CS 586/IE 519: Combinatorial Optimization[1]

Chandra Chekuri[2]

May 6, 2022

[1]These notes are based on scribed lectures from the Spring 2010 edition of the course. CC wrote notes on a tablet which were transcribed into Latex by students in the course. Individual chapters will acknowledge the relevant student contributions. Substantial revisions and corrections have been made during this semester and some new material has been added. The original notes are mostly based on two sources: Schrijver's tome and Michel Goemans's lecutre notes from MIT.

Please report any error and omissions. I will acknowledge and make corrections.

[2]Dept. of Computer Science, University of Illinois, Urbana, IL 61820. Email: chekuri@illinois.edu. The author acknowledges support from several NSF CCF grants over the years. Work on this version of the notes during Spring 2022 is supported by NSF grants CCF-1910149 and CCF-1907937.

Contents

Chapter 1

Introduction and Motivation[1]

Roughly speaking, an optimization problem has the following outline: given an instance of the problem, find the "best" *solution* among all solutions to the given instance. We will be mostly interested in discrete optimization problems where the instances and the solution set for each instance is from a discrete set. This is in contrast to continuous optimization where the input instance and the solution set for an instance can come from a continuous domain.

We assume some familiarity with the computational complexity classes **P**, **NP**, **coNP**. In this class we are mainly interested in polynomial time solvable "combinatorial" optimization problems. Combinatorial optimization problems are a subset of discrete optimization problems although there is no precise formal definition for them. Typically, in combinatorial optimization, we work with structures over a finite ground set E. Solutions correspond to subsets of of 2^E (the power set of E) and one seeks to find a maximum or minimum weight solution for some given weights on E. For example, in the minimum spanning tree problem, the ground set E is the set of edges of a graph $G = (V, E)$ and the solutions are subsets of E that correspond to spanning trees in G.

We will be interested in **NP** optimization problems – **NPO** problems for short. Formally, a problem Q is a subset of Σ^*, where Σ is a finite alphabet such as binary. Each string I in Q is an *instance* of Q. For a string x we use $|x|$ to denote its length. We say that Q is an **NPO** problem if the following hold:

1. for each $x \in \Sigma^*$ there is a polynomial time algorithm that can check if $x \in Q$, i.e., if x is a valid instance of Q
2. for each instance I there is a set $sol(I) \subset \Sigma^*$ such that
   (a) $\forall s \in sol(I), |s| = poly(|I|)$

[1]Based on scribed notes by Alina Ene from 2010.

(b) there exists a poly-time algorithm that on input I, s correctly outputs whether $s \in sol(I)$ or not

3. there is a function $val : \Sigma^* \times \Sigma^* \rightarrow \mathbb{Z}$ s.t. $val(I, s)$ assigns an integer to each instance I and $s \in sol(I)$ and moreover val can be computed by a poly-time algorithm.

Given an **NPO** problem, we say it is a minimization problem if the goal is to compute, given $I \in Q$, $\arg\min_{s \in sol(I)} val(I, s)$. It is a maximization problem if the goal is to compute $\arg\max_{s \in sol(I)} val(I, s)$. A natural *decision* problem associated with an **NPO** problem (say, maximization) is: given I and integer k, is there $s \in sol(I)$ s.t. $val(s, I) \geq k$[2].

Many problems we encounter are **NPO** problems. Some of them can be solved in polynomial time. It is widely believed and conjectured that **P** $\neq$ **NP**, which would mean that there are **NPO** problems (in particular, those whose decision versions are **NP**-complete) that do not have polynomial time algorithms. Assuming **P** $\neq$ **NP**, some important and useful questions are: What problems are in **P**? What characterizes problems in **P**?

These are not easy questions. It has become clear that computation and algorithms are difficult to understand and we are far from being able to characterize the complexity of problems. However, in limited settings we seek to study broad classes of problems and understand some unifying themes. One particular class of problems where this has been possible is the class of constraint satisfaction problems in the Boolean domain. A result of Schaefer completely characterizes which problems are in **P** and which are **NP**-complete. In fact, there is a nice dichotomy. However, the non-Boolean domain is much more complicated even in this limited setting. We refer the interested reader to [7, 17] for surveys on this topic.

In the field of combinatorial optimization some unified and elegant treatment can be given via polyhedra ands the ellipsoid method. The purpose of this course is to expose you to some of these ideas as well as outline some general problems that are known to be solvable in polynomial time. The three ingredients in our study are

1. polynomial time algorithms (which we will refer to as efficient)

2. structural results, especially via min-max characterizations of optimal solutions

3. polyhedral combinatorics

[2]We used integers in our description of **NPO** problems for convenience. We can equivalently define using rationals.

We will illustrate these ingredients as we go along with examples and general results. We refer the reader to Schrijver's tome on this topic [57] as well as other references provided on the course website. In this introductory lecture, we discuss some known examples to highlight the view point we will take on the connections between these three topics. The discussion will be somewhat informal.

1.1 Network Flow

Let $D = (V, A)$ be a directed graph and let $s, t \in V$ be two distinct nodes. Let $c : A \to \mathbb{R}^+$ be a non-negative arc capacity function. An s-t flow is a function $f : A \to \mathbb{R}^+$ that satisfies the following properties.

1. flow conservation: $\sum_{a \in \delta^-(v)} f(a) = \sum_{a \in \delta^+(v)} f(a)$ for all $v \in V - \{s, t\}$
2. capacity constraint: $0 \leq f(a) \leq c(a)$ for all $a \in A$

The value of a given s-t flow f is defined as

$$val(f) = \sum_{a \in \delta^+(s)} f(a) - \sum_{a \in \delta^-(s)} f(a) = \sum_{a \in \delta^-(t)} f(a) - \sum_{a \in \delta^+(t)} f(a)$$

The optimization problem is to find an s-t flow of maximum value.
An s-t cut is a partition of V into (A, B) such that $s \in A$, $t \in B$, and the capacity of this cut is

$$c(A, B) = \sum_{a \in \delta^+(A)} c(a)$$

Clearly, for any s-t flow f and any s-t cut (A, B)

$$val(f) \leq c(A, B) \Rightarrow \max s\text{-}t \text{ flow} \leq \min s\text{-}t \text{ cut capacity}$$

The well-known maxflow-mincut theorem of Menger and Ford-Fulkerson states that

Theorem 1.1. *In any directed graph, the maximum s-t flow value is equal to the minimum s-t cut capacity.*

Ford and Fulkerson proved the above theorem algorithmically. Although their augmenting path algorithm is not a polynomial-time algorithm for general integer capacities, it can be made to run in polynomial time with very slight modifications (for example, the Edmonds-Karp modification to use the shortest augmenting path). Moreover, the algorithm shows that there is an *integer* valued maximum flow whenever the capacities are integer valued. Thus we have

Theorem 1.2. *In any directed graph, the max s-t flow value is equal to the min s-t cut capacity. Moreover, if c is integer valued then there is an integer valued maximum flow.*

This is an example of a polynomial time (good) algorithm revealing structural properties of the problem in terms of a min-max result and integrality of flows. Conversely, suppose we knew the maxflow-mincut theorem but did not know any algorithmic result. Could we gain some insight? We claim that the answer is yes. Consider the decision problem: given G, s, t, is the s-t max flow value at least some given number k? It is easy to see that this problem is in **NP** since one can give a feasible flow f of value at least k as an **NP** certificate[3]. In addition, using the maxflow-mincut theorem we can also see that it is in **coNP**. To show that the flow value is smaller than k, all we need to exhibit is a cut of capacity smaller than k which is guaranteed by the theorem. Therefore the min-max result shows that the problem is in **NP** $\cap$ **coNP**. Most "natural" decision problems in **NP** $\cap$ **coNP** have eventually been shown to have polynomial time algorithms (there are a few well-known exceptions). Moreover, a problem in **NP** $\cap$ **coNP** being **NP**-complete or **coNP**-complete would imply that **NP** = **coNP** which is believed to be unlikely. Thus a min-max result implies that the decision version is in **NP** $\cap$ **coNP** which is strong evidence for the existence of a poly-time algorithm. That does not imply that such an algorithm will come by easily. Examples include matchings in general graphs and linear programming.

Finally, let us consider network flow as a special case of a linear programming problem. We can write it as

$$\begin{aligned}
& \max && \sum_{a \in \delta^+(s)} f(a) - \sum_{a \in \delta^-(s)} f(a) \\
& s.t. && \\
\sum_{a \in \delta^+(v)} f(a) - \sum_{a \in \delta^-(v)} f(a) & = 0 && \forall v \in V, v \neq s, t \\
f(a) & \leq c(a) && \forall a \in A \\
f(a) & \geq 0 && \forall a \in A
\end{aligned}$$

The polynomial time solvability of LP implies a poly-time algorithm for network flow. However, one can say much more. It is known that the matrix defining the above system of inequalities is *totally unimodular* (TUM). From this, it follows that the vertices of the flow-polytope defined by the LP is integral whenever the capacities are integral! In addition, one can show that the dual also has integer vertices since the objective function has integer coefficients. In fact, one can derive the maxflow-mincut theorem from these facts about the polyhedra in

[3]We are ignoring a technicality here that the flow specification be of polynomial size in the input.

question. The polyhedral results also imply other useful facts. For example, we could add lower bounds on the flow

$$\ell(a) \leq f(a) \leq c(a)$$

and the polytope remains an integer polytope. These generlizations allow one to derive various theorems and algorithms about minimum cost flow, circulations, transshipments, and bipartite matchings.

1.2 Bipartite Matchings

Many of you have seen bipartite matchings and various results about them reduced to results on maximum-flow. We can also treat them independently[4]. Let $G = (X \cup Y, E)$ be a bipartite graph with bipartition given by X, Y. Recall that $M \subseteq E$ is a *matching* in a graph if no vertex in G is incident to more than one edge from M.

A vertex cover in $G = (V, E)$ is a subset of vertices $S \subseteq V$ such that for each edge $uv \in E$, u or v is in S. In other words, S covers all edges. We use $\nu(G)$ to indicate the cardinality of a maximum matching in G and $\tau(G)$ for the cardinality of a minimum vertex cover in G.

Claim 1.2.1. *For every G, $\nu(G) \leq \tau(G)$.*

In bipartite graphs, one has the following theorem.

Theorem 1.3 (König's Theorem)**.** *If G is bipartite then $\nu(G) = \tau(G)$.*

The above proves that the max matching and min vertex problems (decision versions) in bipartite graphs are both in **NP** $\cap$ **coNP**. (Note that the vertex cover problem is **NP**-hard in general graphs.) We therefore expect a polynomial time algorithm for $\nu(G)$ and $\tau(G)$ in bipartite graphs. As you may know, one can reduce matching in bipartite graphs to maxflow and König's theorem follows from the maxflow-mincut theorem. One can also obtain a polynomial time augmenting path algorithm for matching in bipartite graphs (implicitly a maxflow algorithm) that proves König's theorem algorithmically.

We now look at the polyhedral aspect. We can write a simple LP relaxation for the maximum matching in a graph G.

$$\begin{aligned} \max \quad & \sum_{e \in E} x(e) \\ \sum_{e \in \delta(u)} x(e) \quad & \leq \quad 1 \qquad \forall u \in V \\ x(e) \quad & \geq \quad 0 \qquad \forall e \in E \end{aligned}$$

[4] A less well-known fact is that one can reduce s-t flow to bipartite b-matchings.

For bipartite graphs the above LP and its dual have integral solutions since the constraint matrix is TUM. One can derive König's theorem and a polynomial time algorithm for maximum (weight) matchings via polyhedral results.

1.3 General Graph Matchings

The constraint matrix of the basic LP for matchings given above is not integral for general graphs, as the following simple graph shows. Let $G = K_3$ be the complete graph on 3 vertices. The solution $x(e) = 1/2$ for each of the 3 edges in G is an optimum solution to the LP of value $3/2$ while the maximum matching in G has size 1.

The algorithmic study of general graph matchings and the polyhedral theory that was developed by Jack Edmonds in the 1960's, and his many foundational results are the start of the field of polyhedral combinatorics. Prior to the work of Edmonds, there was a min-max result for $\nu(G)$ due to Berge which is based on Tutte's necessary and sufficient condition for the existence of a perfect matching. To explain this, for a set $U \subseteq V$, let $o(G-U)$ be the number of odd cardinality components in the graph obtained from G by removing the vertices in U.

Tutte-Berge formula

$$\nu(G) = \min_{U \subseteq V} \frac{1}{2}(|V| + |U| - o(G-U))$$

We will prove the easy direction for now, i.e.,

$$\nu(G) \leq \frac{1}{2}(|V| + |U| - o(G-U)) \qquad \forall U \subseteq V$$

To see this, the number of unmatched vertices is at least $o(G-U) - |U|$, since each odd component in $G-U$ needs a vertex in U. Hence

$$\nu(G) \leq \frac{|V|}{2} - \frac{o(G-U) - |U|}{2} \leq \frac{1}{2}(|V| + |U| - o(G-U))$$

Corollary 1.4. *(Tutte's 1-factor theorem) G has a perfect matching iff $\forall U \subseteq V$ $o(G-U) \leq |U|$.*

The formula shows that the decision version of matching is in **NP** $\cap$ **coNP**. Edmonds gave the first polynomial time algorithms for finding a maximum cardinality matching and also more difficult maximum weight matching problem. As a consequence of his algorithmic work, Edmonds showed the following results on matching polytopes.

Consider the following polytope:

$$\begin{aligned}
\sum_{e \in \delta(v)} x(e) &\leq 1 \qquad \forall v \in V \\
\sum_{e \in E[U]} x(e) &\leq \lfloor \frac{|U|}{2} \rfloor \qquad \forall U, |U| \text{ odd} \\
0 &\leq x(e) \qquad \forall e \in E
\end{aligned}$$

Edmonds showed that the vertices of the above polytope are exactly the characteristic vectors of the matchings of G[5]. Observe that the polytope has an exponential number of constraints. One can ask whether this description of the matching polytope is useful. Clearly if one takes the convex hull of the characteristic verctors of the matchings of G, one obtains a polytope in $\mathbb{R}^E$; one can do this for any combinatorial optimization problem. In general such a polytope may require an exponential number of inequalities to describe it. Edmonds argued that the matching polytope is different since

1. the inequalities are described implicitly in an uniform way
2. his algorithm gave a way to optimize over this polytope

At that time, no poly-time algorithm was known for solving LPs, although LP was known to be in **NP** $\cap$ **coNP**. In 1978, Khachiyan used the ellipsoid algorithm to show that linear programming is in **P**. Very soon, Padberg-Rao, Karp-Papadimitriou, and Grötschel-Lóvasz-Schrijver independently realized that the ellipsoid algorithm has some very important features, and can be used to show the polynomial-time equivalence of optimization and separation for polyhedra; see [30] for a book-length treatment.

Separation Problem for Polyhedron Q
Given n (the dimension of Q), and an upper bound L on the size of the numbers defining the inequalities of Q, and a rational vector $x_0 \in \mathbb{R}^n$, output correctly that $x \in Q$ or a separating hyperplane $ax = b$ such that $ax \leq b \; \forall x \in Q$ and $ax_0 > b$.

Optimization Problem for Polyhedron Q
Given n (the dimension of Q), and an upper bound L on the size of the numbers defining the inequalities of Q, and a rational vector $c \in \mathbb{R}^n$, output correctly one of the following: (i) Q is empty (ii) $\max_{x \in Q} cx$ has no finite solution (iii) a vector

[5]For a subset of edges $A \subseteq E$ the characteristic vector of A is an $|E|$-dimensional vector χ_A where $\chi_A(i) = 1$ if edge e_i is in A and is 0 otherwise; here we assume that the edges in E are numbered from 1 to m. Another typical notation for characteristic vectors is $\mathbf{1}_A$.

x^* such that $cx^* = \sup_{x \in Q} cx$.

Theorem 1.5. *(Grötschel-Lóvasz-Schrijver) There is a polynomial time algorithm for the separation problem over Q iff there is a polynomial time algorithm for the optimization problem over Q.*

The above consequence of the ellipsoid method had/has a substantial theoretical impact in combinatorial optimization. In effect, it shows that an algorithm for a combinatorial optimization problem implies an understanding of the polytope associated with the underlying problem, and vice-versa. For example, the weighted matching algorithm of Edmonds implies that one can separate over the matching polytope. Interestingly, it took until 1982 for Padberg and Rao to find an *explicit* separation algorithm for the matching polytope although one is implied by the preceding theorem.

Bibliographic Notes: Combinatorial Optiization is a mature area with several classical and beatiful results. The book of Schrijver [57] is a comprehensive treatment. The theory of NP-Completeness showed that many natural problems are NP-Complete and this came as perhaps a surprise (and a bit of a disappointment) for those, including Edmonds, who had hoped for an expanding field of **P** time algorithms.

Chapter 2

Matchings in Non-Bipartite Graphs[1]

We discuss matching in general undirected graphs. Given a graph G, $\nu(G)$ denotes the size of the largest matching in G. We follow [57] (Chapter 24). (Based on scribed notes of Matthey Yancy in 2010).

2.1 Tutte-Berge Formula for $\nu(G)$

Tutte (1947) proved the following basic result on perfect matchings.

Theorem 2.1 (Tutte)**.** *A graph $G = (V, E)$ has a prefect matching iff $G - U$ has at most $|U|$ odd components for each $U \subseteq V$.*

Berge (1958) generalized Tutte's theorem to obtain a min-max formula for $\nu(G)$ which is now called the Tutte-Berge formula.

Theorem 2.2 (Tutte-Berge Formula)**.** *For any graph $G = (V, E)$,*

$$\nu(G) = \frac{|V|}{2} - \max_{U \subseteq V} \frac{o(G-U) - |U|}{2}$$

where $o(G - U)$ is the number of components of $G - U$ with an odd number of vertices.

Proof. We have already seen the easy direction that for any U, $\nu(G) \leq \frac{|V|}{2} - \frac{o(G-U)-|U|}{2}$ by noticing that $o(G - U) - |U|$ is the number of nodes from the odd components in $G - U$ that must remain unmatched (one for each odd component).

[1]Based on scribed notes of Matthew Yancey from 2010.

Therefore, it is sufficient to show that $\nu(G) = \frac{|V|}{2} - \max_{U \subseteq V} \frac{o(G-U)-|U|}{2}$. Any reference to left-hand side (LHS) or right-hand side (RHS) will be in reference to this inequality. Proof via induction on $|V|$. Base case of $|V| = 0$ is trivial. We can also assume that G is connected otherwise we can apply induction to each connected component and the theorem easily follows.

Case 1: There exists $v \in V$ such that v is in every maximum matching. Let $G' = (V', E') = G - v$, then $\nu(G') = \nu(G) - 1$ and by induction, there is $U' \subseteq V'$ such that the RHS of the formula is equal to $\nu(G') = \nu(G) - 1$. It is easy to verify that $U = U' \cup \{v\}$ satisfies equality in the formula for G.

Case 2: For every $v \in G$, there is a maximum matching that misses it. Such a graph is called factor-critical and by Lemma 2.1 below, $\nu(G) = \frac{|V|-1}{2}$ and that there is an odd number of vertices in the entire graph. If we take $U = \emptyset$, then the theorem holds. ■

Lemma 2.1. *Let $G = (V, E)$ be a connected graph such that for each $v \in V$ there is a maximum matching in G that misses v. Then, $\nu(G) = \frac{|V|-1}{2}$. In particular, $|V|$ is odd.*

Proof. By way of contradiction, assume there exists two vertices $u \neq v$ and a maximum matching M that avoids them. Among all such choices, choose $M,\ u,\ v$ such that $dist(u, v)$ is minimized (by distance we mean the shortest path length). Since G is connected $dist(u, v)$ is finite. If $dist(u, v) = 1$ then M can be grown by adding the edge uv to it, contradicting M being a maximum matching. Therefore there exists a vertex t, $u \neq t \neq v$, such that t is on a shortest path from u to v. Also, by minimality of distance between u and v we know that t is covered by M; in fact all internal nodes on the shortest path between u and v must be covered by M.

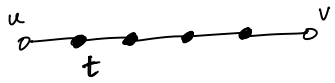


Figure 2.1: u, v are unmatched in M and every internal node on shortest path must be matched by M. Choose t on path arbitrarily.

By the assumption, there is at least one maximum matching that misses t. Choose a maximum matching N that maximizes $N \cap M$ while missing t. N must cover u, or else $N,\ u,\ t$ would have been a better choice than M since $dist(u, t) < dist(u, v)$. Similarly, N covers v. Now $|M| = |N|$ and we have one vertex t covered by M but not covered by N, and and two vertices u, v that are covered by N but not by M. Hence there must be another vertex x covered by M

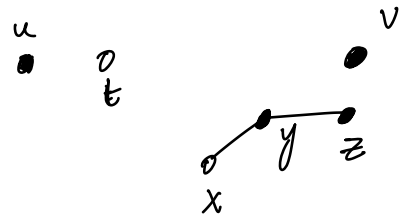


Figure 2.2: u, v matched by N and t, x unmatched by N. $xy \in M$ and $yz \in N$.

but not N that is different from u, v, t. Let $xy \in M$ (since x is covered by M). N is a maximum matching and hence xy cannot be added to it which implies that y is covered by N and also $y \neq t$ (by choice of N). Let $yz \in N$. Note that $z \neq x$.

Consider the matching $N' = N - yz + xy$. We have: $|N'| = |N|$ and N' avoids t and $|N' \cap M| > |N \cap M|$. This is a contradiction to the choice of N. Thus, any maximum matching in G leaves exactly one vertex out and this proves the lemma. ■

Remark 2.1. The preceding proof of the Tutte-Berge formula is graph theoretic and does not yield algorithmic insights, at least directly. We will see an efficient algorithm soon and one can use the properties of the algorithm to obtain an alternative proof of the Tutte-Berge formula. We gave the preceding proof to highlight the historical aspects and the fact that one can be motivated to look for an efficient algorithm by the very existence of a min-max relation.

2.2 Polynomial-time Algorithm for Maximum Cardinality Matching

As we mentioned previously, the Tutte-Berge formula gives strong evidence that maximum cardinality matching has a polynomial-time algorithm. The first such algorithm was discovered by Jack Edmonds. Faster algorithms are now known but the fundamental insight is easier to see in the original algorithm. Given a matching M in a graph G, we say that a node v is M-exposed if it is *not* covered by an edge of M.

Definition 2.3. *A path P in G is M-alternating if every other edge is in M. It can have odd or even length. A path P is M-augmenting if it is M-alternating and both ends are M-exposed.*

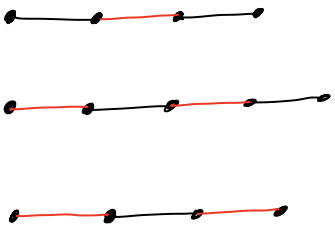


Figure 2.3: Examples of alternating paths with edges in M shown in red.

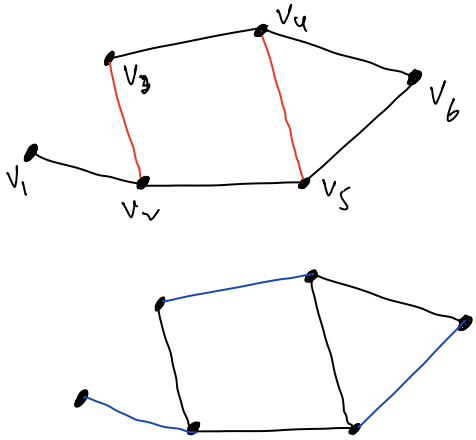


Figure 2.4: The path $v_1, v_2, v_3, v_4, v_5, v_6$ is an M-augmenting path. New matching after augmenting.

Lemma 2.2. *M is a maximum matching in G if and only if there is no M-augmenting path.*

Proof. If there is an M-augmenting path, then we could easily use it to find a matching M' such that $|M'| = |M| + 1$ and hence M is not a maximum matching.

In the other direction, assume that M is a matching that is not maximum by way of contradiction. Then there is a maximum matching N, and $|N| > |M|$. Let H be a subgraph of G induced by the edge set $M \Delta N = (M - N) \cup (N - M)$ (the symmetric difference). Note that the maximum degree of a node in H is at most 2 since a node can be incident to at most one edge from $N - M$ and one edge from $M - N$. Therefore, H is a disjoint collection of paths and cycles. Furthermore, all paths are M-alternating (and N-alternating too). All cycles must be of even length, since they alternate edges from M and N. At least one of the paths must have more edges from N than from M because $|N| > |M|$ and we deleted the same number of edges from N as M. That path must be of odd length and is an M-augmenting path. ■

The above lemma suggests a greedy algorithm for finding a maximum matching in a graph G. Start with a (possibly empty) matching and iteratively augment it by finding an augmenting path, if one exists. Thus the heart of the matter is to find an *efficient* algorithm that given G and matching M, either finds an M-augmenting path or reports that there is none.

Bipartite Graphs: We quickly sketch why it is easy to find augmenting paths in bipartite graphs. Let $G = (V, E)$ with A, B forming the vertex bipartition. Let M be a matching in G. Let X be the M-exposed vertices in A and let Y be the M-exposed vertices in B. Obtain a directed graph $D = (V, E')$ by orienting the edges of G as follows: orient edges in M from B to A and orient edges in $E \setminus M$ from A to B.

