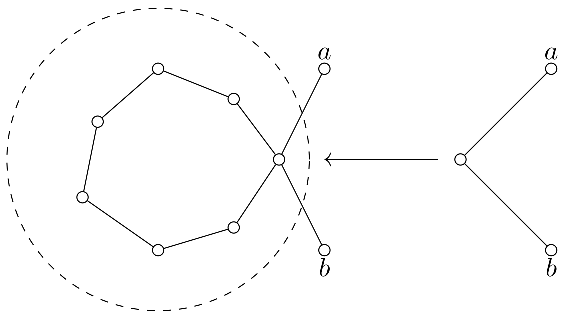
1. Every (S) creates an even pseudo-node;
2. Every (Exp) operation applies to an odd pseudo-node.

For (S) operation, consider $\varphi = \sum_{\text{even pseudo-node v}} |S(v)|$, we note that

- $0 \leq \varphi \leq n$;
- Every (S) step increases φ by ≥ 1;
- (Exp) or (E) do not decrease φ, so there are $\neq n$ (S) steps.

For (E) and (Exp) operations: consider $\varphi' = \sum_{\text{odd pseudo-node v}} |S(v)| + \sum_{v \notin T} \left(|S(v)| - \frac{1}{3}\right)$.

- Each (Exp) decreases φ' by $\geq 2/3$, since (Exp) either applied to v either adds nodes to $B(T)$ or we have a picture below:



and decrease in $|S(v)|$ is $\geq 2/3$ more than incease in $\sum_{u \notin T} |S(u)|$;

- Each (E) decreases φ by $\geq 1/3$ since one node not in $V(T)$ gets added to $B(T)^{new}$;
- (S) does not increase φ'.
- Hence since $0 \leq \varphi \leq n$, there are at most $O(n)$ (Exp) and (E) steps between two augmentations.

□

Lecture 22 - Tuesday, December 02

5.3 T-join

Definition 5.7: T-join

Let $G = (V, E)$ be a graph, $\{c_e\}_{e \in E}$ be edge costs, and $T \subseteq V$ with $|T|$ even. A **T-join in G** is a set $F \subseteq E$ such that $|\delta(v) \cap F|$ is even for $v \notin T$ and odd for $v \in T$.

The Min-cost T-join problem is asking to find a T-join of minimum total cost.

Example 5.3

For $T = \varnothing$, a T-join is $F \subseteq E$ that induces even degree for every $v \in V$.

Definition 5.8: Eulerian

We say that $G = (V, E)$ is **Eulerian** if $|\delta(v)|$ is even for all $v \in V$.

Note 5.7

Here is a fact, if $G = (V, E)$ is Eulerian, then E can be decomposed into edge-disjoint cycles.

Result 5.1

If F is a $\varnothing$-join with $c(F) < 0$, then G has a negative-cost cycle. Hence solving min-cost T-join with $T = \varnothing$ allows us to detect if G has a negative-cost cycle.

Discovery 5.2

Let J_1 be T_1-join and J_2 be T_2-join where $T_1, T_2 \subseteq V$ and $|T_1|, |T_2|$ even. Then $J_1 \Delta J_2$ is a $(T_1 \Delta T_2)$-join.

Proof. Exercise. □

Example 5.4

Let $T = \{a, b\}$, then a minimal T-join is a simple a-b path in G. Hence if $c \geq 0$, then a min-cost T-join is a *shortest a-b path*.

Let $F \subseteq E$ be a T-join, let P be a simple a-b path contained in F. Then, by the above observation, $F \Delta P = F - P$ is a $T \Delta \{a, b\} = \varnothing$-join.

So if G has no negative-cost cycles, then an optimal T-join is a shortest a-b path.

Lemma 5.8

Every (inclusion-wise) minimal T-join is a union of $|T|/2$ edge-disjoint simple paths, every path joins two nodes of T, and these paths do not share any end-points.

