CO450: Combinatorial Optimization

Felix Zhou

Fall 2020, University of Waterloo
from Professor Ricardo Fukasawa's Lectures

©Felix Zhou

©Felix Zhou

Contents

©Felix Zhou

©Felix Zhou

©Felix Zhou

©Felix Zhou

©Felix Zhou

©Felix Zhou

Part I

Minimum Spanning Trees

©Felix Zhou

©Felix Zhou

Chapter 1

Minimum Spanning Trees

We strive towards a minimum cost spanning tree (MST). The first goal is to characterize MSTs and use characterizations to derive algorithms. Finally, we employ linear programming to provide an alternative proof of correctness.

1.1 Spanning Trees

Definition 1.1.1
A subgraph T of $G = (V, E)$ is a spanning tree if
(i) $V(T) = V(G)$
(ii) T is connected
(iii) T is acyclic

©Felix Zhou

Theorem 1.1.1
Let G be connected and a subgraph T where $V(T) = V(G)$. The following are equivalent (TFAE).

(a) T is a spanning tree of G

(b) T is minimally connected

(c) T is maximally acyclic

(d) T is connected and has $n-1$ edges

(e) T is acyclic and has $n-1$ edges

(f) for all $u, v \in V$, there is a unique uv-path in T ($T_{u,v}$)

Theorem 1.1.2
G is connected if and only if for all $A \subseteq V$ with $\varnothing \neq A \neq V$, we have

$$\delta(A) \neq \varnothing.$$

1.2 Minimum Spanning Trees

Problem 1 (Minimum Spanning Tree)
Given a connected graph G and costs $c : E \to \mathbb{R}$, find a spanning tree T of G minimizing

$$c(T) := \sum_{e \in E(T)} c_e.$$

Theorem 1.2.1
Let G be connected, $c : E \to \mathbb{R}$, and T a spanning tree of G. TFAE

(a) T is a MST

(b) for all $uv \in E \setminus E(T)$, all edges e on the subpath $T_{u,v}$ of T have $c_e \leq c_{uv}$

(c) for all $e \in E(T)$, let T_1, T_2 be the two connected components of $T - e$. Then e is a minimum cost edge in $\delta(T_1) = \delta(T_2)$

Proof
$\neg(b) \implies \neg(a)$ Suppose there is some $uv \in E \setminus E(T)$ and $e \in T_{u,v}$ such that

$$c_e > c_{uv}.$$

Then $T + uv - e$ is a spanning tree of strictly smaller cost.

$\neg(c) \implies \neg(b)$ Suppose there is some $e \in T$ where T_1, T_2 are the connected components of $T - e$, such that there is $uv \in \delta(T_1)$ such that

$$c_e > c_{uv}.$$

Then e is on the subpath $T_{u,v}$ but has strictly greater cost than uv.

$(c) \implies (a)$ Suppose T satisfies the (c) and T^* is a MST maximizing

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

If $k = n - 1$, then $T = T^*$ and we are done. Otherwise, there is some

$$uv \in E(T) \setminus E(T*).$$

Let T_1, T_2 be the connected components of $T - uv$. There is some $e^* \in E(T^*_{u,v}) \cap \delta(T_1)$.

Remark that $T' := T^* - e^* + uv$ is a spanning tree with $c_{uv} \leq c_{e^*}$ by assumption and so k was not maximal.

1.3 Kruskal's Algorithm

1.3.1 Pseudocode & Correctness

```
def Kruskal(G):
  H := (V, {})
  while H is not spanning tree:
    e := min cost edge for e if endpoints not in same components of H
    H.add(e)

  return H
```

To see that the above algorithm terminates, first observe that H is always acyclic. Thus while it is not a spanning tree, it necessarily has multiple components, of which the cut in G is not empty. Moreover, the number of components reduces by 1 every iteration. Thus the algorithm terminates in $O(n)$ iterations with a connected, acyclic graph, which is precisely a spanning tree.

An equivalent implementation is the following. We provide it to show that the outputted

©Felix Zhou

spanning tree is a MST.

```
def Kruskal(G):
  H := (V, {})
  E.sort(lambda e: c(e))

  for uv in E:
   if component(u) != component(v):
    H.add(uv)

  return H
```

Now, suppose the algorithm above does not return an MST. Since we showed it returns a spanning tree,

$$\exists uv \in E \setminus E(H), \exists e \in H_{u,v}, c_{uv} < c_e.$$

But then uv would have been processed before e and added to H instead of e. By contradiction, Kruskal's algorithm returns a MST.

1.3.2 Running Time

A naive implementation records the component of each vertex through an array. Checking equality is $O(1)$ time but updates take $O(n)$ time. The total running time is

$$O(m \log m) + O(mn) \subseteq O(mn).$$

Checking equality can be implemented in amortized $O(m \log^*(n))$ total time with union-find data structures. The $\log^*$ function can be thought of as the inverse of the tower exponential function. Specifically, it is sub-logarithmic.

In fact, the analysis can be improved to $O(m\alpha(m, n))$ time where α is the inverse Ackermann function. For all practical inputs, α is essentially a constant.

Thus if we can sort the edges in near linear time, then we can actually find an MST in near linear time.

1.4 Linear Programming Relaxation

The goal of this section to prove the correctness of Kruskal's algorithm through complementary slackness.

We formulate an LP (P_{ST}) for the MST problem by using the characterization of spanning trees being acyclic subgraphs with $n-1$ edges.

$$\begin{aligned} &\min c^T x \\ x(E) &= n-1 = n-\kappa(E) \\ x(F) &\leq n-\kappa(F) && \forall F \subset E \\ x &\geq 0 \end{aligned}$$

Here $\kappa : E \to \mathbb{N}$ denotes the number of connected components of induced by an edge subset.

Recall that since we assume the input is connected and the feasibility region is bounded, the Fundamental Theorem of LPs tells us that there is always an optimal solution.

Proposition 1.4.1
The characteristic vector of the spanning tree produced by Kruskal's algorithm is an optimal solution to the MST LP.

First, let us consider the dual (D_{ST}).

$$\begin{aligned} &\max \sum_{F \subseteq E} (n-\kappa(F)) y_F \\ \sum_{F: e \in F} y_F &\leq c_e && \forall e \in E \\ y_F &\leq 0 && \forall F \subset E \end{aligned}$$

Proof (Proposition)
Let us order the edges based on their costs

$$E = \{e_1, e_2, \ldots, e_m\}$$

with $c_{e_i} \leq c_{e_{i+1}}$ for all $1 \leq i < m$.

Consider the following dual variables

$$\begin{aligned} \bar{y}_{E_i} &:= c_{e_i} - c_{e_{i+1}} \leq 0 && \forall 1 \leq i < m \\ \bar{y}_E &:= c_{e_m} \\ \bar{y}_F &:= 0 && \text{else} \end{aligned}$$

Here

$$E_i := \{e_1, \ldots, e_i\}.$$

It is not hard to see that $\bar{y}$ is feasible for (D_{ST}). In fact, a telescoping sum argument shows us that the dual edge constraints are tight!

©Felix Zhou

Let $\bar{x}$ be the characteristic vector of the spanning tree outputted by Kruskal's algorithm. We claim that

$$T_i := (V, E_i \cap E(T))$$

is a maximally acyclic subgraph of $H_i = (V, E_i)$. Indeed, this is because we add e_i to T_{i-1} if and only if the endpoints are in different components. But if the endpoints are in different components, adding e_i preserves acyclicness. On the other hand, if the endpoints do reside in the same component, the addition of e_i necessarily introduce a cycle since H_{i-1} was maximally acyclic.

This yields that

$$\bar{x}(E_i) = n - \kappa(E_i)$$

for each $1 \leq i \leq m$.

It follows that $(\bar{x}, \bar{y})$ satisfy complementary slackness, since the dual variables excluding the y_{E_i}'s are set to 0. Therefore, the pair are optimal solutions in their respective LPs.

1.5 Greedy Algorithms

Roughly speaking, a greedy algorithm makes decisions based on only local structures. This does not always work for example for the maximum independent set problem.

1.5.1 Maximum Forest

Problem 2 (Maximum Forest)
For an undirected graph G with edge weights $c : E \to \mathbb{R}$, compute an acyclic subgraph F maximizing

$$\sum_{e \in E(F)} c(e).$$

Minimum Spanning Tree Reduction

Consider the following algorithm

1) If G is not connected, minimally connect the components with edges of cost -1
2) Compute a MST T with respect to costs $c'_e := -c_e$ (of the original edges)
3) Delete from T all edges with $c'_e \geq 0$

Proposition 1.5.1
The proposed algorithm outputs a maximum forest.

Proof
Remark that the MST T from 2) is acyclic and thus is a forest. Moreover, deleting edges cannot introduce cycles. Thus the algorithm outputs SOME forest.

We claim that it indeed obtains the weight of a maximum forest. Suppose otherwise and let F^* be any maximum forest.

For each component C of F^*, there is at least one edge of T in $\delta_G(C)$. Repeatedly connected the components of F^* with edges of T. In other words, obtain a spanning tree T^* from the edges of F^*, T. Notice that any edges of $E(T^*) \setminus E(F^*)$ are necessarily non-negatively weighted with respect to c'_e, or else F is not maximal.

Without loss of generality, let us suppose that among all spanning trees obtained from a maximum forest, T^* maximizes

$$|E(T^*) \cap E(T)|.$$

If $T^* = T$, then we are done, since deleting the non-negative edges of T achieves at most the weight of F.

Otherwise, there is some $uv \in E(T^*) \setminus E(T)$. Notice that we actually have $uv \in E(F^*)$ so $c'_{uv} < 0$.

By our characterization of MSTs, all edges e on $T_{u,v}$ satisfy

$$c'_e \leq c'_{uv}.$$

In particular, if T_1^*, T_2^* are the components of $T^* - uv$, then we can pick some

$$f \in T_{u,v} \cap \delta_G(T_1^*)$$

so that

$$T' := T^* - uv + f$$

is a spanning tree with weight at most that of T'.

Clearly T' attains a higher cardinality edge intersection with T. We need only show that it can be obtained from some maximum forest F' to arrive at the desired contradiction.

To see this, take

$$F' := F^* - uv + f.$$

©Felix Zhou

It is clearly acyclic since the supergraph T' is acyclic. Moreover, it has weight at most that of F^* and thus is necessarily a maximum forest.

Now, we need only check that the reduction from disconnected graph to a connected graph is valid. It is clear that any maximum forest does not contain positive edges with respect to c'. Thus any maximum forest in a disconnected input has weight at most that of the returned solution in the augmented graph. Conversely, any returned forest in the augmented graph is a forest of the disconnected input and thus has weight at most that of an optimal solution in the disconnected input.

Direct Approach

```
def Kruskal(G):
  H := (V, {})
  E.sort(lambda e: c(e))

  for uv in E:
   if c(uv) <= 0:
    break
   elif component(u) != component(v):
    H.add(uv)

  return H
```

1.5.2 Minimum Spanning Tree Reduction to Maximum Forest

1) Take $c'_e := -(c_e - M) > 0$ for all $e \in E$
2) Solve the maximum cost forest with respect to c'

Proposition 1.5.2
Assuming that the input is connected, the above algorithm returns a MST with respect to c.

Proof
First remark that any MST with respect to $-c'$ is a MST with respect to c as all spanning trees have $n-1$ edges and thus all spanning trees have their weights shifted by a constant $(n-1)M$. Thus it suffices to argue that the proposed algorithm finds a maximum spanning tree with respect to c'.

©Felix Zhou

Observe that any acyclic subgraph F of G with less than $n-1$ edges is not optimal as there is some spanning tree of G which contains F and has more weight since all edges have positive weight c'. Thus all maximum forests are spanning trees with respect to c' and the weight of a maximum forest is at most that of a maximum spanning tree.

But any spanning tree is also a forest, thus the maximum spanning tree has weight at most that of a maximum forest.

Having considered both directions, we conclude the argument.

©Felix Zhou

©Felix Zhou

Part II

Matroids

©Felix Zhou

©Felix Zhou

Chapter 2

Matroids

We wish to optimize over more abstract senses of acyclic (independence).

2.1 Matroids

2.1.1 Definitions

Let S be a ground set and $\mathcal{I} \subseteq 2^S$.

Definition 2.1.1 (Basis)
A basis of $A \subseteq S$ is an inclusionwise maximal element of $\mathcal{I}$ contained in A.

Definition 2.1.2 (Matroid)
A matroid is a pair $M = (S, \mathcal{I})$ satisfying
(M1) $\varnothing \in \mathcal{I}$
(M2) If $F \in \mathcal{I}$ then $F' \subseteq F \implies F' \in \mathcal{I}$
(M3) For all $A \subseteq S$, every basis of A has the same cardinality

For our purposes, we will restrict ourselves to finite ground sets.

© Felix Zhou

Example 2.1.1 (Graphical/Forest Matroid)
For a graph $G = (V, E)$, let $\mathcal{I}$ be the set of all forests of G. Then

$$M = (S = E, \mathcal{I})$$

is a matroid.

Example 2.1.2 (Uniform Matroid of Rank r)
Let $S := [n]$ and $0 \leq r \leq n$. Put $\mathcal{I}$ as the set of all subsets of S with at most r elements.

$$U_n^r = (S, \mathcal{I})$$

is a matroid.

Example 2.1.3 (Linear Matroid)
Let $N \in \mathbb{F}^{m \times n}$ for some field $\mathbb{F}$ and $S := [n]$. Put

$$\mathcal{I} = \{A \subseteq S : \text{columns indexed by } A \text{ are linearly independent}\}.$$

Then

$$M = (S, \mathcal{I})$$

is a matroid.

2.1.2 Independence Systems

Elements of $\mathcal{I}$ are called *independent sets*. Minimal dependent sets are *circuits*.

If the pair $M = (S, \mathcal{I})$ satisfies (M1), (M2), it is an *independence system*.

Definition 2.1.3 (Rank)
Let $(S, \mathcal{I})$ be an independence system.
The rank of $A \subseteq S$ is

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

In the specific case, $r(S)$ is the rank of M. Moreover, observe that

$$r(A) = |A| \iff A \in \mathcal{I}.$$

Put

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

© Felix Zhou

Proposition 2.1.4
M is a matroid if and only if

$$\rho(A) = r(A)$$

for all $A \subseteq S$.

2.2 Independent Set

Problem 3 (Maximum Weight Independent Set)
Given an independence system $M = (S, \mathcal{I})$ and $c : E \to \mathbb{R}_+$, find $A \in \mathcal{I}$ maximizing

$$c(A) := \sum_{e \in A} c_e.$$

Consider the following greedy algorithm:

1) $F := \varnothing$
2) While there is $e : F \cup \{e\} \in \mathcal{I}$ and $c_e > 0$:
   a) Choose e with largest c_e
   b) $F := F \cup \{e\}$
3) Return F

Theorem 2.2.1 (Rado, Edmonds)
Let M be a matroid and $c \in \mathbb{R}_+^S$. The greedy algorithm finds the maximum weight independent set.

Proof
By Jenkins' theorem to come.

Theorem 2.2.2
Let $M = (S, \mathcal{I})$ be an independence system. The greedy algorithm finds an optimal independent set for all $c \in \mathbb{R}_+^S$ if and only if M is a matroid.

Proof (Rado, Edmonds)
($\neg \impliedby \neg$) Suppose that M is not a matroid. There is some $A \subseteq S$ with bases A_1, A_2

satisfying

$$|A_1| < |A_2|.$$

Define

$$c(e) := \begin{cases} 1, & e \in A_1 \\ 1-\epsilon, & e \in A_2 \setminus A_1 \\ 0, & e \notin A_1 \cup A_2 \end{cases}$$

If we choose $\epsilon > 0$ such that

$$(1-\epsilon)|A_2| > |A_1|,$$

then the greedy algorithm fails to output A_2, which is the maximum weight independent set.

($\Longleftarrow$) By the previous theorem.

Fortunately, the greedy algorithm cannot perform arbitrarily badly.

Theorem 2.2.3 (Jenkins)
Let $(S, \mathcal{I})$ be an independence system. Let $g(S, \mathcal{I})$ be the weight of the independent set found by the greedy algorithm. Then

$$g(S, \mathcal{I}) \geq q(S, \mathcal{I}) \operatorname{OPT}(S, \mathcal{I}).$$

Here

$$q(S, \mathcal{I}) := \min_{A \subseteq S : r(A) \neq 0} \frac{\rho(A)}{r(A)}$$

is the rank quotient.

Proof
Order the ground set by weight

$$S = \{e_1, \ldots, e_n\}, c(e_1) \geq \cdots \geq c(e_n)$$

and define

$$S_j := \{e_1, \ldots, e_j\}.$$

Let $G \in \mathcal{I}$ be a solution obtained by the greedy algorithm, and $\sigma \in \mathcal{I}$ an optimal solution.