The following claim is easy to prove and we leave it as an exercise.

Claim 2.2.1. *There is an M-augmenting path in G if and only if there is an X-Y path in the directed graph D described above.*

Finding an augmenting path and implementing the greedy algorithm is effectively the augmenting path algorithm for maximum flow in the graph obtained from the standard reduction of bipartite matching to s-t max-flow.

Non-Bipartite Graphs: In general graphs it is not straight forward to find an M-augmenting path. As we will see, odd cycles form a barrier and Edmonds discovered the idea of shrinking them in order to recursively find a path. The first observation is that one can efficiently find an alternating *walk*.

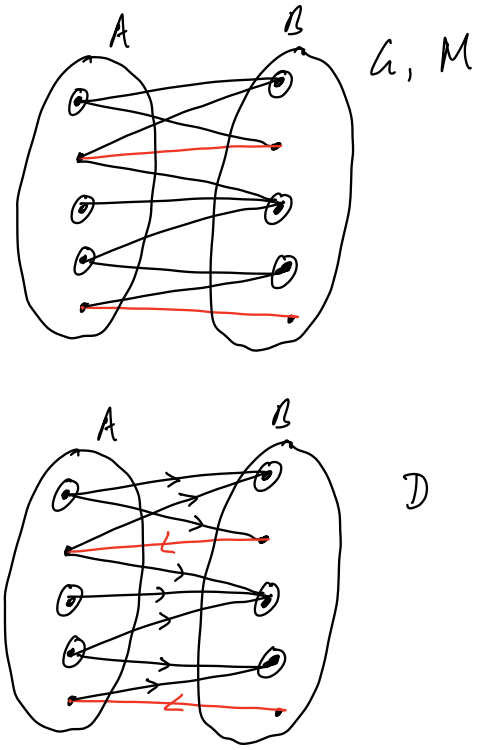


Figure 2.5: Finding M-augmenting path by reduction to directed graph reachability. The M-exposed nodes are shown with a circle over them.

Definition 2.4. *A walk in a graph $G = (V, E)$ is a finite sequence of vertices $v_0, v_1, v_2, \ldots, v_t$ such that $v_i v_{i+1} \in E, 0 \leq i \leq t-1$. The length of the walk is t.*

Note that edges and nodes can be repeated on a walk.

Definition 2.5. *A walk $v_0, v_1, v_2, \ldots, v_t$ is M-alternating walk if for each $1 \leq i \leq t-1$, exactly one of $v_{i-1}v_i$ and $v_i v_{i+1}$ is in M.*

Lemma 2.3. *Given a graph $G = (V, E)$, a matching M, and M-exposed nodes X, there is an $O(|V| + |E|)$ time algorithm that either finds a shortest M-alternating X-X walk of positive length or reports that there is no such walk.*

Proof sketch. Define a directed graph $D = (V, A)$ where $A = \{(u, v) : \exists x \in V, ux \in E, xv \in M\}$. Then a X-X M-alternating walk corresponds to a X-$N(X)$

directed path in D where $N(X)$ is the set of neighbors of X in G (we can assume there is no edge between two nodes in X for otherwise that would be a shortest walk). Alternatively, we can create a bipartite graph with $D = (V \cup V', A)$ where V' is a copy of V and $A = \{(u, v') \mid uv \in E \setminus M\} \cup \{(u', v) \mid uv \in M\}$ and find a shortest X-X' directed path in D where X' is the copy of X in V'. ■

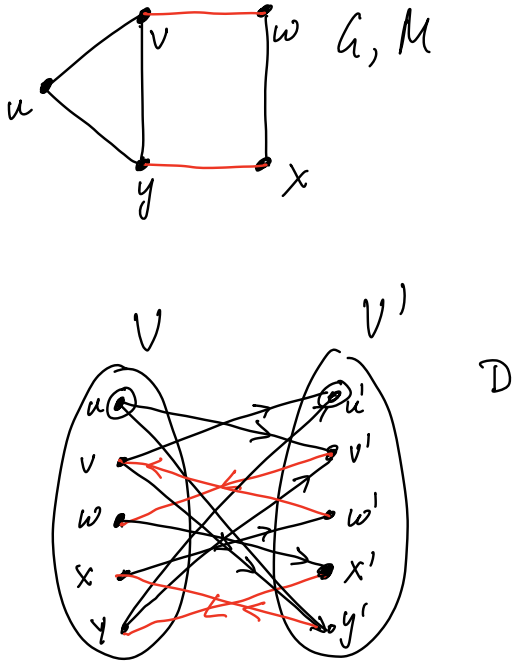


Figure 2.6: Finding M-alternating walk by reduction to directed graph reachability. Making copies of vertices. A shortest alternating walk from the unique M-exposed node u to itself is u, v, w, x, y, u.

What is the structure of an X-X M-alternating walk? Clearly, one possibility is that it is actually a path in which case it will be an M-augmenting path. However, there can be alternating walks that are not paths as shown by the figure below.

One notices that if an X-X M-alternating walk has an even cycle, one can remove it to obtain a shorter alternating walk. Thus, the main feature of an alternating walk when it is not a path is the presence of an *odd* cycle called a

blossom by Edmonds.

Definition 2.6. *An M-flower is an M-alternating walk $v_0, v_1, \ldots, v_t$ such that $v_o \in X$, t is odd and $v_t = v_i$ for some even $i < t$. In other words, it consists of an even length $v_0, \ldots, v_i$ M-alternating path (called the stem) attached to an odd cycle $v_i, v_{i+1}, \ldots, v_t = v_i$ called the M-blossom. The node v_i is the base of the stem and is M-exposed if $i = 0$, otherwise it is M-covered.*

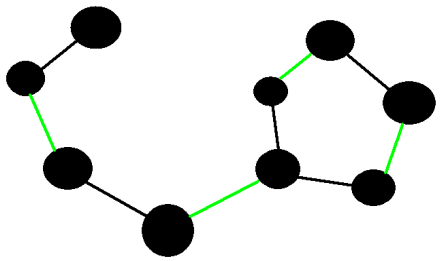


Figure 2.7: A M-flower. The green edges are in the matching

Lemma 2.4. *A shortest positive length X-X M-alternating walk is either an M-augmenting path or contains an M-flower as a prefix.*

Proof. Let $v_0, v_1, \ldots, v_t$ be a shortest X-X M-alternating walk of positive length. If the walk is a path then it is M-augmenting. Otherwise let i be the smallest index such that $v_i = v_j$ for some $j > i$ and choose j to be smallest index such that $v_i = v_j$. If $v_i, \ldots, v_j$ is an even length cycle we can eliminate it from the walk and obtain a shorter alternating walk. Otherwise, $v_0, \ldots, v_i, \ldots, v_j$ is the desired M-flower with v_i as the base of the stem. ■

Exercise 2.1. Describe an example of an X-X M-alternating walk with an even length cycle and show why one can eliminate it to obtain a shorter alternating walk.

Given a M-flower and its blossom B (we think of B as both a set of vertices and an odd cycle), we obtain a graph G/B by shrinking B to a single vertex b and eliminating loops and parallel edges. It is useful to identify b with the base of the stem.

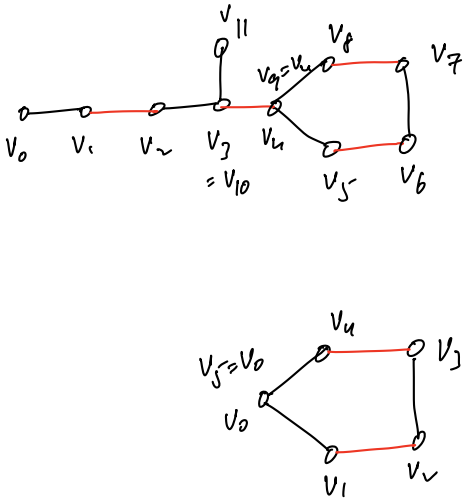


Figure 2.8: M-alternating walks and M-flowers. In the top figure the walk is $v_0, \ldots, v_{11}$ with the M-flower obtained from the prefix $v_0, \ldots, v_9$. In the bottom figure the walk is $v_0, \ldots, v_5$ which is itself an M-flower with degenerate stem; in this case the base is M-exposed.

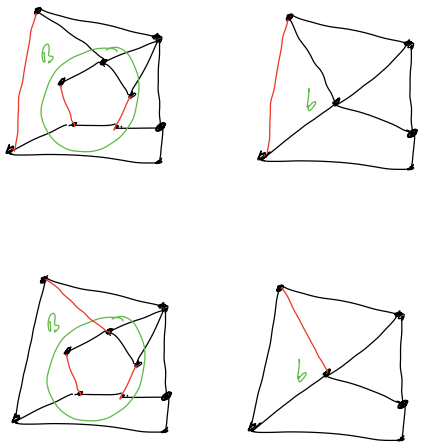


Figure 2.9: Shrinking the blossom. The shrunken vertex b base can be M/B exposed or not.

We obtain a matching M/B in G/B which consists of eliminating the edges of M with both end points in B. We note that b is M/B-exposed iff b is M-exposed.

Theorem 2.7. *M is a maximum matching in G if and only if M/B is a maximum matching in G/B.*

Proof. The next two lemmas cover both directions. ■

To simplify the proof we do the following. Let $P = v_0, \ldots, v_i$ be the stem of the M-flower. Note that P is an even length M-alternating path and if $v_0 \neq v_i$ then v_0 is M-exposed and v_i is M-covered. Consider the matching $M' = M \Delta E(P)$, that is by switching the matching edges in P into non-matching edges and vice-versa. Note that $|M'| = |M|$ and hence M is a maximum matching in G iff M' is a maximum matching. Now, the blossom $B = v_i, \ldots, v_t = v_i$ is also a M'-flower but with a degenerate stem and hence the base is M'-exposed. For the proofs to follow we will assume that $M = M'$ and therefore b is an exposed node in G/B. In particular we will assume that $B = v_0, v_1, \ldots, v_t = v_0$ with t odd.

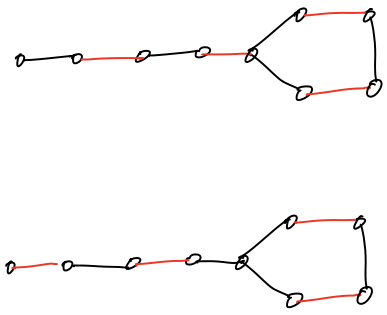


Figure 2.10: Given M-flower we change M to M' such that $|M| = |M'|$ and the blossom is M'-exposed. Helps simplify proof and algorithm.

Proposition 2.2.1. *For each v_i in B there is an even-length M-alternating path Q_i from v_0 to v_i.*

Proof. If i is even then $v_0, v_1, \ldots, v_i$ is the desired path, else if i is odd, $v_0 = v_t, v_{t-1}, \ldots, v_i$ is the desired path. That is, we walk along the odd cycle one direction or the other to get an even length path. ■

Lemma 2.5. *If there is an M/B augmenting path P in G/B then there is an M-augmenting path P' in G. Moreover, P' can be found from P in $O(m)$ time.*

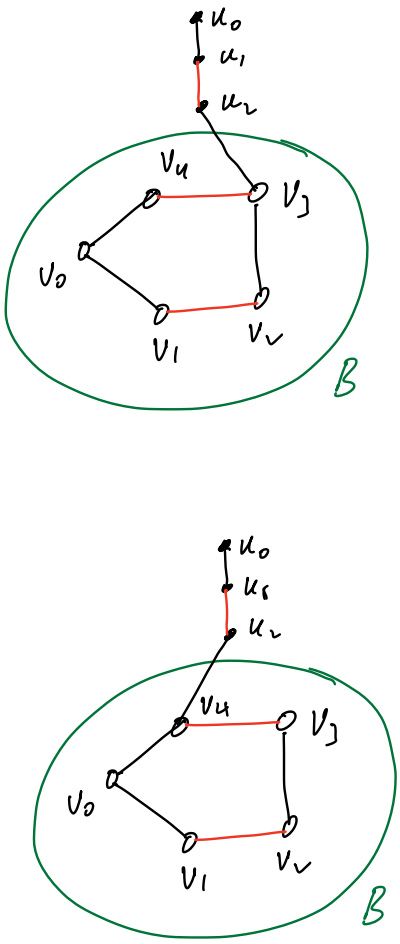


Figure 2.11: M/B-augmenting path in G/B implies an M-augmenting path in G. In the top figure u_2v_3 is the first edge of path P that touches b and we obtain the M-augmenting path $u_0, u_1, u_2, v_3, v4, v_0$ in G. In the bottom figure the path u_2v_4 is the first edge of P that touches b and we obtain the M-augmenting path $u_0, u_1, u_2, v_5, v_3, v_2, v_1, v_0$ in G. The odd cycle allows us to traverse the even length segment to the M-exposed node v_0.

Proof. **Case 1: P does not contain b.** Set $P' = P$.

Case 2: P contains b. b is an exposed node, so it must be an endpoint of P. Without loss of generality, assume b is the first node in P. Then P starts with an edge $bu \notin M/B$ and the edge bu corresponds to an edge v_iu in G where $v_i \in B$.

Obtain path P' by concatenating the even length M-alternating path Q_i from v_0 to v_i from Proposition 2.2.1 with the path P in which b is replaced by v_i; it is easy to verify that is an M-augmenting path in G. ■

Lemma 2.6. *If P is an M-augmenting path in G, then there exists an M/B augmenting path in G/B.*

Proof. Let $P = u_0, u_1, \ldots, u_s$ be an M-augmenting path in G. If $P \cap B = \emptyset$ then P is an M/B augmenting path in G/B and we are done. Since the only exposed node in B is v_0 one of the end points of P must be an exposed node that is not in B; without loss of generality assume that $u_0 \neq v_0$. Let u_j be the first vertex in P that is in B. We claim that $u_0, u_1, \ldots, u_{j-1}, b$ is an M/B augmenting path in G/B. Two cases to verify when $u_j = v_0$ and when $u_j = v_i$ for $i \neq 0$, both are easy. If $u_j = v_0$ then $u_0, u_1, \ldots, u_j$ is itself an M-augmenting path in G since $u_j = v_0$ is M-exposed and since b is M/B-exposed it follows that $u_0, u_1, \ldots, u_{j-1}, b$ is an M/B-augmenting path. If $u_j = v_i$ for $i \neq 0$ then the edge $u_{j-1}v_i \notin M$ since v_i is covered by M. Hence the path $u_0, \ldots, u_{j-1}, b$ is an M/B alternating path that starts with u_0 which is M/B exposed and ends in b which is M/B-exposed and hence M/B-augmenting. ■

From the above lemmas we have the following.

Lemma 2.7. *There is an $O(nm)$ time algorithm that given a graph G and a matching M, either finds an M-augmenting path or reports that there is none. Here $m = |E|$ and $n = |V|$.*

Proof. The algorithm is as follows. Let X be the M-exposed nodes. It first computes a shortest X-X M-alternating walk P in $O(m)$ time — see Lemma 2.3. If there is no such walk then clearly M is maximum and there is no M-augmenting path. If P is an M-augmenting path we are done. Otherwise there is an M-flower in P and a blossom B. We can assume by a simple transformation that the base of B is M-exposed. The algorithm shrinks B and obtains G/B and M/B which can be done in $O(m)$ time. It then calls itself recursively to find an M/B-augmenting path or find out that M/B is a maximum matching in G/B. In the latter case, M is a maximum matching in G. In the former case the M/B augmenting path can be extended to an M-augmenting path in $O(m)$ time as shown in Lemma 2.5. Since G/B has at least two nodes less than G, it follows that his recursive algorithm takes at most $O(nm)$ time. ■

By iteratively using the augmenting algorithm from the above lemma at most $n/2$ times we obtain the following result.

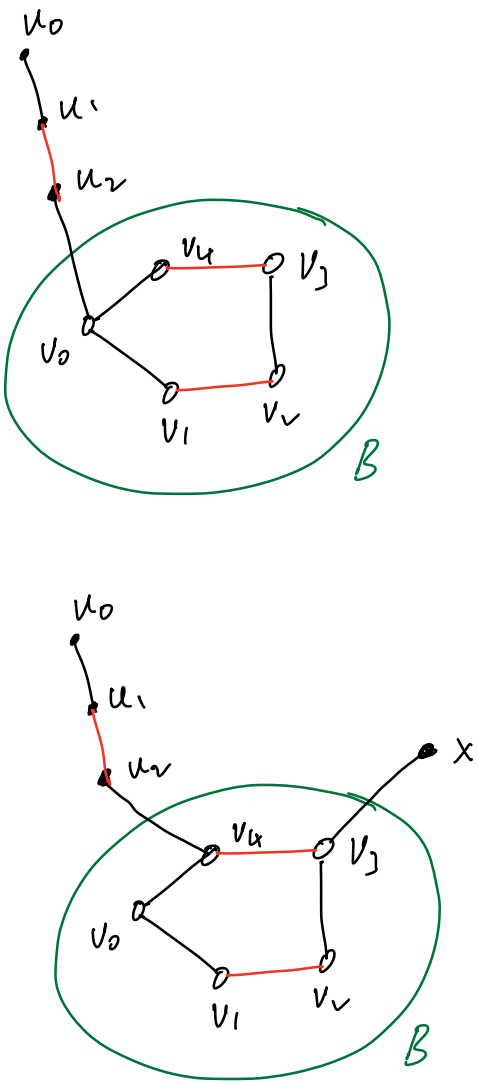


Figure 2.12: M-augmenting path in G implies an M/B-augmenting path in G/B. In the top figure $P = u_0, u_1, u_2, v_0$ which translates to u_0, u_1, u_2, b in G/B. In the second figure $P = u_0, u_2, v_5, v_3, x$ translates also to u_0, u_1, u_2, b in G/B. Note that b is M/B exposed in G/B.

Theorem 2.8. *There is an $O(n^2m)$ time algorithm to find a maximum cardinality matching in a graph with n nodes and m edges.*

The algorithm of Edmonds has been improved in terms of the running time. With some basic book keeping and avoiding recomputing the walk from scracth each iteration, one can obtain an $O(n^3)$ time algorithm. Via data structure tricks such as Union-Find one can improve running time to $O(nm)$. See Chapter 24 in [57] for more details. Micali and Vazirani claimed an $O(m\sqrt{n})$ time algorithm via an involved generalization of the Edmonds-Karp algorithm for bipartite matching. The correctness proof of the Micali-Vazirani algorithm has been difficult to establish (see [65]). However, if one does not worry about an extra logarithmic factor, weighted matching algorithms based on scaling lead to an easier to understand algorithm and proof [21].

Chapter 3

Polyhedra and Linear Programming[1]

This chapter is mainly based on [58]. Missing proofs can be found there.

In this chapter we will cover some basic material on the structure of polyhedra and linear programming. This is a large topic to be covered in a few classes, so pointers will be given for further reading. For algorithmic and computational purposes one needs to work with rational polyhedra. Many basic results, however, are valid for both real and rational polyhedra. We will not make a distinction unless necessary.

3.1 Basics

We will be working only with finite dimensional vector spaces over the reals equipped with the standard notions of Euclidean metric and inner product. We use $\mathbb{R}^n$ for some finite dimension n to denote this space. For $x, y \in \mathbb{R}^n$, $\|x - y\|_2$ defines the Euclidean distance between x and y where $\|z\|_2$ is the Euclidean 2-norm. The Pythagorean theorem shows that this distance induces a metric. A subset $X \subseteq \mathbb{R}^n$ is bounded if there is a finite radius R such that every point in X is at distance at most R from the origin. A set $X \subseteq \mathbb{R}^n$ is closed if the limit of every converging sequence of points in X is contained in X. A set X is compact if it is closed and bounded. We will implicitly use the Bolazano-Wierstrass theorem: every bounded sequence has a convergent subsequence.

Notation: Given two vectors $u, v\mathbb{R}^n$ we need notation for the sum $\sum_{i=1}^n u_i v_i$ which is the inner product of u and v. One notation for this is $\langle u, v \rangle$. However this product also arises in optimization as a linear objective $\sum_i c_i x_i$ where c

[1]Based on notes scribed by Sungjin Im and Ben Moseley in 2010.

is considered a fixed objective direction and x is a variable vector. In these contexts it is perhaps more natural to use notation such as $c^T x$ which comes from linear algebra where we view both c and x as $n \times 1$ column vectors. However, sometimes it is cumbersome to use the transpose notation and one sees cx or $c \cdot x$ as representing the same quantity with the underlying assumption that c is a row vector. We will use the transpose notation when there is scope for confusion and may slip and use the more sloppy notation when it is not confusing.

Definition 3.1. *Let $x_1, x_2, ..., x_m$ be points in $\mathbb{R}^n$. Let $x = \sum_{i=1}^m \lambda_i x_i$, where each $\lambda_{i=1} \in \mathbb{R}$, $1 \leq i \leq m$ is a scalar. Then, x is said to be*

1. *a* linear *combination of the x_i (for arbitrary scalars λ_i).*
2. *an* affine *combination of the x_i if $\sum_{i=1} \lambda_i = 1$.*
3. *a* conical *combination if $\lambda_i \geq 0$ for $1 \leq i \leq m$.*
4. *a* convex *combination if $\sum \lambda_i = 1$ and $\lambda_i \geq 0$ (affine and canonical).*

In the following definitions and propositions, unless otherwise stated, it will be assumed that $x_1, x_2, ..., x_m$ are points in $\mathbb{R}^n$ and $\lambda_1, \lambda_2, ..., \lambda_m$ are scalars in $\mathbb{R}$.

Definition 3.2. *$x_1, x_2, ..., x_m$ are said to be linearly independent if $\sum_{i=1}^m \lambda_i x_i = 0 \Rightarrow \lambda_i = 0 \; \forall i \in [m]$.*

Definition 3.3. *$x_1, x_2, ..., x_m$ are said to be affinely independent if the $m-1$ vectors $(x_i - x_1)$, $i = 2, ..., m$ are linearly independent, or equivalently if $\sum_{i=1}^m \lambda_i x_i = 0$ and $\sum_{i=1}^m \lambda_i = 0 \Rightarrow \forall i \in [m] \; \lambda_i = 0$.*

The following proposition is easy to check and the proof is left as an exercise to the reader.

Claim 3.1.1. *$x_1, x_2, ..., x_m$ are affinely independent if and only if the vectors $\binom{x_i}{1}$, $i = 1, 2, ..., m$, are linearly independent in $\mathbb{R}^{n+1}$.*

Definition 3.4. *A set $X \subseteq \mathbb{R}^n$ is said to be a linear subspace if X is closed under linear combinations of any finite subset of points in X. It is an affine set if it is closed under affine combinations. X is a cone if it is closed under conincal combinations. X is a convex set if it is closed under convex combinations.*

Note that an affine set is a translation of a subspace. That is, X is an affine set if there is a linear subspace Y and a vector b such that $X = \{y + b \mid y \in Y\}$. Given $X \subseteq \mathbb{R}^n$, we let $\texttt{Span}(X)$, $\texttt{Aff}(X)$, $\texttt{Cone}(X)$, and $\texttt{Convex}(X)$ denote the closures of X under linear, affine, conical, and convex combinations, respectively. In any of these sets X is said to be *finitely generated* if X is obtained by taking combinations

starting from a finite set of points. We know form linear algebra that every finite dimensional subspace is generated by a finite sized basis and that all bases have the same cardinality (which is the dimension of the subspace). However, cones and convex sets in finite dimensional spaces need not be finitely generated. Take for example the sphere.

To get an intuitive feel of the above definitions, see Figure 3.1.

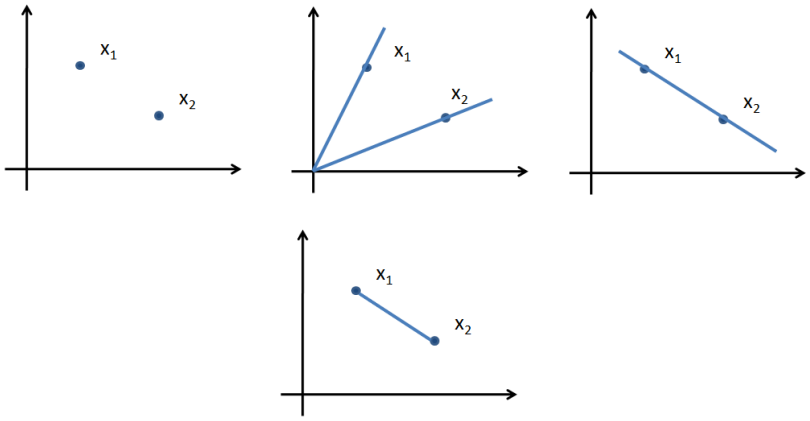


Figure 3.1: The subspace, cone set, affine set, and convex set of x_1, x_2 (from left to right). Note that the subspace is $\mathbb{R}^2$ and the cone set includes all points inside and on the two arrows.