Proof. Clearly, the union of the states paths is a T-join. Let J be a minimal T-join. Let H be a component of (V, J). Then H must contain some $u \in T$ since J is minimal, and so it must also contain some other node $v \in T$ because otherwise it would have an odd number of nodes of odd degrees. Let P be some simple u-v path in H, so P is a $\{u, v\}$-join. Then $J \Delta P = J - P$ is a $T \Delta \{u, v\} = T - \{u, v\}$-join. Repeating this process, we can find simple paths $P_1, \ldots, P_{|T|/2}$ of the stated form contained in J. Hence $P_1 \cup \cdots \cup P_{|T|/2}$ is a T-join. Recall that J is a minimal T-join, so we are done. □

Corollary 5.3

Let $c \geq 0$, there is an optimal T-join that is the union of $|T|/2$ edge-disjoint shortest paths, which join the nodes of T in pairs.

5.3.1 Solving T-join via Matching ($c \geq 0$)

Therefore, solving the min-cost T-join problem can be done via solving for the min-cost matching problem.

Definition 5.9: Metric Completion

Given $G = (V, E)$ a graph, edge costs $\{c_e\}_{e \in E}$, the **metric completion** of (G, c) is $(\overline{G}, \overline{c})$ where $\overline{G} = (V, \overline{E})$ is the complete graph on V, and $\overline{c}_{u,v}$ = shortest u-v distance in G for all $u, v \in V$.

```
1 Consider $\overline{G}[T] = (T, \overline{E}(T))$;
2 Find min-cost perfect matching $M$ in $\overline{G}[T]$;
3 Let $P_1, \ldots, P_{|M|}$ be the paths in $G$ corresponding to egdes in $M$;
4 return $P_1 \Delta \cdots \Delta P_{|M|}$
```

Algorithm 10: Algorithm for min-cost T-join ($c \geq 0$)

Theorem 5.6

We have the following result:

1. $c(M)$ is equal to the cost of the optimal T-join;
2. $J \coloneqq P_1 \Delta \cdots \Delta P_{|M|}$ is a min-cost T-join.

5.3.2 Arbitrary c Case

Let $N = \{e \in E : c_e < 0\}$ and let $T_N = \{v \in V : |\delta(v) \cap N| \text{ is odd}\}$. By construction, N is a T_N-join. Let J be a T-join, so by observation, $J \Delta N$ is a $(T \Delta T_N)$-join. Also,

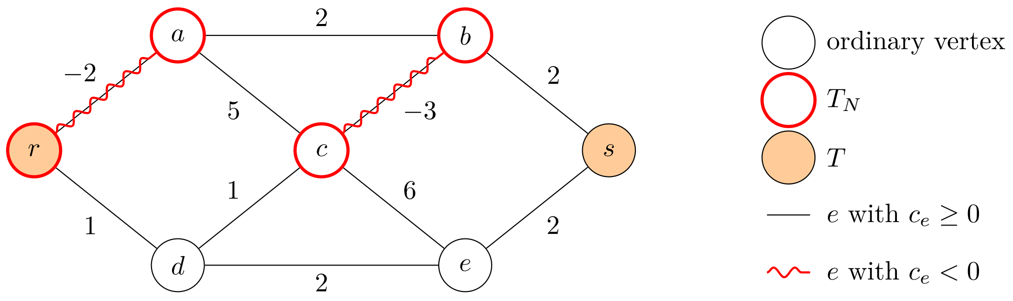
$$\begin{aligned} c(J) &= c(J \cap N) + c(J - N) \\ &= c(N) + \Big[- c(N - J) + c(J - N) \Big] \\ &= c(N) + \sum_{e \in J \Delta N} |c_e| \end{aligned}$$

This suggests that J is a minimum c-cost T-join if and only if $J \Delta N$ is a minimum $|c|$-cost $(T \Delta T_N)$-join. Hence we have the following algorithm.