Put

$$G_j := G \cap S_j, \sigma_j := \sigma \cap S_j.$$

©Feng Zhou

We can express

$$\begin{aligned} c(G) &= \sum_{j=1}^{n} c(e_j)(|G_j| - |G_{j-1}|) \\ &= \sum_{j=1}^{n} |G_j|(c(e_j) - c(e_{j+1})) \qquad\qquad c(e_{n+1}) := 0 \\ &= \sum_{j=1}^{n} |G_j|\Delta_j. \end{aligned}$$

Recall that the greedy algorithm computes a maximal independent subset of S_j. Thus G_j is a basis of S_j and

$$\begin{aligned} & |G_j| \geq \rho(S_j) \\ & \Longrightarrow \\ & c(G) \geq \sum_{j=1}^{n} \rho(S_j)\Delta_j \\ & \qquad \geq \sum_{j=1}^{n} q(S, \mathcal{I}) r(S_j)\Delta_j \\ & \qquad \geq \sum_{j=1}^{n} q(S, \mathcal{I})|\sigma_j|\Delta_j \\ & \qquad = q(S, \mathcal{I})c(\sigma). \end{aligned}$$

It is easy to see that the greedy algorithm terminates in polynomial time assuming that we can test indepence efficiently.

2.3 Alternative Characterizations

2.3.1 Matroids

Theorem 2.3.1
Let $M = (S, \mathcal{I})$ be an independence system. Then M is a matroid if and only if it satisfies

$$(M3') : \forall X, Y \in \mathcal{I}, |X| > |Y| \implies \exists x \in X \setminus Y, Y \cup \{x\} \in \mathcal{I}.$$

©FelixZhou

Proof
$(M3' \implies M3)$ Fix a basis B of A. If any other basis B' is smaller, we can add an element to it from $B \subseteq A$ and remain independent. If any other basis is bigger, then our original basis could have been augmented with element of $B' \subseteq A$. Both contradict the definition of bases.

$(M3 \implies M3')$ Let $X, Y \in \mathcal{I}$ such that $|X| > |Y|$. Consider

$$A := X \cup Y.$$

Y is clearly not a basis of A since X is an independent subset of A with greater cardinality. Thus Y is not inclusion-wise maximal and we can choose some element of A to augment it. But our only choice is from $X \setminus Y$, yielding the desired result.

Example 2.3.2
For a graph G, let $W \subseteq V$ be a stable set. Put $k : V \to \mathbb{Z}_+$ and consider the following independence system

$$S = E, \mathcal{I} = \{F \subseteq E : |\delta(v) \cap F| \leq k_v, \forall v \in W\}.$$

It is easy to show that $(X, \mathcal{I})$ is a matroid through (M3').

2.3.2 Circuits & Bases

We can implicitly describe a matroid by indicating the set of circuits of M, denoted $\mathcal{C}$. Then

$$A \in \mathcal{I} \iff \nexists C \in \mathcal{C} : C \subseteq A.$$

This begs the question of which subsets of S can form $\mathcal{C}$ for some matroid?

Theorem 2.3.3
Let $M = (S, \mathcal{I})$ be a matroid. Then for all $A \in \mathcal{I}$ and $e \in S$

$$A \cup \{e\}$$

contains at most 1 circuit.

Proof
Let A be a minimal counterexample with respect to cardinality. There is some e such

that $A \cup \{e\}$ has two distinct circuits C_1, C_2.

We must have $e \in C_1 \cap C_2$, or else A already contains a circuit.

By the minimality of A,

$$A \cup \{e\} = C_1 \cup C_2.$$

Moreover, neither C_1, C_2 can be a subset of the other and we can choose

$$e_1 \in C_1 \setminus C_2, e_2 \in C_2 \setminus C_1.$$

Consider $A' := (C_1 \cup C_2) \setminus \{e_1, e_2\}$. If A' has a circuit C, then $C \neq C_1, C_2$ as we deleted an element from both those circuits. On the other hand

$$A' + e_1 = (A - e_2) + e$$

contains C_1 and C since $e_2 \notin C_1$. Thus A was not a minimal counterexample and we could have take

$$A - e_2.$$

By contradiction, $A' \in \mathcal{I}$. In addition, A is a basis of $C_1 \cup C_2$ as it is maximally independent. But so is A', as adding either e_1, e_2 results in a circuit. Thus A, A' are bases with

$$|A'| < |A|,$$

contradicting the definition of a matroid.

Theorem 2.3.4
Let $\mathcal{C} \subseteq 2^S$. Then $\mathcal{C}$ is the set of circuits of a matroid if and only if
(C1) $\varnothing \notin \mathcal{C}$
(C2) If $C_1, C_2 \in \mathcal{C}$ are such that $C_1 \subseteq C_2$, then $C_1 = C_2$
(C3) If $C_1 \neq C_2 \in \mathcal{C}$ and $e \in C_1 \cap C_2$, then there is a circuit C with $C \subseteq (C_1 \cup C_2) \setminus \{e\}$

Proof
($\implies$) (C1), (C2) are clear. Suppose (C3) is violated. Thus

$$A := (C_1 \cup C_2) \setminus \{e\} \in \mathcal{I}.$$

In particular, $A \cup \{e\}$ has 2 distinct circuits, contradicting the previous theorem.

($\impliedby$) Define

$$\mathcal{I} := \{A \subseteq S : \nexists C \in \mathcal{C}, C \subseteq A\}.$$

(M1), (M2) follows directly by definition. Suppose (M3) is false. We can choose A_1, A_2, bases of $A \subseteq S$ such that $|A_1| < |A_2|$, maximizing

$$|A_1 \cap A_2|.$$

Since A_1 is a basis, we can pick $e \in A_1 \setminus A_2$. Thus $A_2 \cup \{e\}$ contains a circuit C. Suppose $A_2 \cup \{e\}$ contains a another circuit C'. Observe that $e \in C'$ or else $C' \subseteq A_2 \in \mathcal{I}$. But then (C3) implies that there is a circuit in $C \cup C' \setminus \{e\} \subseteq A_2$. Hence C is the unique circuit of $A_2 \cup \{e\}$.

Since $A_1 \in \mathcal{I}$, we may choose some $f \in C \setminus A_1$. Then

$$A_3 := (A_2 \cup \{e\}) \setminus \{f\} \in \mathcal{I}$$

since we removed an element from the unique circuit.

But then, by greedily augmenting A_3 to a basis if neceesary, we may assume without loss of generality that A_3 is a basis of A with

$$|A_3| > |A_1|.$$

However, observe that

$$|A_3 \cap A_1| > |A_2 \cap A_1|,$$

which contradicts the choice of A_2, A_1.

Alternatively, we can characterize the independent sets of a matroid by listing its bases. The independent sets are the subsets of bases.

Theorem 2.3.5
Let $\mathcal{B} \subseteq 2^S$. $\mathcal{B}$ is the set of bases of a matroid $(S, \mathcal{I})$ if and only if
(B1) $\mathcal{B} \neq \varnothing$
(B2) For any $B_1, B_2 \in \mathcal{B}$ with $x \in B_1 \setminus B_2$, there is some $y \in B_2 \setminus B_1$ such that

$$(B_1 \setminus \{x\}) \cup \{y\} \in \mathcal{B}.$$

(basis exchange property)

© Yixin Zhou

Theorem 2.3.6
$\mathcal{B}$ is the set of bases of a matroid $(S, \mathcal{I})$ if and only if (B1) and (B2') below holds:
For any $B_1, B_2 \in \mathcal{B}$, where $y \in B_2 \setminus B_1$, there is some $x \in B_1 \setminus B_2$ such that

$$(B_1 \setminus \{x\}) \cup \{y\} \in \mathcal{B}.$$

©Felix Zhou

©Felix Zhou

Chapter 3

Polymatroids

Let us study some polyhedrals which are related matroids.

3.1 Motivation

Let $M = (S, \mathcal{I}), c \in \mathbb{R}_+^S$. Let $x \in \mathbb{R}^S$ be decision variables. Put (P_M) as the LP

$$\begin{aligned} \max\, & c^T x \\ x(A) &\leq r(A) && \forall A \subseteq S \\ x &\geq 0 \end{aligned}$$

Observe that any independent $J \in \mathcal{I}$ implies x^J is feasible for (P_M).

Theorem 3.1.1
Let $M = (S, \mathcal{I})$ be a matroid and G the solution returned by the greedy algorithm. Then x^G is optimal for (P_M).

Proof
To come later.

©Felix Zhou

3.2 Submodular Functions

Definition 3.2.1
$f : 2^S \to \mathbb{R}$ is submodular if for all $A, B \subseteq S$,

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

Example 3.2.1
For a graph G, let the ground set be V.

Then

$$f(A) = |\delta(A)|$$

is a submodular function.

Proposition 3.2.2
Let $M = (S, \mathcal{I})$ be a matroid. Then $r(A)$ is submodular.

Proof
Let $J_\cap$ be a basis of $A \cap B$. Extend $J_\cap$ to a basis J_B of B. Extend J_B to a basis $J_\cup$ of $A \cup B$.

Put

$$J' := J_\cup \setminus (J_B \setminus J_\cap)$$

and remark that $J' \in \mathcal{I}$ and $J' \subseteq A$.

It follows that

$$\begin{aligned} r(A) + r(B) &\geq |J'| + |J_B| \\ &= |J_\cup| - (|J_B| - |J_\cap|) + |J_B| \\ &= r(A \cup B) + r(A \cap B). \end{aligned}$$

©Felix Zhou

3.3 Polymatroids

Definition 3.3.1 (Polymatroid)
Let $f : 2^S \to \mathbb{R}$ be submodular. Then

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

is a polymatroid.

Proposition 3.3.1
We may assume for that the submodular function corresponding to a polymatroid satisfies $f(\varnothing) = 0$ and monotonicity.

3.4 Optimizing over Polymatroids

Let (P_f) be the primal LP

$$\begin{aligned} &\max c^T x && \\ x(A) &\leq f(A) && \forall A \subseteq S \\ x &\geq 0 && \end{aligned}$$

where f is submodular and monotone with $f(\varnothing) = 0$.

The dual (D_f) is then

$$\begin{aligned} &\min \sum_{A \subseteq S} f(A) y_A && \\ \sum_{A : e \in A} y_A &\geq c(e) && \forall e \in S \\ y &\geq 0 && \end{aligned}$$

3.4.1 Greedy Algorithm

Let the ground set by

$$S = \{e_1, \ldots, e_n\}$$

such that

$$c_{e_1} \geq \cdots \geq c_{e_k} > 0 \geq c_{e_{k+1}} \geq \cdots \geq c_{e_n}.$$

Moreover, put

$$S_j := \{e_1, \ldots, e_j\}.$$

The primal greedy algorithm will pick the solution

$$x(e_j) = \begin{cases} f(S_j) - f(S_{j-1}), & j = 1, \ldots, k \\ 0, & j > k \end{cases}$$

where $f(S_0) := 0$.

The dual greedy algorithm will choose

$$y(A) = \begin{cases} c(e_j) - c(e_{j+1}) \geq 0, & A = S_j, j = 1, \ldots, k-1 \\ c(e_k) & A = S_k \\ 0, & \text{else} \end{cases}$$

Theorem 3.4.1
x, y produced by the primal / dual greedy algorithm are optimal for (P_f) / (D_f) respectively.

Proof
We argue using complementary slackness conditions. First we show that x is feasible in the primal LP.

Take $A \subseteq S$. We will show for $0 \leq j \leq k$ inductively that

$$x(A \cap S_j) \leq f(A \cap S_j).$$

The base case where $j = 0$ and $S_j := \varnothing$ by convention holds.

$$0 = x(\varnothing) \leq f(\varnothing) = 0.$$

Otherwise, suppose the statement holds for some $j \geq 0$. If $e_j \notin A$, then

$$x(A \cap S_{j+1}) = x(A \cap S_j) \leq f(A \cap S_j) = f(A \cap S_{j+1})$$

so the inequality holds. On the other hand, if $e_j \in A$, then

$$\begin{aligned} x(A \cap S_{j+1}) &= x(A \cap S_j) + f(S_{j+1}) - f(S_j) \\ &\leq f(A \cap S_j) + f(S_{j+1}) - f(S_j) \\ &\leq f(A \cap S_{j+1}). \end{aligned} \tag{$*$}$$

($*$) To see this inequality, notice that

$$\begin{aligned} f(A \cap S_{j+1}) + f(S_j) &\geq f(A \cap S_{j+1} \cap S_j) + f(A \cap S_{j+1} \cup S_j) \\ &= f(A \cap S_j) + f(S_{j+1}). \end{aligned}$$

By construction $x(A) = x(A \cap S_k) \leq f(A \cap S_k) \leq f(A)$ by monotonicity. Thus x is indeed feasible in the primal LP.

Now consider the dual LP. Fix any $e_j \in S$ for $1 \leq j \leq k$. Those are the only possibly violated constraints since $y \geq 0$. Then

$$\sum_{A: e_j \in A} y(A) = c(e_k) + \sum_{i=j}^{k-1} c(e_i) - c(e_{i+1}) = c(e_j).$$

Thus the dual constaint for $e \in S$ is tight if and only if $c(e) > 0$.

Finally, we argue that x, y satisfies the complementary slackness conditions. Observe $x(e) > 0$ can only happen for $e = e_j, 1 \leq j \leq k$. But for those elements, we showed the dual contraints are tight.

Suppose now that the dual variable $y_A > 0$ for some $A \subseteq S$ is non-zero. This can only happend when $A = S_j, 1 \leq j \leq k$. We claim that $x(S_j) = f(S_j)$ for $1 \leq j \leq k$. To see this simply compute

$$\begin{aligned} x(S_j) &= \sum_{i=1}^{j} x(e_i) \\ &= \sum_{i=1}^{j} f(S_i) - f(S_{i-1}) \\ &= f(S_j). \end{aligned}$$

Thus x, y are feasible and satisfy complementary slackness conditions, implying that they are optimal in their respective LPs.

Corollary 3.4.1.1
Let $M = (S, \mathcal{I}), c \in \mathbb{R}^S$ and $J \in \mathcal{I}$. Then J is an inclusionwise minimal, maximum weight independent set if and only if

a) $e \in J$ implies $c_e > 0$

b) $e \notin J, J \cup \{e\} \in \mathcal{I}$ implies $c_e \leq 0$

c) $e \notin J, f \in J$, and $(J \cup \{e\}) \setminus \{f\} \in \mathcal{I}$ implies $c_e \leq c_f$

©Felix Zhou

Proof
($\Longrightarrow$) This direction is clear.

($\Longleftarrow$) Consider (P_r), the polymatroid with respect to the rank function of M. Define y to be a solution from greedy dual algorithm. Put x^J as the characteristic vector of J.

First, we argue that the three conditions imply that x^J, y satisfy the complementary slackness conditions. It would follow that x^J has maximum weight.

The definition of the dual yields that

$$\sum_{A:e_j \in A} y_A = c(e_j)$$

for all $j \leq k$. a) implies that $x^J(e_j) = 0$ for all $j > k$. Thus for all $j = 1, \ldots, n$

$$x(e_j) = 0 \vee \sum_{A:e_j \in A} y_A = c(e_j).$$

Pick $y_A > 0$. By construction, $A = S_j$ for some $j \leq k$. Consider

$$x^J(S_j) = |J \cap S_j| = |J_j|.$$

Suppose that $|J_j| < r(S_j)$. In other words, J_j is NOT a basis of S_j. This implies that there is some $e \in S_j \setminus J$ such that

$$J_j \cup \{e\} \in \mathcal{I}.$$

Case I: $J \cup \{e\} \in \mathcal{I}$: b) implies that $c_e \leq 0$. But then $e \in S_j, j \leq k$ implies $c_e > 0$. This is impossible.

Case II: $J \cup \{e\} \notin \mathcal{I}$: Extend $J_j \cup \{e\}$ to a basis J' of $J \cup \{e\}$. Notice that J is a basis of $J \cup \{e\}$. In particular, $|J'| = |J|$ and both are subsets of $J \cup \{e\}$.

There is some $f \in J \setminus J' \subseteq J \setminus S_j$ such that

$$J' = (J \cup \{e\}) \setminus \{f\} \in \mathcal{I}.$$

By c), $c_e \leq c_f$. But $y(S_j) = c(e_j) - c(e_{j+1}) > 0$ and $f \notin S_j$ implies that $c(e_j) > c(e_{j+1}) \geq c(f)$. This is a contradiction.

We have thus shown that $x^J(J_j) = r(S_j)$. Thus for all $A \subseteq S$,

$$y_A = 0 \vee x^J(S_j) = |J_j|.$$

x^J certainly then has maximum weight.