Definition 3.5. *Given a set $X \subseteq \mathbb{R}^n$, the affine dimension of X is the maximum number of affinely independent points in X.*

3.2 Polyhedra, Polytopes, and Cones

Definition 3.6 (Hyperplane, Halfspace)**.** *A* hyperplane *in $\mathbb{R}^n$ is the set of all points $x \in \mathbb{R}^n$ that satisfy $a \cdot x = b$ for some $a \in \mathbb{R}^n$ and $b \in \mathbb{R}$. A* halfspace *is the set of all points x such that $a \cdot x \leq b$ for some $a \in \mathbb{R}^n$ and $b \in \mathbb{R}$.*

Definition 3.7 (Polyhedron)**.** *A* Polyhedron *in $\mathbb{R}^n$ is the intersection of finitely many halfspaces. It can be equivalently defined to be the set $\{x \mid Ax \leq b\}$ for a matrix $A \in \mathbb{R}^{m \times n}$ and a vector $b \in \mathbb{R}^{m \times 1}$.*

Definition 3.8 (Polyhedral cone)**.** *A* polyhedral cone *is $\mathbb{R}^n$ the intersection of finitely many halfspaces that contain the origin, i.e. $\{x \mid Ax \leq 0\}$ for a matrix $A \in \mathbb{R}^{m \times n}$.*

There are two ways of defining polytopes.

Definition 3.9 (Polyotpe)**.** *A* polytope *is a* bounded *polyhedron.*

Definition 3.10 (Polyotpe)**.** *A* polytope *is the convex hull of a finite set of points.*

The equivalence of the two definitions is a non-trivial and fundamental theorem! One can view polytopes as having two representations: via inequalities and via vertices. There can be an exponential gap between the representation sizes. For one direction consider the Boolean hypercube in n dimensions which is the convex hull of the 2^n vectors $\{0,1\}^n$. This is easily represented as a polyhedron with $2n$ inequalities (how?). Similarly there are examples of m points in n dimensions such that the number of inequalities required to describe it is exponential.

Note that a polyhedron is a convex and closed set. It is illuminating to classify a polyhedron into the following four categories.

1. Empty set (when the system $Ax \leq b$ is infeasible.)
2. Polytope (when the polyhedron is bounded.)
3. Cone
4. (Combination of) Cone and Polytope

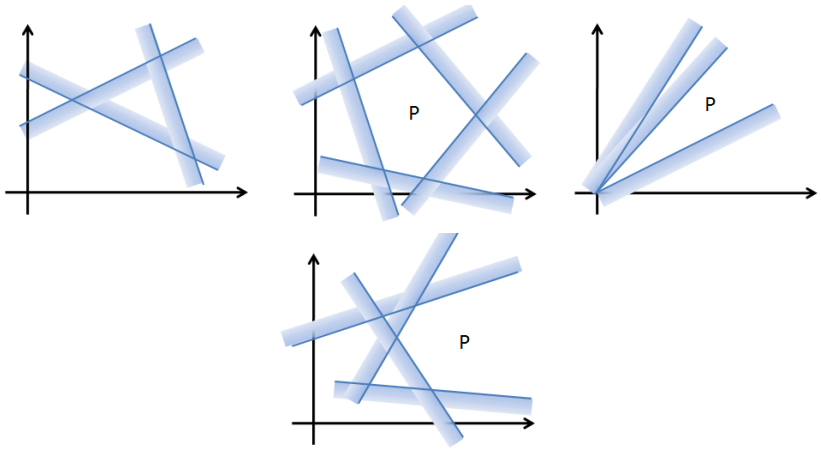
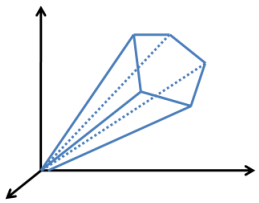


Figure 3.2: Examples of polyhedra left to right: Empty, Polytope, Cone, Combination of cone and polytope.

What “combination of cone and polytope" means will become clear soon in Theorem 3.13. For the examples, the reader is referred to Figure 3.2. In 2-D, a



cone can have only two “extreme rays," while in 3-D there is no bound on the number of extreme rays it can have.

For the most of part, we will be largely concerned with polytopes, but we need to have a better understanding of polyhedra first. Although it is geometrically “obvious" that a polytope is the convex hull of its “vertices," the proof is quite non-trivial. We will state the following three theorems without proof.

Theorem 3.11. *A bounded polyhedron is the convex hull of a finite set of points.*

Theorem 3.12. *A polyhedral cone is generated by a finite set of vectors. That is, for any $A \in \mathbb{R}^{m\times n}$, there exists a finite set X such that $\{x = \sum_i \lambda_i x_i \mid x_i \in X, \lambda_i \geq 0\} = \{x \mid Ax \leq 0\}$.*

Theorem 3.13. *A polyhedron $\{x \mid Ax \leq b\}$ can be written as the Minkowski sum of a polytope Q and a cone C, i.e. $P = Q + C = \{x + y \mid x \in Q, y \in C\}$.*

One can (geometrically) obtain the Minkowski sum of a polytope Q and a cone C by sweeping the origin of the cone C over the polytope Q. If the polyhedron P is pointed (has at least one “vertex"), the decomposition is, in fact, modulo scaling factor unique. Further the cone C above is $\{x \mid Ax \leq 0\}$, or equivalently the set of unbounded directions in P. The cone C is called the characteristic cone or the recession cone of P.

Exercise 3.1. Let $P = \{x \mid Ax \leq b, x \in \mathbb{R}^n\}$ be a polyhedron. Let $z \in P$. Prove that for $y \in C$ and any $\alpha \geq 0$, $z + \alpha y \in P$.

Many facts about polyhedra and linear programming rely on (in addition to convexity) variants of Farkas’ lemma that characterizes when a system of linear inequalities do not have solution. The simplest proof for one variant is via Fourier-Motzkin elimination that is independently interesting and related to the standard Gauss-Jordan elimination for solving system of linear equations.

3.3 Fourier-Motzkin Elimination

Let $P = \{x \mid Ax \leq b\} \subseteq \mathbb{R}^n$ be a polyhedron. For k in $[n]$,
we let $P^k = \{(x_1, .., x_{k-1}, x_{k+1}, ..., x_n) \mid (x_1, x_2, ..., x_n) \in P\}$ be the *projection* of P

along the x_k-axis.

Theorem 3.14. *P^k is a polyhedron.*

Proof. We derive a set of inequalities that describe P^k. We do this by considering the inequalities in $Ax \leq b$ and eliminating the variables x_k as follows. Partition the inequalities in $Ax \leq b$ into three sets:

$$S^+ = \{i \in [m] \mid a_{ik} > 0\}, S^- = \{i \in [m] \mid a_{ik} < 0\}, \text{and } S^0 = \{i \in [m] \mid a_{ik} = 0\}.$$

Define a new set of inequalities consisting of S_0 and one new inequality for each pair (i, ℓ) in $S^+ \times S^-$:

$$a_{ik}(\sum_{j=1}^{n} a_{\ell j} x_j) - a_{\ell k}(\sum_{j=1}^{n} a_{ij} x_j) \leq a_{ik} b_\ell - a_{\ell k} b_i.$$

Note that the combined inequality does not have x_k. We also observe that since $a_{ik} > 0$ and $a_{\ell k} < 0$ the combined inequality is obtained as a *non-negative* combination of two inequalities of the original system. It is therefore clear that the new system of inequalities is a set of valid inequalities for P[2].

We now have a total of $|S^0| + |S^+||S^-|$ new inequalities. Let $P' = \{x' \in \mathbb{R}^{n-1} \mid A'x' \leq b'\}$ where $A'x' \leq b'$ is the new system of inequalities in variables $x_1, x_2, ..., x_{k-1}, x_{k+1}, ..., x_n$. We prove the theorem by showing that $P^k = P'$.

We first show the easier direction: $P^k \subseteq P'$. Consider any point $z \in P^k$. By definition of P^k, there exists $y \in P$ such that $Ay \leq b$ and projection of y onto the k'th axis is z. The system $A'x' \leq b'$ is valid for P and hence y satisfies this system and hence also z (since the new system does not have the variable x_k).

We now show that $P' \subseteq P^k$. Note that $Ax \leq b$ can be rewritten as

$$a_{i1} x_1 \leq b_i - \sum_{j=2}^{n} a_{ij} x_j, i \in [m]. \tag{3.1}$$

Without loss of generality, assume $k = 1$. Consider any $z = (z_2, z_3, ..., z_n) \in P'$. We want to show that there exists $z_1 \in \mathbb{R}$ such that $Az' \leq b$, where $z' = (z_1, z_2, ..., z_n)$. To simplify notation, for $i \in [m]$, let $C_i = b_i - \sum_{j=2}^{n} a_{ij} z_j$. Observe that z satisfies all inequalities in S^0, since the new system includes those constraints. Thus it suffices to show that

$$\begin{aligned} \exists z_1 \text{ s.t.} \quad & a_{i1} z_1 \leq C_i, \forall i \in S^+ \cup S^-. \\ \Leftrightarrow \quad & \max_{\ell \in S^-} \frac{C_\ell}{a_{\ell 1}} \leq z_1 \leq \min_{i \in S^+} \frac{C_i}{a_{i1}}. \end{aligned}$$

[2]It is convenient to assume, without loss of generality, that the coefficient of x_k in each of the inequalities of S_1 is 1 and is equal to -1 in each of the inequalities of S_2. Then each new inequality is obtained as simply the sum of one inequality from S_1 and one from S_2.

It is easy to observe that this is equivalent to

$$\begin{array}{rrcll} & \frac{C_\ell}{a_{\ell 1}} & \leq & \frac{C_i}{a_{i1}} & \forall (i,\ell) \in S^+ \times S^- \\ \Leftrightarrow & 0 & \leq & a_{i1}C_\ell - a_{\ell 1}C_i & \forall (i,\ell) \in S^+ \times S^- \\ \Leftrightarrow & A'z & \leq & b' & \end{array}$$

And we know that $A'z \leq b'$ since $z \in P'$, completing the proof. ■

We state two useful corollaries the first of which is based on observation we made in the proof.

Corollary 3.15. *P^k is a polyhedron and moreover $P^k = \{z \in \mathbb{R}^n \mid A'z \leq b'\}$ where each inquality in $A'z \leq b'$ is a non-negative combination of the inequalities of P.*

Corollary 3.16. *P^k is empty iff P is empty.*

Remark 3.1. Fourier-Motzkin elimination gives an exponetial time algorithm to check if a polyhedron defined by a system of inequalities $\{x \mid Ax \leq b\}$ is empty. We keep eliminating variables until we reduce the problem to a one-dimension problem where it is easy, however, the number of inequalities grows as we eliminate variables. Polynomial-time solvability of linear programming implies that there are other more efficient ways of checking whether P is empty.

From Fourier-Motzkin elimination we get an easy proof of one variant of Farkas' lemma.

Theorem 3.17 (Theorem of Alternatives)**.** *Let $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. For the system $Ax \leq b$, exactly one of the following two alternatives hold:*

- *The system is feasible.*
- *There exists $y \in \mathbb{R}^m$ such that $y \geq 0$, $y^T A = 0$ and $y^T b < 0$.*

What the theorem says is that if the system of inequalities $Ax \leq b$ is infeasible then there is a proof (certificate) of this which can be obtained by taking non-negative combination of the inequalities (given by $y \geq 0$) to derive a contradiction of the following form: $0 = y^T A \leq y^T b < 0$.

Proof of Theorem 3.17. Suppose that there exists a vector $y \geq 0$ such that $y^T A = 0$ and $y^T b < 0$. If there is any feasible vector z such that $Az \leq b$ then it easily follows that $0 \leq y^T Az \leq y^T b$, since $y \geq 0$, which is a contradiction to the fact that $y^T b < 0$. Thus there can be no feasible solution to $Ax \leq b$.

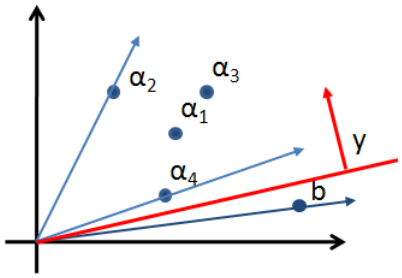
Conversely, suppose $Ax \leq b$ is infeasible. Let $P = \{x \mid Ax \leq b\}$. We use Fourier-Motzkin procedure to eliminate variables $x_1, x_2, \ldots, x_n$ (we can choose any arbitrary order) to obtain polyhedra $P = Q_0, Q_1, Q_2, \ldots, Q_{n-1}, Q_n$. Note that

Q_{i+1} is non-empty iff Q_i is, and that Q_{n-1} has only one variable and Q_n has none. The inequalities describing Q_i are non-negative combination of the inequalities of P as we observed earlier. Thus, Q_n is empty iff we have derived an inequality of the form $0 \leq C$ for some $C < 0$ at some point in the process. That inequality gives the desired $y \geq 0$. ■

Two variant of Farkas' lemma that are useful can be derived from the theorem of alternatives.

Theorem 3.18. *$Ax = b, x \geq 0$ has no solution iff $\exists y \in \mathbb{R}^m$ such that $y^T A \geq 0$ and $y^T b < 0$.*

The preceding theorem has a nice geometric interpretation. Let $\alpha_1, \alpha_2, \ldots, \alpha_n$ be the columns of A viewed as vectors in $\mathbb{R}^m$. Then $Ax = b, x \geq 0$ has a solution if and only if b is in the cone generated by $\alpha_1, \alpha_2, \ldots, \alpha_n$; here the conic combination is given by $x \geq 0$. So b is either in the $\mathsf{Cone}(\alpha_1, \alpha_2, \ldots, \alpha_n)$ or there is a hyperplane separating b from $\alpha_1, \alpha_2, \ldots, \alpha_n$.



In fact the theorem can be strengthened to show that the hyperplane can be chosen to be one that spans $t - 1$ linearly independent vectors in $\alpha_1, \alpha_2, \ldots, \alpha_n$, where $t = \mathsf{rank}(\alpha_1, \alpha_2, \ldots, \alpha_n, b)$.

Proof of Theorem 3.18. It is easy to see that if $\exists y \in \mathbb{R}^m$ such that $y^T A \geq 0$ and $y^T b < 0$ then the original system has no solution. We prove the harder direction.

We can rewrite $Ax = b, x \geq 0$ as

$$\begin{bmatrix} A \\ -A \\ -I \end{bmatrix} x \leq \begin{bmatrix} b \\ -b \\ 0 \end{bmatrix}$$

Hence by the Theorem of Alternatives, $Ax = b, x \geq 0$ is not feasible only if there exists a $y' = \begin{bmatrix} u & v & w \end{bmatrix}^T$, where u, v are vectors in $\mathbb{R}^m$ and w is vector in $\mathbb{R}^n$

such that

$$\begin{aligned} u, v, w &\geq 0 \\ u^T A - v^T A - w &= 0 \\ u^T b - v^T b &< 0 \end{aligned}$$

Let $y = u - v$. Note that $y \in \mathbb{R}^m$ is not necessarily positive. From the second and third inequalities, we can easily obtain $y^T A = w \geq 0$ and $y^T b < 0$. ■

Another variant of Farkas' lemma is as follows and the proof is left as an exercise.

Theorem 3.19. *$Ax \leq b, x \geq 0$ has a solution iff $y^T b \geq 0$ for each row $y \geq 0$ with $y^T A \geq 0$.*

Another interesting and very useful theorem is Carathéodory's theorem

Theorem 3.20 (Carathéodory)**.** *Let $x \in \texttt{Convexhull}(X)$ for a finite set X of points in $\mathbb{R}^n$. Then $x \in \texttt{Convexhull}(X')$ for some $X' \subseteq X$ such that vectors in X' are affinely independent. In particular, $|X'| \leq n + 1$.*

A conic variant of Carathéodory's theorem is as follows.

Theorem 3.21. *Let $x \in \texttt{Cone}(X)$ where $X = \{x_1, x_2, ..., x_m\}$, $x_i \in \mathbb{R}^n$. Then $x \in \texttt{Cone}(X')$ for some $X' \subseteq X$ where vectors in X' are linearly independent. In particular, $|X'| \leq n$.*

Proof. Since $x \in \texttt{Cone}(X)$, $x = \sum_i \lambda_i x_i$ for some $\lambda_i \geq 0$. Choose a combination with mininum support, i.e. the smallest number of non-zero λ_i values. Let $X' = \{\lambda x_i \mid \lambda_i > 0\}$ and $I = \{i \mid \lambda_i > 0\}$. If vectors in X' are linearly independent, we are done. Otherwise, $\exists \alpha_i, i \in I$ s.t. $\sum_{i \in I} \alpha_i \lambda_i x_i = 0$. By scaling we can assume that $\forall i \in I, \alpha_i \leq 1$, and $\exists j \in I$ s.t. $\alpha_j = 1$. Then,

$$x = \sum_{i \in I} \lambda_i x_i = \sum_{i \in I} \lambda_i x_i - \sum_{i \in I} \alpha_i \lambda_i x_i = \sum_{i \in I} \lambda_i (1 - \alpha_i) x_i.$$

Letting $\lambda'_i = \lambda_i(1 - \alpha_i)$ we see that $\lambda'_i \geq 0$ for all i and $\lambda'_j = 0$. Therefore, from the preceding we have that $x = \sum_{i \in I \setminus \{j\}} \lambda'_i x_i$ which means that x can be written as a conical combination of vectors in $X' - \{x_j\}$, contradicting the choice of X'. ■

Exercise 3.2. Prove the affine version of Carathéodory's theorem from the conical version (or directly).

3.4 Linear Programming

Linear programming is an optimization problem of the following form.

$$\begin{aligned} \max \quad & c^T x \quad \text{(Primal-LP)} \\ & Ax \leq b \end{aligned}$$

where $c \in \mathbb{R}^n$ and $A \in \mathbb{R}^{m \times n}$. Here

In other words, we wish to optimize a linear objective function over a *polyhedron*; by negating c we can also minimize. This is a natural standard form of LP since we wish to highlight the fact that the feasible space is a polyhedron. Given an LP, there are three possibilities:

1. The polyhedron is *infeasible*.
2. The objective function can be made arbitrarily large in which case we say it is *unbounded*.
3. There is a finite optimum value in which case we say it is *bounded*.

Another important standard form of LP is

$$\begin{aligned} \max \quad & c^T x \quad \text{(Primal-LP)} \\ & Ax = b \\ & x \geq 0 \end{aligned}$$

The above is an interesting standard form since the form highlights the difference between linear system solving and LPs. In linear system solving we only have $Ax = b$ without the constraint that $x \geq 0$.

Exercise 3.3. Consider the problem $\max c^T x$ subject to $Ax = b$. Use insights from basic linear algebra and linear system solving to obtain an efficient algorithm for this problem.

Each linear program has its associated “dual" linear program. The LP we refer to by “dual" depends on the “starting" LP, which is called as the primal LP; in fact the dual of dual LP is exactly the same as the primal LP. Let us say that the following LP is the primal LP here.

$$\begin{aligned} \max \, & c^T x \\ & Ax \leq b \end{aligned}$$

We can “derive" the dual by thinking about how we can obtain an upper bound on the optimal value for the primal LP. Given the system $Ax \leq b$, any

inequality obtained by non-negative combination of the inequalities in $Ax \leq b$ is a valid inequality for the system. We can represent a non-negative combination by a $m \times 1$ row vector $y^T \geq 0$.

Thus $y^T Ax \leq y^T b$ is a valid inequality whenever $y \geq 0$. Take any vector $y \geq 0$ such that $y^T A = c$. Then such a vector gives us an upperbound on the LP value since $y^T Ax = cx \leq y^T b$ is a valid inequality. Therefore one can obtain an upperbound by minimizing over all $y \geq 0$ such that $y^T A = c$. Therefore the objective function of the primal LP is upperbounded by the optimum value of

$$\begin{aligned} \min \quad & y^T b \quad \text{(Dual-LP)} \\ & y^T A = c \\ & y \geq 0 \end{aligned}$$

The above derivation of the Dual LP immediately implies the Weak Duality Theorem.

Theorem 3.22 (Weak Duality)**.** *If x' and y' are feasible solutions to Primal-LP and Dual-LP then $c^T x' \leq (y')^T b$.*

Corollary 3.23. *If the primal-LP is unbounded then the Dual-LP is infeasible.*

Exercise 3.4. Prove that the dual of the Dual-LP is the Primal-LP.

The main result in the theory of linear programming is the following Strong Duality Theorem which is essentially a min-max result.

Theorem 3.24 (Strong Duality)**.** *If Primal-LP and Dual-LP have feasible solutions, then there exist feasible solutions x^* and y^* to them such that $c^T x^* = (y^*)^T b$.*

Proof. Note that by weak duality we have that $c^T x' \leq (y')^T b$ for any feasible pair of x' and y'. Thus to show the existence of x^* and y^* it suffices to show that the system of inequalities below has a feasible solution whenever the two LPs are feasible.

$$\begin{aligned} c^T x &\geq y^T b \\ Ax &\leq b \\ y^T A &= c \\ y &\geq 0 \end{aligned}$$

We rewrite this as

$$\begin{aligned} Ax &\leq b \\ y^T A &\leq c \\ -y^T A &\leq -c \\ -y &\leq 0 \\ -c^T x + y^T b &\leq 0 \end{aligned}$$

and apply the Theorem of Alternatives. Note that we have inequalities in $n + m$ variables corresponding to the x and y variables. By expressing those variables as a vector $z = \begin{bmatrix} x \\ y \end{bmatrix}$, we have

$$\begin{bmatrix} A & 0 \\ 0 & A^T \\ 0 & -A^T \\ 0 & -I \\ -c^T & b^T \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \leq \begin{bmatrix} b \\ c \\ -c \\ 0 \\ 0 \end{bmatrix}$$

Note that only one inequality ties the two sets of variables. The total number of constraints in the above system is $m + 2n + m + 1$. If the above system does not have a solution then there exists a row vector $\begin{bmatrix} s^T & p^T & q^T & u^T & \alpha \end{bmatrix} \geq 0$, where s is a $m \times 1$ vector, p, q are $n \times 1$ vectors, u is a $m \times 1$ vector and α is a scalar such that

$$\begin{aligned} s^T A - \alpha \cdot c &= 0 \\ pA^T - qA^T - u + \alpha \cdot b &= 0 \end{aligned}$$

and

$$s^T b + p^T c - q^T c < 0.$$

We replace $p - q$ by w and note that now w is not necessarily positive. Hence we obtain that if the strong duality does not hold then there exist vectors $s, u \in \mathbb{R}^m, w \in \mathbb{R}^n, v \in \mathbb{R}$ such that

$$\begin{aligned} s, u, \alpha &\geq 0 \\ s^T A - \alpha c &= 0 \\ wA^T - u + \alpha b &= 0 \\ s^T b + w^T c &< 0 \end{aligned}$$

We consider two cases.

Case 1: $\alpha = 0$. (note that α is a scalar.) What does this case mean? This means that the inequality $c^T x \geq y^T b$ is not playing a role in the infeasibility proof of the system. Recall that this is the only inequality that ties the x and y variables together. Thus the infeasibility must be present in either the original primal sytem or the dual system but they have feasible solutions and hence this cannot happen. We formally prove this now.

When $\alpha = 0$ we obtain the simplified set.

$$\begin{aligned} s &\geq 0 \\ s^T A &= 0 \\ wA^T &= u \\ s^T b + wc^T &< 0 \end{aligned}$$

We have a feasible dual solution y^* that satisfies $(y^*)^T A = c$. Since $s^T A = 0$, and $s \geq 0$, $y^* + \beta s$ is a feasible solution for the dual for any scalar $\beta \geq 0$. Similarly knowing that there exists a feasible solution $Ax^* \leq b$, and $Aw = 0$, it follows that $x^* - \beta w$ is feasible for the primal for any scalar $\beta \geq 0$. Applying the Weak Duality Theorem, we have that $\forall \beta \geq 0$,

$$\begin{aligned} & c^T(x^* - \beta w) \leq (y^* + \alpha s)^T b \\ \Rightarrow \quad & c^T x^* - (y^*)^T b \leq \beta(s^T b + c^T w) \end{aligned}$$

However, the LHS is fixed while the RHS can be made arbitrarily small because $s^T b + c^T w < 0$ and β can be chosen arbitrarily large. This is a contradiction.

Case 2: $\alpha > 0$. Let $s' = \frac{1}{\alpha}(s)$, $w' = \frac{1}{\alpha}(w)$, and $u' = \frac{1}{\alpha}(u)$. Then, we have

$$\begin{aligned} s', u' &\geq 0 \\ (s')^T A &= c \\ w'A^T - u' &= -b \Rightarrow -A(w')^T = b - u' \leq b \ \text{ [Since } u' \geq 0.] \\ (s')^T b + c^T w' &< 0 \end{aligned}$$

From the inequalities above, we observe that s' is dual feasible and $-w'$ is primal feasible. Thus by the Weak Duality, we have $-c^T w' \leq (s')^T b$, contradicting that $s'b + c^T w' < 0$.

■

Complementary Slackness is a very useful consequence of Strong Duality.