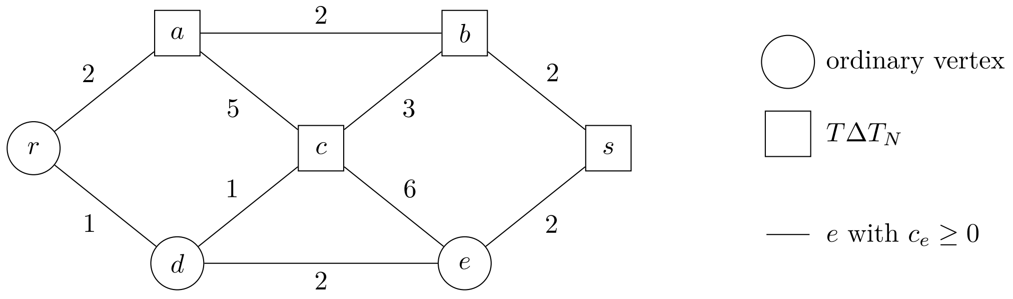
```
1 Find $N$ and $T_N$;
2 Find a min $|c|$-cost $(T \Delta T_N)$-join $J'$;
3 return $J' \Delta N$ (which is a $(T \Delta T_N) \Delta T_N = T$-join)
```

Algorithm 11: Algorithm for min-cost T-join (arbitray c)

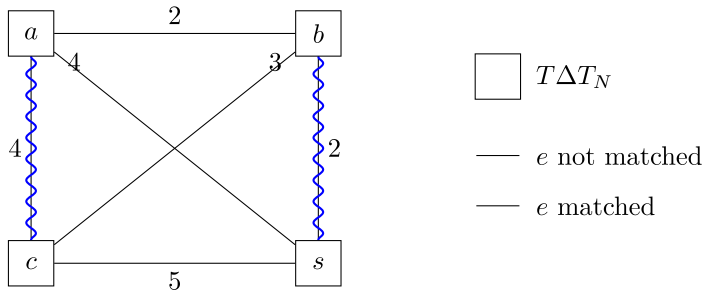
Here is an example running the algorithm:



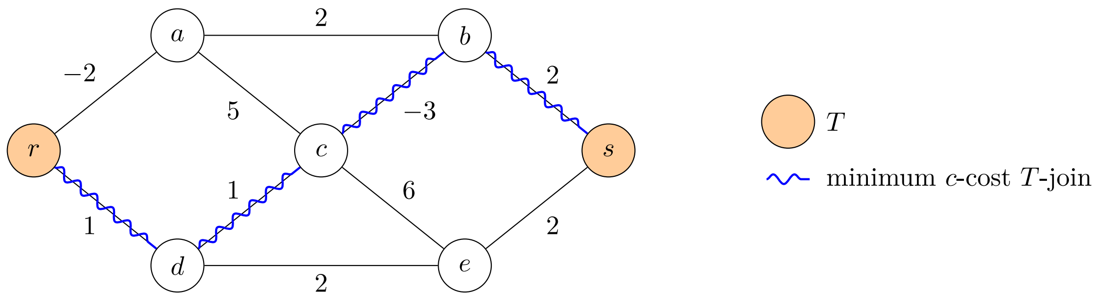
Identifying $T \Delta T_N$, and changing all edge costs to their absolute values, we have the following picture:



To find the minimum $|c|$-cost $(T \Delta T_N)$-join, we have the following diagram proceeded as described in chapter 5.3.1:



Using this to trace back to the edges in the original graph, we obtain the following minimum c-cost T-join:



6 Exercises

Exercises I collected from anywhere everywhere. glhf!

6.1 Greedy Algorithm, Minimum Spanning Trees, and Matroids?

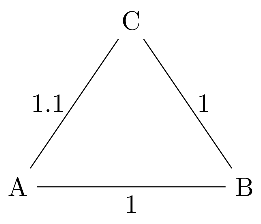
Exercise 6.1

Suppose that for every cut of the graph, there is a unique light edge crossing the cut. Show that the graph has a unique minimum spanning tree. Does the converse hold? If not, please give a counterexample.

Proof. Fix any cut $(S, V \setminus S)$ and let e be its unique light edge, we claim that e is contained in every minimum spanning tree of G.