Lastly, a) says that J is inclusionwise minimal.

Chapter 4

Matroid Construction

Given a particular matroid, we would like to determine valid operations which result in other matroids.

4.1 Matroid Construction

4.1.1 Deletion

For a matroid $M = (S, \mathcal{I})$, fix $B \subseteq S$. Then

$$M \setminus B := (S', \mathcal{I}')$$

given by $S' := S' \setminus B$ and $\mathcal{I}' := \{A \subseteq S \setminus B : A \in \mathcal{I}\}$ is a matroid.

The reason for this is that bases which do not include elements of B are preserved.

4.1.2 Truncation

Let $k \in \mathbb{Z}_+$. Define

$$\mathcal{I}' := \{A \in \mathcal{I}, |A| \leq k\}.$$

Then $M' = (S, \mathcal{I}')$ is a matroid.

The reason for this is that non-basis augmentation still holds (M3').

©Felix Zhou

4.1.3 Disjoint Union

Suppose $M_i = (S_i, \mathcal{I}_i)$ are matroids for $1 \leq i \leq k$ such that

$$S_i \cap S_j = \varnothing.$$

Then

$$M_1 \oplus \cdots \oplus M_k := (S, \mathcal{I})$$

with $S = \bigcup_{i=1}^{k} S_i$ and

$$\mathcal{I} := \{A \subseteq S : A \cap S_i \in \mathcal{I}_i\}$$

is a matroid.

To see this note that if $A \subseteq S$, then all bases of A have the same cardinality due to all bases of $A \cap S_i$ having the same size.

4.1.4 Contraction

For $B \subseteq S$, let J be a basis of B. Then

$$M/B := (S', \mathcal{I}')$$

where $S' = S \setminus B$, and

$$\mathcal{I}' := \{A \subseteq S' : A \cup J \in \mathcal{I}\}$$

is a matroid.

Proposition 4.1.1
Let M be the forest matroid of $G = (V, E)$. Fix $B \subseteq E$, then M/B is the forest matroid of G/B.

Theorem 4.1.2
M/B is a matroid which is independent of the choice of J. Moreover

$$r_{M/B}(A) = r_M(A \cup B) - r_M(B).$$

Proof
(M1), (M2) are clear.

(M3) Let $A \subseteq S' := S \setminus B$. Let J' to be an M/B basis of A. Thus

$$J \cup J' \in \mathcal{I}.$$

We claim that $J \cup J'$ is an M basis of $A \cup B$. Suppose not, there is element $e \in A \cup B$ such that

$$J \cup J' \cup \{e\} \in \mathcal{I}.$$

But $e \notin B$ since J is maximally independent in B. Nor can we have $e \in A$ or else J' is not maximally independent in M/B.

It follows that

$$|J \cup J'| = r_M(A \cup B)$$

This in turn implies

$$|J'| = r_M(A \cup B) - |J| = r_M(A \cup B) - r_M(B).$$

4.1.5 Duality

Consider

$$M^* := (S, \mathcal{I}^*)$$

where

$$\mathcal{I}^* := \{A \subseteq S : S \setminus A \text{ has a basis of } M\} = \{A \subseteq S : r(S \setminus A) = r(S)\}.$$

Theorem 4.1.3
M^* is a matroid with rank function

$$r^*(A) = |A| + r_M(S \setminus A) - r_M(S).$$

Proof
(M1), (M2) are clear.

(M3) Let $A \subseteq S$ and J^* be an M^*-basis of A. Put J as a M-basis of $S \setminus A$. We can extend J to an M-basis J' of $S \setminus J^*$.

We claim that J' is an M-basis of S. Indeed $J^* \in \mathcal{I}^*$ implies that $r(S \setminus J^*) = r(S)$. Thus $|J'| = r(S \setminus J^*) = r(S)$ as required.

Next, we claim that $A \setminus J^* \subseteq J'$. Suppose otherwise that there is some

$$e \in (A \setminus J^*) \setminus J'.$$

Notice that $J' \subseteq S \setminus (J^* \cup \{e\})$. Since J' is an M-basis of S,

$$J^* \cup \{e\} \in \mathcal{I}^*.$$

This is the desired contradiction since J^* was a M^*-basis of A.

It follows by our previous claim that

$$\begin{aligned} |J'| &= |A \setminus J^*| + |J| \\ &= |A| - |J^*| + |J| \\ &\iff \\ |J^*| &= |A| - |J'| + |J| \\ &= |A| - r_M(S) + r_M(S \setminus A). \end{aligned}$$

Example 4.1.4
Consider the forest matroid M of a planar graph G. Let M^* be the dual matroid.

$$\begin{aligned} A \in \mathcal{I}^* &\iff r(E \setminus A) = r(E) \\ &\iff G[E \setminus A] \text{ has a spanning tree} \\ &\iff V(E \setminus A) \text{ is connected.} \end{aligned}$$

Cycles in G^* correspond to edge sets $\delta(S)$ in G. Thus minimal dependent sets in M^* are cycles and M^* is the forest matroid of G^*.

©Felix Zhou

Chapter 5

Matroid Intersection

5.1 Matroid Intersection

Problem 4 (Weighted Matroid Intersection)
Let $M_1 = (S, \mathcal{I}_1), M_2 = (S, \mathcal{I}_2)$ be matroids over the same ground set and $c : S \to \mathbb{R}_+$. Find $A \in \mathcal{I}_1 \cap \mathcal{I}_2$ maximizing

$$c(A).$$

The unweighted matroid intersection problem consists of uniform weights of 1.

Example 5.1.1
Let $G = (V, E)$ with bipartition V_1, V_2. Let

$$\mathcal{I}_i := \{A \subseteq E : |A \cap \delta(v)| \leq 1, \forall v \in V_i\}$$

for $i = 1, 2$.

This is then exactly the maximum cardinality bipartite matching problem!

Let $J \in \mathcal{I}_1 \cap \mathcal{I}_2$ and $A \subseteq S$. Let us try to account for the size of J by splitting it into a M_1 component and M_2-component.

$$|J| = |J \cap A| + |J \cap \bar{A}|.$$

©Felix Zhou

Theorem 5.1.2 (Matroid Intersection, Edmonds)
Suppose $M_i = (S, \mathcal{I}_i), i = 1, 2$ are matroids. Then

$$\max\{|J| : J \in \mathcal{I}_1 \cap \mathcal{I}_2\} = \min\{r_1(A) + r_2(\bar{A}) : A \subseteq S\}.$$

Consider the previous example of the bipartite matching matroid intersection problem. Let M be a maximum matching of G. By the matroid intersection theorem, there is some $A \subseteq E$ such that

$$|M| = r_1(A) + r_2(E \setminus A).$$

Let B_1 be a M_1-basis of A, and B_2 and M_2-basis of $E \setminus A$. Put

$$U_i := B_i \cap V_i, i = 1, 2.$$

as the vertices from V_i incident to an edge of B_i.

Proposition 5.1.3
(i) $|U_1| = r_1(A)$.
(ii) $|U_2| = r_2(E \setminus A)$.
(iii) $U_1 \cup U_2$ is a vertex cover of G.

Proof
Observe that by definition, every vertex of $A \cap V_1$ is incident to at most one vertex of B_1. But every vertex of $A \cap V_1$ is incident to at least one edge by maximal independence. Thus

$$|U_1| = |B_1| = r_1(A).$$

The proof of (ii) is similar.

Every edge e either lives in A or $E \setminus A$. In the case of the former, a vertex in U_1 is incident it. In the case of the latter, a vertex in U_2 is incident to it. In either case,

$$U_1 \cup U_2$$

is a vertex cover.

Theorem 5.1.4 (König)
For a bipartite graph G,

$$\nu(G) = \tau(G).$$

©felixzhou

5.2 Matroid Intersection Algorithm

5.2.1 Motivation

In the setting of bipartite matching, we are essentially looking for augmenting paths.

Specifically, we wish to find some

$$P : e_1, f_1, \ldots, e_m, f_m, e_{m+1}$$

such that

$$\begin{aligned} e_i &\notin J && 1 \le i \le m \\ f_i &\in J && 1 \le i \le m \\ J \cup \{e_1\} &\in \mathcal{I}_2 \\ J \cup \{e_{m+1}\} &\in \mathcal{I}_1 \\ J \cup \{e_i\} \setminus \{f_i\} &\in \mathcal{I}_1 && 1 \le i \le m \\ J \cup \{e_{i+1}\} \setminus \{f_i\} &\in \mathcal{I}_2 && 1 \le i \le m \end{aligned}$$

We refer to the condition above as $(\star)$.

Given such a P, we can define

$$J' := J \Delta P = J \cup \{e_1, \ldots, e_{m+1}\} \setminus \{f_1, \ldots, f_m\}.$$

Lemma 5.2.1
If P is an inclusion wise minimal subset of S satisfying $(\star)$, then

$$J' \in \mathcal{I}_1 \cap \mathcal{I}_2.$$

Proof
First observe that $J \cup \{e_i\} \notin \mathcal{I}_1$ for all $1 \le i \le m$. Otherwise, we can take

$$P : e_1, f_1, \ldots, e_{i-1}, f_{i-1}, e_i$$

as a strict subset satisfying $(\star)$.

Put

$$A := J \cup \{e_1, \ldots, e_{m+1}\}$$

and

$$A_i := A \setminus \{f_m, \ldots, f_i\}.$$

Let C_i be the (unique) M_1-circuit in $J \cup \{e_i\}$ for $1 \leq i \leq m$.

We claim that $C_i \subseteq A_{i+1}$. Otherwise, there is some $k > i$ such that

$$J \cup \{e_i\} \setminus \{f_k\} \in \mathcal{I}_1.$$

So

$$e_1, f_1, \ldots, e_i, f_k, e_{k+1}, \ldots, f_m, e_{m+1}$$

would also satisfy $(\star)$.

Now,

$$C_i \subseteq A_{i+1} = A_i \cup \{f_i\} \implies C_i \setminus \{f_i\} \subseteq A_i.$$

From $(\star)$, $C_i \setminus \{f_i\} \in \mathcal{I}_1$ means that we can extend it to an M_1-basis of A_i, say B_i. But $B_i \cup \{f_i\} \supseteq C_i$ and

$$B_i \subseteq A_i \subseteq A_{i+1} = A_i \cup \{f_i\}.$$

It follows that B_i is a M_1-basis of A_{i+1}. So $r_1(A_i) = r_1(A_{i+1})$ for every i and

$$r_1(J') = r_1(A).$$

Now, $J \cup \{e_{m+1}\} \in \mathcal{I}_1$ from $(\star)$ means that

$$r_1(J') = r_1(A) \geq |J| + 1 = |J'|.$$

Therefore, $J' \in \mathcal{I}_1$.

A symmetric argument shows that

$$J' \in \mathcal{I}_2.$$

5.2.2 Augmenting Paths

We create a graph to model the algorithm.

Suppose we have as input

$$S = \{e_1, \ldots, e_m\}, J \in \mathcal{I}_1 \cap \mathcal{I}_2.$$

Let the node set be

$$S \cup \{r, s\}.$$

This is the ground set with a source node r and sink node s.

©FelixZhou

We define the arc set as follows. For each e_i such that

$$e_i \notin J, J \cup \{e_i\} \in \mathcal{I}_2,$$

add an arc re_i.

For each e_j such that

$$e_j \notin J, J \cup \{e_j\} \in \mathcal{I}_1,$$

add an arc $e_j s$.

Moreover, for $e, f \in S$, we add the arc ef if either

$$e \notin J, f \in J, J \cup \{e\} \notin \mathcal{I}_1, J \cup \{e\} \setminus \{f\} \in \mathcal{I}_1$$

or alternatively the arc fe if

$$e \notin J, f \in J, J \cup \{e\} \notin \mathcal{I}_2, J \cup \{e\} \setminus \{f\} \in \mathcal{I}_2.$$

It is not hard to see that a rs-alternating dipath corresponds precisely to a path satisfying $(\star)$. In addition, we can simply find the shortest such path to get an inclusion-wise minimal path.

Lemma 5.2.2
If there is not path satisfying $(\star)$. then J is a maximum cardinality element of $\mathcal{I}_1 \cap \mathcal{I}_2$.

Proof
Let U be the elements of S reachable from r by a dipath.

If $U = \varnothing$, then J is an inclusion wise maximal element of $\mathcal{I}_2$ and thus maximum cardinality element in $\mathcal{I}_1 \cap \mathcal{I}_2$.

Otherwise, there is some $e \in U \setminus J$. Indeed, by definition there is an arc re_i for which

$$e_i \notin J, J \cup \{e_i\} \in \mathcal{I}_2.$$

Since es is not an arc, then $J \cup \{e\} \notin \mathcal{I}_1$.

Let C be the M_1-circuit of $J \cup \{e\}$. Since there is no ef with $f \in S \setminus U$, we must have

$$C \subseteq U \implies C \subseteq (J \cap U) \cup \{e\}.$$

Note that $J \cap U \in \mathcal{I}_1$ so $J \cap U$ is an M_1-basis of U.

Apply the same argument to show that $J \cap \bar{U}$ is a M_2-basis of J. Thus

$$|J| = |J \cap U| + |J \cap \bar{U}| = r_1(U) + r_2(\bar{U})$$

©felixzhou

and J attains the upper bound.

This indeed leads to a polynomial time algorithm for solving the matroid intersection problem, assuming that we have a polynomial time independence oracle.

In fact, the weighted version can also be tackled in a similar way within polynomial time.

We can also solve this using the LP

$$\begin{aligned} \max & c^T x & \\ x(A) &\leq r_1(A) & \forall A \subseteq S \\ x(A) &\leq r_1(A) & \forall A \subseteq S \\ x &\geq 0 & \end{aligned}$$

5.3 Generalizations

5.3.1 3-Way Intersection

It is worthy to note that the 3-way matroid intersection problem is $\mathcal{NP}$-hard.

$$\max |A| : A \in \mathcal{I}_1 \cap \mathcal{I}_2 \cap \mathcal{I}_3.$$

5.3.2 Matroid Partitioning

Definition 5.3.1 (Partitionable)
Let $M_i = (S, \mathcal{I}_i), 1 \leq i \leq k$ be matroids. We say $J \subseteq S$ is partitionable if

$$J = J_1 \dot{\cup} \ldots \dot{\cup} J_k$$

with $J_i \in \mathcal{I}_i, 1 \leq i \leq k$.

Theorem 5.3.1 (Matroid Partitioning; Edmonds, Fulkerson)
We have

$$\max\{|J| : J \text{ is partitionable}\} = \min_{A \subseteq S} \left\{ |\bar{A}| + \sum_{i=1}^{k} r_i(A) \right\}.$$

Qi Zhou

Proof
Let $S = \{e_1, \ldots, e_n\}$. Put S^i as a copy of S for each $1 \leq i \leq k$.

For $J \subseteq \bigcup_{i=1}^{k} S^i$, let J^0 be the corresponding set of elements in S

$$J^0 = \{e \in S : \exists i \in [k], e^i \in J\}.$$

For each i, let

$$M_i' := (S^i, \{J \subseteq S^i : J^0 \in \mathcal{I}_i\}).$$

Let

$$N_a := M_1' \oplus \cdots \oplus M_k'.$$

Put $S' := \bigcup_{i=1}^{k} S_i$ and

$$\mathcal{I}_b := \{A \subseteq S' : \forall e \in S, A \text{ has at most one copy of } e\}.$$

Finally, let $N_b = (S', \mathcal{I}_b)$.

Observe that J is independent in both N_a, N_b if and only if J^0 is partitionable. Moreover, in such case,

$$|J| = |J^0|.$$

It follows by the matroid intersection theorem that

$$\max\{|J^0| : J^0 \text{ is partitionable}\} = \min_{B \subseteq S'} \{r_a(B) + r_b(S' \setminus B)\}.$$

We claim that there is a minimizer B of the form

$$\bigcup_{e \in B^0} \{e^1, \ldots, e^k\}.$$

Suppose there is some $e^j \in B$ and let $e^k \in S' \setminus B$. Consider

$$B' := B \setminus \{e^j\}.$$

Let D be a M_b-basis of $S' \setminus B$, and notice that $D \subseteq S' \setminus B'$. If $D \cup \{e^j\} \in \mathcal{I}_b$, then

$$D \cup \{e^k\} \in \mathcal{I}_b$$

as well, which contradicts the maximality of D within $S' \setminus B$.

It follows that D is also a basis of $S' \setminus B'$ and

$$r_b(S' \setminus B') = r_b(S' \setminus B).$$

©Felix Zhou

Moreover, $r_a(B') \leq r_a(B)$ so B' is also a minimizer.