Theorem 3.25 (Complementary Slackness)**.** *Let x^*, y^* be feasible solutions to the primal and dual LPs. Then x^*, y^* are optimal solutions if and only if $\forall i \in [m]$, either $y_i^* = 0$ or $a_i x^* = b_i$.*

Proof. Let $A_1, A_2, \ldots, A_m$ be the row vectors of A. Suppose that the given condition is satisfied. Then we have $(y^*)^T Ax^* = \sum_{i=1}^m y_i^*(A_i x^*) = \sum_{i=1}^m y_i^* b_i = (y^*)^T b$. Also we know that $c^T x^* = ((y^*)^T A)x^*$ since y^* is a feasible solution for the dual. Thus we have $c^T x^* = (y^*)^T b$, and by weak duality, we conclude that x^*, y^* are optimal.

Conversely, suppose that x^* and y^* both are optimal.

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

where the first equality is due to strong duality, second is due to dual feasibility of y^* and the third is due to primal feasibility of x^*. Thus we obtain the equality $(y^*)^T Ax^* = (y^*)^T b$, that is $\sum_{i=1}^m y_i^*(A_i x^*) = \sum_{i=1}^m y_i^* b_i$; note that this is a vector equation. Also $y^* \geq 0$. This forces the desired condition, since $A_i x^* \leq b_i$, $y_i^* \geq 0$ for each $i \in [m]$ due to feasibility of x^*, y^*. ■

3.5 Implicit equalities and Redundant Constraints

Let $P = \{x \mid Ax \leq b\}$ is a *polyhedron* in $\mathbb{R}^n$ where A is a $m \times n$ matrix and b is a $m \times 1$ matrix.

The set of directions along which one can go to infinity in P is described by the charectiristic cone of P (also called the recession cone of P).

Definition 3.26. **charcone**$(P) = \{x \mid Ax \leq 0\}$.

Definition 3.27. **linspace**$(P) = \{x \mid Ax = 0\} =$ **charcone**$(P) \cap -$**charcone**(P). *In words,* **linspace**(P) *is the set of all directions c such that there is a line parallel to c fully contained in P.*

Definition 3.28. *A polyhedron P is* pointed *if and only if* **linspace**$(P) = \{0\}$*, that is* **linspace**(P) *has dimension* 0*.*

Exercise 3.5. Describe a polyhedron in two dimensions that is *not* pointed.

Definition 3.29. *An inequality $a_i x \leq b_i$ in $Ax \leq b$ is an* implicit equality *if $a_i x = b_i$ $\forall x \in P$.*

Definition 3.30. *A constraint row in $Ax \leq b$ is* redundant *if removing it does not change the polyhedron. The system $Ax \leq b$ is* irredundant *if no constraint is redundant.*

Let $I \subseteq \{1, 2, \ldots, m\}$ be the index set of all implicit equalities in $Ax \leq b$. Then we can partition A into $A^= x \leq b^=$ and $A^+ x \leq b^+$. Here $A^=$ consists of the rows of A with indices in I and A^+ are the remaining rows of A. Therefore, $P = \{x \mid A^= x = b^=, A^+ x \leq b^+\}$. In other words, P lies in an affine subspace defined by $A^= x = b^=$. In some setting it is convenient to assume that each row in $A^+ x \leq b^+$ is irredundant.

Exercise 3.6. Prove $\exists x \in P$ such that $A^{=}x = b^{=}$ and $A^{+}x < b^{+}$.

Definition 3.31. *The dimension,* $\mathbf{dim}(P)$*, of a polyhedron* P *is the maximum number of affinely independent points in* P *minus* 1.

Notice that by definition of dimension, if $P \subseteq \mathbb{R}^n$ then $\mathbf{dim}(P) \leq n$. If $P = \emptyset$ then $\mathbf{dim}(P) = -1$, and $\mathbf{dim}(P) = 0$ if and only if P consists of a single point. If $\mathbf{dim}(P) = n$ then we say that P is *full-dimensional*.

Exercise 3.7. Show that $\mathbf{dim}(P) = n - \mathbf{rank}(A^{=})$.

The previous exercise implies that P is full-dimensional if and only if there are no implicit inequalities in $Ax \leq b$.

Definition 3.32. $\mathbf{affhull}(P) = \{x \mid A^{=}x = b^{=}\}$

The following lemma says that inequalities in $A^{=}x \leq b^{=}$ already imply that they are equalities.

Lemma 3.1. $\mathbf{affhull}(P) = \{x \mid A^{=}x = b^{=}\} = \{x \mid A^{=}x \leq b^{=}\}$.

3.6 Faces of Polyhedra

Definition 3.33. *An inequality* $\alpha x \leq \beta$*, where* $\alpha \neq 0$*, is a* valid *inequality for a polyhedron* $P = \{x \mid Ax \leq b\}$ *if* $\alpha x \leq \beta \ \forall x \in P$*. The inequality is a* supporting hyperplane *if it is valid and has a non-empty intersection with* P*.*

Definition 3.34. *A* face *of a polyhedron* P *is the intersection of* P *with* $\{x \mid \alpha x = \beta\}$ *where* $\alpha x \leq \beta$ *is a valid inequality for* P*.*

We are interested in non-empty faces. Notice that a face of a polyhedron is also a polyhedron. A face of P is an *extreme point* or a *vertex* if it has dimension 0. It is a *facet* if the dimension of the face is $\mathbf{dim}(P) - 1$. The face is an *edge* if it has dimension 1.

Another way to define a face is to say that F is a face of P if $F = \{x \in P \mid A'x = b'\}$ where $A'x = b'$ is a subset of the inequalities of $Ax \leq b$. In other words, $F = \{x \in P \mid a_i x = b_i, i \in I\}$ where $I \subseteq \{1, 2, \ldots, m\}$ is a subset of the rows of A.

Now we will show that these two definitions are equivalent. The equivalence helps in various proofs with one or the other more convenient.

Theorem 3.35. *Let* $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^n$*. Let* $P = \{x \mid Ax \leq b\}$ *be a polyhedron. Let* F *be a face defined by a valid inequality* $\alpha x \leq \beta$*. Then* $\exists I \subseteq \{1, 2, \ldots, m\}$ *such that* $F = \{x \in P \mid a_i x = b_i, i \in I\}$*.*

Proof. Let $F = \{x \mid x \in P, \alpha x = \beta\}$ where $\alpha x \leq \beta$ is a supporting hyperplane.

Claim 3.6.1. *F is the set of all optimal solutions to the LP*

$$\max \alpha x$$
$$Ax \leq b.$$

Proof of Claim. Since $\alpha x \leq \beta$ is a valid inequality for P the optimum value for the LP is at most β. Since F is non-empty the optimum is achievable. By definition of F, $y \in F$ iff $Ay \leq b$ and $\alpha y = \beta$. Thus any point in F is an optimum solution to the LP and vice versa. ■

As we saw the above LP has a bounded optimal value β. This implies that the dual LP has a feasible solution y^* of the same value. Let $I = \{i \mid y^*_i > 0\}$. Let Z be the set of all optimal solutions to the primal. Let $z \in Z$. By complimentary slackness for z and y^*, we have that $y^*_i > 0$ implies $a_i z = b_i$ for all $z \in Z$. Therefore Z satisfies,

$$a_i x = b_i \qquad i \in I$$
$$a_i x \leq b_i \qquad i \notin I$$

Again, by complementary slackness any z that satisfies the above is optimal (via y^*) and, hence, $F = \{x \in P \mid a_i x_i = b_i, i \in I\}$. ■

Now we consider the converse.

Theorem 3.36. *Let $P = \{x \mid Ax \leq b\}$ be a non-empty polyhedron where $A \in \mathbb{R}^{m \times n}$ is a $m \times n$ matrix, $b \in \mathbb{R}^m$ is a $m \times 1$ matrix. Let $I \subseteq \{1, 2, \ldots, m\}$ and $F = \{x \in P \mid a_i x = b_i, i \in I\}$. If F is non-empty, then there is a valid inequality $\alpha x \leq \beta$ such that $F = P \cap \{x \mid \alpha x = \beta\}$.*

Proof. Let $\alpha = \sum_{i \in I} a_i$ be a row vector and $\beta = \sum_{i \in I} b_i$. We claim that $F = P \cap \{x \mid \alpha x = \beta\}$. To see this we observe that the inequality $\alpha x \leq \beta$ is a valid inequality for P since it is a non-negative combination of inequalities of the system $Ax \leq b$. Consider any $z \in F$. We have $z \in P$ and $a_i z = b_i$ for each $i \in I$. This implies that $\alpha z = \sum_{i \in I} a_i z = \sum_{i \in I} b_i = \beta$. Thus $F \subseteq P \cap \{x \mid \alpha x = \beta\}$. Conversely if $\alpha z = \beta$ and $z \in P$ then we claim that $a_i z = b_i$ for all $i \in I$; this is easy to verify. ■

Corollary 3.37.

1. *Each face is a polyhedron.*
2. *The number of faces of $P = \{x \mid Ax \leq b\}$ where A is a $m \times n$ matrix is at most 2^m.*

3. *If F is a face of P and $F' \subseteq F$ then F' is a face of P if and only if F' is a face of F.*

4. *The intersecton of two faces is either a face or is empty.*

The faces of a polyhedron naturally induce a poset and in fact they form a lattice.

3.6.1 Facets

Definition 3.38. *A* facet *of P is an inclusion-wise maximal face distinct from P. Equivalently, a face F of P is a facet if and only if* $\mathbf{dim}(F) = \mathbf{dim}(P) - 1$.

If P is full-dimensional then facets are hyperplanes but in general a facet may be a lower-dimensional object if P is not full-dimensional. Suppose P is a line segement in $n > 2$ dimensions. Then the facets are in fact the two end points of the line segment which are 0 dimensional objects.

One can show the following intuitive theorem about facets.

Theorem 3.39. *Let $P = \{x \mid Ax \leq b\} = \{x \mid A^{=}x \leq b^{=}, A^{+}x \leq b^{+}\}$. If $A^{+}x \leq b^{+}$ is* irredundant *then there is a one to one correspondence between the facets of P and the inequalities in $A^{+}x \leq b^{+}$. That is, for any facet F of P there is a unique irredundant inequality $a_i x \leq b_i$ from $A^{+}x \leq b^{+}$ such that $F = \{x \in P \mid a_i x = b_i\}$, and vice-versa.*

Corollary 3.40. *Each face of P is the intersection of some of the facets of P.*

Corollary 3.41. *A polyhedrom P has no facet if and only if P is an affine subspace.*

Exercise 3.8. Prove the above two corollaries assuming Theorem 3.39.

3.6.2 Minimal Faces and Vertices

A face is inclusion-wise minimal if it does not contain any other face. From Corollary 3.41 and the fact that a face of a polyhedron is a polyhedron the next proposition follows. We leave the proof as an exercise.

Claim 3.6.2. *A face of F of P is minimal if and only if F is an affine subspace.*

Theorem 3.42. *A set F is minimal face of P if and only if $\emptyset \neq F$, $F \subseteq P$ and $F = \{x \mid A'x = b'\}$ for some subsystem $A'x \leq b'$ of $Ax \leq b$.*

Proof. Suppose F is a face and $F = \{x \mid A'x = b'\}$ then by the previous proposition it is minimal. For the converse direction suppose F is a minimal face of P. Since F is a face we can define F as $F = \{x \mid A''x \leq b'', A'x = b'\}$ where $A''x \leq b''$ and $A'x \leq b'$ are two subsystems of $Ax \leq b$. We can assume that $A''x \leq b''$ is as small as possible and therefore, irredundant. If there is an irredundant inequality then F has a facet which contradicts the minimality of F. Hence there can be no irredundant inequality which implies that $F = \{x \mid A'x = b'\}$. ■

Claim 3.6.3. *If F is a minimal face of P then it is a translate of* **linspace**(P)*. Hence all minimal faces have the same dimension.*

Proof. Let F be a minimal face of P where $F = \{x \mid A'x = b'\}$. Then $\{x \mid A'x = 0\} \supseteq \{x \mid Ax = 0\} =$ **linspace**(P). Moreover, the rank of A' is equal to the rank of A. Otherwise there would exist an inequality $a_i x \leq \beta_i$ in $Ax \leq b$ where a_i is not in the linear hull of the rows of A'. Thus we have, $F \subseteq \{x \mid A'x = b', a_i x \leq \beta_i\}$ since F is a face of P but since a_i is not in the linear hull of A' we have strict containement $\{x \mid A'x = b', a_i x \leq \beta_i\} \subset \{x \mid A'x = b'\} = F$, a contradiction. ■

A *vertex* or an *extreme point* of P is a (minimal) face of dimension 0. That is, a single point. A polyhedron is *pointed* if and only if it has a vertex. Note that since all minimal faces have the same dimension, if P has a vertex than all minimal faces are vertices. Since a minimal face F of P is defined by $A'x = b'$ for some subsystem $A'x \leq b'$ of $Ax \leq b$, if a vertex of P is the *unique solution* to $A'x = b'$ then **rank**$(A') = n$. We can then assume that A' has n rows. Vertices are also called *basic feasible solutions*.

Corollary 3.43. *A polyhedron $\{x \mid Ax \leq b, x \in \mathbb{R}^n\}$ has a vertex only if A has a column rank n.*

Exercise 3.9. Prove that $\{x \mid Ax \leq b, x \in \mathbb{R}^n\}$ has a vertex if it is non-empty and A has column rank n.

3.6.3 Decomposition of Polyhedra

Recall that we had earlier stated that,

Theorem 3.44. *Any polyhedron P can be written as $Q + C$ where Q is a convex hull of a finites set of vectors and $C = \{x \mid Ax \leq 0\}$ is the* **charcone** *of P.*

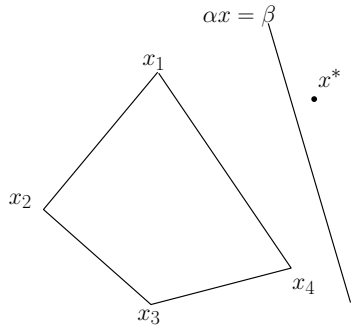
We can say a little bit more now. Given P, let $F_1, F_2, \ldots, F_h$ be its minimal faces. Choose $x_i \in F_i$ arbitrarily. Then $P =$ **convexhull**$(x_1, x_2, \ldots, x_h) + C$. In particular, if P is pointed then $x_1, x_2, \ldots, x_h$ are vertices of P and hence $P =$ **convexhull**(vertices$(P)) + C$.

We will prove the above for polytopes.

Theorem 3.45. *A polytope (bounded polyhedron) is the convex hull of its vertices*

Proof. First observe that a bounded polyhedron is necessarily pointed; otherwise its minimal faces affine subspaces with dimensional at least 1 which implies that it is not bounded. Let $X = \{x_1, x_2, \ldots, x_h\}$ be the vertices of P. Clearly **convexhull**$(X) \subseteq P$. We prove the converse. Suppose $x^* \in P$ does not belong to **convexhull**(X).

Claim 3.6.4. *There exists a hyperplane $\alpha x = \beta$ such that $\alpha x_i \leq \beta \; \forall x_i \in X$ and $\alpha x^* > \beta$.*



Now consider the LP $\max \alpha x$ such that $x \in P$. The set of optimal solutions to this LP is a face of F and the optimum value is strictly larger than β since $x^* \in P$ by assumption and $\alpha x^* > \beta$. By Claim 3.6.4 $X \cap F = \emptyset$. Since F is a face of P, it has a vertex of P since P is pointed. The contradicts that X is the set of all vertices of P. ■

One consequence of the decomposition theorem is the following.

Theorem 3.46. *If $P = \{x \mid Ax \leq b\}$ is pointed then for any $c \neq 0$ the LP*

$$\max cx$$
$$Ax \leq b$$

is either unbounded, or there is a vertex x^* such that x^* is an optimal solution.

Proof. Suppose the given LP has a finite optimum of value β. Then the inequality $\alpha x \leq \beta$ is a supporting hyperplane and hence there is an optimum solution on the face F defined by the intersection of P and his supporting hyperplane. Thus there is a minimal face of F' of P contained in P which contains an optimum solution. Since P is pointed any minimal face is a vertex. ■

Exercise 3.10. Give an example of polyhedron in two dimensions which is not pointed and an objective direction such that the optimum value is finite.

3.7 Complexity of Linear Programming

Recall that LP is the problem

$$\begin{aligned} &\max c^T x \\ &Ax \leq b \\ &x \in \mathbb{R}^n \end{aligned}$$

As a computational problem we assume that the inputs c, A, B are given as rational numbers along with the integer dimension n. Thus the input consists of $n + m \times n + n$ rational numbers. Given an instance I we use **size**(I) to denote the number of bits in the binary representation of I. We use it loosely for other quantities such as numbers, matrices, etc. We have that **size**(I) for an LP instance is,

$$\textbf{size}(c) + \textbf{size}(A) + \textbf{size}(b) \leq (m \times n + 2n)\textbf{size}(L)$$

where L is the largest number in c, A, b.

Linear system solving: We first consider the problem of solving a linear system of the form $Ax = b$. Recall that Gaussian elimination is a standard algorithm that we learn which takes $O(n^3)$ arithmetic operations in the worst-case but it is less obvious that is in fact a true polynomial time algorithm since one also has to worry about the size of the numbers.

Lemma 3.2. *Given a $n \times n$ rational matrix* **size**$(\det(A)) =$ **poly**$($**size**$(A))$.

Proof. First assume that A is integer valued. Let S_n be the set of $n!$ permutations of $[n]$. For a permuation $\sigma \in S_n$ let $\text{sign}(\sigma) \in \{-1, 1\}$ be its sign. A well know characterization of the determinant is:

$$\det(A) = \sum_{\sigma \in S_n} \text{sign}(\sigma) \prod_{i=1}^{n} A_{i\sigma(i)}.$$

From this one can conlude that $|\det(A)| \leq n! \prod_{ij}(|A_{ij}| + 1)$ when all entries of A are integers. Therefore

$$\log(|\det(A)|) = O(n \log n) + O(\sum_{ij} \log(|A_{ij}| + 1)).$$

This shows the desired claim when A is integer valued. For the rational case Let $A = (\frac{p_{ij}}{q_{ij}})_{i,j=1}^{n}$ where for each i, j the values of p_{ij} and q_{ij} are relatively prime and $q_{ij} > 0$. Let q be the least common multiple of q_{ij} values. Then $q \leq \prod_{i,j} q_{i,j}$. Let $A' = qA$. Then A' has integer values and $\det(A') = \det(A)/q^n$. We leave it as an exercise to see that $\det(A)$ can be expressed as a rational number c/d where c and d are of size polynomial in the size of A. ■

Corollary 3.47. *If A has an inverse then* $\mathbf{size}(A^{-1}) = \mathbf{poly}(\mathbf{size}(A))$.

Corollary 3.48. *If $Ax = b$ has a feasible solution then there exists a solution x^* such that* $\mathbf{size}(x^*) = \mathbf{poly}(\mathbf{size}(A, b))$.

Proof. Follows by applying Cramer's rule for the solution of linear systems. ■

Using Gaussian elimination carefully and the above we can show the following. The reason to be careful is that we need to ensure that the size of the number in the intermediate calculations are also of size polynomial in the input size.

Theorem 3.49. *There is a polynomial time algorithm that given a rational linear system $Ax = b$, decides whether it has a feasible solution and outputs one if it has. Moreover, one can determine if A has a unique feasible solution.*

Now that we have understood the setting of linear system solving we consider LP. First, we consider we consider the decision problem of whether $Ax \leq b$ has a feasible solution.

Theorem 3.50. *If a linear system $Ax \leq b$ has a feasible solution then there exists a solution x^* such that* $\mathbf{size}(x^*) = \mathbf{poly}(\mathbf{size}(A, b))$.

Proof. Consider a minimal face F of $P = \{x \mid Ax \leq b\}$. We have seen that $F = \{x \mid A'x = b'\}$ for some subsystem $A'x \leq b'$ of $Ax \leq b$. By Corollary 3.48 $A'x = b'$ has a solution of size $\mathbf{poly}(\mathbf{size}(A', b'))$. ■

Corollary 3.51. *The problem of deciding whether $\{x \mid Ax \leq b\}$ is* non-empty *is in* **NP**. *The problem of deciding whether $\{x \mid Ax \leq b\}$ is* empty *os om* $\mathbf{co} - \mathbf{NP}$.

Proof. To prove that $Ax \leq b$ has a feasible solution it suffices to exhibit a vector $z \in \mathbb{R}^n$ and there is an efficient verifier that checks that $Az \leq b$. The complexity of this checking depends on the size of z and from the preceding theorem we know that there exists a z whose size is polynomial in the size of the input A, b. This proves that the problem of checking non-emptiness is in **NP**.

By Farkas lemma, if $Ax \leq b$ is empty only if $\exists y \geq 0$ such that $y^T A = 0$ and $y^T b = -1$. Thus, checking if $Ax \leq b$ is empty is equivalent to checking if the system $y^T A = 0, y^T b = -1, y \geq 0$ is feasible. ■

Thus we have seen that deciding whether $Ax \leq b$ is feasible is in $\mathbf{NP} \cap \mathbf{coNP}$.

Now consider the optimization problem.

$$\max \alpha x$$
$$Ax \leq b$$

A natural decision problem associated with the above optimization problem is to decide if the optimum value is at least some given rational number β. Thus the input to this problem is c, A, b, β.

Exercise 3.11. Prove that the preceding decision problem is in **NP** $\cap$ **coNP**.

Another useful fact is the following.

Lemma 3.3. *If the optimum value of the LP* $\max\{c^T x \mid Ax \leq b\}$ *is finite then the optimum value is of size polynomial in the input size.*

Proof sketch. If it is finite then the primal and the dual have finite values. We consider the system of inequalities.

$$cx = yb$$

$$Ax \leq b$$

$$yA = c$$

$$y \geq 0.$$

A solution to this system has size at most **size**(A, b, c) by Lemma 3.2. ■

Exercise 3.12. Show that the decision problem of deciding whether the LP $\max\{c^T x \mid Ax \leq b\}$ is unbounded is in **NP** $\cap$ **coNP**.

The optimization problem for

$$\max c^T x$$
$$Ax \leq b$$

requires an algorithm that correctly outputs one of the following

1. $Ax \leq b$ is infeasible
2. the optimal value is unbounded
3. a solution x^* such that $c^T x^*$ is the optimum value

A related search problem is given $Ax \leq b$ either output that $Ax \leq b$ is infeasible or a solution x^* such that $Ax^* \leq b$.

Lemma 3.4. *Prove that the optimization problem and the search problem are polynomial time equivalent.*

Proof sketch: It is to see that the optimization problem is more genereal than the search problem. The interesting direction is to reduce optimization to search. Given an LP $\max\{c^T x \mid Ax \leq b\}$ we first check if it in feasible via the search procedure. Then we check whether the value is unbounded by considering feasibility of the dual LP (note that the dual is of size polynomial in the primal). If both primal and dual are feasible then we can use the idea to combine the primal and dual into a single system of inequalties and any feasible solution to the combined system gives an optimum solution to the primal. ■

3.8 Polynomial-time Algorithms for LP

The simplex method is a well-known and practical method for LP solving. Originally due to Dantzig in the US and others in USSR, it has several variants and hence we refer to it as a method. It continues to be the method of choice in many software packages. However all known variants have been shown to run in exponential time in the worst case. Whether there is a variant that runs in (strongly) polynomial-time is a major open problem.

Khachiyan's Ellipsoid algorithm was the first polynomial-time algorithm for LP. Although an impractical algorithm, it had (and continues to have) a major theoretical impact. It showed that one does not need the full system $Ax \leq b$ in advance. For a linear system $Ax \leq b$ there is a solution x^* such that x^* is a solution to $A'x \leq b'$ for some subsystem $A'x \leq b'$ such that rank of A' is equal to the rank of A. This implies hat A' can be chosen to have at most n rows. Thus the size of a solution depends only on n and the maximum sized entry in A. As we discussed in the first chapter, the Ellipsoid method can be used to show the equivalence of separation and optimization.

Subsequently, Karmarkar described a different polynomial-time algorithm based on the interior point method. This is much more useful in practice, especially for certain large linear programs and can beat the well-known simplex method. However, we note that Karmarkar's algorithm works only for explicitly given LPs while the Ellipsoid method works for a more general class of implicit LPs which come up in combinatorial optimization from a theoretical point of view.

The known polynomial-time algorithms for LP are not strongly-polynomial. That is, the number of arithmetic operations depends on bit size of the numbers in the input (and not just on the combinatorial parameters n, m). A major open problem is whether there exists a strongly-polynomial-time algorithm for LP. Tardós [61] showed that the running time for explicitly given LPs can be made polynomial in n, m and $size(A)$; in other words the number of arithmetic operations do not depend on the bit size of the objective vector c and the

right hand side vector b. In several combinatorial applications the entries in A are small (typically in $\{0, 1, -1\}$) and hence her result automatically implies a strongly polynomial time algorithm for all such LPs which are given explicitly. She obtained the first strongly polynomial time algorithm for the fundamental problem of min-cost flow via the LP technique [62] before other more practical and direct methods were found [29]. There are many technical developments since then to extend and improve these results.