Proof of claim: Take an arbitrary MST T. If $e \in T$, then we are done, if not, we consider $T + e$, who contains exactly one cycle C. We know that there is another edge $f \in C$ with strictly larger cost since e is the unique light edge, replacing f with e yields us a strictly cheaper spanning tree, a contradiction.

Now we can use this to prove that every minimum spanning tree is unique. Conversely, the statement does not hold. Here is a counterexample:



done. □

Exercise 6.2

Show that the dual "matroid" is a matroid. Recall that given a matroid $M = (U, \mathcal{I})$, then dual matroid is defiend as

$$M^* = (U, \mathcal{I}^* = \{J \subseteq U : U - J \text{ contains an } M\text{-basis}\})$$

Exercise 6.3

Give a counterexample to show that an independent system with all maximal independent sets of the same size may not be a matroid.

Proof. Let the ground set $U = \{1, 2, 3, 4\}$ and define $\mathcal{I}$ to be

$$\mathcal{I} \coloneqq \{\{\varnothing\}, \{1\}, \{2\}, \{3\}, \{4\}, \{1, 2\}, \{3, 4\}\}$$

Clearly $(U, \mathcal{I})$ is not a matroid because it does not have the exchange property (we can take $A \coloneqq \{1\}$ and $B \coloneqq \{3, 4\}$ for example). □

Exercise 6.4

A family $\mathcal{F}$ of sets is said to be *laminar* if, for any two sets $A, B \in \mathcal{F}$, we have that either (i) $A \subseteq B$, or (ii) $B \subseteq A$ or (iii) $A \cap B = \emptyset$. Suppose that we have alaminar family $\mathcal{F}$ of subsets of E and an integer $k(A)$ for every set $A \in \mathcal{F}$. Show that $(E, \mathcal{I})$ defines a matroid (a *laminar matroid*) where:

$$\mathcal{I} = \{X \subseteq E : |X \cap A| \leq k(A) \text{ for all } A \in \mathcal{F}\}.$$

Exercise 6.5

The dual matroid of a graphic matroid is called a cographic matroid. What are the independent sets, bases and cycles of these matroids?

Exercise 6.6

Let $(E, \mathcal{I})$ be a matroid, and $F \subseteq E$.

1. F is called a *closed set* (or *flat*) if it has the following property:

$$r(F \cup \{e\}) > r(F), \quad \forall e \in E \setminus F.$$

   Describe the closed sets of graphic matroids using graphtheoretical terms.

2. F is said to be *inseparable* if it cannot be partitioned into two (non-empty) subsets F_1, F_2 such that

$$r(F) = r(F_1) + r(F_2).$$

   Prove that the inseparable sets of a graphic matroid are exactly those sets F of edges for which the subgraph $G' = (V(F), F)$ is 2-node-connected.

   Remark. A (connected) graph is called 2-node-connected (or simply 2-connected) if it remains connected after deleting any node and all the adjacent edges.

Exercise 6.7

Parallel extension of an element

Let $M = (N, \mathcal{I})$ be a matroid and let $e \in N$ be a non-loop. Introduce a new element $f \notin N$, and define a family $\mathcal{I}'$ of subsets of $N \cup \{f\}$ by declaring $X \subseteq N \cup \{f\}$ independent if and only if

$$f \notin X \text{ and } X \in \mathcal{I} \qquad \text{or} \qquad f \in X \text{ and } (X \setminus \{f\}) \cup \{e\} \in \mathcal{I}.$$

We call the resulting set system $M' = (N \cup \{f\}, \mathcal{I}')$ the *parallel extension of M at e*.

1. Prove that M' is a matroid. (10 marks)

2. Describe the bases and circuits of M' in terms of the bases and circuits of M. In particular, show that: (10 marks)

- every basis of M' is obtained from a basis of M either by keeping it as is (if it contains e) or by replacing e with f; and
- the circuits of M' are precisely:
  - all circuits of M that do not contain e;
  - the 2-element set $\{e, f\}$; and
  - for every circuit C of M with $e \in C$, the set $(C \setminus \{e\}) \cup \{f\}$.