But then

$$r_a(B) = \sum_{i=1}^{k} r_i(B^0)$$

and

$$r_b(S' \setminus B) = |S \setminus B^0|.$$

©Felix Zhou

Part III

Matchings

©Felix Zhou

©Felix Zhou

Chapter 6

Matchings

6.1 Matchings

Definition 6.1.1
For a graph G, a subset $M \subseteq E$ is a matching if

$$\forall v \in V, |\delta(v) \cap M| \leq 1.$$

We say a vertex v is M-covered if it is incident with an edge of M. Otherwise, v is M-*exposed*.

6.1.1 Augmenting Paths

Definition 6.1.2 (Alternating Path)
For a graph G and a matching $M \subseteq E$, a path

$$P : v_1, \ldots, v_k$$

is M-alternating if

$$v_{i-1}v_i \in M \iff v_iv_{i+1} \notin M.$$

Furthermore, we say a M-alternating path is M-*augmenting* if v_1, v_k are M-exposed.

Recall that the symmetric difference between two sets A, B is

$$A \Delta B := (A \cup B) \setminus (A \cap B).$$

© Felix Zhou

Theorem 6.1.1 (Berge)
Let M be a matching of G. Then M is a maximum cardinality matching if and only if there does not exist a M-alternating path.

Proof
($\Longrightarrow$) If P is a M-augmenting path, then

$$M' := M \Delta E(P)$$

is a strictly larger matching.

($\Longleftarrow$) Suppose there is a larger matching M'. Consider

$$G[M \Delta M'].$$

Every vertex has degree at most 2 and thus G' is a disjoint union of alternating paths and cycles, with respect to both M, M'.

Since $|M'| > |M|$, at least one of the components of G' has more edges from M'. This can only happen in one of the paths, say P. The first and last edge of P must come from M', thus P is the desired augmenting path.

6.1.2 Alternating Trees

The idea is to continually augment matchings to find a maximum matching. Start at a M-exposed vertex r and compute a breadth-first search tree where odd distanced vertices (A) were "discovered" through non-matching edges and vice versa for even distanced vertices (B).

If we discover a matched vertex, then we can extend the tree by two vertices. Otherwise, if we discover an unmatched vertex, then we can augment our matching and initialize our tree from another exposed vertex.

This idea "almost" works.

6.2 Bipartite Graphs

Let us first consider an algorithm for bipartite graphs.

©Felix Zhou

```
def Bipartite_Matching(G):
  M := {}
  r in V
  T := ({r}, {})
  A := {} # odd distanced vertices
  B := {r} # even distanced vertices

  while there is vw in E: v in B(T) and w not in V(T):
    if w is M-covered by wx:
      use vw to extend T
      A.add(w)
      B.add(x)
    else:
      use vw to augment M
      if there is another M-exposed vertex r in V:
        T := ({r}, {})
        A := {}
        B := {r}
      else:
        return M
  return "no perfect matching"
```

6.2.1 Hall's Theorem

Theorem 6.2.1 (Hall)
Let G be a graph with bipartition $A \cup B$. There is a matching satisfying A if and only if

$$\forall X \subseteq A, |N(S)| \geq |X|.$$

Proof
($\neg \Longleftarrow \neg$) If there is $X \subseteq A$ such that

$$|X| > |N(X)|,$$

then not all vertices of X can be covered.

($\Longleftarrow$) Let us argue by induction on $|A|$. If $|A| \leq 1$, the result holds trivially.

Suppose that for every non-trivial subset $X \subseteq A$,

$$|N(X)| > |X|.$$

©Fei Zhou

Pick $uv \in E$ with $u \in A, v \in B$. Consider

$$G' := G - \{u, v\}.$$

This has a bipartition $A' := A - u, B' := B - v$.

Now, for all $X \subseteq A'$,

$$|N'_G(X)| \geq |N_G(X)| - 1 \geq |X|$$

and so Hall's conditions holds. By induction, there is a matching M' covering A'. It follows that

$$M' + uv$$

covers A.

Otherwise, there is some non-trivial $X \subseteq A$ such that

$$|N(X)| = |X|.$$

We know by induction that there is a matching M^* in $G[X \cup N(X)]$ covering X, since Halls conditions holds for that subgraph. Now, consider

$$G' := G[A \setminus X \cup N \setminus N(X)].$$

We wish to argue that Hall's condition still holds.

Indeed,

$$\begin{aligned} |N_{G'}(Y)| &= |N_G(X \cup Y)| - |N_G(X)| \\ &= |N_G(X \cup Y)| - |X| \\ &\geq |X \cup Y| - |X| \\ &= |Y|. \end{aligned}$$

Hence by induction, there is a matching satisfying $A - X$ in G'. Taken with the X-saturating matching in $G[X \cup N(X)]$, this yields the desired result.

Corollary 6.2.1.1
Let G be a graph with bipartition $A \cup B$. There is a perfect matching if and only if

$$|A| = |B| \wedge \forall X \subseteq A, |X| \leq |N(X)|.$$

©Felix Zhou

6.2.2 Correctness

Proposition 6.2.2
The bipartite matching algorithm correctly detects there is no perfect matching.

Proof
Let T, B, A be the state of the algorithm just before termination. We claim that

$$N(B(T)) = A(T)$$

as well as

$$|B(T)| > |A(T)|.$$

This shows that G has no perfect matching by the corollary to Hall's theorem.

The first claim follows by the loop condition of our algorithm. Moreover, notice that every non-root vertex in T is matched to a vertex of T of the opposite distance parity. Thus there is exactly one more element in $B(T)$ than $A(T)$.

It is clear that the bipartite matching algorithm runs in polynomial time. We can augment the matching at most $\frac{n}{2}$ times, with each augmentation requiring linear time to compute the alternating tree.

6.3 Vertex Covers

Definition 6.3.1 (Vertex Cover)
$U \subseteq V$ is a vertex cover if for each $e \in E$

$$|e \cap U| \geq 1.$$

Let τ be the size of a minimum vertex cover and ν the size of a maximum matching. It is clear that

$$\nu(G) \leq \tau(G).$$

Theorem 6.3.1 (König)
If G is bipartite, then

$$\nu(G) = \tau(G).$$

6.4 General Graphs

Let us now consider the scenario where odd cycles might exist.

Fix $A \subseteq V$ and consider $G - A$. Put

$$H_1, \ldots, H_k$$

as the odd components of $G - A$.

Suppose $k = \text{odd}(G - A)$. Then

$$\nu(G) \leq \frac{1}{2}\left(|V| - k + |A|\right).$$

Notice that if A is a vertex cover, then

$$\frac{1}{2}(|V| - k + |A|) = \frac{1}{2}(|V| - |V| + |A| + |A|) = |A|.$$

Thus our bound is AT LEAST as good as the vertex cover bound.

In fact, the following theorem holds.

Theorem 6.4.1 (Tutte-Berge Formula)
Let G be a graph.

$$\nu(G) = \frac{1}{2}\min\{|V| - \text{odd}(G - A) + |A| : A \subseteq V\}.$$

There is a slightly weaker statement.

Theorem 6.4.2
G has a perfect matching if and only if

$$\text{odd}(G - A) \leq |A|$$

for all $A \subseteq V$.

Proof
First observe for $A = \varnothing$, the statement becomes

$$\text{odd}(G) = 0$$

which is necessary for G to have a perfect matching.

Now, by the Tutte-Berge formula, G has a perfect matching if and only if

$$\nu(G) = \frac{n}{2} \iff n = \min_{A \subseteq V}\{n - \text{odd}(G - A) + |A|\}$$

if and only if

$$\min_{A \subseteq V}\{|A| - \text{odd}(G - A)\} = 0.$$

But for $A = \varnothing$,

$$|A| - \text{odd}(G - A) = 0.$$

The statement follows since we have deduced the minimum is at most 0. Thus the minimum is 0 if and only if the statement holds.

6.4.1 Tutte-Berge Formula

We now seek to prove the Tutte-Berge formula.

Definition 6.4.1 (Essential)
We say $u \in V$ is essential if u is M-covered in EVERY maximum cardinality matching M.

Otherwise, $u \in V$ is *inessential*.

Odd Cycles

Let V be an odd cycle and $G' := G/C$. Put $\mathcal{C}$ as the identified vertex. We will allow parallel edges but remove loops.

The idea is that a matching in G' can be extended to a matching in G with the same number of exposed vertices.

Proposition 6.4.3
Let $G = (V, E)$, C an odd cycle, and $G' := G/C$. Let M' be a matching in G'.
There exists a matching M of G such that the number of M-exposed vertices in G is equal to the number of M'-exposed vertices of G'.

It follows that

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

Unfortunately, equality does not always hold.

Definition 6.4.2 (Tight)
We say an odd cycle C is tight if

$$\nu(G) = \nu(G') + \frac{|C| - 1}{2}.$$

Tutte-Berge Formula

Lemma 6.4.4
If $uv \in E$ is such that u, v are both not essential, there is a tight odd cycle C containing uv.
Moreover, $\mathcal{C}$ is an inessential vertex of G'.

Proof
Observe no maximum matching exposes both u, v. Let M_1 be a maximum matching exposing u and M_2 a maximum matching exposing v. Consider

$$G[M_1 \cup M_2].$$

v is the endpoint of some path P. Walk from v until the other endpoint, say u'. If u' is M_2 exposed, P is a M_2-augmenting path. But if u' is a M_1-exposed vertex other than u, we can produce a M_1-augmenting path by appending uv to the beginning of P.

Thus $u' = u$ and $C := P + uv$ is a tight odd cycle containing uv.

To see that $\mathcal{C}$ is inessential in G', observe that $M_2 - E(C)$ is a maximum matching of G' not covering $\mathcal{C}$. This is because the inequality above yields

$$|M_2 - E(C)| = \nu(G) - \frac{|E(C)| - 1}{2} \geq \nu(G').$$

We are now ready to prove the Tutte-Berge Formula.

Proof (Tutte-Berge Formula)
Our goal is to produce a matching M and $A \subseteq V$ with exactly

$$\text{odd}(G - A) - |A|$$

M-exposed vertices. Then since the Tutte-Berge formula trivially provides an upper bound, we will have shown that we obtain the upper bound so equality must follow.

We argue by induction on $m = |E|$. The case where $m = 0$ is trivial.

Choose some $uv \in E$ arbitrarily. Suppose that one of the endpoints are essential, say v.

$\underline{v \text{ is Essential:}}$ Let $G' := G - v$. Then by definition, $\nu(G') < \nu(G)$.

By induction, there is a matching M' in G' and $A' \subseteq V - v$ with

$$|M'| = \frac{1}{2}(n - 1 - \text{odd}(G' - A') + |A'|).$$

Let M be a matching of G with

$$|M| = \nu(G).$$

Choose $e \in \delta(v) \cap M$. There must be such an edge since v is essential.

Then $\bar{M} := M - e$ is a matching in G'. It follows that

$$|\bar{M}| = |M| - 1 \leq |M'|.$$

But $|M'| \leq |M| - 1$ since v is essential. Thus

$$|M'| = |M| - 1.$$

Define

$$A := A' + v.$$

Notice that $\text{odd}(G - A) = \text{odd}(G' - A')$ by definition. It follows that

$$\begin{aligned} |M| - 1 &= \frac{1}{2}(n - 1 - \text{odd}(G' - A') + |A'|) \\ |M| &= \frac{1}{2}(n - 1 - \text{odd}(G - A) + |A| - 1) + 1 \\ &= \frac{1}{2}(n - \text{odd}(G - A) + |A|). \end{aligned}$$

as required.

$\underline{u, v \text{ are Both Not Essential:}}$ Choose a tight odd cycle C containing uv where $\mathcal{C}$ is inessential in $G' = G/C$. There is a matching M' of G' and $A' \subseteq V(G')$ where

$$|M'| = \frac{1}{2}(|V(G')| - \text{odd}(G' - A') + |A'|).$$

©Felix Zhou

Moreover, we may assume $\mathcal{C}$ is M'-exposed since $\mathcal{C}$ is inessential.

We claim that $\mathcal{C} \notin A'$. Indeed, since

$$|M'| = \text{odd}(G' - A') = |A'|,$$

it must be that every vertex of A' is matched to a vertex in an odd component of $G' - A'$. But $\mathcal{C}$ is inessential and therefore cannot be in A'.

We now know that $\mathcal{C} \notin A'$, thus any component of $G' - A'$ containing $\mathcal{C}$ will be a component of $G - A'$ of the same parity. Hence

$$\text{odd}(G' - A') = \text{odd}(G - A').$$

Extend M' to M with $\frac{|C|-1}{2}$ edges from C. But this does not change the number of exposed vertices since $\mathcal{C}$ contributed 1 in M' and the odd cycle C also contributes exactly 1 exposed vertex. Then

$$\text{odd}(G' - A') - |A'| = \text{odd}(G - A') - |A'|$$

M-exposed vertices. This concludes the proof.

©Felix Zhou

Chapter 7

Matching Algorithms

7.1 The Blossom Algorithm

Definition 7.1.1 (Frustrated)
We say an M-alternating tree T is frustrated if for all $uv \in E$ such that $u \in B(T)$, we have $v \in A(T)$.

Proposition 7.1.1
If T is frustrated, then G has no perfect matching.

Proof
Since all neighbors of $B(T)$ are in $A(T)$, $G - A(T)$ has at least $|B(T)|$ odd components.

Thus

$$\text{odd}(G - A(T)) \geq |B(T)| > |A(T)|$$

and the result follows from Tutte's Matching Theorem.

Unfortunately, it is possible that some vertices of $B(T)$ are adjacent to each other. Thus we cannot always find a frustrated M-alternating tree.

Let $u, v \in B(T)$ such that $uv \in E$. Then $T + uv$ has a unique odd cycle C (*Blossom*). Shrink the Blossom and let $G' := G/C$.

Informally, we can then continue our algorithm to extend our M-alternating tree in G'. Given a matching in G', we can extend it to one in G.

©Eric Zhou

7.1.1 Repeated Shrinking

Definition 7.1.2 (Derived Graph)
We say the graph obtained after (repeatedly) shrinking Blossoms a derived graph.

We will write $S(v)$ as the set of vertices which have been shrunk to form $v \in V(G)$. Recursively,

$$S(v) = \begin{cases} v, & v \in V(G) \\ \bigcup_{w \in C} S(w), & v = v_C \text{ for some blossom } C \end{cases}$$

Notice that $|S(v)|$ is always odd since we delete/add odd number of vertices for every shrink operation.

Proposition 7.1.2
Let G' be a derived graph from G and M' a matching of G'.
If T' is a M'-alternating frustrated tree of G' such that all pseudonodes are in $B(T')$, then G has no perfect matching.

Proof
Observe that

$$\text{odd}(G' - A(T')) \geq |B(T')| > |A(T')|$$

as before. As a reminder, this is because every neighbor of a vertex in $B(T')$ resides in $A(T')$.

But now, upon uncontraction of the Blossoms of $B(T')$, each odd component of $G' - A(T')$ remains an odd component of $G - A(T')$. Thus

$$\text{odd}(G - A(T')) \geq |B(T')| > |A(T')|$$

and the result follows once again by Tutte's Matching Theorem.

Proposition 7.1.3
Let G' be a derived graph from G, M' a matching of G', T' an M'-alternating tree, and $uv \in E(G')$ with $u, v \in B(T')$.
Put C' as the unique Blossom in $T' + uv$. Then $M'' := M' \setminus E(C')$ is a matching for $G'' = G'/C'$ and

$$T'' = (V(T') \setminus V(C') \cup \{v_{C'}\}, E(T') \setminus E(C'))$$

is an M''-alternating tree in G'' with $v_{C'} \in B(T'')$.

7.1.2 Perfect Matching Algorithm

```
def Pefect_Matching(G: Graph, M: Matching):
    M_prime := M
    G_prime := G
    r := M-exposed node
    T := ({r}, {})
    A := {} # odd distanced vertices
    B := {r} # even distanced vertices

    while there is vw in E such that v in B(T) and w not in A(T):
      if w not in V(T) and is M_prime-exposed:
        use vw to augment M_prime
        extend M_prime to a matching M of G
        M_prime := M
        G_prime := G

        if no M_prime-exposed node in G_prime:
          return perfect matching M_prime
        else:
          r := M_prime exposed node
          T := ({r}, {})
      elif w not in V(T) and is M_prime-covered by wx:
        use vw to extend T
        A.add(w)
        B.add(x)
      else: # w in B(T)
        use vw to shrink unique Blossom and update M_prime, T

    return G_prime, M_prime and "no perfect matching"
```

Theorem 7.1.4
The Blossom algorithm performs $O(n)$ augmentations, $O(n^2)$ shrinks, and $O(m)$ tree extensions.
Furthermore, it correctly determines if G has a perfect matching.