Schrijver's book [58] is a methodical and technical introduction covering structural and algorithmic aspects. The book by Grötschel, Lovász and Schrijver [30] covers the Ellipsoid method and its implications for combinatorial optimization. There are many fine books on linear programming. The book by Matousek and Gärtner [49] is an accessible and recent introduction.

Chapter 4

Integer Programming and Integer Polyhedra

Many discrete optimization problems are naturally modeled as an integer (linear) programming (ILP) problem. An ILP problem is like an LP problem in that it has a linear object and a set of linear constraints but we require the solution to be an integer vector rather than a real-valued vector.

$$\begin{aligned} &\max cx \\ &Ax \leq b \\ &x \in \mathbb{Z}^n \end{aligned} \tag{4.1}$$

It is easy to show that ILP is NP-hard via a reduction from say SAT. The decision version of ILP is the following: Given rational matrix A and rational vector b, does $Ax \leq b$ have an integral solution x?

Theorem 4.1. *Decision version of ILP is in NP, and hence it is NP-Complete.*

To prove that ILP is in NP requires technical work: we need to show that if there is integer vector in $Ax \leq b$ then there is one whose size is polynomial in $size(A, b)$. We proved this for LP by appealing to the fact that if there is a solution to a linear system $Ax = b$, where A, b have rational entries, then there is one whose size is polynomial $size(A, b)$. It is harder to prove it for integer vectors and requires some understanding of the algorithmic aspects of lattices and the geometry of numbers [47, 58].

A special case of interest is when the number of variables, n, is a fixed constant but the number of constraints, m, is part of the input. The following theorem is known, first established by Lenstra [44] in 1983.

Theorem 4.2. *For each fixed n, there is a polynomial time algorithm for ILP in n variables.*

4.1 Integer Polyhedra

Given a rational polyhedron $P = \{x|Ax \leq b\}$, we use P_I to denote the convex hull of all the integer vectors in P; this is called the integer hull of P. It is not hard to see that if P is a polytope then P_I is also a polytope. A technically more involved result is the following.

Theorem 4.3. *For any rational polyhedron P, P_I is a polyhedron.*

Definition 4.4. *A rational polyhedron P is an integer polyhedron if and only if $P = P_I$.*

Theorem 4.5. *The following are equivalent:*

1. *$P = P_I$ i.e., P is integer polyhedron.*
2. *Every face of P has an integer vector.*
3. *Every minimal face of P has an integer vector.*
4. *$\max\{cx \mid x \in P\}$ is attained by an integer vector when the optimum value is finite.*

Proof. (i)$\Longrightarrow$(ii): Let F be a face of P, then $F = P \cap H$, where H is a supporting hyperplane, and let $x \in F$. From $P = P_I$, x is a convex combination of integral points in P. All the points in this convex combination must belong to H and thus to F.
(ii)$\Longrightarrow$(iii): it is direct from (ii).
(iii)$\Longrightarrow$(iv): Let $\delta = \max\{cx : x \in P\} < +\infty$, then $F = \{x \in P : cx = \delta\}$ is a face of P, which has an integer vector from (iii).
(iv)$\Longrightarrow$(i): Suppose there is a vector $y \in P \setminus P_I$. Then there is an inequality $\alpha x \leq \beta$ valid for P_I while $\alpha y > \beta$ (a hyperplane separating y and P_I). It follows that $\max\{\alpha x \mid x \in P_I\} \leq \beta$ while $\max\{\alpha x \mid x \in P\} > \beta$ since $y \in P \setminus P_I$. Then (iv) is violated for $c = \alpha$. ∎

Corollary 4.6. *A pointed polyhedron P is an integer polyhedron iff each vertex of P is an integer vector.*

Another useful theorem that characterizes integral polyhedra, in full generality due to Edmons and Giles is the following.

Theorem 4.7. *A rational polyhedron P is integral if and only if $\max\{cx \mid Ax \leq b\}$ is an integer for each integral vector c for which the maximum is finite.*

4.2 Integer Polyhedra and Combinatorial Optimization

Combinatorial optimization can typically be modeled as an ILP problem in a natural fashion although there is not necessarily a unique way to model them. Say the ILP is of the form $\max\{x \mid Ax \leq b, x \in \mathbb{Z}^n\}$. By lucky coincidence if the underlying LP $\max\{x \mid Ax \leq b, x \in \mathbb{R}^n\}$ is an integer polyhedron then we can solve the ILP in polynomial time provided we can solve the LP in polynomial time! If the LP has a polynomial number of constraints then it can be solved in polynomial time. In other cases, even when the LP has an exponential number of constraints, we may still be able to solve it in polynomial time via the Ellipsoid method if the LP admits an efficient separation oracle.

Why should such a coincidence happen? As we will see shortly there is a class of problems which exhibit this coincidence via the notion of totally unimodular matrices. As we discussed in the introductory lecture we can invert the question as follows. Consider a typical combinatorial optimization problem. Consider an instances I of size n and let $\mathcal{S}(I)$ be the set of feasible solutions. One can often easily define an *implicit* integer polytope $P(I)$ as the convex hull of the characteristic vectors of the set $\mathcal{S}(I)$. For example let the problem correspond to finding maximum weight matchings in a graph $G = (V, E)$. We can define $\mathcal{S}(G)$ to be the set of all matchings of G and each matching M corresponds to a $|E|$-dimensional $\{0, 1\}$ vector χ_M (the characteristic vector of M). The convex hull of these vectors forms the matching polytope of G. It is clearly an integer polytope by definition. Since it is a rational polytope we can seek to optimize over this polytope which would enable us to solve the maximum weight matching problem; the Ellipsoid method shows that optimization over this polytope is polynomial-time equivalent to separating over this polytope. Thus, integer polyhedra naturally arise from a combinatorial optimizaton problem in an implicit fashion. One can use combinatorial and polyhedral insights to efficiently solve the underlying LP. And any algorithm to efficiently solve the optimization problem implies understanding of this polytope.

Chapter 5

Totally Unimodular Matrices and Applications[1]

Totally Unimodular Matrices give rise to integer polyhedra with several fundamental applications in combinatorial optimization.

Definition 5.1. *A matrix A is totally unimodular (TUM) if the determinant of each square submatrix of A is in $\{0, 1, -1\}$.*

The definition automatically implies that each entry of a TUM is in $\{0, 1, -1\}$.

Claim 5.0.1. *If A is TUM and U is a non-singular square submatrix of A, then U^{-1} is a matrix with entries in $\{0, 1, -1\}$.*

Proof. $U^{-1} = \dfrac{\text{adj}\, A}{det(U)}$ where $\text{adj}\, U$ is the adjoint matrix of U. Recall that $\text{adj}\, U = C^T$ where C is cofactor matrix of U. $C_{i,j}$ is the determinant of the sub-matrix of U obtained by removing row i and column j from U. Since A is TUM, $C_{i,j} \in \{0, -1, 1\}$ for all i, j and $\det(U) \in \{-1, 1\}$ since U is non-singular. Therefore, U^{-1} is an integer matrix with entries in $\{0, -1, 1\}$. ∎

Theorem 5.2. *If A is TUM then for all integral vectors b, the polyhedron $P = \{x \mid Ax \leq b\}$ is an integer polyhedron.*

Proof. We will assume without loss of generality that P is non-empty. Consider any minimal face F of P. $F = \{x \mid A'x = b'\}$ for some subsystem $A'x \leq b'$ of $Ax \leq b$ where A' having full row rank. F is non-empty since P is non-empty. Consider the system $A'x = b'$ which has a feasible solution and since it has full row rank, A' has $m' \leq n$ rows. Then $A' = [U \quad V]$, where U is a $m' \times m'$

[1]Based on notes scribed by GuoJun Qi and Siva Theja Maguluri from 2010.

matrix of full row and column rank after potentially rearranging columns of A'. Note that U is a is a square submatrix of A with full rank and hence it is invertible. Therefore $A'x = b'$ has a solution $y = U^{-1}b$ and by Claim 5.0.1 U^{-1} is an integral matrix and hence y is an integer vector when b is an integer vector. Thus every minimal face of P has an integer vector which implies that P is an integer polyhedron. ■

The following simple claim is useful in various proofs on TU matrices. We leave the proof as an exercise.

Claim 5.0.2. *A is TUM $\iff$ A' obtained by multiplying any row or column by -1 is TUM.*

We claim several important corollaries.

Corollary 5.3. *If A is TUM then for all integral vector a, b, c, d, the polyhedron $\{x \mid u \leq x \leq b, c \leq Ax \leq d\}$ is an integer polyhedron.*

Proof. Suppose A is TUM. We claim that the matrix $A' \begin{bmatrix} I \\ -I \\ A \\ -A \end{bmatrix}$ is also TUM. Consider any sub-matrix U of A'. We need to show that $\det(U) \in \{0, -1, 1\}$. This can be easily proven by using the formula for computing the determinant via cofactors and using the preceding claim as needed. ■

The constraints $a \leq x \leq b$ are called *box constraints* thus total unimodularity implies that adding integral box constraints retains integrality of the polyhedron which is very useful.

Claim 5.0.3. *A is TUM $\iff$ A^T is TUM.*

Corollary 5.4. *Suppose A is TUM and b, c are integral vectors. Then $\max\{cx \mid Ax \leq b, x \geq 0\} = \min\{yb \mid yA \leq c, y \geq 0\}$ are attained by integral vectors x^* and y^*, if the optimum value is finite.*

Proof. The polyhedron $\{y \mid y \geq 0, yA \leq c\}$ is integral since A^T is TUM and also $\begin{bmatrix} A^T \\ -I \end{bmatrix}$. Thus, the primal and dual LPs are integer polyhedrons and hence they have integer optima whenever the optimum value is finite. ■

There are several characterizations of TUM matrices. We give a few useful ones below. See [58] (Chapter 19) for a proof.

Theorem 5.5. *Let A be a matrix with entries in $\{0, +1, -1\}$. Then the followings are equivalent.*

1. *A is TUM.*
2. *For all integral vector b, $\{x \mid Ax \leq b, x \geq 0\}$ is an integer polyhedron.*
3. *For all integral vectors a, b, c, d, $\{x \mid a \leq x \leq b, c \leq Ax \leq d\}$ is an integer polyhedron.*
4. *Each collection of column S of A can be split into two sets S_1 and S_2 such that the sum of columns in S_1 minus the sum of columns in S_2 is a vector with entries in $\{0, +1, -1\}$.*
5. *Each nonsingular submatrix of A has a row with an odd number of nonzero components.*
6. *No square submatrix of A has determinant $+2$ or -2.*

$(i) \iff (ii)$ is the Hoffman-Kruskal's theorem. $(ii) \implies (iii)$ follows from the fact that A is TUM $\implies \begin{bmatrix} I \\ -I \\ A \\ -A \end{bmatrix}$ is TUM. $(i) \iff (iv)$ is Ghouila-Houri's theorem.

Solving LPs with TUM matrices: LPs with TUM matrices can be solved in strong polynomial time via Tardös's algorithm [61, 62]. The generic result is useful to know from a theoretical point of view; for the typical applications there are problem-specific combinatorial algorithms that are much faster and practical.

5.1 Examples and Network Matrices

Several important matrices that arise in combinatorial optimization are TUM.

Example 1: Bipartite Graphs. Let $G = (V, E)$ an undirected graph. Let M be the $\{0, 1\}$ edge-vertex incidence matrix defined as follows. M has $|E|$ rows, one for each edge and $|V|$ columns, one for each vertex. $M_{e,v} = 1$ if e is incident to v otherwise it is 0. The claim is that M is TUM iff G is bipartite.

To see bipartiteness is needed, consider the matrix $\begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$ for a triangle which is an odd cycle. Its determinant is 2.

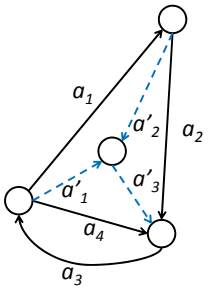


Figure 5.1: Network matrix is defined by a directed tree (dotted edges) and a directed graph on the same vertex set.

Exercise 5.1. Show that edge-vertex adjacency matrix of any odd cycle has determinant 2.

Example 2: Directed Graphs. Let $D = (V, A)$ be a directed graph. Let M be an $|E| \times |V|$ arc-vertex adjacency matrix defined as

$$M_{a,v} = \begin{cases} 0, \text{ if } a \text{ is not incident to } v \\ +1, \text{ if } a \text{ enters } v \\ -1, \text{ if } a \text{ leaves } v \end{cases} \tag{5.1}$$

M is TUM. This was first observed by Poincaré [1900].

Example 3: Consecutive 1's: A matrix A with is a consecutive 1's matrix if it is a matrix with entries in $\{0, 1\}$ such that in each row the 1's are in a consecutive block. This naturally arises as an incidence matrix of a collection of intervals and a set of points on the real line. One can also consider consecutive 1's in each column and those are also TUM.

The above three claims of matrices are special cases of network matrices (due to Tutte).

Definition 5.6. *A network matrix is defined from a directed graph $D = (V, A)$ and a directed tree $T = (V, A')$ on the same vertex set V. The matrix M is $|A'| \times |A|$ matrix such that for $a = (u, v) \in A$ and $a' \in A'$*

$$M_{a,a'} = \begin{cases} 0, \textit{ if the unique path from } u \to v \textit{ in } T \textit{ does not contain } a' \\ +1, \textit{ if the unique path from } u \to v \textit{ in } T \textit{ passes through } a' \textit{ in forward direction} \\ -1, \textit{ if the unique path from } u \to v \textit{ in } T \textit{ passes through } a' \textit{ in backward direction} \end{cases}$$

The network matrix corresponding to the directed graph and the tree in Figure 5.1 is given below. The dotted edge is T, and the solid edge is D.

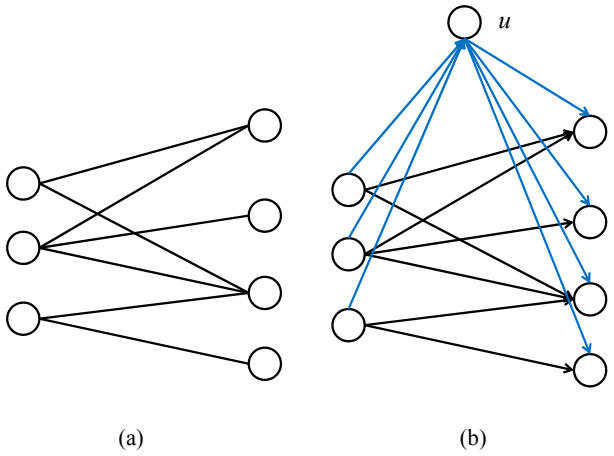


Figure 5.2

$$M = \begin{array}{c} \\ a'_1 \\ a'_2 \\ a'_3 \end{array} \begin{array}{c} \begin{array}{cccc} a_1 & a_2 & a_3 & a_4 \end{array} \\ \begin{bmatrix} 1 & 0 & -1 & 1 \\ -1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 1 \end{bmatrix} \end{array}$$

Theorem 5.7 (Tutte)**.** *Every network matrix is TUM.*

We will prove this later. First we show that the previous examples can be cast as special cases of network matrices.

Bipartite graphs. Say $G = \{X \cup Y, E\}$ as in Figure 5.2(a). One can see that edge-vertex adjacency matrix of G as the network matrix induced by a directed graph $G = (X \cup Y \cup \{u\}, A)$ where u is a new vertex and A is the set of arcs defined by orientating the edges of G from X to Y. $T = (X \cup Y \cup \{u\}, A')$ where $A' = \{(v, u) | v \in X\} \cup \{(u, v) | v \in Y\}$ as in Figure 5.2(b).

Directed graphs. Suppose $D = (V, A)$ is a directed graph. Consider the network matrix induced by $D = (V \cup \{u\}, A)$ and $T = (V \cup \{u\}, A')$ where u is a new vertex and where $A' = \{(v, u) | v \in V\}$.

Consecutive 1's matrix. Let A be a consecutive 1's matrix with m rows and n columns. Assume for simplicity that each row has at least one 1 and let ℓ_i and r_i be the left most and right most columns of the consecutive block of 1's in row i. Let $V = \{1, 2, \ldots, n\}$. Consider $T = (V, A')$ where $A' = \{(i, i+1) \mid 1 \leq i < n\}$ and $D = (V, A)$ where $A = \{(\ell_i, r_i) \mid 1 \leq i \leq n\}$. It is easy to see that A is the network matrix defined by T and A.

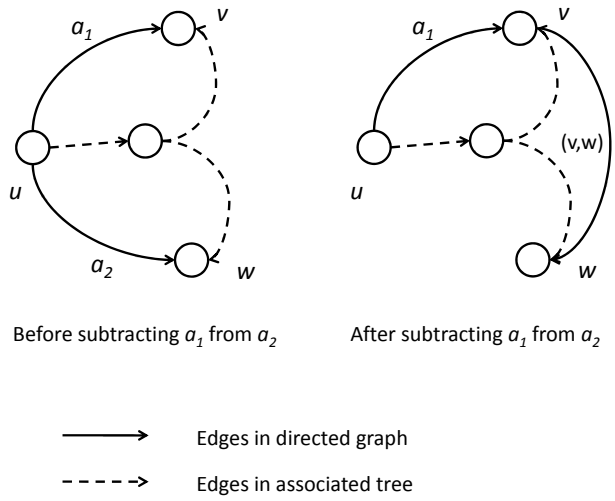


Figure 5.3

Now we prove that every network matrix is TUM. We need a preliminary lemma.

Lemma 5.1. *Every submatrix M' of a network matrix M is also a network matrix.*

Proof. If M is a network matrix, defined by $D = (V, A)$ and $T = (V, A')$, then removing a column in M corresponds to removing an arc $a \in A$. Removing a row corresponds to identifying/contracting the end points of an arc a' in T. ■

Claim 5.1.1. *A is TUM $\iff$ A' obtained by multiplying any row or column by -1 is TUM.*

Corollary 5.8. *If M is a network matrix, M is TUM $\iff$ M' is TUM where M' is obtained by reversing an arc of either T or D.*

Proof of Theorem 5.7. By Lemma 5.1, it suffices to show that any square network matrix C has determinant in $\{0, +1, -1\}$. Let C be a $k \times k$ network matrix defined by $D = (V, A)$ and $T = (V, A')$. We prove by induction on k that $det(C) \in \{0, 1, -1\}$. Base case with $k = 1$ is trivial since entries of C are in $\{0, 1, -1\}$.

Let $a' \in A'$ be an arc incident to a leaf u in T. By reorienting the arcs of T, we will assume that a' leaves u and moreover all arcs A incident to u leave u (see Corollary 5.8).

Let $a_1, a_2, \cdots, a_h$ be arcs in A leaving u (If no arcs are incident to u then $det(C) = 0$). Assume without loss of generality that a' is the first row of C and that $a_1, a_2, \cdots, a_h$ are the first h columns of C.

Claim 5.1.2. *Let C' be obtained by subtracting column a_1 from column a_2. C' is the network matrix for $T = (V, A')$ and $D = (V, A - a_2 + (v, w))$ where $a_1 = (u, v)$ and $a_2 = (u, w)$.*

We leave the proof of the above as an exercise — see Figure 5.3.

Let C'' be the matrix obtained by subtracting column of a_1 from each of $a_2, \cdots, a_h$. From the above claim, it is also a network matrix. Moreover, $det(C'') = det(C)$ since determinant is preserved by these operations. Now C'' has 1 in the first row in column one (corresponding to a_1) and 0's in all other columns. Therefore, $det(C'') \in \{0, +1, -1\}$ by expanding along the first row and using induction for the submatrix of C'' consisting of columns 2 to k and rows 2 to k. ■

Some natural questions on TUM matrices are the following.

(i) Are there TUM matrices that are not a network matrix (or its transpose)?

(ii) Given a matrix A, can one check efficiently whether it is a TUM matrix?

The answer to (i) is negative as shown by the following two matrices given by Hoffman[1960]

$$\begin{bmatrix} 1 & -1 & 0 & 0 & -1 \\ -1 & 1 & -1 & 0 & 0 \\ 0 & -1 & 1 & -1 & 0 \\ 0 & 0 & -1 & 1 & -1 \\ -1 & 0 & 0 & -1 & 1 \end{bmatrix}$$

and Bixby[1977].

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Amazingly, in some sense, these are the only two exceptions. Seymour, in a deep and difficult technical theorem, showed via matroid theory methods that any TUM matrix can be obtained by "gluing" together network matrices and the above two matrices via some standard operations that preserve total unimodularity. His descomposition theorem also led to a polynomial time algorithm for checking if a given matrix is TUM. There was an earlier polynomial time algorithm to check if a given matrix is a network matrix. See [58] (Chapters 20 and 21) for details.

5.2 Integer Decomposition Property

A polyhedron P has the *integer decomposition property* if $\forall$ integers $k \geq 1$ and $x \in P$, kx is integral implies $kx = x_1 + x_2 + \ldots + x_k$ for integral vectors $x_1, \ldots, x_k$ in P. Baum and Trotter showed the following:

Theorem 5.9 (Baum and Trotter)**.** *A matrix A is TUM iff $P = \{x \mid x \geq 0, Ax \leq b\}$ has the integer decomposition property for all integral verctors b.*

Proof. We show one direction, the one useful for applications. Suppose A is TUM, consider $P = \{x \mid x \geq 0, Ax \leq b\}$. Let $y = kx^*$ be an integral vector where $x^* \in P$. We prove by induction on k that $y = x_1 + x_2 + \ldots . x_k$ for integral vectors $x_1, x_2, \ldots, x_k$ in P.

Base case for $k = 1$ is trivial.

For $k \geq 2$, consider the polyhedron $P' = \{x \mid 0 \leq x \leq y; Ay - kb + b \leq Ax \leq b\}$. P' is an integral polyhedron since A is TUM and $Ay - kb + b$ and b are integral. The vector $x^* \in P'$ and hence P' is not empty. Hence there is an integral vector $x_1 \in P'$. Moreover $y' = y - x_1$ is integral and $y' \geq 0$, $Ay' \leq (k-1)b$.

By induction $y' = x_2 + \ldots + x_{k-1}$ where $x_2, \ldots, x_{k-1}$ are integral vectors in P. $y = x_1 + \ldots + x_k$ is the desired combination for y. ∎

Remark 5.1. A polyhedron P may have the integer decomposition property even if the constraint matrix A is not TUM. The point about TUM matrices is that the property holds for all integral right hand side vectors b.

5.3 Applications of TUM Matrices

We saw that network matrices are TUM and that some matrices arising from graphs are network matrices. TUM matrices give rise to integral polyhedra, and in particular, simultaneously to the primal and dual in the following when A is TUM and c, b are integral vectors.

$$\max\{cx \mid x \geq 0, Ax \leq b\} = \min\{yb \mid y \geq 0, yA \geq c\}$$

We can derive some min-max results and algorithms as a consequence.

5.3.1 Bipartite Graph Matchings

Let $G = (V, E)$ be a bipartite graph with $V = V_1 \uplus V_2$ as the bipartition. We can write an integer program for the maximum cardinality matching problem as

$$\begin{aligned} \max \quad & \sum_{e \in E} x(e) \\ x(\delta(u)) &\leq 1 \; \forall u \in V \\ x(e) &\geq 0 \; \forall e \in E \\ x &\in \mathbb{Z} \end{aligned}$$

We observe that this is a ILP problem $\max\{1 \cdot x \mid Mx \leq 1, x \geq 0, x \in \mathbb{Z}\}$ where M is the edge-vertex incidence matrix of G. Since M is TUM, we can drop the integrality constraint and solve the *linear program* $\max\{1 \cdot x \mid Mx \leq 1, x \geq 0\}$ since $\{x \mid Mx \leq 1, x \geq 0\}$ is an integral polyhedron. The dual of the above LP is

$$\begin{aligned} \min \sum_{u \in V} y(u) & \\ y(u) + y(v) &\geq 1 \quad uv \in E \\ y &\geq 0 \end{aligned}$$

in other words $\min\{y \cdot 1 | yM \geq 1, y \geq 0\}$ which is is also an integral polyhedron since M^T is TUM. We note that this is the min-cardinality vertex cover problem

Note that the primal LP is a polytope and hence has a finite optimum solution. By duality, and integrality of the polyhedra, we get that both primal and dual have integral optimum solutions x^* and y^* such that $1 \cdot x^* = y^* \cdot 1$. We get as an immediate corollary König's Theorem.

Theorem 5.10 (König)**.** *In a bipartite graph the cardinality of a maximum matching is equal to the cardinality of a minimum vertex cover.*

Also, by poly-time solvability of linear programming, there is a polynomial time algorithm for maximum matching and minimum vertex cover in bipartite graphs, and also their weighted versions. Note that we have much more efficient combinatorial algorithms for these problems.