3. Suppose M is representable over a field $\mathbb{F}$ by a matrix A whose columns are indexed by N. Let a_e be the column of A corresponding to e, and let A' be the matrix obtained from A by appending a new column labelled f that is equal to a_e. Prove that A' is a representation of M' over $\mathbb{F}$. (5 marks)

Exercise 6.8

Matchings via matroid intersection

Let $G = (V = L \cup R, E)$ be a bipartite graph with bipartition (L, R). Consider the following independence systems on the ground set E of edges:

$$\mathcal{I}_1 := \{F \subseteq E : \text{no two edges in } F \text{ share a vertex in } L\},$$

$$\mathcal{I}_2 := \{F \subseteq E : \text{no two edges in } F \text{ share a vertex in } R\}.$$

We can show that $M_1 = (E, \mathcal{I}_1)$ and $M_2 = (E, \mathcal{I}_2)$ are matroids and a subset $F \subseteq E$ is a matching in G if and only if F is independent in both M_1 and M_2. Let r_1 and r_2 be the rank functions of M_1 and M_2, respectively. You may use without proof the matroid intersection theorem in the form

$$\max\{|I| : I \in \mathcal{I}_1 \cap \mathcal{I}_2\} = \min_{X \subseteq E}\big(r_1(X) + r_2(E \setminus X)\big).$$

Use the Matroid Intersection theorem to prove Hall's marriage theorem: there exists a matching that covers every vertex in L if and only if

$$|N(S)| \geq |S| \qquad \text{for every } S \subseteq L,$$

where $N(S)$ denotes the set of neighbors of S in R.

Proof. Consider any minimizer A for $\min_{A \subseteq E}\big(r_1(A) + r_2(E - A)\big)$. Define

$$L_A := \{v \in L : \delta(v) \cap A \neq \varnothing\}$$
$$R_{E-A} := = \{v \in R : \delta(v) \cap (E - A) \neq \varnothing\}$$

For $v \in L \setminus L_A$, we know that there exists $e = vw \in E - A$ where $w \in R_{E-A}$. This suggests that $|L \setminus L_A| \leq |N(L \setminus L_A)| \leq |R_{E-A}|$. Therefore,

$$\max\{|I| : I \in \mathcal{I}_1 \cap \mathcal{I}_2\} = |L_A| + |R_{E-A}| \geq |L_A| + |L \setminus L_A| = |L|$$

as desired. ☐

Exercise 6.9

Minimum distance via circuits

Let $\mathbb{F}$ be a field and let H be an $m \times n$ matrix over $\mathbb{F}$. Consider:

- the linear code
$$C := \{x \in \mathbb{F}^n : Hx = 0\},$$
whose parity-check matrix is H;
- the matroid $M(H)$ on ground set $N = \{1, \dots, n\}$ whose independent sets are those index sets of columns of H that are linearly independent over $\mathbb{F}$.

For a vector $x \in \mathbb{F}^n$, let $\operatorname{supp}(x) := \{i \in N : x_i \neq 0\}$ and let $\operatorname{wt}(x) := |\operatorname{supp}(x)|$ denote its (Hamming) weight. The *minimum distance* of C is

$$d(C) := \min\{\operatorname{wt}(x) : x \in C,\ x \neq 0\}.$$

1. Prove that if $x \in C$ is a nonzero codeword with support S, then S contains a circuit of $M(H)$. Deduce that (10 marks)
$$d(C) \ \geq\ \min\{|C| : C \text{ is a circuit of } M(H)\}.$$
2. Prove the converse inequality: show that if $S \subseteq N$ is a minimal subset of columns of H that is linearly dependent, then there exists a nonzero codeword $x \in C$ whose support is contained in S. Conclude that (10 marks)
$$d(C) = \min\{|C| : C \text{ is a circuit of } M(H)\}.$$
3. Work over the field $\mathbb{F} = \mathbb{F}_2$ and consider the 3×7 matrix (5 marks)
$$H = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}.$$
This is a parity-check matrix of the binary $[7,4]$ Hamming code. By analysing the circuits of $M(H)$, compute the minimum distance $d(C)$ of this code. (You do *not* need to list *all* circuits, but you must justify that no smaller circuit exists.)