Proof
The correctness follows from our previous propositions about frustrated trees, as that is exactly what our algorithm returns if no perfect matching is found.

Each augmentation increases $|M'|$ by 1, thus there are at most $O(n)$ augmentations.

©Felix Zhou

> Between augmentation steps, shrinking reduces the size of G' by at least 2 vertices. Thus there are $O(n \cdot n)$ total shrinks.
>
> Finally, each edge is added to the tree at most once, hence there are $O(m)$ tree extensions.

7.1.3 Maximum Cardinality Matching Algorithm

```
def Maximum_Cardinality_Matching(G: Graph, M: Matching):
    M_prime := M
    G_prime := G
    T_prime := ({}, {})

    while there is an M_prime exposed node r of G_prime:
      T := ({r}, {})
      A := {} # odd distanced vertices
      B := {r} # even distanced vertices

      while there is vw in E such that v in B(T) and w not in A(T):
        if w not in V(T) and is M_prime-exposed:
          use vw to augment M_prime
          extend M_prime to a matching M of G
          M_prime := M
          G_prime := G

          if no M_prime-exposed node in G_prime:
            return perfect matching M_prime
          else:
            r := M_prime exposed node
            T := ({r}, {})
        elif w not in V(T) and is M_prime-covered by wx:
          use vw to extend T
          A.add(w)
          B.add(x)
        else: # w in B(T)
          use vw to shrink unique Blossom and update M_prime, T

      T_prime.add(T)
      G_prime.remove(V(T))
      M_prime.remove(E(T))

    return M_prime
```

©Felix Zhou

Proposition 7.1.5
The Blossom algorithm correctly computes a maximum cardinality matching.

Proof
Let $T_1, \ldots, T_k$ be the trees in $\mathcal{T}$ and M the final matching.

For each T_i, there is only one M-exposed vertex in T_i, namely, its root r_i. Thus there are k M-exposed vertices.

Put

$$A := \bigcup_{i=1}^{k} A(T_i).$$

By the inner while loop, $N(B(T_i)) \subseteq A(T_i)$ for each i. Thus each vertex in $B(T_i)$ is an odd component of $G - A$.

It follows that

$$\text{odd}(G - A) \geq \sum_{i=1}^{k} |B(T_i)| \geq \sum_{i=1}^{k} 1 + |A(T_i)| = |A| + k.$$

But

$$|M| = \frac{n-k}{2} \geq \frac{1}{2}\left(n - \text{odd}(G - A) + |A|\right).$$

Recall that the reverse inequality always holds for any matching and we are done.

Observe that this is an algorithmic proof of the Tutte-Berge formula.

7.2 Gallai-Edmonds Partition

Definition 7.2.1 (Gallai-Edmonds Partition)
Let $G = (V, E)$.
Put B as the set of inessential vertices. Let C be the neighbors of B in $V \setminus B$. Finally, set D as the rest of the vertices.
(B, C, D) is the Gallai-Edmonds Partition/Decomposition of G.

We should think of C as the minimizer of the Tutte-Berge formula. $G[B]$ only has odd components and in a maximum matching M, $M \cap E(D)$ is a perfect matching of $G[D]$.

Proposition 7.2.1
Let $T_1, \ldots, T_k$ be the frustrated trees found in the maximum cardinality Blossom algorithm.
Then

$$C = \bigcup_{i=1}^{k} A(T_i)$$
$$B = \bigcup_{i=1}^{k} \left(\bigcup_{v \in B(T_i)} S(v) \right)$$
$$D = V \setminus (B \cup C)$$

yields a Gallai-Edmonds Partition.

Observe that this implies all components of $G[B]$ are odd since they are isolated vertices or (repeatedly) expanded blossoms. In addition, C is the minimizer of the Tutte-Berge formula. It also means that $G[D]$ only has even components since all odd components of $G - A$ are precisely $G[B]$. Moreover, this implies that the Gallai-Edmonds decomposition can be computed in polynomial time.

Proof
We saw all vertices in $\bigcup_{i=1}^{k} A(T_i)$ are essential. This is because the Blossom algorithm finds a minimizer of the Tutte-Berge formula, of which all vertices are essential.

For all vertices in $\bigcup_{i=1}^{k} \left(\bigcup_{v \in B(T_i)} S(v) \right)$, there is an even M-alternating path from an M-exposed vertex r to it (a root of some frustrated tree). Take such an even path P and observe that

$$M' := M \Delta E(P)$$

is a matching with $|M'| = |M|$ exposing v. It follows that v is inessential.

Finally, we know that $G[D]$ only has even components. Put $v \in D$. We claim that

$$\nu(G - v) < \nu(G).$$

Indeed, we have already shown that A is a minimizer to the Tutte-Berge formula. Thus there are exactly k exposed vertices in a maximum matching. But there are at least k exposed vertices of B, thus no vertex of D can be exposed in a maximum matching. Thus shows that $G[D]$ contains a perfect matching as required.

Notice that $v \in D$ is NOT adjacent to a vertex of B while $v \in C$ is always matched (adjacent) to a vertex in B.

Chapter 8

Weighted Matchings

8.1 Minimum Weight Perfect Matching

Problem 5 (Minimum Weight Perfect Matching)
Given $G = (V, E)$ and $e : E \to \mathbb{R}$, find a perfect matching M minimizing

$$c(M) = \sum_{e \in M} c(e).$$

Consider the following LP

$$\begin{aligned} \min \sum_{e \in E} c_e x_e & \\ x(\delta(v)) = 1 & \qquad \forall v \in V \\ x \geq 0 & \end{aligned}$$

It dual is given by

$$\begin{aligned} \max \sum_{v \in V} y_v & \\ y_u + y_v \leq c_{uv} & \qquad \forall uv \in E \end{aligned}$$

Unfortunately the LP relaxation is not integral due to the possible existence of odd cycles.

©Feiran Zhou

8.1.1 Bipartite Graphs

Theorem 8.1.1 (Birkhoff)
Let G be bipartite and $c \in \mathbb{R}^E$.
Then G has a perfect matching if and only if the primal LP is feasible.
Moreover, if the primal LP is feasible, then for all minimum cost perfect matchings M^*,

$$\text{OPT} = c(M^*).$$

Proof
($\implies$) This direction is trivial.

($\impliedby$) We omit the proof that feasibility implies perfect matching and leave is as an exercise in the assignment.

It remains to show that the optimal solution is the cost of a minimum cost perfect matching. We will do so using complementary slackness.

Let $\bar{y}$ be feasible for the dual. Pick

$$E^= := \{uv \in E : \bar{y}_u + \bar{y}_v = c_{uv}\}.$$

If $G^= := (V, E^=)$ has a perfect matching M, then $\chi^M, \bar{y}$ satisfy complementary slackness and we are done.

Otherwise, update $\bar{y}$ as below.

Suppose the perfect matching algorithm terminated with a M-frustrated tree in $G^=$. Define

$$\epsilon = \min\{c_{uv} - \bar{y}_u - \bar{y}_v : u \in B(T), v \notin V(T)\}.$$

Update $\bar{y}$ as

$$\bar{y}_u := \begin{cases} \bar{y}_u + \epsilon, & u \in B(T) \\ \bar{y}_u - \epsilon, & u \in A(T) \\ \bar{y}_u, & u \notin V(T) \end{cases}$$

This ensures that $\bar{y}$ is still feasible for the dual. Moreover, $M \subseteq E^=$ for the new $E^=$ by construction. In fact, $E(T) \subseteq E^=$ since we increased/decreased the vertex potentials based on parity of distance from the root. We also ensure that at least one edge in $\delta(V(T))$ is added to $E^=$.

If there are any vertices $u \notin V(T)$, then this ensures at least one M-exposed vertex is found and we can augment our matching.

©FelixZhou

> Otherwise, by the termination condition of the Blossom algorithm, every $vw \in E$ with $v \in B(T)$ satisfies $w \in A(T)$ and G has no perfect matching.

```
def Min_Weight_Bipartite_Perfect_Matching(G: Graph, M: Matching, y:
    FeasibleDual):
  r := M-exposed node
  T := ({r}, {})
  A := {} # odd distanced vertices
  B := {r} # even distanced vertices

  while True:
    # find a perfect matching
    while there is vw in E_= such that v in B(T) and w not in V(T):
      if w is M_prime-exposed:
        use vw to augment M_prime
        if no M-exposed node in G:
          return M
        else:
          r := M_prime exposed node
          T := ({r}, {})
          A.clear()
          B.clear()
      else:
        use vw to extend T
        A.add(w)
        B.add(x)
    # update dual solution
    if all uv in E with v in B(T) is w in A(T):
      return "no perfect matching"
    else:
      eps := min(c(uv) for v in V(B) and w not in V(T))
      for v in B(T):
        y(v) += eps
      for v in A(T):
        y(v) -= eps
```

The correctness of the algorithm follows from Birkhoff's theorem. Observe that inner while loop is the bipartite matching algorithm and terminates in polynomial time.

On the other hand, if the outer loop does not terminate, then it is guaranteed we have a M-augmenting path in $G^{=}$ in the next iteration. Thus the outer loop runs for at most n iterations.

All in all, the algorithm correctly finds a minimum weight perfect matching in a bipartite graph within polynomial time.

©Felix Zhou

8.1.2 General Graphs

Observe that the issue is the existence of odd cycles. We add additional constraints to the primal LP that all odd vertex sets must have an outgoing edge.

$$\begin{aligned} \min \sum_{e \in E} c_e x_e & \\ x(\delta(v)) = 1 & \qquad \forall v \in V \\ x(\delta(S)) \geq 1 & \qquad \underbrace{\forall S \subseteq V, |S| \equiv 1 \mod 2, |S| \geq 3}_{=:\theta} \\ x \geq 0 & \end{aligned}$$

It dual is given by

$$\begin{aligned} \max \sum_{v \in V} y_v + \sum_{S \in \theta} y_S & \\ y_u + y_v + \sum_{S \in \theta : uv \in \delta(S)} y_S \leq c_{uv} & \qquad \forall uv \in E \\ y_S \geq 0 & \qquad \forall S \in \theta \end{aligned}$$

The complementary slackness conditions are given by

$$\begin{aligned} x(\delta(S)) = 1 \vee y_S = 0 \\ x_{uv} = 0 \vee \bar{c}_{uv} = 0 \end{aligned}$$

The idea is to construct a perfect matching M with

$$M \subseteq E^{=} = \{e \in E : \bar{c}_e = 0\}$$

AND

$$\forall S \in \theta, y_S > 0 \implies |M \cap \delta(S)| = 1.$$

Notice that $E^{=}$ is implicitly dependent on some dual solution.

The updates to our dual solution is more complicated. Let

$$\epsilon_1 := \min\{\bar{c}_{uv} : u \in B(T), v \notin V(T)\}$$

and

$$\epsilon_2 := \min\left\{\frac{1}{2}\bar{c}_{uv} : u, v \in B(T)\right\}.$$

©Felix Zhou

We take $\epsilon := \min(\epsilon_1, \epsilon_2)$.

Then set

$$\bar{y}_u = \begin{cases} \bar{y}_u + \epsilon, & u \in B(T) \\ \bar{y}_u - \epsilon, & u \in A(T) \\ \bar{y}_u, & u \notin V(T) \end{cases}$$

as before.

This brings up the issue of how to shrink a Blossom C in our subroutine to find a perfect matching in $G^=$.

We now need to keep parallel edges. The identified vertex v_C is essentially some odd cycle with vertex set $V(C) = S \in \theta$. Thus we set $y_S = 0$ and change the cost of an edge $v_C v$ as

$$c'_{v_C v} := c_{uv} - y_u$$

where $u \in S, v \notin S$.

Proposition 8.1.2
Let $\bar{y}$ be feasible in the dual, with $\bar{y}_S = 0$ for all $S \in \theta$. Put G', c' be derived from G, c by shrinking the blossom C with

$$E(C) \subseteq E^=.$$

Let M' be a perfect matching of G' and y' feasible for the derived dual, where M', y' satisfy the complementary slackness conditions with $y_{v_C} \geq 0$.
Extend M' to a perfect matching $\hat{M}$ of G and define $\hat{y}$ as

$$\hat{y}_v := \begin{cases} \bar{y}_v & v \in V(C) \\ y'_v & v \in V(G') \setminus v_C \end{cases}$$

and

$$\hat{y}_S := \begin{cases} y'_{v_C} & S = S(v_C) \\ y'_D & S = S(D), D \in \theta(G') \\ 0, & \text{else} \end{cases}$$

Then $\hat{y}$ is feasible for the original dual and $\hat{M}, \hat{y}$ satisfy the complementary slackness conditions.

We do not wish to track all the $y_S > 0$ for $S \in \theta$ so we do not expand all pseudonodes upon augmenting our matching / failing to find a perfect matching in some derived graph. Instead, we proceed with algorithm in the derived graph and update potentials.

While working in the derived graph, we must choose our ϵ even more carefully. It is possible

there is some pseudonode $v_C \in A(T)$ for which we must maintain

$$y_C \geq 0.$$

Thus we have a third parameter

$$\epsilon_3 := \min\{y_{v_C} : v_C \text{ is a pseudonode}\}.$$

We take

$$\epsilon := \min(\epsilon_1, \epsilon_2, \epsilon_3).$$

Now, it is possible if $\epsilon = \epsilon_3$ that $E^=$ stays the same upon updating the dual. Thus in this case, we expand only the v_C's satisfying $y_{v_C} = 0$. Notice we are still able to avoid tracking many $y_S, S \in \theta$, since we only expand if the odd dual variable is set to 0.

By expanding a pseudonode, we mean replacing some v_C with the vertices of C and re-attaching edges to their previous endpoints. We need to readjust edge weights for $uv \in \delta(C)$ by adding back the previously subtracted y_u where $u \in V(C)$. Let ux, vy be the edges of T incident with v_C where $u, v \in V(C)$. Extend M' in the only way possible such that every cycle vertex is saturated. Finally, let P be the even uv-path in C and update T with edges

$$E(T) \cup E(P).$$

©Felix Zhou

```
def Min_Weight_Perfect_Matching(G: Graph, M: Matching, y: FeasibleDual):
  r := M-exposed node
  T := ({r}, {})
  A := {} # odd distanced vertices
  B := {r} # even distanced vertices

  while True:
    # find a perfect matching
    if there is vw in E_= such that v in B(T) and w not in V(T):
      if w is M_prime-exposed:
        use vw to augment M_prime
        if no M_prime-exposed node in G:
          extend to perfect matching M in original graph
          return M
        else:
          r := M_prime exposed node
          T := ({r}, {})
          A.clear()
          B.clear()
      else:
        use vw to extend T
        A.add(w)
        B.add(x)
    elif there is vw in E_= where u, v in B(T):
      use vw to shrink G
      update M_prime
      update T
      update c # costs
    elif there is pseudonode v in A(T) with y(v) = 0:
      expand v
      update M_prime
      update T
      update c # costs
    elif all uv in E with v in B(T) is w in A(T) and no pseudonode in A(T):
      # frustrated tree
      return "no perfect matching"
    else: # update dual solution
      eps1 := min(c(uv) for v in V(B) and w not in V(T))
      eps2 := min(c(uv)/2 for uv in E_= where u, v in B(T))
      eps3 := min(y_v for pseudonode y_v in A(T))
      eps = min(eps1, eps2, eps3)
      # update dual solution
      for v in B(T):
        y(v) += eps
      for v in A(T):
        y(v) -= eps
```

©Felix Zhou

The correctness of this algorithm has been covered by the build-up. We essentially argue that each step preserves that the partial matching is always a subset of $E^=$. To see polynomial time termination, the steps are similar to that of the cardinality matching algorithm. One slightly tricky observation is that unshrinking only happens if there is some pseudonode in $A(T)$, which came as a result of some matching augmentation in a derived graph.

There are at most $O(n)$ matching augmentations. We bound the running time of the algorithm between augmentations. We extend an alternating tree at most $O(m)$ times. Between tree extensions, there may be a series of shrinking/expansion operations. Any shrunken Blossom is not expanded until the next augmentation. Thus at worst, we expand all previously contracted cycles and shrink the graph for a total of $O(n)$ operations. Finally, the dual updates cannot happen in succession thus it is at most the total number of other operations, which does not affect asymptotic complexity. The total runtime is thus

$$O(mn^2).$$

8.2 Maximum Weight Matching

There are direct Blossom algorithm variations which solve this problem. Instead, we show an elegant reduction from the maximum weight perfect matching problem.