Weighted case: Consider maximum weight bipartite matching where $w : E \to \mathbb{Z}$ are given edge weights. Since the polytope is integral we can solve the maximum weight problem by simply solving the underlying LP relaxation. Now consider the case when the weights are integral. We then have $\max\{wx \mid Mx \leq 1, x \geq 0\} = \min\{y \cdot 1 \mid yM \geq w, y \geq 0\}$ has integer primal and dual solutions x^* and $y^* \cdot 1$ since w is integral. For the primal x^* corresponds to a maximum w-weight matching. In the dual, via complementary slackness, we have

$$y^*(u) + y^*(v) = w(v)$$

for all $x^*(uv) > 0$ The dual values y^* can be thought of as a solution to a generalization of vertex cover: we wish to cover each edge $e \in E$ $w(e)$ times and we are allowed to take integer copies of vertices. The duality relation implies that max w-weight matching value is equal to the min w-vertex cover value and this is a generalization of König's theorem and is the so-called Egervary theorem. Sometimes the two theorems are jointly called König-Egervary theorem.

Perfect matchings: A matching M in a graph $G = (V, E)$ is said to *saturate* a subset $S \subseteq V$ of vertices if $M \cap S = S$ (that is every vertex in S is matched). M is perfect if M saturates V. A well-known theorem in graph theory is Hall's marriage theorem which gives a necessary and sufficient condition for the existence of a perfect matching. It is typically given in terms of a matching that saturates one side of the bipartite graph.

Theorem 5.11 (Hall)**.** *Let $G = (V, E)$ be a bipartite graph with X, Y as the vertex sets of the bipartition. Then there is a matching that saturates X iff $|N(S)| \geq |S| \forall S \subseteq X$ where $N(S)$ is the set of neighbors of S.*

Exercise 5.2. Derive Hall's theorem from König's Theorem.

We state a more general version of Hall's theorem below.

Theorem 5.12. *Let $G = (X \cup Y, E)$ be a bipartite graph. Let $R \subseteq U$. Then there is a matching that covers R iff there exists a matching M that covers $R \cap X$ and a matching that covers $R \cap Y$. Therefore, a matching covers R iff $|N(S)| \geq |S| \forall S \subseteq R \cap X$ and $\forall S \subseteq R \cap Y$.*

Exercise 5.3. Prove above theorem.

Matching and Perfect Matching Polytopes: The polytope $\{x \mid Mx \leq 1, x \geq 0\}$ is the convex hull of the characteristic vectors of the matchings in G and $\{x \mid Mx = 1, x \geq 0\}$ is the convex hull of the perfect matchings of G. One easy consequence is the following theorem.

Theorem 5.13 (Birkhoff - Von Neumann)**.** *Let A be a $n \times n$ doubly stochastic matrix. Then A can be written as a convex combination of permutation matrices.*

A doubly stochastic matrix is a square non-negative matrix in which each row and column sum is 1. A permutation matrix is a square $\{0, 1\}$ matrix that has a single 1 in each row and column. Each permulation matrix corresponds to a permutation σ in S_n, the set of all permutations on an n-element set.

Exercise 5.4. Prove the above theorem using the perfect matching polytope description for bipartite graphs. Show that one can find a convex combination that has at most m matrices where m is the number of non-zeroes in the given doubly-stochastic matrix.

Min-cost perfect matching: In min-cost perfect matching we are given a graph $G = (V, E)$ and edge costs $c : E \to \mathbb{Z}$ (costs can be negative) and the goal is to find a minimum-cost perfect matching. It is easy to see that for bipartite graphs we can solve this problem via the LP $\min\{cx \mid Mx = 1, x \geq 0\}$.

Exercise 5.5. Show that we can assume that all edge costs are non-negative when considering min-cost perfect matching. Show that min-cost perfect matching can be efficiently reduced to max-weight matching and vice-versa.

***b*-matchings:** b-matchings generalize matchings. Given an integral vector $b : V \to \mathbb{Z}^+$, a b-matching is a set of edges such that the number of edges incident to a vertex v is at most $b(v)$. From the fact that the matrix M is TUM, one can obtain various properties of b-matchings by observing that the polyhedron

$$\begin{aligned} Mx &\leq b \\ x &\geq 0 \end{aligned}$$

is integral for integral b.

5.3.2 Single Commodity Flows and Cuts

We can derive various useful and known facts about single commodity flows and cuts using the fact that the directed graph arc-vertex incidence matrix is TUM.

Consider the s-t maximum-flow problem in a directed graph $D = (V, A)$ with capacities $c : A \to \mathbb{R}^+$. We can express the maximum flow problem as an LP with variables $x(a)$ for flow on arc a.

$$\begin{aligned} \max \sum_{a \in \delta^+(s)} x(a) - \sum_{a \in \delta^-(s)} x(a) & \\ \sum_{a \in \delta^+(v)} x(a) - \sum_{a \in \delta^-(v)} x(a) &= 0 \quad \forall v \in V - \{s, t\} \\ x(a) &\leq c(a) \quad \forall a \in A \\ x(a) &\geq 0 \quad \forall a \in A \end{aligned}$$

Note that the polyhedron defined by the above is of the form $\{x \mid M'x = 0, 0 \leq x \leq c\}$ where M' is the arc-vertex incidence matrix of D with the columns corresponding to s, t removed. M' is a submatrix of M, the arc-vertex incidence matrix of D which is TUM, and hence M' is also TUM. Therefore, the polyhedron above is integral for integral c. One immediate corollary is that for integral capacities, there is a maximum flow that is integral. We now derive the maxflow-mincut theorem as a consequence of the total unimodularity of M.

The dual to the maximum-flow LP above has two sets of variables. $y(a), a \in A$ for the capacity constraints and $z(v), v \in V - \{s,t\}$ for the flow conservation constraints. We let $w(a)$ be the weight vector of the primal. Note that

$$w(a) = \begin{cases} 1 & \text{if } a = (s,v) \text{ for some } v \in V \\ -1 & \text{if } a = (v,s) \text{ for some } v \in V \\ 0 & \text{otherwise} \end{cases}$$

For simplicity assume that there is no arc (s,t) or (t,s). If there are such arcs we can sub-divided them with an internal node; this assumption is only to simplify the description of the dual. Then the dual is:

$$\begin{aligned} \min \sum_{a \in A} c(a)y(a) & & & & \\ z(u) - z(v) + y(u,v) &\geq 0 & (u,v) \in A & \quad \{u,v\} \cap \{s,t\} = \emptyset \\ -z(v) + y(s,v) &\geq 1 & \forall (s,v) \in A & \\ z(v) + y(s,v) &\geq -1 & \forall (v,s) \in A & \\ z(v) + y(v,t) &\geq 0 & \forall (v,t) \in A & \\ -z(v) + y(t,v) &\geq 0 & \forall (t,v) \in A & \\ y &\geq 0 & & \end{aligned}$$

Note that z are unconstrained variables. In matrix form, the primal is $\max\{wx \mid M'x = 0, 0 \leq x \leq c\}$ and the dual is $\min\{yc \mid y \geq 0; \exists z : y + zM' \geq w^T\}$. Since w is integral and M' is TUM, dual is an integral polyhedron. Primal is bounded polyhedron and hence primal and dual have optimal solution x^* and (y^*, z^*) such that $wx^* = y^*c$ and y^*, z^* is integral.

We give two related but slightly different proofs that prove that there is an s-t mincut whose cost is at most $\sum_{a \in A} c(a)y^*(a)$.

In the first proof consider $A' = \{a \in A \mid y^*(a) \geq 1\}$. It is easy to see that $c(A') = \sum_{a \in A'} = c(a) \leq \sum_{a \in A} c(a)y^*(a)$. We now argue that there is no path in $G - A'$ which implies that A' induces an s-t cut. Suppose there is a path $p = s, v_1, v_2, \ldots, v_k, t$ in $G - A'$ where $k \geq 1$ (note that we assume that (s,t) is not in A for simplicity). Since y^* is integral, $y^*(a) = 0$ for all $a \in p$. Considering the dual constraint for the arc (s, v_1) we see that $z(v_1) \leq -1$ since $y(s, v_1) = 0$. For each arc (v_i, v_{i+1}) in p, via the dual constraint and the fact that $y^*(v_i, v_{i+1}) = 0$, we obtain that $z(v_{i+1}) \leq z(v_i)$. These set of inequalities imply that $z(v_k) \leq -1$. However, applying the dual constraint to the arc (v_k, t) we obtain that $z(v_k) \geq 0$ which is a contradiction to the feasibility of the dual solution.

We now describe the second proof. We can extend z to have variables $z(s)$ and $z(t)$ with $z(s) = -1$ and $z(t) = 0$. Then the dual has a cleaner form,

$\max\{yc \mid y \geq 0, \exists z : y + zM \geq 0\}$. Note that M here is the full arc-vertex incidence matrix of D. Thus we have x^* and integral (y^*, z^*) such that $wx^* = y^*c$ and $y^* + z^*M \geq 0$.

Let $U = \{v \in V | z^*(v) < 0\}$. Note that $s \in U$ and $t \notin U$ and hence $\delta^+(U)$ is a s-t cut.

Claim 5.3.1. $c(\delta^+(U)) \leq y^*c = \sum_{a \in A} y^*(a)c(a)$

Proof. Take any arc $(u, v) \in \delta^+(U)$. We have $z^*(u) - z^*(v) + y^*(u, v) \geq 0$ for each (u, v). Since $u \in U$ and $v \notin U$, $z^*(u) < 0$ and $z^*(v) \geq 0$. Since z^* is integral, we have

$$\begin{aligned} y^*(u, v) &\geq 1 \\ \implies c(\delta^+(U)) &\leq \sum_{a \in \delta^+(U)} c(a)y^*(a) \\ &\leq \sum_{a \in A} c(a)y^*(a) \text{ since } y^* \geq 0 \end{aligned}$$

■

Therefore, U is a s-t cut of capacity at most $y^*c = wx^*$ but wx^* is the value of a maximum flow. Since the capacity of any cut upper bounds the maximum flow, we have that there exists a cut of capacity equal to that of the maximum flow. We therefore, get the following theorem,

Theorem 5.14. *In any directed graph $G = (V, A)$ with non-negative arc capacities, $c : E \to \mathbb{Q}^+$, the s-t maximum-flow value is equal to the s-t minimum cut capacity. Moreover, if $c : E \to \mathbb{Z}^+$, then there is an integral maximum flow.*

Interpretation of the dual values: A natural interpretation of the dual is the following. The dual values, $y(a)$ indicate whether a is cut or not. The value $z(v)$ is the shortest path distance from s to v with $y(a)$ values as the length on the arcs. We want to separate s from t. So, we have (implicitly) $z(s) = -1$ and $z(t) = 0$. The constraints $z(u) - z(v) + y(u, v) \geq 0$ enforce that the z values are indeed shortest path distances. The objective function $\sum_{a \in A} c(a)y(a)$ is the capacity of the cut subject to separating s from t.

Circulations and lower and upper bounds on arcs: More general applications of flows are obtained by considering both lower and upper bounds on the flow on arcs. In these settings, circulations are more convenient and natural.

Definition 5.15. *For a directed graph $D = (V, A)$, a circulation is a function $f : A \to \mathbb{R}^+$ such that $\sum_{a \in \delta^-(v)} f(a) = \sum_{a \in \delta^+(v)} f(a) \forall v \in V$*

Given non-negative lower and upper bounds on the arcs, $l : A \to \mathbb{R}^+$ and $u : A \to \mathbb{R}^+$, we are interested in circulations that satisfy the bounds on the arcs. In other words, the feasibility of the following:

$$\begin{gathered} l(a) \leq x(a) \leq u(a) \\ x \text{ is a circulation} \end{gathered}$$

The above polyhedron is same as $\{x \mid Mx = 0, l \leq x \leq u\}$ where M is the arc-vertex incidence graph of D, which is TUM. Therefore, if l, u are integral then the polyhedron is integral. Checking if there is a feasible circulation in a graph with given l and u is at least as hard as solving the maximum flow problem.

Exercise 5.6. Given $D, s, t \in V$ and a flow value F, show that checking if there is an $s - t$ flow of value F can be efficently reduced to checking if a given directed graph has a circulation respecting lower and upper bounds.

The converse is also true however; one can reduce circulation problems to regular maximum-flow problems, though it takes a bit of work.

Min-cost circulation is the problem: $\min\{cx \mid l \leq x \leq u, Mx = 0\}$. We therefore obtain that

Theorem 5.16. *The min-cost circulation problem with lower and upper bounds can be solved in (strongly) polynomial time. Moreover, if l, u are integral then there exists an integral optimum solution.*

The analogue of max flow-min cut theorem in the circulation setting is Hoffman's circulation theorem.

Theorem 5.17. *Given $D = (V, A)$ and $l : A \to \mathbb{R}^+$ and $u : A \to \mathbb{R}^+$, there is a feasible circulation $x : A \to \mathbb{R}^+$ iff*

1. *$l(a) \leq c(a)\ \forall a \in A$ and*
2. *$\forall U \subseteq V, l(\delta^-(U)) \leq c(\delta^+(U))$.*

Moreover, if l, u are integral then there is an integral circulation.

Exercise 5.7. Prove Hoffman's theorem using TUM property of M and duality.

***b*-Transshipments:** One obtains slightly more general objects called transshipments as follows:

Definition 5.18. *Let $D = (V, E)$ be a directed graph and $b : A \to \mathbb{R}$. A b-transshipment is a function $f : A \to \mathbb{R}^+$ such that $\forall u \in V, f(\delta^-(u)) - f(\delta^+(u)) = b(u)$ i.e, the excess inflow at u is equal to $b(u)$.*

We think of nodes u with $b(u) < 0$ as supply nodes and $b(u) > 0$ as demand nodes. Note that $b = 0$ captures circulations. Once can generalize Hoffman's circulation theorem.

Theorem 5.19. *Given $D = (V, A)$, $b : V \to \mathbb{R}^+$ and $l : A \to \mathbb{R}^+$and $u : A \to \mathbb{R}^+$,there exists a b-transshipment respecting l, u iff*

1. $l(a) \leq u(a)\ \forall a \in A$ *and*
2. $\sum_{v \in V} b(v) = 0$ *and*
3. $\forall S \subseteq V$, $u(\delta^+(S)) \geq l(\delta^-(S)) + b(S)$.

Moreover, if b, l, u are integral, there is an integral b-transshipment.

Exercise 5.8. Derive the above theorem from Hoffman's circulation theorem.

5.3.3 Interval graphs

A graph $G = (V, E)$ on n nodes is an interval graph if there exist a collection $\mathcal{I}$ of n closed intervals on the real line and a bijection $f : V \to \mathcal{I}$ such that $uv \in E$ iff $f(u)$ and $f(v)$ intersect. Given an interval graph, an interesting problem is to find a maximum weight independent set in G where $w : A \to \mathbb{R}^+$ is a weight function. This is same as asking for the maximum weight non-overlapping set of intervals in a collection of intervals.

We can write an LP for it. Let $\mathcal{I} = \{I_1, \ldots, I_n\}$

$$\begin{aligned} \max \sum_{i=1}^{n} w_i x_i & \\ \sum_{I_i : p \in I_i} x_i &\leq 1 \quad \forall p \text{ in } \mathbb{R} \\ x_i &\geq 0 \qquad 1 \leq i \leq n \end{aligned}$$

Note that the constraints can be written only for a finite set of points which correspond to the end points of the intervals. These are the natural "clique" constraints: each maximal clique in G corresponds to a point p and all the intervals containing p. Clearly an independent set cannot pick more than one node from a clique.

The LP above is $\max\{wx \mid x \geq 0, Mx \leq 1\}$. If we sort the points then M is a consecutive ones matrix (each column has consecutive 1s), and hence TUM. Therefore, the polyhedron is integral. We therefore have a polynomial time algorithm for the max-weight independent set problem in interval graphs. This

problem can be easily solved efficienctly via dynamic programming. However, we observe that we can also efficiently solve $\max\{wx \mid x \geq 0, Mx \leq b, x \in \mathbb{Z}^n\}$for any integer b and this is not easy to see via other methods.

To illustrate the use of integer decomposition properties of polyhedra, we derive a simple and well known fact.

Claim 5.3.2. *Suppose we have a collection of intervals $\mathcal{I}$ such that $\forall p \in \mathbb{R}$ the maximum number of intervals containing p is at most k. Then $\mathcal{I}$ can be partitioned into $\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_k$ such that each $\mathcal{I}_k$ is a collection of non-overlapping intervals. In other words, if G is an interval graph then $\omega(G) = \chi(G)$ where $\omega(G)$ is the clique-number of G and $\chi(G)$ is the chromatic number of G.*

One can prove the above easily via a greedy algorithm. We can also derive this by considering the independent set polytope $\{x \mid x \geq 0, Mx \leq 1\}$. We note that $x^* = \frac{1}{k}.\overline{1}$ is feasible for this polytope if no point p is contained in more than k intervals. Since P has the integer decomposition property, $y = kx^* = \overline{1}$ can be written as $x_1 + \ldots + x_k$ where x_i is integral $\forall 1 \leq i \leq k$ and $x_i \in P$. This gives the desired decomposition. The advantage of the polyhedral approach that one obtains a more general theorem by using an arbitrary integral b in the polytope $\{x \mid x \geq 0, Mx \leq b\}$ and this has applications; see for example [68], and generalizations [51, 59].

Chapter 6

Network Flow: A Quick Overview

Network flow is a large subject with many books and articles devoted to it. We refer the reader to [3, 57, 67] for book length treatments. See also notes of Kent Quanrud (Chapters 1 to 3). We confine our attention to single-commodity flows in this chapter. The goal is to highlight a few structural and algorithmic aspects of flows. The discussion will be breezy with several claims left as exercises. We discussed polyhedral aspects of the standard edge-based flows in the chapter on TUM matrices. Here will focus on some combinatorial and algorithmic aspects, and also point out the path-based flow formulation and its utility.

6.1 Preliminaries

Throughout we will only work with directed graphs. Let $G = (V, A)$ be a directed graph with non-negative edge capacities $c : A \to \mathbb{R}_+$. We will frequently use s, t to denote two distinct nodes in G. It is common to refer to an edge-capacitated directed graphs as a *flow network*.

We will start with an edge-based definition of flows.

Definition 6.1. *Let $G = (V, A)$ be a capacitated directed graph and let $s, t \in V$ with $s \neq t$. A real-valued function $f : A \to \mathbb{R}$ is an s-t flow if it satisfies the following conditions:*

- flow conservation *at all nodes $v \notin \{s, t\}$:* $\sum_{a \in \delta^+(v)} f(a) = \sum_{a \in \delta^-(v)} f(a)$
- capacity constraints *on all arcs:* $0 \leq f(a) \leq c(a)$ *for all $a \in A$.*

Definition 6.2. *Given an s-t flow f, the* value *of the flow is defined as the net flow leaving the source s:* $\sum_{a \in \delta^+(s)} f(a) - \sum_{a \in \delta^-(s)} f(a)$. *We use val($f$) or $|f|$.*

Exercise 6.1. Suppose f is an s-t flow and let $S \subset V$ such that $s \in S, t \in V \setminus S$. The flow out of S is defined as $\sum_{a \in \delta^+(S)} f(a) - \sum_{a \in \delta^-(S)} f(a)$. Prove that this is equal to $\text{val}(f)$.

Definition 6.3. *Given a edge-capacitated graph $G = (V, A)$ a function $f : A \to \mathbb{R}$ is a* circulation *if it satisfies flow conservation at every node of G. Given upper and lower bounds on the arcs $\ell : A \to \mathbb{R}$ and $c : A \to \mathbb{R}$ a circulation respects the bounds if $\ell(a) \leq f(a) \leq c(a)$ for all arcs a.*

Remark 6.1. An s-t flow f can have zero value but can be a non-trivial circulation. A circulation is an u-v flow of value 0 for any nodes u, v.

Remark 6.2. It is convenient/necessary to manipulate flows and circulations by adding and subtracting them. In doing these operations the bound constraints may not hold (including non-negativity). In such settings we may use the terminology of flows/circulations if they satisfy the conservation constraints. It is sometimes convenient to assume that the underlying directed graph is a complete directed graph.

Path based definition of flow and flow decomposition: An alternative and important view of flows is via paths. Given $G = (V, A)$ and nodes s, t let $\mathcal{P}_{s,t}$ denote the set of all s-t paths. We may use $\mathcal{P}$ to simplify notation when s, t are clear form the context. Then one can define an s-t flow as a function $f : \mathcal{P}_{s,t} \to \mathbb{R}_+$ that assigns a flow of value $f(p)$ for each path $p \in \mathcal{P}_{s,t}$. We can define $\text{val}(f)$ for a path based flow as $\sum_{p \in \mathcal{P}_{s,t}} f(p)$. One can define circulations similarly as a function $f : \mathcal{C} \to \mathbb{R}_+$ where $\mathcal{C}$ is the set of all directed cycles in G. Given a path based s-t flow f one naturally obtains an edge based s-t flow g where $g(a) = \sum_{p \in \mathcal{P}: a \in p} f(p)$. It is easy to check that the value of f and g are the same. At first glance the path based notion of flow seems not so useful since it is defined over $\mathcal{P}_{s,t}$ whose size can be exponential in the size of the input graph. Nevertheless, one sees that the paths are impicitly encoded by a graph, and in many scenarios one has a flow that is non-negative only on a polynomial number of paths.

Flow decomposition is the process of taking an edge-based flow f and decomposing it into a path based flow. Similarly one can decompose an edge-based circulation into a cycle based circulation.

Lemma 6.1. *Let $f : A \to \mathbb{R}_+$ be an s-t flow. Then there is an efficient algorithm that outputs a path flow $g : \mathcal{P}_{s,t} \to \mathbb{R}_+$ of the same value and g has non-zero flow on at most m paths where $m = |A|$. Moreover, for all $a \in A$, $f(a) \geq \sum_{p \ni a} g(p)$.*

Proof sketch. Consider an edge based flow f. We construct g as follows; we implicitly assign $g(p) = 0$ for all p initially. We can assume, without loss of generality, that $f(a) > 0$ for all a (otherwise we remove such arcs). If $\text{val}(f) = 0$

then we can return empty g. Otherwise there is an s-t path p in G with all arcs in p having non-zero flow (why?). Let a' be the arc in p with the smallest f value. We set $g(p) = f(a')$ and reduce $f(a)$ for all $a \in p$ by $g(p)$ to obtain a new flow f'. It is easy to observe that f' is a valid flow and that $\text{val}(f) = \text{val}(f') + f(a')$. We recursively compute a flow decomposition g' of f' in $G - a'$ (since $f'(a') = 0$); by induction $\text{val}(g') = \text{val}(f')$ and support of g' has at most $m - 1$ paths. We update g by adding g' to it and we return g. Note that the support of g is at most m and $\text{val}(g) = \text{val}(f)$. It is easy to see that this decomposition can be efficiently computed. ■

We leave the following two lemmas as exercises.

Lemma 6.2. *Let f be an edge-based circulation in G that respects non-negative lower and upper bounds $\ell : A \to \mathbb{R}_+$ and $c : A \to \mathbb{R}_+$. Then there is a $g : \mathcal{C} \to \mathbb{R}_+$ such that (i) for all $a \in A$ we have $f(a) = \sum_{C \in \mathcal{C}: a \in C} g(C) = f(a)$ and (ii) support of g is at most $|A|$. Moreover such a g can be computed efficiently.*

Definition 6.4. *An s-t flow is acyclic if there is no cycle in the support of f. In other words the arcs with non-negative flow form a directed acyclic graph (DAG).*

Claim 6.1.1. *Given any s-t flow $f : A \to \mathbb{R}_+$ there is an acyclic flow $g : A \to \mathbb{R}_+$ such that (i) $g(a) \leq f(a)$ for all $a \in A$ and (ii) $val(g) = val(f)$. Moreover one can compute such a g in polynomial time.*

Proof sketch. Start with f. If there is a cycle C in the support of f we reduce the flow on all arcs in the cycle by $\min_{a \in C} f(a)$. This reduces the flow on at least one arc to zero and does not affect the value. We repeat this until we obtain an acyclic flow. ■

Lemma 6.3. *Any s-t flow f can be efficiently decomposted into a collection of at at most m paths and cycles such that $f(a)$ is equal to the total flow on the paths and cycles in the decomposition. More formally there is a $g : \mathcal{P}_{s,t} \cup \mathcal{C} \to \mathbb{R}_+$ such that (i) for all $a \in A$, $f(a) = \sum_{p \in \mathcal{P}_{s,t}: a \in p} g(p) + \sum_{C \in \mathcal{C}: a \in C} g(C)$ and (ii) support of g is at most m, and (iii) such a g can be computed efficiently from f.*

Exercise 6.2. What is the difference between Lemma 6.1 and Lemma 6.3?

Remark 6.3. Given a flow f the representation size of a flow decomposition via paths/cycles is $O(nm)$ since we need to specify m paths each of which can have size n. This is indeed necessary in the worst case when capacities can be arbitrary (can you construct an easy example?). Here $n = |V|$ and $m = |E|$. This is in contrast to the edge flow represenation which requires only m numbers. The $O(nm)$ bound is some times called the flow-decomposition barrier since several maximum flow algorithms based on augmenting paths implicitly compute a flow

decomposition along the way. How fast can one compute a flow decomposition. The obvious algorithm implied in the proof implies a bound of $O(m^2)$ (we assume $m \geq n$) since each iteration requires finding an s-t path and there are a total of m iterations. A careful implementation via dynamic tree data structure allows one to obtain a running time of $O(nm \log(m/n))$.