6.2 Flows and Cuts

Exercise 6.10

A conference organizer wants to set up a review plan. There are m submitted papers and n reviewers. Each reviewer has made p papers as prefer to review. Each paper should have at least q review reports. Find a method to determine whether such a review plan exists or not.

Proof. We construct a flow network $G = (V, E)$ as follows.

$$V = \{s, t\} \cup \{r_1, \dots, r_n\} \cup \{p_1, \dots, p_m\}.$$

Edges and capacities:

- For each reviewer i, add edge (s, r_i) with capacity $c(s, r_i) = p$.
- For each preferred pair (reviewer i prefers paper j), add edge (r_i, p_j) with capacity $c(r_i, p_j) = 1$.
- For each paper j, add edge (p_j, t) with capacity $c(p_j, t) = q$.

Compute a maximum s–t flow f. A feasible review plan exists iff $|f| = mq$. □

Exercise 6.11

Suppose there exist two distinct maximum flows f_1 and f_2. Show that there exist infinitely many maximum flows.

Proof. Define $f_\lambda = \lambda f_1 + (1 - \lambda) f_2$ for $\lambda \in [0, 1]$, we can verify that all f_λ is a feasible flow. □

Exercise 6.12

Consider a flow network $G = (V, E)$ with a source s, a sink t and nonnegative capacities. Suppose a maximum flow f is given. If an arc is broken, find a fast algorithm to compute a new maximum flow based on f. A favorite algorithm will run in $O(|E| \log |V|)$ time.

Exercise 6.13

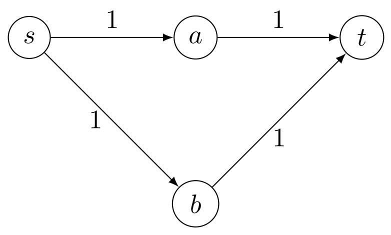
Consider bipartite graph $G = (U, V, E)$. Let $\mathcal{H}$ be the collection of all subgraphs H that for every $u \in U$, H has at most one edge incident to u. Let $E(H)$ denote the edge set of H and $\mathcal{I} = \{E(H) \mid H \in \mathcal{H}\}$. Show that (a) $(E, \mathcal{I})$ is a matroid and (b) all matchings in G form an intersection of two matroids.

Exercise 6.14

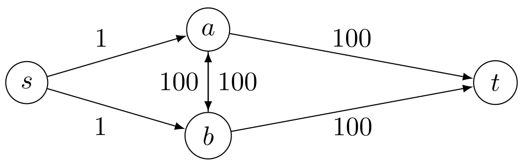
Prove or disprove (by counterexample) following statement.

1. If a flow network has unique maximum flow, then it has unique minimum s-t cut.
2. If a flow network has unique minimum s-t cut, then it has unique maximum flow.
3. A maximum flow must associate with a minimum s-t cut such that the flow passes through the minimum s-t cut.
4. A minimum s-t cut must associate with a maximum flow such that the flow passes through the minimum s-t cut.

Proof. First is false, here is a counterexample:



Second is false, here is a counterexample:



The third and the fourth are correct. ☐

Exercise 6.15

Feasible transshipment via cuts

Let $G = (V, E)$ be a directed graph with edge capacities $u : E \to \mathbb{R}_{\geq 0}$. Each vertex $v \in V$ has a *supply/demand* $b(v) \in \mathbb{R}$ such that $\sum_{v \in V} b(v) = 0$. We interpret $b(v) > 0$ as a supply of $b(v)$ units that must leave v and $b(v) < 0$ as a demand of $-b(v)$ units that must arrive at v. A *transshipment flow* is a function $f : E \to \mathbb{R}_{\geq 0}$ such that

$$\begin{aligned} 0 \leq f(e) &\leq u(e) && \text{for all } e \in E, \\ \sum_{e \in \delta^+(v)} f(e) - \sum_{e \in \delta^-(v)} f(e) &= b(v) && \text{for all } v \in V. \end{aligned}$$

1. Construct an auxiliary network $G' = (V', E')$ by adding a super-source s and a super-sink t as follows: (8 marks)

   - For every v with $b(v) > 0$, add an edge (s, v) with capacity $b(v)$.
   - For every v with $b(v) < 0$, add an edge (v, t) with capacity $-b(v)$.
   - Keep all original edges E with their capacities $u(e)$.