8.2.1 Maximum Weight Perfect Matching Reduction

Let $G = (V, E), c : E \to \mathbb{R}$ be an instance of the maximum weight matching problem. Let G' be a copy of G with the exact same edge costs. Put $\bar{G}$ as the graph on vertices $V(G) \cup V(G')$ and edges

$$E(G) \cup E(G') \cup \{vv' : v \in V(G)\}.$$

The new edges vv' all have cost 0.

It is clear that $\bar{G}$ has a perfect matching.

Proposition 8.2.1
Let $\bar{M}$ be a maximum weight perfect matching in $\bar{G}$. Then

$$M := \bar{M} \cap E(G)$$

is a maximum weight matching in G.

Proof
It is clear that M is a matching.

Indeed, let M^* be a maximum weight matching in G. We can copy M^* and take the edges vv' for M^*-exposed vertices to construct a perfect matching in $\bar{G}$ with cost

$$2c(M^*).$$

Since the edges in $\delta(V(G))$ have weight 0, the edges within $E(G), E(G')$ contribute to all the weights. In particular, we must have $c(M) \geq c(\bar{M} \cap E(G'))$ or else $\bar{M}$ was not maximal (take a copy of $M \cap E(G')$ in G instead).

It follows that

$$2c(M) \geq c(\bar{M}) \geq 2c(M^*) \implies c(M) \geq c(M^*)$$

as desired.

8.2.2 Linear Programming Formulation

We conclude this section by noting the existence of a primal dual algorithm originating from the following LP.

$$\begin{aligned} \max & \sum_{e \in E} c_e x_e \\ x(\delta(v)) &\leq 1 && \forall v \in V \\ x(E(S)) &\leq |S| - 1 && \forall S \subseteq V : |S| \text{ odd}, |S| \geq 3 \\ x &\geq 0 \end{aligned}$$

©Felix Zhou

©Felix Zhou

Part IV

T-Joins

©Felix Zhou

©Felix Zhou

Chapter 9

T-Joins

9.1 T-Joins

Definition 9.1.1 (Euler Tour)
Given a connected graph G (potentially with parallel edges), an Euler tour is a closed walk visiting every edge exactly once.

Theorem 9.1.1
A connected graph G has an Euler tour if and only if every vertex has even degree.

Problem 6 (Postman Tour)
A postman tour is a closed walk traversing every edge at least once.
Given $c : E \to \mathbb{R}_{\geq 0}$, find a minimum cost postman tour.

Observe that if G has an Euler tour, it is optimal!

Consider the following logic: Let $x_e \in \mathbb{Z}_{\geq 0}$ for all $e \in E$. Put G^x as the graph obtained by making $1 + x_e$ copies of e.

© Alex Zhou

The idea is to find x such that G^x has an Eulerian tour. This translates to the following:

$$\begin{aligned} &\min \sum_{e \in E} c_e x_e \\ \sum_{e \in \delta(v)} (1 + x_e) &= 0 \mod 2 \qquad v \in V \qquad (*) \\ x_e &\geq 0 \\ x &\in \mathbb{Z}^E \end{aligned}$$

$(*)$ We alternatively write this as

$$x(\delta(v)) \equiv |\delta(v)| \mod 2$$

for all $v \in V$.

Notice that when written this way, it becomes clear that we can restrict $x_e \in \{0, 1\}$, since $x_e - 2$ does not change the parity and can only decrease the cost.

Definition 9.1.2 (Postman Set)
$J \subseteq E$ such that

$$|J \cap \delta(v)| \equiv |\delta(v)| \mod 2$$

for all $v \in V$.

By our remark above, we reduced the problem to finding a postman set.

Definition 9.1.3 (T-Join)
Let $T \subseteq V$ be such that $|T|$ is even.
$J \subseteq E$ is a T-join if it is a postman set with respect to T.

$$|J \cap \delta(v)| \equiv |T \cap \{v\}| \mod 2$$

for all $v \in V$.

In other words, the vertices of (V, J) with odd degree are precisely T.

Problem 7 (Minimum Cost T-Join)
Given $c : E \to \mathbb{R}$ and $G = (V, E)$ with an even vertex subset $T \subseteq V$.
Find a T-join of G minimizing $c(J)$.

© Eric Zhou

9.2 Non-Negative Edge Weights

Observe that the definition does not required connectedness or $c \geq 0$.

What do minimal T-joins look like?

Proposition 9.2.1
Let J' be a T'-join of G. Then J is a T-join of G if and only if

$$J \Delta J'$$

is a $(T \Delta T')$-join of G.

Proof
$(\implies)$ Let $\bar{J} = J \Delta J'$ and fix $v \in V$.

Case I: $v \in T \setminus T'$ Then $|J \cap \delta(v)|$ is odd and $|J' \cap \delta(v)|$ is even. Hence their sum is odd!

But $J \Delta J'$ removes an even number of those since we double count them in the sum and

$$|\bar{J} \cap \delta(v)|$$

is odd.

Case II: $v \in T' \setminus T$ This is identical to Case I.

Case III: $v \in T \cap T'$ Both $|J \cap \delta(v)|, |J' \cap \delta(v)|$ is odd. Hence their symmetric sum is even.

Case IV: $v \in \overline{T \cup T'}$ Both $|J \cap \delta(v)|, |J' \cap \delta(v)|$ is even. Hence their symmetric sum is even.

Thus by definition, $J \Delta J'$ is a $T \Delta T'$-join.

$(\impliedby)$ We utilize the associativity and commutativity of the symmetric difference. Specifically, let us apply the forward direction with

$$J' = J', J = J \Delta J'$$
$$T' = T', T = T \Delta T'$$

Then $J' \Delta J \Delta J' = J$ is a $T' \Delta T \Delta T' = T$-join.

Proposition 9.2.2
J is a minimal T-join if and only if it is the union of edges of $\frac{|T|}{2}$ edge-disjoint paths joining disjoint pairs of vertices in T.

Proof
Let $|T| = 2k$. We argue by induction on k.

The base case of $k = 0$ trivially holds as the only minimal T-join is $\varnothing$. Now suppose $k \geq 1$.

($\implies$) We argue that J contains such an edge set. Then the result follows by minimality.

Let $u \in T$, and K be the connected component of (V, J) containing u. There is necessarily some $v \in T \setminus u$ such that $v \in K$, otherwise the sum of degrees is not even.

Let P be a uv-path in (V, J). Consider $J' = J \setminus E(P)$. Observe that $J'\Delta J = E(P)$ is a $\{u, v\}$-join, as the only odd degree vertices of $(V, E(P))$ are the endpoints u, v. Thus an application of the previous proposition yields that J' is a $T \setminus \{u, v\}$-join.

But $|J'| < 2k$. Hence by induction, J' must be a union of edges of edge-disjoint paths. Thus $J = J' \dot{\cup} E(P)$ is a union of edges of edge-disjoint paths as required.

($\impliedby$) This is clear, since no subset of J can be a T-join.

Proposition 9.2.3
Suppose $c \geq 0$. There exists a minimum cost T-join that is the union of

$$\frac{|T|}{2}$$

edge-disjoint shortest c-paths joining vertices of T in (distinct) pairs.

Proof
Let J be a minimal minimum cost T-join. Put P as a uv-path with $E(P) \subseteq J$ and $u, v \in T$.

Suppose towards a contradiction that P' is a uv-path with

$$c(P') < c(P).$$

Notice that $E(P), E(P')$ are $\{u, v\}$-joins.

It follows that

$$J' := J \Delta E(P) \Delta E(P')$$

is a $T\Delta\{u,v\}\Delta\{u,v\} = T$-join.

By our previous characterization of minimal T-joins,

$$\begin{aligned} c(J') &= c(J \setminus E(P)) + c(E(P')) - 2c((J \setminus E(P)) \cap E(P')) \\ &\leq c(J) - c(E(P)) + c(E(P')) \\ &< c(J) \end{aligned}$$

which is a contradiction.

Let $w : T^2 \to \mathbb{R}_{\geq 0}$ be the shortest path metric in G. Put G' as the complete weighted graph with vertex set T.

Proposition 9.2.4
The minimum cost T-join when $c \geq 0$ can be computed by computing a minimum cost perfect matching in G'.

Proof
Let M be the minimum weight perfect matching in G'. Let

$$\{u_i, v_i\}_{i=1}^{\frac{|T|}{2}}$$

be the edges in M.

Put P_i as the shortest path in G for $1 \leq i \leq \frac{|T|}{2}$. Thus

$$E(P_1)\Delta \ldots \Delta E(P_{\frac{|T|}{2}})$$

is a T-join of cost at most

$$\sum_{i=1}^{\frac{|T|}{2}} w(u_i, v_i).$$

By the previous proposition, any minimum T-join corresponds to a perfect matching in G' and has cost at least that sum. Optimality follows.

9.3 General Edge Weights

We seek to reduce this to the non-negative T-join problem.

Put

$$N := \{e \in E : c_e < 0\}$$
$$T' = \{v \in V : v \text{ has odd degree in } (V, N)\}.$$

Then N is a T'-join by definition.

But J is a T-join if and only if $J \Delta N$ is a $(T \Delta T')$-join. Moreover,

$$\begin{aligned} c(J) &= c(J \setminus N) + c(J \cap N) \\ &= c(J \setminus N) - c(N \setminus J) + c(N \setminus J) + c(J \cap N) \\ &= c(J \setminus N) - c(N \setminus J) + c(N) \\ &= \sum_{e \in J \Delta N} |c_e| + c(N) \end{aligned}$$

Thus to minimize $c(J)$, it suffices to find a minimum cost $(T \Delta T')$-join with respect to costs $|c|$.

9.4 Linear Programming Formulations

A subset $S \subseteq V$ is *T-odd* if $|S \cap T|$ is odd. Moreover, if S is T-odd, we say $\delta(S)$ is a *T-cut*.

Let $S \subseteq V$ be T-odd and J a T-join. If $J \cap \delta(S) = \varnothing$, then the subgraph of (V, J) induced by S has an odd number of odd degree vertices which is impossible. Thus any T-join must cross this cut at least once.

$$\begin{aligned} \min \sum_{e \in E} c_e x_e & \qquad (P) \\ x(\delta(S)) \geq 1 & \qquad \forall S \in \theta := \{T\text{-odd sets } S\} \\ x \geq 0 & \end{aligned}$$

Theorem 9.4.1
Let G be a graph and $T \subseteq V$ with even cardinality. Suppose $c : E \to \mathbb{R}_{\geq 0}$.
Then the minimum cost of a T-join is equal to the optimal value of (P).

©Felix Zhou

The dual of the above LP is

$$\begin{aligned} & \max \sum_{S \in \theta} \alpha_S && (D) \\ \sum_{S \in \theta : e \in \delta(S)} & \alpha_S \leq c_e && \forall e \in E \\ & \alpha \geq 0 \end{aligned}$$

Recall the perfect matching LP is

$$\begin{aligned} \min \sum_{u,v \in E'} & d(u,v) w_{uv} && (P_M) \\ w(\delta(v)) &= 1 && \forall v \in V \\ w(\delta(A)) &\geq 1 && \forall A \in \theta_M := \{A \subseteq V : |A| \geq 3, |A| \text{ odd}\} \\ w &\geq 0 \end{aligned}$$

Its dual is

$$\begin{aligned} & \max \sum_{v \in V} \beta_v + \sum_{A \in \theta_M} \gamma_A && (D_M) \\ \beta_u + \beta_v + \sum_{A \in \theta_M : uv \in \delta(A)} & \gamma_A \leq d(u,v) && \forall u, v \in E' \\ & \gamma_A \geq 0 \end{aligned}$$

Proof
Let J^* be an optimal T-join. We have seen that

$$c(J^*) \geq \zeta^*$$

where ζ^* is the optimal value of (P).

Case I: $T = V$ Let G' be the graph used to solve the minimum cost T-join with costs d used to solve the minimum cost T-join problem.

We have previously shown that $c(J^*)$ is the optimal value to (P_M). But it is also the optimal value to (P_D). We need only show that it is also optimal for (P) and we would be done.

The idea is as follows. Note $E \subseteq E'$ and $d(u,v) \leq c_{uv}$ for all $uv \in E$. From an optimal solution to (D_M), we can build a solution to (D) of the same cost and vice versa.

©Felix Zhou

The only caveat is that some of the β variables can be negative. However, it can be shown that they are in fact non-negative but we omit the proof.

Case II: $T \subseteq V$ Build $\hat{G}$ with a copy $\hat{v}$ of v for all $v \in V \setminus T$. Add $v\hat{v}$ edges with cost 0.

Set $\hat{T} := \hat{V}$ and find a minimum cost $\hat{T}$-join of $\hat{G}$, say $\hat{J}$. We argue there is a minimum cost T-join developed from $\hat{T}$.

Observe that

$$J_0 := \{v\hat{v} : v \in V \setminus T\}$$

is a minimum cost $\hat{V} \setminus T$-join. But recall this implies that

$$J := \hat{J} \Delta J_0$$

is a $\hat{T}\Delta\hat{V} \setminus T = T$-join. Moreover, it has cost

$$c(J) = c(\hat{J} \setminus J_0) + c(J_0 \setminus J) = c(\hat{J}).$$

Vice versa, given a T-join J, we can take

$$J \Delta J_0$$

to be a $\hat{T}$-join with cost

$$c(J \setminus J_0) + c(J_0 \setminus J) = c(J).$$

It follows that J is a minimum cost T-join.

This shows that

$$\mathrm{OPT}(D(\hat{G})) = c(\hat{J}) = c(J) \geq \mathrm{OPT}(P(G)) = \mathrm{OPT}(D(G)).$$

If we construct a dual solution for G from a dual solution to $\hat{G}$ of the same cost. This would show that

$$\mathrm{OPT}(D(G)) \geq \mathrm{OPT}(D(\hat{G}))$$

and the previous inequality would become an equality.

First observe that if S is T-odd, then the set $\hat{S} := (S \cap T) \cup \{v, \hat{v} : v \in S \setminus T\}$ satisfies

$$|\hat{T} \cap \hat{S}| \equiv |T \cap S| \equiv 1 \mod 2.$$

In addition, if there is an edge $v\hat{v} \in \delta(S)$ for any $S \in \hat{\theta}$,

$$0 \leq \alpha_S \leq \sum_{S \in \hat{\theta} : e \in \delta(S)} \alpha_S \leq 0$$

so $\alpha_S = 0$.

It follows that there is a correspondence between non-zero variables for T and $\hat{T}$-odd sets.

$$\{\alpha_S : S \in \theta\} \rightleftharpoons \{\alpha_S : S \in \theta, S \subseteq T \vee S = (S \cap T) \cup \{v, \hat{v} : v \in S \setminus T\}\}.$$

The corresponding dual claim follows.

©Felix Zhou

We conclude this section by remarking the existence of an LP formulation when the costs are not necessarily non-negative but it is more involved.

©Felix Zhou

©Felix Zhou

Part V

Flows & Cuts

©Felix Zhou

©Felix Zhou

Chapter 10

Flows & Cuts

10.1 Maximum Flow

Consider a directed graph $D = (V, A)$ and $x \in \mathbb{R}^A$ as well as $r, s \in V$.

Definition 10.1.1 (Flow)
x is an rs-flow if

$$f_x(v) := x(\delta^-(v)) - x(\delta^+(v)) = 0$$

for all $v \in V \setminus \{r, s\}$. Here $+$ indicates outgoing arcs and $-$ indicates incoming arcs.

Given capacities $\ell \leq \mu \in \mathbb{R}^A$, we say an rs-flow is *feasible* if

$$\ell_a \leq x_a \leq \mu_a$$

for all $a \in A$.

The *value* of a feasible rs-flow is $f_x(s)$.

Problem 8 (Maximum Flow)
Find a rs-flow of maximum value.

We will assume $\ell = 0$ and

$$\delta^-(r) = \delta^+(r) = \varnothing.$$

©Felix Zhou

Proposition 10.1.1
If x is a feasible rs-flow and $\delta^+(R)$ is an rs-cut, then

$$x(\delta^+(R)) - x(\delta^-(R)) = f_x(s).$$

Proof
x is feasible thus $f_x(v) = 0$ for all $v \in V - r - s$.

By computation,

$$\begin{aligned} x(\delta^+(R)) - x(\delta^-(R)) &= \sum_{v \in R} x(\delta^+(v)) - x(\delta^-(v)) \\ &= f_x(s) + \sum_{v \in R - s} x(\delta^+(v)) - x(\delta^-(v)) \\ &= f_x(s). \end{aligned}$$

Corollary 10.1.1.1
Let x be a feasible rs-flow and $\delta^+(R)$ an rs-cut. Then

$$f_x(s) \leq \mu(\delta^+(R)).$$

Proof
By computation,

$$\begin{aligned} f_x(s) &= x(\delta^+(R)) - x(\delta^-(R)) \\ &\leq x(\delta^+(R)) \\ &\leq \mu(\delta^+(R)). \end{aligned}$$

Definition 10.1.2 (Incrementing Path)
Suppose P is a path using some arcs in the "forward" as well as "backward" direction. We say that P is x-incrementing if

$$x_a < \mu_a$$

for all forward arcs and

$$x_a > 0$$

for all backward arcs.