6.1.1 Maximum Flow and the Residual Network

The input to the maximum s-t flow problem is a flow network $G = (V, A)$ with non-negative edge capacities $c : A \to \mathbb{Z}_+$ and two nodes s, t. The goal is to find an s-t flow of maximum value.

LP Formulations: We saw that it can be cast as an LP problem via the edge formulation and we reproduced it below. Consider the s-t maximum-flow problem in a directed graph The variables are $x(a), a \in A$ which stand for the flow value on the arcs.

$$\begin{aligned}
\max \sum_{a \in \delta^+(s)} x(a) - \sum_{a \in \delta^-(s)} x(a) & \\
\sum_{a \in \delta^+(v)} x(a) - \sum_{a \in \delta^-(v)} x(a) &= 0 \quad \forall v \in V - \{s, t\} \\
x(a) &\leq c(a) \quad \forall a \in A \\
x(a) &\geq 0 \quad \forall a \in A
\end{aligned}$$

The path formulation also leads to an LP with a variable $x(p)$ for each path p.

$$\begin{aligned}
\max \sum_{p \in \mathcal{P}_{s,t}} x(p) & \\
\sum_{p \ni a} x(p) \leq c(a) & \qquad a \in A \\
x(p) \geq 0 & \qquad p \in \mathcal{P}_{s,t}
\end{aligned}$$

Note that the two polyhedrons are very different! The arc-based formulation is an integer polyhedron when the capacities are integral while the path based polyhedron is not although the two have the same optimum value. Is there any advantage of the exponential sized path based formulation? There are indeed some, and the change in perspective is useful in some applications when considering more general versions of flows.

Residual network: The residual graph/network plays a fundamental role in the study of network flows. Given a flow network $G = (V, A)$ and a valid s-t flow $f : A \to \mathbb{R}_+$ the residual negraph $G_f = (V, A')$ is defined as follows. We describe it in a constructive fashion. For each arc $(u, v) \in A$ with $0 \leq f(a) < c(a)$ there we add a forward arc (u, v) with residual capacity $c'(u, v) = c(a) - f(a)$ and a reverse arc (v, u) with capacity $c'(v, u) = f(a)$. For an arc (u, v) with $f(u, v) = c(u, v)$ (which is called a saturated arc) we only add a reverse arc (v, u) with $c'(v, u) = f(a) = c(u, v)$. Via the construction, every arc in G_f has a positive capacity.

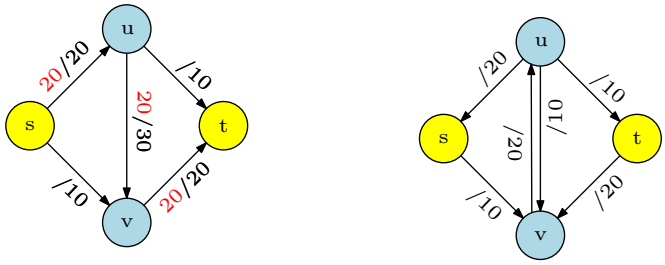


Figure 6.1: Flow in a network and the residual network.

Remark 6.4. In a directed graph $G = (V, A)$, for a pair of nodes u, v, both arcs (u, v) and (v, u) can be in A. A pair of arcs (u, v) and (v, u) is called a digon and forms a cycle. If $f(u, v) > 0$ and $f(v, u) > 0$ then the residual network can create two parallel arcs in each direction. These can be merged into a single arc by combining the capacities. Alternatively, when considering s-t flows we can reduce the flow along the digon to make sure that at least one of the two arcs have zero flow; this transformation is not feasible if arcs have lower bounds.

Once we compute a flow f, the residual graph G_f captures precisely the remaining problem. This can be formalized via the following two lemmas which are somewhat tedious to formally prove but are very intuitive. The two lemmas look similar but read them carefully.

Lemma 6.4. *Suppose f is an s-t flow in G. Let g be* any *s-t flow in G_f. Then $f + g$ is an s-t flow in G of value $val(f) + val(g)$.*

Lemma 6.5. *Suppose f is an s-t flow in G and let g be* any *s-t flow in G. Then $g - f$ is an s-t flow in the residual network G_f of value $val(g) - val(f)$ (which can be negative indicating that it is a t-s flow).*

Corollary 6.5. *A flow f is a maximum s-t flow iff there is no s-t path in G_f.*

Proof. Suppose there is a flow g such that $\text{val}(g) > \text{val}(f)$ then by Lemma 6.5 there is a flow in G_f of value $\text{val}(g) - \text{val}(f) > 0$ in G_f which implies that there is an s-t path in G_f. Thus no path in G_f implies that f is maximum. Suppose there is a path in G_f then there is a non-zero flow g in G_f since every edge in G_f has strictly positive capacity. Hence $g + f$ is a flow in G with $\text{val}(g + f) > \text{val}(f)$ and hence f is not a maximum flow. ■

Remark 6.5. Note that the preceding argument does not invoke the notion of cuts.

The following establishes the maxflow-mincut theorem.

Lemma 6.6. *Let f be a flow such that there is no s-t path in G_f. Let A be the set of nodes reachable from s in G_f. Then $val(f) = \sum_{a \in \delta^+(A)} c(a)$ and hence is a minimum s-t cut whose value is equal to $val(f)$.*

Proof sketch. Let A be set of nodes reachable from s in G_f. This implies that there is no arc $(x, y) \in \delta^+_{G_f}(A)$ for otherwise y would be reachable from s too. This implies that every arc $(u, v) \in \delta^+_G(A)$ must be saturated by f, and every arc $(u, v) \in \delta^-_G(A)$ must have zero flow, otherwise (u, v) would be in $\delta^+_{G_f}(A)$ by the definition of G_f. Thus the flow out of A in G must be equal to $\text{val}(f)$ which implies that it is a minimum cut. ■

6.2 Augmenting Path Algorithms

Lemmas 6.4 and 6.5 give a simple algorithm for finding a maximum flow. Start with any flow f (in particular a flow of value 0). Create G_f. If G_f has no s-t path (which is easy to check) then f is maximum. Otherwise find any non-zero flow g in G_f. Update f to be $g + f$ to obtain a flow of more value. The most obvious way to find a flow of non-zero value in a flow network G is to take *any* path p from s to t with non-zero edge capacities and send flow of value equal to $\min_{a \in p} c(a)$ on it. This is the Ford-Fulkerson augmenting path algorithm that repeatedly finds a path in G_f and *augments* along that path. Typically the algorithm is described as augmenting in G along the path p in G_f which wraps the idea of finding a flow in G_f and adding it to flow f in one step. However, it is conceptually useful to realize that one can find any flow g in G_f and add it to the flow f in G. The termination of the basic algorithm is not guaranteed if the capacities are irrational. If all capacities are integer then the augmenting path algorithm maintains the fact that the flow at each stage is integral (if it starts with $f(a) = 0$ for all $a \in A$)). Termination is easily guaranteed since each augmentation increases flow by an integer amount which is at least 1, and the maximum flow is at most the minimum cut which is an integer value. The FF algorithm also proves that one has an integral maximum flow

whenevever capacities are all integers. The basic Ford-Fulkerson algorithm can take F iterations where F is the maximum flow value. Each iteration can be implemented in $O(m)$ time and hence on obtains an $O(mF)$ time algorithm. In particular, if G is a simple directed graph, $F \leq n$ and hence we obtain an $O(mn)$ time algorithm. However, when the graph can have large (integer) capacities, F depends on the edge capacities and the Ford-Fulkerson algorithm need not run in polynomial time. Note that the algorithm is not fully specified since the procedure to choose the s-t path p in G_f is not explicitly stated; if one can choose the path p adversarially there are examples showing that the algorithm can take an exponential time when capacities are large.

There is a huge amount of research on fast maximum flow algorithm and there have been several recent breakthroughs via sophisticated mix of discrete and continuous optimization methods combined with advances in dynamic data structures. Here we describe a few variants of the augmenting path algorithm that capture some nice ideas. We will not discuss scaling based algorithms.

6.2.1 Augmenting along high-capacity paths

Suppose we find find an s-t path p in G_f. The maximum amount of flow that we can send along p is given by its *bottleneck* capacity which is $\min_{a \in p} c(a)$. Thus, it makes sense to find a path p that has maximum bottleneck capacity; this is the path that allows us to send the maximum amount of flow if we can only use one path.

Exercise 6.3. Decribe an adaptation of Dijkstra's shortest path algorithm or another method to find the s-t path with maximum bottleneck capacity in $O(m \log n)$ time.

One can prove that the algorithm terminates in polynomial time.

Theorem 6.6. *The algorithm that augments along the path with maximum bottleneck capacity in the residual graph terminates in $O(m \log F)$ iterations when all capacities are integer valued. Here F is the maximum flow value.*

We sketch the proof. Suppose f is the current flow and let f^* is a maximum flow. Then by Lemma 6.5 there is a flow g of value $\text{val}(f^*) - \text{val}(f)$ in G_f. By flow decomposition g can be decomposed into at most m paths in G_f. This implies that one of these paths must carry a flow of value at least $\text{val}(g)/m = \text{val}(f^*) - \text{val}(f) = (F - \text{val}(f))/m$. This implies that the algorithm that augments along the maximum bottleneck capacity path increases the flow value of f by at least $(F - \text{val}(f))/m$. Note that in the first iteration the increase is at least F/m but then as it proceeds and $\text{val}(f)$ gets closer to F, the guarantee on the increase gets smaller. Nevertheless we can analyze the number of iterations as follows.

We start with flow value 0 and maximum flow value F. How many iterations does it take so that we reach a flow of value at least $F/2$? As long as the current flow f satisfies $\text{val}(f) < F/2$, the amount we increase in each iteration is at least $(F - \text{val}(f))/m \geq F/(2m)$. Thus, we can have at most $2m$ iterations where $\text{val}(f) \leq F/2$.

Once we reach a flow f with $\text{val}(f) \geq F/2$ we see that we can analyze the process with respect to G_f. In G_f the maximum flow left is at most $F/2$. It takes $\leq 2m$ iterations to get to flow of value at least $F/4$ in G_f by the same reasoning. Following this, one can see that after $(2m)i$ iterations the maxflow in the residual graph is at most $F/2^i$. Thus after $2m \log F$ iterations the flow in the residual network is at most 1 and we will be done in one more iteration since each augmentation increases the flow but at least one unit.

One can do a more direct analysis that show an upper bound of $1 + m \ln F$ iterations.

6.2.2 Shortest augmenting path: a strongly polynomial-time algorithm

Edmonds and Karp described and analyze a variant of the Ford-Fulkerson algorithm which is very simple and natural. In each iteration augment along the shortest s-t path in the residual network G_f. By shortest we mean the path with the fewest number of edges. They proved the following.

Theorem 6.7 (Edmonds-Karp, 1972)**.** *The shortest augmenting path algorithm terminates in $O(mn)$ iterations and hence the total running time of a simple implementation of this algorithm is $O(m^2n)$. Here m is the number of edges and n is the number of nodes in the flow network.*

Thus the algorithm is *strongly polynomial* time algorithm. The algorithm will terminate even for irrational capacities as long as we can do arithmetic over the given capacities. The analysis is based on the following two lemmas.

Lemma 6.7. *Suppose we augment along a shortest s-t path in G_f then the shortest s-t path length does not decrease in the new residual graph after the augmentation.*

Lemma 6.8. *There can at most m iterations of the shortest augmenting path algorithm before the shortest path s-t path length in the residual graph increases by at least 1.*

We sketch the proofs of the preceding lemmas. They are based on simple properties of shortest paths that are typically computed by the breadth-first-search (BFS) algorithm. BFS from a node s in a directed graph $G = (V, A)$ classifies nodes according to distance. Let L_i be the set of nodes at distance exactly i from s with $L_0 = \{s\}$. Let $t \in L_d$ where d is the shortest path distance

from s to t. The following proprties are easy to see and standard in understanding BFS.

- An arc (u, v) is a forward arc if $u \in L_i$ and $v \in L_{i+1}$
- An arc (u, v) is a backward arc if $u \in L_i$ and $v \in L_j$ for some $j < i$
- An arc may have both end points inside the same layer and we ignore them.
- Every s-t shortest path uses only forward edges

We make two simple claims.

Claim 6.2.1. *Suppose a forward edge is* deleted. *Then the shortest path distance from s to any node does not decrease.*

Claim 6.2.2. *Suppose we add a backward edge (u, v) to G where $u \in L_i$ and $v \in L_j$ for some $j < i$. Then the shortest path distance from s to any node does not decrease.*

Suppose p is an s-t shortest path in the residual network G_f. Then what happens after we augment along p? We remove edges in p that are saturated. For each edge in p we add a corresponding reverse arc. Since p is a shortest path its edges are all forward edges in the BFS layers. By the preceding claims we see that after augmentation the shortest s-t path length in the residual network does not decrease. This proves Lemma 6.7. Now we sketch the proof of Lemma 6.8. Suppose the shortest path distance from s to t is d. Each augmentation saturates at least one forward edge which is deleted and becomes a backward edge. If the s-t path length stays at d then no backward edge can be in the shortest s-t path (why?). However, there can be at most m iterations before s-t distance increases since each iteration deletes at least one forward edge.

6.2.3 Blocking Flows

Dinic defined the notion of blocking flows based on the analysis of the shortest augmenting path algorithm. Suppose the current shortest s-t path distance in G_f is d. Augmenting along a single path can take m iterations before the distance increases by one. Instead of augmenting along a single path suppose we find a flow g in G_f such that g *blocks* all s-t shortest paths of length at most d. Then after adding g to f in the residual graph the shortest path distance increases by one. The shortest path distance between s and t can increase at most n times.

Definition 6.8. *Given a flow network G an s-t flow f is a blocking flow if for every s-t path p there is at least one arc in p that is saturated by f. In other words f is a* maximal *flow in that one cannot add greedily add flow to f in G.*

Note that a blocking flow is maximal in G and need not be a maximum flow. How can we find a blocking flow? One can do a simple greedy algorithm where in each iteration we find an s-t path and augment along the path to saturate one additional arc; the capacities along the path are reduced and those that saturated are not considered in future path selections. Note that we are not computing a residual network in this process, and it is clear that the greedy algorithm will terminate in $O(m)$ iterations; a naive implementation takes $O(m^2)$ time. The utility of blocking flows is given by the next lemma.

Lemma 6.9. *Let G be a flow network and let the s-t distance be d. Suppose f is a blocking flow in G then the s-t shortest distance in G_f is strictly more than d.*

Corollary 6.9. *A maximum s-t flow can be computed via $O(n)$ blocking flow computations.*

The preceding lemma and corollary shows that if we find a blocking flow g in G_f and augment along g then we will need $O(n)$ such iterations. The total time would be $O(m^2 n)$ since each blocking flow takes $O(m^2)$ time. This has not led to any improvement in the running time over the shortest augmenting path algorithm! A useful observation is that instead of finding a blocking flow in the entire graph G_f, it suffices to find it only in the layered graph consisting of the forward edges in a BFS computation.

Lemma 6.10. *Let G be flow network and let the s-t shortest path distance be d. Let H be the layered subgraph of G obtained by retaining only the forward edges between layers of a BFS computation from s (and omitting the layers beyond L_d). Suppose f is a blocking flow in H and consider f as a flow in G — the shortest s-t path distance is strictly larger than d in G_f.*

Computing blocking flow in layered DAGs can be done faster than the naive greedy algorithm. One can use some simple graph search methods to avoid recomputing shortest paths from scratch in each iteration. These lead to the following.

Theorem 6.10. *In unit-capacity layered graphs a blocking flow can be computed in $O(m)$ time. In capacitated graphs a blocking flow can be computed in $O(mn)$ time.*

Corollary 6.11. *One can compute maximum flow in unit capacity graphs in $O(mn)$ time and in capacitated graphs in $O(mn^2)$ time.*

Note that we obtained an improvement in general capacitated graphs from $O(m^2 n)$ to $O(mn^2)$.

Via dynamic tree data structures Goldberg and Tarjan showed that blocking flows can be compted in capacitated graphs in almost linear time.

Theorem 6.12 (Goldberg and Tarjan)**.** *In a capacitated flow network a blocking flow can computed in $O(m \log(n^2/m))$ time.*

Corollary 6.13. *There is an $O(mn \log(n^2/m)$-time algorithm for max flow.*

In recent work, Orlin obtained an $O(mn)$ time algorithm which is the fastest strongly polynomial time algorithm for arbitrary capacities.

Combining blocking flows with Ford-Fulkerson: In unit-capacity graphs and simple graphs one can combine blocking flows with simple observations about maxflow being small when the shortest path distance is long. When flow value is small we can switch to the basic Ford-Fulkerson algorithm which turns out to be more efficient in that regime. By balancing parameters one obtains faster algorithms. We refer the reader to network flows books for more details. As a particular application we mention the reduction of maximum cardinality bipartite matching to network flow. Applying this basic idea implies an algorithm with running time $O(m\sqrt{n})$ for maximum bipartite matching which was noted by Hopcroft and Karp.

6.3 Minimum Cost Flow

In min-cost flow we are given a capacitated flow network $G = (V, A)$, source s and sink t and also costs/weights $w(a), a \in A$ on the arcs (they can be negative). Formally the cost of a flow f, denoted by cost(f) is defined as $\sum_{a \in A} w(a)f(a)$. The goal is to find a minimum-cost maximum flow. Alternatively, one can ask for a minimum cost flow of some given value F. One can write the second version as an LP below.

$$\begin{aligned}
&\min \sum_{a \in A} w(a)x(a) \\
\sum_{a \in \delta^+(s)} x(a) - \sum_{a \in \delta^-(s)} x(a) &\geq F \\
\sum_{a \in \delta^+(v)} x(a) - \sum_{a \in \delta^-(v)} x(a) &= 0 \quad \forall v \in V - \{s, t\} \\
x(a) &\leq c(a) \quad \forall a \in A \\
x(a) &\geq 0 \quad \forall a \in A
\end{aligned}$$

The polytope is integral whenever F and the capacities are integer valued. Here we are interested in a discussion of combinatorial algorithm and some basic results. We observe that when $F = 1$ the problem is closely connected to the shortest path problem.

Exercise 6.4. Show that the s-t shortest path problem in a directed graph with non-negative edge lengths is equivalent to computing the s-t mincost flow of value 1 in the same graph with non-negative costs.

The costs can be negative and in fact one sees that shortest path computation even with negative lengths can be reduced to minimum cost flow.

Exercise 6.5. Suppose $G = (V, A)$ is a network with edge lengths which can be negative. Show that the s-t shortest path problem in G is equivalent to computing the s-t mincost flow of value 1 in the same graph when there is no negative length cycle in G. Show that if the min-cost s-t flow of value 1 in G with capacities set to 2 has strictly less cost than cost of flow with capacities 1 iff G has a negative length cycle.

Circulations are quite natural in discussing min-cost flow for two reasons. First, as we observed above, if edges can have negative lengths then the min-cost flow need not be acyclic. Second, even when we start out with non-negative costs, the natural residual flow network creates negative costs. For this reason, it is common for people to work with min-cost circulations instead of min-cost flow. There are advantages and disadvantages to this. For the sake of brevity and intuitive clarity we will stick with min-cost flow

Residual network: We generalize the definition of the residual network to incorporate costs. Suppose $G = (V, A)$ is a flow network with capacities $c : A \to \mathbb{Z}_+$ and edge costs $w : A \to \mathbb{Z}$. Let $f : A \to \mathbb{R}$ be an s-t flow in G. Recall the definition of the residual network $G_f = (V, A')$. For each arc (u, v) with $0 \leq f(a) < c(a)$ we keep the arc (u, v) with residual capacity $c'(a) = c(a) - f(a)$ and add a reverse arc (v, u) with capacity $c'(v, u) = f(a)$. If (u, v) is saturated, that is $f(a) = c(a)$ we only add the reverse arc. When edges have costs we simply set the cost of the reverse arc to be $-w(a)$ and the cost of the forward arc to be the same as the original one. This is natural since sending flow on the reverse arc corresponds to reducing flow on the original arc. The following lemmas capture the calculus of flows with costs.

Lemma 6.11. *Suppose f is an s-t flow and let g be an s-t flow in G_f. Then $f + g$ is an s-t flow in G such that $val(g + f) = val(g) + val(f)$ and $cost(g + f) = cost(g) + cost(f)$.*

Lemma 6.12. *Suppose f and g are s-t flows in G. Then $g - f$ is an s-t flow such that $val(g - f) = val(g) - val(f)$ and $cost(g - f) = cost(g) - cost(f)$.*

Optimality characterization: The key to understanding min-cost flow is the following intuitive characterization.

Lemma 6.13. *Suppose f is an s-t flow of value F. Then f is a min-cost s-t-flow of value F iff there is no negative cost cycle in G_f.*

Proof sketch. Suppose there is a negative cost cycle C in G_f. Then we can send a non-negative amount of flow along C and the cost of this flow < 0. Let g be this flow which is in fact a circulation since it is along a cycle. Thus $f + g$ is an s-t flow of the same value F, and the cost of $f + g$ is reduced because cost of g is negative.

Let g be the min-cost flow among all flows with value F. Suppose $cost(f) > cost(g)$. Consider the flow $g - f$ in G_f. Since val(g) = val(f) we have $g - f$ is a flow of value 0 in G_f which means that it is a circulation, and $cost(g - f) < 0$. We can decompose the circulation $g - f$ into a collection of cycles. At least one of these cycles have to have negative cost since the total cost is negative. Thus if f is not a min-cost flow then there is a negative length cycle in G_f. ■

Based on the preceding optimality criterion there are two classes of algorithm one can think of. One is to increase the flow value from 0 to F while inductively maintaining the optimality of the cost at each stage. The other is to find a flow value F and then improve its cost. We briefly discuss these approaches.

6.3.1 Successive Shortest Path Algorithm

Here we will assume that costs are non-negative and capacities are integer valued; in fact we will assume that they are unit. Suppose we want to find a flow of value 1. This is simply the shortest path problem so we can think of this as augmenting flow by one unit along the shortest path. Now suppose we want to find a flow of value 2 by increasing the flow by one more unit. In the standard Ford-Fulkerson augmenting path algorithm we will computed the residual graph G_f and find an augmenting path. Instead we will find a shortest path in G_f and augment along that path. Note that even if the original graph did not have negative costs the residual graph G_f will have negative costs! But can it have a negative length cycle?

Theorem 6.14. *Let G be flow network with unit capacities and non-negative costs. Suppose f is a min-cost s-t flow of value k in G. Then there is no negative length cycle in G_f. If there is an s-t path in G_f then augmenting along a shortest s-t path in G_f yields a min-cost flow of value $k + 1$.*

6.3.2 Cycle cancelling and a strongly polynomial time algorithm

Now we discuss the cycle-cancelling approaches. Suppose we want to compute a min-cost s-t flow of value F. We will assume that capacities and costs are integer valued (but costs can be negative now). We first compute an arbitrary flow f of value F (or certify that there is no flow of value F in G). We will further assume that F is integer valued. Recall that f is a min-cost flow iff there is no

negative length cycle C in G_f. We can check whether G_f has a negative length cycle C via say the Bellman-Ford algorithm. If it does not then f is a min-cost flow. Otherwise by sending flow of at least one unit along C we obtain a new flow g with less cost than f. We can repeat this until no improvement is possible. When all costs and capacities are integer valued this process terminates since cost is reduced by one unit in each iteration and we maintain the invariant that the flow is integer valued. However this is not a polynomial time algorithm when costs or capacities can be large. A natural question is whether one can choose the negative length cycle C is some clever fashion. One obstacle is that some of the natural candidates are NP-Hard to compute; for instance finding the most negative length cycle generalizes the Hamiltonian cycle problem.

Even though maxflow had a simple strongly polynomial-time algorithm from 1972, finding one for min-cost flow was not obvious. As we mentioned in earlier chapters, Eva Tardös was the first to obtain a strongly polynomial time algorithm via LP techniques. The first "combinatorial" strongly polynomial time algorithm for min-cost flow is due to Goldberg and Tarjan. Their algorithm checks whether G_f has a negative length cycle C and if so it finds the minimum *mean length* cycle C and augments along that cycle. The mean length of a cycle C is defined as $\frac{\sum_{a \in C} w(a)}{|C|}$. One can find a minimum mean length cycle in strongly polynomial time via dynamic programming. See if you can figure it out.

Lemma 6.14. *There is a strongly polynomial-time algorithm for computing the minimum mean length cycle.*

The non-trivial result of Goldberg and Tarjan is the following.

Theorem 6.15. *Suppose f is a flow of value F. Augmenting along the minimum mean length cycle in each iteration terminates with a min-cost flow in $O(m^2 n \log n)$ iterations.*

For integer costs the algorithm can be shown to terminate in $O(mn \log(nW))$ iterations where W is the maximum absolute value of the edge costs.

Chapter 7

Gomory-Hu Tree for Connectivity in Graphs[1]

Most of this chapter is based on Chapter 15 from [57] with some additional discussion on submodularity.