   Let $B := \sum_{v: b(v) > 0} b(v)$ be the total supply. Prove that there exists a feasible transshipment flow in G if and only if G' has an s–t flow of value B saturating every edge out of s and into t.

2. For a subset $S \subseteq V$, define the capacity of the outgoing cut by

   $$c(\delta^+(S)) := \sum_{e=(x,y) \in E,\ x \in S,\, y \notin S} u(e).$$

   Use the max-flow min-cut theorem on G' to prove the following: (12 marks)

Theorem. A feasible transshipment flow exists in G if and only if for every subset $S \subseteq V$,

$$\sum_{v \in S} b(v) \ \leq \ c(\delta^+(S)).$$

(Hint: show that any s–t cut in G' corresponds to some subset $S \subseteq V$ and interpret the cut capacity.)

3. Specialize the theorem in part (b) to the case where all supplies are at a single vertex s (so $b(s) > 0$, $b(v) \leq 0$ for $v \neq s$) and all demands are at a single vertex t. Show that the condition reduces to the usual max-flow/min-cut condition between s and t. (5 marks)

Exercise 6.16

Shipping as much as possible before a deadline

Let $G = (V, E)$ be a directed graph with distinct vertices $s, t \in V$. Each edge $e = (u, v)$ has unit capacity and an integer transit time$\tau(e) \in \{1, 2\}$. Time is discrete: $0, 1, 2, \ldots$. We want to send as many unit *packets* as possible from s (departing at time 0) to t so that each packet arrives by time T. Each packet travels along a directed path in G, moving forward one edge at a time according to the transit times, and at any time step at most one packet may use any given edge.

1. Build the *time-expanded network* $G^T = (V^T, E^T)$ as follows: (10 marks)

   - For each vertex $v \in V$ and time $k = 0, 1, \ldots, T$, create a vertex $(v, k) \in V^T$.
   - For each edge $e = (u, v) \in E$ with $\tau(e) = 1$ and each k with $0 \leq k \leq T - 1$, add an edge $((u, k), (v, k + 1))$ of capacity 1.
   - For each edge $e = (u, v) \in E$ with $\tau(e) = 2$ and each k with $0 \leq k \leq T - 2$, add an edge $((u, k), (v, k + 2))$ of capacity 1.
   - For each vertex $v \in V$ and each k with $0 \leq k \leq T - 1$, add a *waiting edge* $((v, k), (v, k + 1))$ of infinite capacity.

   The source is $(s, 0)$ and the sink is (t, T). Prove that integer $((s, 0), (t, T))$-flows in G^T of value F correspond bijectively to ways of sending F packets from s to t that arrive by time T in the original dynamic model.

2. Use the max-flow min-cut theorem on G^T to show that the maximum number of packets that can arrive at t by time T is equal to the minimum capacity of an $(s, 0)$–(t, T) cut in G^T. Give an interpretation of such a cut as a set of *time-stamped edges* in the original network whose removal blocks all packets from meeting the deadline. (10 marks)

3. Assume now that all transit times are $\tau(e) = 1$ and T is fixed. Show that the maximum number of packets that can arrive by time T is exactly the maximum number of pairwise edge-disjoint s–t paths of length at most T in G. (Hint: describe the correspondence between such paths and flow paths in G^T.) (5 marks)

6.3 Matchings

Exercise 6.17

Show that the vertex-cover problem in bipartite graphs can be solved in polynomial-time.