If in addition, P is a rs-path, then we say P is a x-augmenting path.

Proposition 10.1.2
If there is a x-augmenting path, then x is NOT a maximum flow.

10.2 Minimum Cut

Theorem 10.2.1 (Maximum-Flow Minimum-Cut)
If there is a maximum flow x, then

$$\max\{f_x(s) : x \text{ is a feasible } rs\text{-flow}\} = \min\{\mu(\delta^+(R)) : \delta^+(R) \text{ is an } rs\text{-cut}\}.$$

Proof
Let x be a maximum flow. Pick R to be the set of vertices reachable by a rv x-incrementing path.

Remark that $r \in R$ but $s \notin R$ by maximality. Furthermore, for all $vw \in \delta^+(R)$,

$$x_{vw} = \mu_{vw}$$

or else $w \in R$. Similarly, if $vw \in \delta^-(R)$, then $x_{vw} = 0$ or else $v \in R$.

With this, we have

$$f_x(s) = x(\delta^+(R)) = \mu(\delta^+(R)).$$

Notice that we can also show this with strong LP duality.

Theorem 10.2.2
A feasible rs-flow x is maximum if and only if there is no x-augmenting path.

Proof
Forward direction is done. The reverse direction is by remarking the existence of a minimum capacity rs-cut.

Theorem 10.2.3
If $\mu \in \mathbb{Z}_+^A$ and there is a maximum flow, then there is a maximum flow that is integral.

Proof
First remark that there is a cut with finite capacity, namely $\delta(s)$. Thus there is a finite upper bound all flows and a maximum integral flow exists.

Put x as a maximum integral flow and assume it is not a maximum flow. There is a x-augmenting rs-path. But the residual capacities are integral and thus at least 1. This contradicts the assumption that x was a maximum integral flow.

10.3 Ford-Fulkerson Algorithm

The idea is to construct a residual digraph $D_x := (V, A_x)$ with capacity c_x.

Here $vw \in A_x$ if $vw \in A$ and $x_{vw} < \mu_{vw}$. We set $c_x(vw) = \mu_{vw} - x_{vw}$.

In addition, $vw \in A_x$ if $wv \in A$ and $x_{wv} > 0$. We set $c_x(vw) = x_{wv}$.

Observe that there is a x-augmenting rs-path if and only if there is a rs-dipath in D_x. Moreover, we can increment flow along such a path by the minimum of residual capacities along the path.

We can find a rs-dipath in D_x in $O(m)$ time. Unfortunately, if M is the value of the maximum flow, we may need M augmentations to arrive at a maximum flow.

Theorem 10.3.1 (Dinits '70; Edmonds & Karp '72)
If P is chosen to be the shortest rs-dipath in D_x, then there are at most

$$nm$$

augmentations.

Let $d_x(v, w)$ be the length of the shortest vw-dipath in D_x. Put

$$P : v_0, \ldots, v_k$$

as the shortest rs-dipath in D_x and x' be a feasible rs-flow obtained after augmenting x using P.

©Fuxin Zhou

Lemma 10.3.2
For all $v \in V$,

$$d_{x'}(r, v) \geq d_x(r, v)$$

and

$$d_{x'}(v, w) \geq d_x(v, s).$$

Notice the distances are by convention ∞ if no such path exists.

Proof
Suppose towards a contradiction that there is some $v \in V$,

$$d_{x'}(r, v) < d_x(r, v).$$

Choose such a v with minimal $d_{x'}(r, v)$. Notice that $v \neq r$ or else both distances are 0.

Put P' as the rv-dipath in $D_{x'}$ with length $d'_x(r, v)$. Let w be the vertex immediately before v in P'. It follows that $(\star)$

$$d_x(r, v) > d_{x'}(r, v) = d_{x'}(r, w) + 1 \geq d_x(r, w) + 1.$$

If $wv \in A_x$, then

$$d_x(r, v) \leq d_x(r, w) + 1 < d_x(r, v)$$

which is a contradiction.

Thus $wv \notin A_x$ but $wv \in A_{x'}$. Hence wv or vw is an arc in P. It must be that $vw \in E(P)$.

It follows that $v = v_{i-1}, w = v_i$ for some $i \in [k]$ and by $(\star)$,

$$d_x(r, v_{i-1}) \geq d_x(r, v_i) + 1.$$

But this is not possible since we chose P to be the shortest path. Specifically, since P visits v, w,

$$d_x(r, v) = d_x(r, w) - 1.$$

©Felix Zhou

Lemma 10.3.3
If $d_{x'}(r,s) = d_x(r,s)$,

$$\tilde{A}_{x'} \subsetneq \tilde{A}_x.$$

Here

$$\tilde{A}_x := \{uv \in A : \text{either } vw \text{ or } wv \text{ are in a shortest } x\text{-augmenting } rs\text{-path}\}.$$

Proof
Containment Let $k := d_x(r,s)$ and pick $vw \in \tilde{A}_{x'}$.

If vw is in a shortest rs-dipath within $D_{x'}$,

$$d_{x'}(r,v) = i-1, d_{x'}(w,s) = k-i$$

and hence

$$d_{x'}(r,v) + d_{x'}(w,s) = k-1.$$

By the previous claim,

$$d_x(r,v) + d_x(w,s) \leq k-1.$$

If $vw \notin \tilde{A}_x$, then $x_{vw} = x'_{vw}$ as flow did not change implies $vw \in A_x$. But then there is a rs-dipath of length k including vw, which is a contradiction. Thus $vw \in \tilde{A}_x$.

A symmetric argument for the case wv is in a shortest rs-dipath within $D_{x'}$ completes the containment proof.

Strict Containment To see that the containment is strict, let P be a path used to change $x \to x'$. There is some $vw \in A$ such that

$$vw \in P, x'_{vw} = \mu_{vw} \vee wv \in P, x'_{vw} = 0.$$

Suppose $x'_{vw} = \mu_{vw}$. We have

$$d_x(r,v) = i-1, d_x(w,s) = k-i, vw \in \tilde{A}_x, vw \notin D_{x'}.$$

An x'-augmenting path cannot use vw. Hence if $vw \in \tilde{A}_{x'}$, there is an x'-augmenting path using wv.

But

$$\begin{aligned} d_{x'}(r,w) + d_{x'}(v,s) &\geq d_x(r,w) + d_x(v,s) \\ &= (i-1+1) + (k-i+1) \\ &= k+1 \end{aligned}$$

and the length of the shortest rs-dipath in $D_{x'}$ using wv is not a shortest rs-dipath. Thus $vw \notin \tilde{A}_{x'}$ as desired.

The second case is similar.

Proof (Theorem 10.3.1)
The first claim shows that the the algorithm terminates in at most $n-1$ stages, where in each stage, $d_x(r,s)$ remains constant.

The second claim shows that each stage has at most m iterations.

This completes the argument.

10.4 Applications

10.4.1 Bipartite Matching

Consider the reduction from bipartite matching to maximum flow. Let $G = (V, E)$ be bipartite with bipartition $V = A \cup B$. Create new vertices r, s and add edges rA, Bs. Finally, direct all edges from $r \to s, A \to B$.

Put capacities in the original graph as ∞ and capacities in the new arcs as 1. It is easy to see that the arcs with non-zero flow within a maximum flow form a maximum matching.

With some more work, we can show that a minimum cut corresponds to a minimum vertex cover. This provides an alternative proof to König's theorem.

10.4.2 Flow Feasibility

Given $D = (V, A), \mu \in \mathbb{R}_+^A$ and $b \in \mathbb{R}^V$ such that $b(V) = 0$.

©Felix Zhou

Problem 9 (Flow Feasibility)
Does there exist $x \in \mathbb{R}^E$ such that

$$f_x(v) = b_v$$

for all $v \in V$ such that

$$0 \leq x_a \leq \mu_a$$

for all $a \in A$.

Create new vertices r, s. For each v where $b(v) < 0$, add the arc rv with capacity $-b(v)$. For each v where $b(v) > 0$, add the arc vs with capacity $b(v)$.

There is a feasible flow if and only if the maximum flow in the auxiliary graph attains

$$\sum_{v:b(v)>0} b(v).$$

By the max-flow min-cut theorem, this happens if and only if for all $S \subseteq V$

$$\mu(\delta^+(S \cup \{r\})) \geq \sum_{v:b(v)>0} b(v).$$

Observe that

$$\mu(\delta^+(S \cup \{r\})) = \sum_{v \in S: b_v > 0} b_v + \sum_{v \notin S: b_v < 0} (-b_v) + \mu(\delta_D^+(S)).$$

In other words,

$$\sum_{v \notin S: b_v > 0} b_v + \sum_{v \notin S: b_v < 0} b_v \leq \mu(\delta_D^+(S)).$$

Reworded once more,

$$\forall S \subseteq V, b(S) \leq \mu(\delta_D^+(\bar{S})).$$

Intuitively, all this says is that the total demand for all subsets of nodes does not exceed the maximum capacity of incoming arcs.

10.5 Undirected Minimum Cut

Problem 10 (Undirected Minimum Cut)
Given an undirected graph $G = (V, E)$ and $\mu \in \mathbb{R}^E_{\geq 0}$, find $S \subseteq V$ such that $\varnothing \neq S \subsetneq V$ minimizing

$$\mu(\delta(S)).$$

Write

$$\lambda(G)$$

to be the weight of the minimum cut. Put

$$\lambda(G, v, w)$$

to be the weight of the minimum vw-cut.

We can replace each undirected edge with a forward and backward arc of the same capacity, and compute

$$\lambda(G) = \min_{v,w \in V(G)} \lambda(G, v, w).$$

This requires $O(n^2)$ maximum flow computations.

10.5.1 Gomory-Hu Trees

Construction

Pick r, s arbitrarily and compute a minimum rs-cut. Put R, S as its *shores*, ie

$$\lambda(G, v, w) = \mu(\delta(R)), S := V \setminus R.$$

Let us store the results in a tree.

$$T = (\{R, S\}, \{RS\})$$

where RS has edge label $\lambda(G, v, w)$.

In general, suppose our current Gomory-Hu tree T has vertices corresponding only to singletons. We are then done. Otherwise, there is a vertex $A \in V(T)$ corresponding to at least 2 $a_1, a_2 \in A$.

Let G_A be the graph obtained from G by contracting each of the connected components of $T - A$. To be accurate, we take the union of vertices of G represented by each component of $T - A$ and contract them in G.

Compute

$$\lambda(G_A, a_1, a_2) = \mu(\delta(X))$$

and let

$$A_1 := X \cap A, A_2 := \bar{X} \cap A.$$

©Felix Zhou

Split A into A_1, A_2 within T, and label the edge A_1A_2 with

$$\lambda(G_A, a_1, a_2).$$

As for the connected components of $T \setminus A$, the corresponding contracted super-vertex belongs either to X or $\bar{X}$ in G_A. Connect each component to either A_1 or A_2 accordingly.

Retrieving Minimum Cut

To retrieve $\lambda(G, c, d)$, simply take the minimum weight edge in $T_{c,d}$!

Thus with $n-1$ maximum flow computations, we can compute a nice data structure to store all vw-cuts!

Correctness

Proposition 10.5.1
The function $\mu(\delta(A))$ is submodular. That is,

$$\mu(\delta(A)) + \mu(\delta(B)) \geq \mu(\delta(A \cup B)) + \mu(\delta(A \cap B))$$

For all $A, B \subseteq V$.

Proof
Fix $A, B \subseteq V$. Let us carefully decompose edge sets in consideration.

Consider the following disjoint edge-sets.

$$\begin{aligned}
E_{AB} &:= \{ab \in E : a \in A \setminus B, b \in B \setminus A\} \\
E_{AX} &:= \{ax \in E : a \in A \setminus B, x \in V \setminus (A \cup B)\} \\
E_{BX} &:= \{bx \in E : b \in B \setminus A, x \in V \setminus (A \cup B)\} \\
E_{YX} &:= \{yx \in E : y \in A \cap B, x \in V \setminus (A \cup B)\} \\
E_{YB} &:= \{yb \in E : y \in A \cap B, b \in B \setminus A\} \\
E_{YA} &:= \{ya \in E : y \in A \cap B, a \in A \setminus B\}
\end{aligned}$$

They are all pairwise disjoint since $V \setminus (A \cup B), B \setminus A, A \setminus B, A \cap B$ are disjoint sets and we consider the six edge sets grouped by endpoints in the disjoint sets.

We can write

$$\begin{aligned}\delta(A) &= E_{AX} \cup E_{YX} \cup E_{YB} \cup E_{AB}\\ \delta(B) &= E_{BX} \cup E_{YX} \cup E_{YA} \cup E_{AB}\\ \delta(A \cup B) &= E_{AX} \cup E_{BX} \cup E_{YX}\\ \delta(A \cap B) &= E_{YA} \cup E_{YB} \cup E_{YX}\end{aligned}$$

Thus

$$\mu(\delta(A)) + \mu(\delta(B)) - \mu(\delta(A \cup B)) - \mu(\delta(A \cap B)) = 2\mu(E_{AB}) \geq 0$$

as desired.

Lemma 10.5.2
Let $\delta_G(S)$ be a minimum rs-cut and let $v, w \in S$.
Then there exists a minimum vw-cut $\delta_G(T)$ such that $T \subseteq S$.

Proof
Let $\delta(X)$ be a minimum vw-cut. Without loss of generality, by relabeling if necessary, we may assume that

$$s \in S \cap X.$$

Case I: $r \in X$ In this case, we have

$$s \in S \cap X, r \in \bar{S} \cap X.$$

By submodularity, we have

$$\begin{aligned}\mu(\delta(S)) + \mu(\delta(\bar{X})) &\geq \mu(\delta(S \cap \bar{X})) + \mu(\delta(S \cup \bar{X}))\\ &\geq \mu(\delta(S \cap \bar{X})) + \mu(\delta(S)). \qquad \delta(S \cup \bar{X}) \text{ is } rs\text{-cut}\end{aligned}$$

Thus $\delta(S \cap \bar{X})$ is the desired minimum vw-cut with $S \cap \bar{X} \subseteq S$.

Case II: $r \in \bar{X}$ This case is similar, except we have

$$s \in S \cap X, r \in \bar{S} \cap \bar{X}.$$

We can similarly apply submodularity with S, X.

©Felix Zhou

Lemma 10.5.3
Let $G = (V, E), \mu \in R_+^E$ and $s, t \notin B$.
If there exists a minimum st-cut $\delta(X)$ with $X \cap B = \varnothing$, then

$$\lambda(G, s, t) = \lambda(G/B, s, t).$$

Proof
Since $X \cap B = \varnothing$, we also have $X \subseteq V(G/B)$. Thus

$$\begin{aligned}\lambda(G, s, t) &= \mu(\delta_G(X)) \\ &= \mu(\delta_{G/B}(X)) \\ &\geq \lambda(G/B, s, t).\end{aligned}$$

Let $Y \subseteq V(G/B)$ define a minimum st-cut in G/B with $v_B \notin Y$. Hence Y is an st-cut in G. Thus

$$\begin{aligned}\lambda(G/B, s, t) &= \mu(\delta_{G/B}(Y)) \\ &= \mu(\delta_G(Y)) \\ &\geq \lambda(G, s, t).\end{aligned}$$

Suppose T is a GH tree at any point during the construction algorithm. Let f_e be its labels and RS any edge in T. We say RS *has a representative* if

$$\exists r \in R, s \in S : \lambda(G, r, s) = f_{RS}$$

AND the two connected component of $T - RS$ induce the cut of weight $\lambda(G, r, s)$.

Lemma 10.5.4
Let $G = (V, E), \mu \in \mathbb{R}_+^E$ and $p, q, r \in V$.
Then

$$\lambda(G, p, q) \geq \min\{\lambda(G, q, r), \lambda(G, p, r)\}.$$

Equivalently, the two smallest among

$$\lambda(G, p, q), \lambda(G, q, r), \lambda(G, p, r)$$

are equal.

Proof
To see this let $P \subseteq V$ with $p \in P, q \notin P$ induce a minimum pq-cut. If $r \in P$, then $\delta(P)$

is also a rq-cut. Otherwise, if $r \notin P$, then $\delta(P)$ is also a pr-cut.

Thus the inequality follows.

Lemma 10.5.5
Every edge in $E(T)$ has a representative at all times.

Proof
We argue by induction that each edge has a representative.

This is clearly true when initially there are no edges in the tree. It is also true when we build the initial edge.

Now let $x, y \in R$ be a node which represented more than one vertex and X, Y define a cut in G_R with

$$Y := V(G_R) \setminus X, x \in X, y \in Y, \mu(\delta_{G_R}(X)) = \lambda(G_R, x, y).$$

This is the result of our computation for the next step.

It suffices to show that

$$\lambda(G_R, x, y) = \lambda(G, x, y).$$

Indeed, this immediately shows the first part of the definition of being represented. Moreover, the definition of how the algorithm re-distributes edges in $\delta_T(R)$ yields the second part of the definition.

Consider some other component B of $T - R$. Since the unique edge in $\delta_T(B)$ had a representative b_1, b_2, its label was

$$\lambda(G, b_1, b_2) = \mu(\delta_G(B)).$$

Apply our first lemma with $S = \bar{B}$ to see that there is a minimum xy-cut $\delta_G(U)$ with $U \subseteq \bar{B}$. In other words, there is a minimum xy-cut U in G with $U \cap B = \varnothing$.

The second lemma yields that

$$\lambda(G, x, y) = \lambda(G/B, x, y).$$

We can repeat this argument for each component and realize that

$$\lambda(G, x, y) = \lambda(G_R, x, y).$$

So the new edge XY has a representative.

©Felix Zhou

Now, any edge not in $\delta_T(R)$ still has a representative as their endpoints have not changed. It remains to consider some redistributed edge in $\delta_T(R)$ with label

$$\lambda(G, b, g)$$

such that $b \notin R, g \in R$. Without loss of generality, suppose the edge was distributed with new endpoint X. If $g \in X$, then the edge still has a representative. Let us consider the case when $g \in Y$.

We claim in this other case that

$$\lambda(G, b, g) = \lambda(G, b, x).$$

Hence the redistributed edge still has a representative. We have proven that $\lambda(G, x, y) = \mu(\delta_G(S))$ where S is the component of $T - XY$ containing x.

By the first lemma, there is a minimum bx-cut $\delta_G(W)$ satisfying $W \subseteq S$. Specifically, $S \cap Y = \varnothing$. Apply the second lemma and let $G' := G/Y$. We have

$$\lambda(G, b, x) = \lambda(G', b, x).$$

Now, $g \in Y$ implies that any $v_Y b$-cut in G' is a bg-cut in G. Hence

$$\lambda(G', v_Y, b) \geq \lambda(G, b, g).$$

Moreover, any xv_Y-cut in G' is a xy-cut in G. Thus

$$\lambda(G', v_Y, x) \geq \lambda(G, x, y).$$

Finally, the minimum xy-cut in G is also a bg-cut in G. So

$$\lambda(G, x, y) \geq \lambda(G, b, g).$$

Putting everything together,

$$\begin{aligned}
\lambda(G, b, x) &= \lambda(G', b, x) \\
&\geq \min\{\lambda(G', v_Y, b), \lambda(G', v_Y, x)\} \\
&\geq \min\{\lambda(G, b, g), \lambda(G, x, y)\} \\
&= \lambda(G, b, g).
\end{aligned}$$

Note that since $b \in B, x \notin B$,

$$\mu(\delta_G(B)) = \lambda(G, b, g) \geq \lambda(G, b, x).$$

Thus we have equality.

©Felix Zhou

Theorem 10.5.6
Let T be the final Gomory-Hu tree. Then for all $r, s \in V$, $\lambda(G, r, s)$ is equal to the smallest label of an edge in $T_{r,s}$.
Also, if e^* is such an edge, then

$$\lambda(G, r, s) = \mu(\delta(H)),$$

where H is one of the two components of $T - e^*$.

Proof
Let

$$T_{r,s} = v_0, e_1, v_1, e_2, \ldots, e_k, v_k.$$

and write f_e to be the labels of edges in T.

By the previous lemma,

$$\lambda(G, v_{i-1}, v_i) = f_{e_i}$$

for all $i \in [k]$.

We claim that

$$\lambda(G, r, s) \geq \min_{i \in [k]} \lambda(G, v_{i-1}, v_i).$$

This shows both the first statement because the minimum $v_{i-1}v_i$-cut induced by components of $T - e_i$ are also rs-cuts. as well as the second statement since each edge e_i has a representative in $\{v_{i-1}\}, \{v_i\}$.

We by induction on k. If $k = 1$, the result is trivial. Fix $k \geq 2$. By induction,

$$\lambda(G, r, v_{k-1}) \geq \min_{i \in [k-1]} \lambda(G, v_{i-1}, v_i).$$

Furthermore, by the third lemma,

$$\lambda(G, r, s) \geq \min\{\lambda(G, r, v_{k-1}), \lambda(G, v_{k-1}, s)\} \geq \min_{i \in [k]} \lambda(G, v_{i-1}, v_i)$$

as required.

By induction, we conclude the proof.

©Felix Zhou

©Felix Zhou

Part VI

Other Topics

©Felix Zhou

©Felix Zhou

Chapter 11

Randomized Algorithms

The idea is to use randomization in a clever way to achieve better or more practical performance, as well as simpler algorithms.

11.1 Global Minimum Cut

Karger's algorithms solves this problem.

1) While G has more than 2 vertices:
   (a) Choose e with probability $\frac{\mu_e}{\mu(E)}$
   (b) $G := G/e$
2) Return cut separating 2 vertices

We remark here that $\mu(E)$ changes as the number of edges within G decreases.

It is clear this algorithm terminates in polynomial time.

Theorem 11.1.1
Karger's algorithm returns a minimum cut with probability at least

$$\frac{2}{n(n-1)}.$$

©Felix Zhou

Proof
Let $A \subseteq E$ be a minimum cut. The algorithm returns A if none of its edges are contracted.

Suppose i edges have been contracted so far. Let G^i be the graph with $n-i$ vertices left. Since A is a minimum cut,

$$\mu(A) \leq \mu(\delta_{G^i}(v))$$

for all $v \in V(G^i)$.

It follows that

$$\mu(A) \leq \sum_{v \in V(G^i)} \frac{\mu(\delta_{G^i}(v))}{n-i} = \frac{2\mu(E(G^i))}{n-i}.$$

The probability of NOT picking an edge in A is

$$1 - \frac{\mu(A)}{\mu(E(G^i))} \geq 1 - \frac{2}{n-i} = \frac{n-i-2}{n-i}.$$

By a telescoping sum, the probability of success is at least

$$\begin{aligned}\prod_{i=0}^{n-3} \frac{n-i-2}{n-i} &= \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-2} \cdot \cdots \cdot \frac{1}{3} \\ &= \frac{2}{n(n-1)}\end{aligned}$$

as required.

11.1.1 Boosting

Although a single run of Karger's algorithm has a high change of failure. Running the algorithm q independent times yields a failure rate of at most

$$\begin{aligned}\left(1 - \frac{2}{n(n-1)}\right)^q &\leq \left(1 - \frac{2}{n^2}\right)^q \\ &\leq e^{-\frac{2q}{n^2}} && 1 + x \leq e^x \\ &\leq e^{-k} && q \in \Theta(kn^2)\end{aligned}$$

©Felix Zhou

Chapter 12

Approximation Algorithms

There are many $\mathcal{NP}$-hard problems. We want to find a "pretty" good solution in polynomial time.

12.1 K-cuts

Problem 11 (K-cuts)
Given $G = (V, E)$ and $\mu \in \mathbb{R}_+^E, k \in \mathbb{Z}_+$, find $A \subseteq E$ minimizing $\mu(A)$ such that $(V, E \setminus A)$ has at least k components.

Consider the following approximation algorithm leveraging Gomory-Hu trees.

1) Compute a Gomory-Hu tree T with edge labels f_e
2) Order edges of T such that $f_{e_1} \leq f_{e_2} \leq \cdots \leq f_{e_{n-1}}$
3) Remove $e_1, \ldots, e_{k-1}$, giving k connected components of T: $V_1, \ldots, V_k$

Let $A^* \subseteq E$ be an optimal solution with connected components

$$V_1^*, \ldots V_k^*.$$

We may assume without loss of generality that there are precisely k components.

Theorem 12.1.1
The proposed algorithm yields a $2 - \frac{2}{k}$-approximation.

©Rex Zhou

Proof
Contract V_i^* into v_i^* in T. Now drop edges arbitrarily to get a tree T'.

Each edge $v_i^* v_j^*$ in T' corresponds to an edge ab in T with $a \in V_i^*, b \in V_j^*$. hence

$$f_{v_i^* v_j^*} = f_{ab} \leq \mu(\delta_G(V_j^*)).$$

Consider T' as a tree rooted at some v_r^* maximizing $\mu(\delta_G(V_r^*))$. Apply the inequality above to the child endpoint of every edge. Every node except the root is counted once. Hence

$$\begin{aligned}
\sum_{e \in E(T')} f_e &\leq \sum_{j=1}^{k} \mu(\delta_G(V_j^*)) - \mu(\delta_G(V_r^*)) \\
&\leq \sum_{j=1}^{k} \mu(\delta_G(V_J^*)) - \sum_{j=1}^{k} \frac{\mu(\delta_G(V_j^*))}{k} \\
&= \left(1 - \frac{1}{k}\right) 2\mu(A^*).
\end{aligned}$$

But we picked the cheapest $k-1$ edges in T which is a lower bound for $\sum_{e \in E(T')} f_e$. This concludes the proof.

12.2 Set Cover

Problem 12 (Set Cover)
Given elements $G = \{1, \ldots, m\}$ and a collection of subsets $\{S_1, \ldots, S_n\}$ of G, each with a cost $c_j \geq 0$. find $\Delta \subseteq \{1, \ldots, n\}$ minimizing $c(\Delta)$ such that

$$\bigcup_{j \in \Delta} S_j = G.$$

©Felix Zhou

Consider the LP formulation

$$\begin{aligned} \min \sum_{j=1}^{n} c_j x_j & \qquad (P) \\ \sum_{j:e\in S_j} x_j \geq 1 & \qquad \forall i \in [m] \\ x \geq 0 & \end{aligned}$$

and notice that any optimal solution satisfies $x^* \in [0,1]^n$.

12.2.1 Randomized Rounding

Define the following algorithm:

1) For each $j \in [n]$, select S_j with probability x_j^* independently.
2) Return Δ as the selected S_j's

Lemma 12.2.1
The probability i is covered by Δ is at least

$$1 - \frac{1}{e}.$$

Proof
By independence, this probability that i is uncovered is at most

$$\begin{aligned} \prod_{j:i\in S_j} (1 - x_j^*) &\leq \prod_{j:i\in S_j} e^{-x_j^*} \\ &= e^{-\sum_{j:i\in S_j} x_j^*} \\ &\leq e^{-1} \end{aligned}$$

Let us run the algorithm $2\ln n$ times and output the UNION of all sets that were picked at any given iteration. The probability that any i is not covered is at most

$$e^{-2\ln n} = \frac{1}{n^2}.$$

Thus by a union bound, the probability that there is an uncovered i is at most $\frac{1}{n}$.

©Feiran Zhou

The expected cost of a single run of the algorithm is

$$\begin{aligned} E[c(\Delta)] &= \sum_{j=1}^{n} c_j P(S_j \in \Delta) \\ &= \sum_{j=1}^{n} c_j x_j^* \\ &= \mathrm{OPT}(P) \\ &\leq \mathrm{OPT}(\text{Set Cover}). \end{aligned}$$

Hence the final solution has expected cost at most $2 \ln n \cdot \mathrm{OPT}$!

12.2.2 Primal-Dual Approach

Consider the dual of (P)

$$\begin{aligned} \max \sum_{i=1}^{m} y_i & \qquad (D) \\ \sum_{i \in S_j} y_i \leq c_j & \qquad \forall j \in [n] \\ y \geq 0 & \end{aligned}$$

The idea is to find integral x^* and any y^* feasible to (P), (D) satisfying the relaxed CS conditions

(i) $\forall j, x_j^* > 0 \implies \sum_{i \in S_j} y_i^* \geq c_j$
(ii) $\forall i, y_i^* > 0 \implies \sum_{j : i \in S_j} x_j^* \leq f$

Here $1 < f$.

©Felix Zhou

Then

$$\begin{aligned}\sum_{j=1}^{n} c_j x_j^* &= \sum_{j=1}^{n} x_j^* \left(\sum_{i \in S_j} y_i^* \right) \\ &= \sum_{i=1}^{m} y_i^* \left(\sum_{j : i \in S_j} x_j^* \right) \\ &\leq f \sum_{i=1}^{m} y_i^* \cdot 1 \\ &= f \cdot \mathrm{OPT}\end{aligned}$$

Consider the following algorithm:

1) Initialize $x^* := 0, y^* = 0$
2) While there is some i such that $\sum_{j : i \in S_j} x_j^* < 1$
   a) Pick an i and raise y_i^* until $\sum_{i \in S_j} y_i^* = c_j$ for some j
   b) Let $x_j^* = 1$ for all j where $\sum_{i \in S_j} y_i^* = c_j$
3) Output x^*

The algorithm always progresses in each iteration since the only way an item is not covered is if no previously chosen set covers it. Hence there is some unchosen set which covers it and the corresponding set variable can be raised.

Theorem 12.2.2
The proposed algorithm is an f-approximation algorithm where f is the maximum number of sets in which an element appears.

©Felix Zhou

©Felix Zhou

Chapter 13

Integer Programming

13.1 Minimum Bounded Degree Spanning Tree

Problem 13 (Minimum Bounded Degree Spanning Tree)
Given $G = (V, E)$, costs $c \in \mathbb{R}^E$, and $k \in \mathbb{Z}_+$ for $k \geq 2$, find a spanning tree T of minimum cost where

$$\delta_T(v) \leq k$$

for all $v \in V$.

This problem is $\mathcal{NP}$-hard as it is equivalent to the Hamiltonian Path problem for $k = 2$.

Fix $k \geq 2$ and let $\text{OPT}(k)$ be the value of a k-spanning tree. We wish to produce a spanning tree T with

$$\delta_T(v) \leq k_v + 2$$

for all $v \in V$ and

$$c(T) \leq \text{OPT}(k).$$

Consider the bounded degree spanning tree polytope

$$\begin{aligned} \min \sum_{e \in E} c_e x_e & \qquad (LP) \\ x(E) &= n - 1 \\ x(E(S)) &\leq |S| - 1 \qquad \forall S \subsetneq V \\ x(\delta(v)) &\leq k \qquad \forall v \in V \\ x &\geq 0 \end{aligned}$$

©Feizhou

Although this LP is exponential in size, we can actually an optimal solution x^* in polynomial time through integer programming techniques.

Let $E^* := \{e \in E : x_e^* > 0\}$ be the support of an optimal solution x^*. We may assume that x^* is an extreme point which yields the result that

$$|E^*(U)| \leq 2|U| - 1$$

for all $U \subseteq V$.

By the homework, there is an orientation of the edges E^* such that the indegree of every vertex is at most 2.

With matroid intersection, we can find the desired T. Put A^* as an orientation of E^* such that $D^* = (V, A^*)$ satisfies

$$|\delta_{D^*}^-(v)| \leq 2$$

for all $v \in V$.

For any $B \subseteq A^*$, let $E(B)$ be the corresponding set of edges in E^*. Define $M_1 := (E^*, \mathcal{I}^*)$ where

$$\mathcal{I}^* := \{F \subseteq E^* : |F \cap E(\delta_{D^*}^+(v))| \leq k, \forall v \in V\}.$$

Lemma 13.1.1
If $F \subseteq \mathcal{I}^*$, then

$$|F \cap \delta_G(v)| \leq k + 2.$$

Proof
The degree of each vertex is the sum of out and in degrees, which is at most $k + 2$.

Let M_2 be the graphic matroid of G^*. Let $\bar{c} := -c + M > 0$ and remark that T is a minimum cost spanning tree for c if and only if T is a maximum cost spanning tree for $\bar{c}$.

Thus compute the maximum weight independent set T in $\mathcal{I}_1 \cap \mathcal{I}_2$ and return T.

Lemma 13.1.2
T is a spanning tree.

Proof
The weights are strictly positive.

It only remains to argue that

$$c(T) \leq \mathrm{OPT}(k).$$

By results in matroid intersection and polyhedral theory, x^T is an optimal solution to

$$\begin{aligned} \min c^T x & \qquad (LP') \\ x(E(S)) \leq |S| - 1 & \qquad \forall S \subseteq V \\ x(E) = n - 1 & \\ x(E(\delta^+_{D^*}(v))) \leq k & \qquad \forall v \in V \\ x \geq 0 & \end{aligned}$$

But the optimal x^* we found before is feasible for (LP') and hence

$$c(T) \leq c^T x^* \leq \mathrm{OPT}(k)$$

as desired.

©Felix Zhou