Exercise 6.18

Show that in any graph $G = (V, E)$ (not necessarily bipartite), the size of any maximal matching M (i.e. a matching M in which one cannot add an edge while keeping it a matching) is at least half the size of a maximum matching M^*.

Exercise 6.19

Consider a bipartite graph $G = (V, E)$ with bipartition (A, B): $V = A \cup B$. Assume that, for some vertex sets $A_1 \subseteq A$ and $B_1 \subseteq B$, there exists a matching M_A covering all vertices in A_1 and a matching M_B covering all vertices in B_1. Prove that there always exists a matching covering all vertices in $A_1 \cup B_1$.

Exercise 6.20

Consider the following 2-person game on a (not necessarily bipartite) graph $G = (V, E)$. Players 1 and 2 alternate and each selects a (yet unchosen) edge e of the graph so that e together with the previously selected edges form a simple path. The first player unable to select such an edge loses. Show that if G has a perfect matching then player 1 has a winning strategy.

Exercise 6.21

Deficiency formula for bipartite matchings

Let $G = (L \cup R, E)$ be a bipartite graph with bipartition (L, R). For a subset $S \subseteq L$ let $N(S)$ denote its neighborhood in R, and define the *deficiency* of S by

$$\delta(S) := |S| - |N(S)|.$$

Let $\Delta(G) := \max_{S \subseteq L} \delta(S)$ be the maximum deficiency.

1. Prove that every matching M in G satisfies (5 marks)

$$|M| \ \leq \ |L| - \Delta(G).$$

   (Hint: Fix $S \subseteq L$ and count how many vertices of L can be matched.)

2. Show that there exists a matching M in G with (10 marks)

$$|M| = |L| - \Delta(G).$$

   (Hint: You may use Hall's theorem and/or König's theorem without proof.)

3. Deduce the alternative "min–max" form (5 marks)

$$\max\{\,|M| : M \text{ is a matching in } G\,\} \;=\; \min_{S\subseteq L}\bigl(|L\setminus S| + |N(S)|\bigr).$$

Exercise 6.22

Maximizing matched important vertices

Let $G = (L \cup R, E)$ be a bipartite graph. Suppose that a subset $L_{\text{imp}} \subseteq L$ of the left vertices are declared *important*, and the remaining vertices $L\setminus L_{\text{imp}}$ are *unimportant*. We are interested in matchings that match as many important vertices as possible (we do not care whether unimportant vertices are matched).

1. For a given matching M, let (8 marks)

$$\text{sat}(M) := |\{\ell \in L_{\text{imp}} : \ell \text{ is matched in } M\}|$$

   be the number of matched important vertices. Show that for every subset $S \subseteq L_{\text{imp}}$ and every matching M,

$$\text{sat}(M) \;\le\; |L_{\text{imp}} \setminus S| + |N(S)|.$$

   Conclude that

$$\max\{\,\text{sat}(M) : M \text{ is a matching}\,\} \;\le\; \min_{S\subseteq L_{\text{imp}}}\bigl(|L_{\text{imp}}\setminus S| + |N(S)|\bigr).$$

2. Consider the induced bipartite graph $G_{\text{imp}} = (L_{\text{imp}} \cup R, E_{\text{imp}})$ obtained by deleting all vertices in $L \setminus L_{\text{imp}}$ and their incident edges. Use exercise 6.21 for G_{imp} to show that there exists a matching M in the original graph G such that (12 marks)

$$\text{sat}(M) = \min_{S\subseteq L_{\text{imp}}}\bigl(|L_{\text{imp}}\setminus S| + |N(S)|\bigr).$$

   (Hint: start with a maximum matching in G_{imp} and extend it to a matching in G without decreasing $\text{sat}(M)$.)

3. Give a short interpretation of the formula (5 marks)

$$\max\{\,\text{sat}(M)\,\} = \min_{S\subseteq L_{\text{imp}}}\bigl(|L_{\text{imp}}\setminus S| + |N(S)|\bigr)$$

   in terms of "how badly Hall's condition can fail" on the set of important vertices.

Index