Connectivity is a fundamental topic in graph theory. Given an undirected graph $G = (V, E)$ and two nodes s, t the *edge-connectivity* between s and t is the maximum number of edge disjoint paths between s and t. We denote it by $\alpha_G(s, t)$. Via Menger's theorem (equivalently the maxflow-mincut theorem), $\alpha_G(s, t)$ is the same as the minimum number of edges whose deletion disconnects s from t. Another way to express this is via cuts: $\alpha_G(s, t) = \min_{U \subset V, s \in U, t \in V - S} |\delta(U)|$. Note that in undirected graphs $\alpha_G(s, t) = \alpha_G(t, s)$ due to symmetry. Connectivity in directed graphs and for vertex connectivity are also important and well-studied but we will confine our attention in this chapter to undirected graphs; we drop the term undirected for the rest of the chapter. When G has non-negative edge capacities $c : E \rightarrow \mathbb{R}_+$, the connectivity notion generalizes naturally where $\alpha_G(s, t)$ is the capacity of the minimum cut separating s from t. One can compute $\alpha_G(s, t)$ via maximum flow computation. There are also other ways to compute it An important and useful notion is that of the *global* minimum cut of G. We can define it two equivalent ways: $\min_{s,t \in V, s \neq t} \alpha_G(s, t) = \min_{\emptyset \subset U \subset V} c(\delta(U))$. Sometime the global mincut value is referred to as the connectivity of G when G has unit capacities: it is the minimum number of edges whose removal disconnects the graph into two components.

How can we compute the global minimum cut? The naive way, based on the definition, is via computing the minimum cut for all $\binom{n}{2}$ pairs of vertices. However, one can do better, due to symmetry. Pick an arbitrary vertex s and compute the min s-t cut for all $t \in V - \{s\}$ and take the minimum. This only

[1] Based on scribed notes of David Morrison from 2010.

requires $n-1$ minimum cut computations. Can we do better? Karger, in his seminal work, showed two different ways to compute the global minimum cut of a graph without flows; the first is via random contraction [37], and the second is via tree packings which yields a near-linear time randomized algorithm [36]. A remarkable and simple result in very recent work shows how to do this computation using only $O(\log n)$ s-t maximum flow computations [45]; the precise statement is a bit more technical and we refer the reader to the paper.

At first glance it appears that in a given graph G there can be $\binom{n}{2}$ different minimum cut values, one for each pair. In a beautiful result, Gomory and Hu showed that in fact there are only $(n-1)$ distinct s-t minimum cut values in any graph. Moreover there is a single tree that captures all the cut values and minimum cuts. Moreover this tree can be computed via $(n-1)$ minimum cut computations. There have been several recent results yielding faster algorithms.

7.1 A Detour through Submodularity

We have seen that $\alpha_G(s,t)$ can be computed via maximum flow and/or via LP techniques. A different and fundamental way to compute $\alpha_G(s,t)$ is via a connection to *submodularity*.

Definition 7.1. *Given a finite set N, a real-valued set function $f : 2^N \to \mathbb{R}$ is* **submodular** *if for all $A, B \in 2^N$, $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$.*

An alternate definition based on the idea of "decreasing marginal value" is the following:

Definition 7.2. *Given a finite set N, a real-valued set function $f : 2^N \to \mathbb{R}$ is submodular if $f(A+v) - f(A) \geq f(B+v) - f(B)$ for all $A \subseteq B$ and $v \in N \setminus B$.*

Exercise 7.1. Prove the equivalence of the two definitions.

There are several special cases of submodular functions that will be of interest:

1. Monotone submodular functions that satisfy $f(A) \leq f(B)$ for all $A \subseteq B$.
2. Non-negative submodular set functions: $f(A) \geq 0$ for all $A \subseteq N$.
3. Symmetric submodular functions where $f(A) = f(N \setminus A)$ for all $A \subseteq N$.
4. Normalized: $f(\emptyset) = 0$.

We will cover submodularity in more depth later on but we consider the connection to the graph cut function here.

Lemma 7.1. *Let $G = (V, E)$ be a graph with non-negative capacity function $c : E \rightarrow \mathbb{R}^+$. Then $f : 2^V \; to \mathbb{R}^+$ defined by $f(A) = c(\delta(A))$ (i.e., the capacity of a cut induced by a set A) is non-negative, submodular and symmetric.*

Proof. To see this, notice that $f(A) + f(B) = a + b + 2c + d + e + 2f$, for any arbitrary A and B, and a, b, c, d, e, f are as shown in figure 7.1. Here, a (for example) represents the total capacity of edges with one endpoint in A and the other in $V \setminus (A \cup B)$. Also notice that $f(A \cup B) + f(A \cap B) = a + b + 2c + d + e$, and since all values are positive, we see that $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$, satisfying definition 7.1.

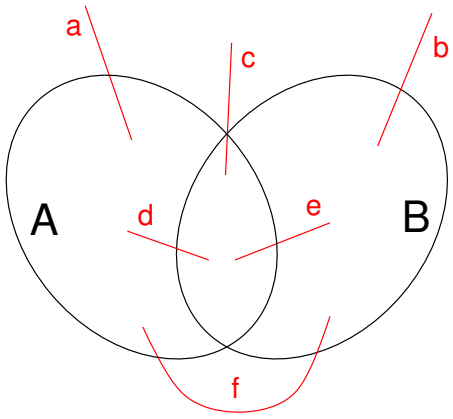


Figure 7.1: Given a graph G and two sets $A, B \subseteq V$, this diagram shows all of the possible classes of edges of interest in G. In particular, there could be edges with both endpoints in $V \setminus (A \cup B)$, A, or B that are not shown here.

■

Exercise 7.2. Suppose f and g are two submodular functions over same ground set N. Prove that $f + g$ is submodular. Argue that if G has a single edge e then the cut function is submodular and use this and the preceding fact to obtain an alternate proof of the submodularity of the cut function of a graph.

Exercise 7.3. Let $G = (V, A)$ be directed graph. Show that the directed cut function $|\delta^+(S)|, S \subseteq V$ is submodular. Note that this is not necessarily symmetric.

Symmetric submodular functions satisfy an additional property.

Definition 7.3. *A real-valued set function $f : 2^N \rightarrow \mathbb{R}_+$ is called* **posi-modular**, *if $f(A) + f(B) \geq f(A - B) + f(B - A)$ for $A, B \subseteq N$.*

Lemma 7.2. *If $f : 2^N \rightarrow \mathbb{R}$ is submodular and symmetric then it is posi-modular.*

Proof.

$$\begin{aligned} f(A) + f(B) = f(V - A) + f(B) &\geq f((V - A) \cap B) + f((V - A) \cup B) \\ &= f(B - A) + f(V - (A - B)) \\ &= f(B - A) + f(A - B). \end{aligned}$$

We use symmetry in the first and last lines above and submodularity in the first line. ■

Submodular set function minimization: Consider a submodular set function $f : 2^N \to \mathbb{R}$. How do we specify f? Note that the function specifies a value for an exponential number of sets. As we saw, the graph cut function is submodular. Thus, in many settings the set function is implicitly defined in some setting. In such settings we will assume that f is available via what is called a *value oracle*: this means that there is a procedure/black box that given $A \subseteq N$ outputs the value $f(A)$. For instance if f is the cut function of a graph G there is a procedure that can efficiently compute $|\delta(A)|$ for any given set A of vertices. A classical and fundamental result in combinatorial optimization that we will see later is the following.

Theorem 7.4. *Given N and a submodular set function $f : 2^N \to \mathbb{Z}$ via a value oracle there is a strongly polynomial-time algorithm[2] that computes $\min_{S \subseteq N} f(S)$ and a set achieving the minimum.*

Via the above general result one obtains an alternate algorithm for computing the minimum s-t cut.

Corollary 7.5. *There is a strongly polynomial time algorithm that given an capacitated directed graph $G = (V, A)$ and s, t outputs $\alpha_G(s, t)$.*

Proof. Consider the function $f : 2^{V-\{s,t\}} \to \mathbb{R}_+$ where $f(A) = |\delta^+(A + s)|$. Note that f is submodular and the minimum of this derived set function is the s-t minimum cut value. ■

Several properties of graph cuts are more easily understood as properties of submodularity. There are two advantages to this. One can simplify certain (parts of) proofs via submodularity and this helps delineate where and whether one is exploiting the specifics of the graph cut function. Second, some results can be shown to hold beyond graphs and then one has a more general result that

[2]Technically speaking the running time consists of two parts. One is the number of calls to the value oracle and the other is the number of arithmetic operations. Both are strongly polynomial in $|N|$. Thus the overall run time is strongly polynomial if one assumes that the value oracle is itself a strongly polynomial time algorithm.

allows for other applications. In the context of Gomory-Hu trees we will see that we only use submodularity and symmetry, and hence there is a Gomory-Hu tree for the cut function defined by an *any* non-negative symmetric submodular function. We will mention an application to hypergraphs later.

Definition 7.6. *Given a symmetric set function $f : 2^V \to \mathbb{R}_+$ and distinct nodes $s, t \in V$ we let $\alpha_f(s,t) = \min_{U \subset V : s \in U, v \in V - U} f(A)$ denote the s-t cut with respect to the function f. Note that α_f is symmetric.*

We state a simple property for cut values that holds in the abstract setting.

Lemma 7.3. *For any symmetric set function $f : 2^V \to \mathbb{R}_+$ and three distinct nodes a, b, c we have $\alpha_f(a,c) \geq \min\{\alpha_f(a,b), \alpha_f(b,c)\}$.*

Proof. Consider a mincut U that separates a from c, that is $a \in U, c \in V - U$ and $\alpha_f(a,c) = f(U)$. Where is b? If $b \in V - U$ then $\alpha_f(a,b) \leq f(U) = \alpha(a,c)$ since U separates a from b. Otherwise, $b \in U$ and hence $\alpha(b,c) \leq f(U) = \alpha(a,c)$. ■

The preceding basic property of the cuts induced by symmetric function suffice to prove that there can be at most $n-1$ distinct mincut values and that there is a compact representation [32, 34] of the cut value. However, one cannot optimize cut values for arbitrary symmetric functions via value oracles and hence it is mainly a structural result. For symmetric submodular functions the results can be made algorithmic and one derives additional properties of the cut tree that also allow us to obtain the value as well as the cuts themselves. The stronger properties are not known to hold true for arbitrary symmetric functions.

7.2 Algorithmic Proof of Gomory-Hu Tree

We start with a definition of a Gomory-Hu tree that satisfies a key property which implies additional stronger properties.

Definition 7.7. *Let $G = (V, E)$ be a graph with non-negative edge-capacities $c : E \to \mathbb{R}_+$. A tree $T = (V(G), E_T)$ is a* **Gomory-Hu tree** *if for all $st \in E_T$, $\delta(W)$ is a minimum s,t cut in G, where W is one component of $T - st$.*

The natural question is whether such a tree even exists; we will return to this question shortly. However, if we are given such a tree for an arbitrary graph G, we know that this tree obeys some very nice properties. In particular, we can label the edges of the tree with the values of the minimum cuts, as the following theorem shows (an example of this can be seen in figure 7.2):

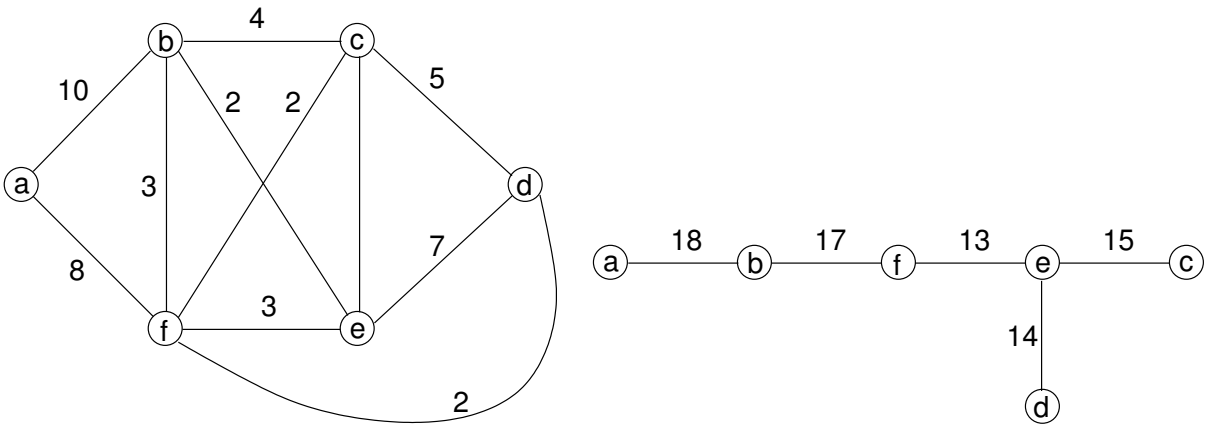


Figure 7.2: A graph G with its corresponding Gomory-Hu tree. Taken from [64].

Theorem 7.8. *Let T be a Gomory-Hu tree for a graph $G = (V, E)$. Then, for all $u, v \in V$, let st be the edge on the unique path in T from u to v such that $\alpha_G(s,t)$ is minimized. Then,*

$$\alpha_G(u,v) = \alpha_G(s,t)$$

and the cut $\delta(W)$ induced by $T - st$ is a u, v minimum cut in G. Thus $\alpha_G(s,t) = \alpha_T(s,t)$ for each $s, t \in V$ where the capacity of an edge st in T is equal to $\alpha_G(s,t)$.

Proof. We first note that α_G obeys a triangle inequality. That is, $\alpha_G(a,b) \geq \min(\alpha_G(a,c), \alpha_G(b,c))$ for any undirected graph G and vertices a, b, c (to see this, note that c has to be on one side or the other of any a, b cut).

Consider the path from u to v in T. We note that if $uv = st$, then $\alpha_G(u,v) = \alpha_G(s,t)$. Otherwise, let $w \neq v$ be the neighbor of u on the u-v path in T. By the triangle inequality mentioned above, $\alpha_G(u,v) \geq \min(\alpha_G(u,w), \alpha_G(w,v))$. If $uw = st$, then $\alpha_G(u,v) \geq \alpha_G(s,t)$; otherwise, by induction on the path length, we have that $\alpha_G(u,v) \geq \alpha_G(w,v) \geq \alpha_G(s,t)$.

However, by the definition of Gomory-Hu trees, we have that $\alpha_G(u,v) \leq \alpha_G(s,t)$, since the cut induced by $T - st$ is a valid cut for u, v. Thus, we have $\alpha_G(u,v) = \alpha_G(s,t)$ and the cut induced by $T - st$ is a u, v minimum cut in G. ■

Remark 7.1. Gomory-Hu trees can be (and are often) defined by asking for the property described in Theorem 7.8. However, the proof shows that the basic requirement in Definition 7.7 implies the other property.

The preceding theorem shows that we can represent compactly all of the minimum cuts in an undirected graph. Several non-trivial facts about undirected

graphs can be seen transparently via the existence of the Gomory-Hu tree. The only remaining question is "Does such a tree exist? And if so, how does one compute it efficiently?" We will answer both questions by giving a constructive proof of Gomory-Hu trees for any undirected graph G.

We now prove the following lemma, which will be instrumental in constructing Gomory-Hu trees.

Lemma 7.4. *Let $\delta(W)$ be an s,t minimum cut in a graph G with respect to a capacity function c. Then for any $u,v \in W, u \neq v$, there is a u,v minimum cut $\delta(X)$ where $X \subseteq W$.*

Proof. Let $\delta(X)$ be any u,v minimum cut that crosses W. Suppose without loss of generality that $s \in W, s \in X$, and $u \in X$. If one of these are not the case, we can invert the roles of s and t or X and $V \setminus X$. Then there are two cases to consider:

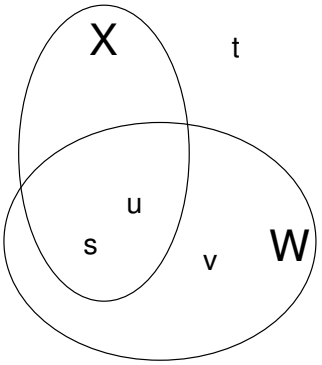


Figure 7.3: $\delta(W)$ is a minimum s,t cut. $\delta(X)$ is a minimum u,v cut that crosses W. This diagram shows the situation in Case 1; a similar picture can be drawn for Case 2

Case 1: $t \notin X$ (see figure 7.3). Then, via submodularity of the cut function,

$$c(\delta(X)) + c(\delta(W)) \geq c(\delta(X \cap W)) + c(\delta(X \cup W)) \tag{7.1}$$

But notice that $\delta(X \cap W)$ is a u,v cut, so since $\delta(X)$ is a minimum u-v cut, we have $c(\delta(X \cap W)) \geq c(\delta(X))$. Also, $X \cup W$ is an s-t cut (since $t \notin X$), so $c(\delta(X \cup W)) \geq c(\delta(W))$. Thus, equality holds in equation (7.1), and $X \cap W$ is a minimum u,v cut.

Case 2: $t \in X$. Since c is posi-modular, we have that

$$c(\delta(X)) + c(\delta(W)) \geq c(\delta(W \setminus X)) + c(\delta(X \setminus W)) \tag{7.2}$$

However, $\delta(W \setminus X)$ is a u, v cut, so $c(\delta(W \setminus X)) \geq c(\delta(X))$. Similarly, $\delta(X \setminus W)$ is an s, t cut, so $c(\delta(X \setminus W)) \geq c(\delta(W))$. Therefore, equality holds in equation (7.2), and $W \setminus X$ is a u, v minimum cut.

■

The preceding argument shows that minimum cuts can be **uncrossed**, a technique that is useful in many settings. In order to construct a Gomory-Hu tree for a graph, we need to consider a stronger definition to apply induction.

Definition 7.9. *Let $G = (V, E), R \subseteq V$. Then a* **Gomory-Hu tree for R in G** *is a pair consisting of $T = (R, E_T)$ and a partition $(C_r \mid r \in R)$ of V associated with each $r \in R$ such that*

1. *For all $r \in R$, $r \in C_r$*
2. *For all $st \in E_T$, $T - st$ induces a minimum cut in G between s and t defined by*

$$\delta(U) = \bigcup_{r \in X} C_r$$

where X is the vertex set of a component of $T - st$.

Notice that a Gomory-Hu tree for G is simply a generalized Gomory-Hu tree with $R = V$. Intuitively, we associate with each vertex v in the tree a "bag" that contains all of the vertices that have to appear on the same side as v in some minimum cut. This allows us to define the algorithm GomoryHuAlg which is a simple divide and conquer algorithm. It is based on the key lemma. We pick an arbitrary pair r_1, r_2 from R and compute a mincut W that separates r_1 and r_2. Let $R_1 = R \cap W$ and $R_2 = R \setminus R_1 = R \cap (V - W)$. By the key lemma, for any $u, v \in R_1$ there is a u-v mincut X completely contained in W and similarly if $u, v \in R_2$ there is a mincut is completely contained in $V - W$. This justifies divide and conquer where we shrink the other side of the cut to a single vertex. The key issue is how to attach the two trees together and it requires some care.

Claim 7.2.1. *Let $u, v \in R_1$. Then $\alpha_{G_1}(u, v) = \alpha_G(u, v)$. Similarly, let $u, v \in R_2$ then $\alpha_{G_2}(u, v) = \alpha_G(u, v)$.*

Proof. Follows from Lemma 13.3. ■

Algorithm 1 GomoryHuAlg(G, R)

if $|R| = 1$ **then return** $T = (\{r\}, \emptyset), C_r = V$
else
Let $r_1, r_2 \in R$, and let $\delta(W)$ be an r_1, r_2 minimum cut

▷ Create two subinstances of the problem
$G_1 = G$ with $V \setminus W$ shrunk to a single vertex, v_1; $R_1 = R \cap W$
$G_2 = G$ with W shrunk to a single vertex, v_2; $R_2 = R \setminus W$

▷ Now we recurse
$T_1, (C_r^1 \mid r \in R_1) =$ GomoryHuAlg(G_1, R_1)
$T_2, (C_r^2 \mid r \in R_2) =$ GomoryHuAlg(G_2, R_2)

Let r' be the vertex such that $v_1 \in C_{r'}^1$,
Let r'' be the vertex such that $v_2 \in C_{r''}^2$,
▷ Note that r', r'' are not necessarily r_1, r_2!

▷ See figure 7.4
$T = (R_1 \cup R_2, E_{T_1} \cup E_{T_2} \cup \{r'r''\})$
$(C_r \mid r \in R) =$ ComputePartitions$(R_1, R_2, C_r^1, C_r^2, r', r'')$ **return** T, C_r
end if

Theorem 7.10. *GomoryHuAlg returns a valid Gomory-Hu tree for a set R.*

Proof. We need to show that any $st \in E_T$ satisfies the "key property" of Gomory-Hu trees. That is, we need to show that $T - st$ induces a minimum cut in G between s and t. The base case is trivial. Then, suppose that $st \in T_1$ or $st \in T_2$. By Claim 7.2.1 and induction we see that the key property is satisfied for all edges in T_1 and T_2.

Thus, the only edge we need to care about is the edge we added from r' to r''. Note $T - (r', r'')$ corresponds to the cut $(W, V - W)$ in G. Thus, to prove correctness of T it suffices to prove that the cut $\delta(W)$ is a r'-r'' mincut in G. In particular it suffices to argue that $\alpha_G(r', r'') \geq \alpha_G(r_1, r_2)$. Note that $\alpha_G(r', r'') \leq \alpha_G(r_1, r_2)$ since $\delta(W)$ is a valid cut that separates r' form r''. First, consider the simple case when $\alpha_G(r_1, r_2)$ is minimum over all pairs of vertices in R. In this case, we see that in particular, $\alpha_G(r_1, r_2) \leq \alpha_G(r', r'')$, so we are done.

However, we chose r_1, r_2 arbitrarily so we need a more general argument. Suppose there is a r'-r'' minimum cut $\delta(X)$ such that $c(\delta(X)) < (\delta(W))$. Assume without loss of generality that $r' \in X, r'' \notin X$. Note that $r_1, r' \in W$. X does not separate r_1 from r_2, otherwise we will contradict the fact that $\delta(W)$ is a mincut

Algorithm 2 ComputePartitions($R_1, R_2, C^1_r, c^2_r, r', r''$)

▷ We use the returned partitions, except we remove v_1 and v_2 from $C_{r'}$ and $C_{r''}$, respectively
For $r \in R_1, r \neq r', C_r = C^1_r$
For $r \in R_2, r \neq r'', C_r = C^2_r$
$C_{r'} = C^1_{r'} - \{v_1\}, C_{r''} = C^2_{r''} - \{v_2\}$ **return** $(C_r \mid r \in R)$

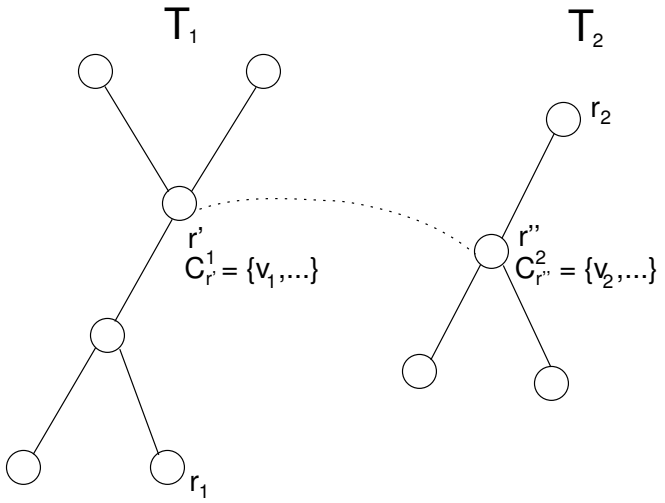


Figure 7.4: T_1 and T_2 have been recursively computed by GomoryHuAlg. Then we find r' and r'' such that v_1 (the shrunken vertex corresponding to $V \setminus W$ in T_1) is in the partition of r', and similarly for r'' and v_2. Then, to compute T, we connect r' and r'', and recompute the partitions for the whole tree according to ComputePartitions.

for them. Therefore, both $r_1, r_2 \in X$ or neither. We will assume $r_1, r_2 \notin X$. The other case is similar. Thus $r' \in X$ and $r_1 \notin X$. Thus X separates r' from r_1. Via the key lemma, there is a mincut separating r' and r_1 which is completely contained in W. This means we can assume without loss of generality that $X \subseteq W$ and hence $\alpha_{G_1}(r', r_1) \leq c(\delta(X)) < c(\delta(W)) = \alpha_G(r_1, r_2)$.

Now, consider the path from r' to r_1 in T_1. There exists an edge uv on this path such that the weight of uv in T_1, $w_1(uv)$, is $\alpha_{G_1}(r', r_1)$. Because T_1 is a Gomory-Hu tree for G_1, uv induces an r_1, r_2 cut in G of capacity $w_1(uv)$ (since $v_1 \in C^1_{r'}$). But this contradicts the fact that W is a r_1, r_2 minimum cut.

This proves that T is a Gomory-Hu tree for G, and hence by induction, GomoryHuAlg is correct. ■

We obtain the following corollary:

Corollary 7.11. *A Gomory-Hu tree for $R \subseteq V$ in G can be computed in the time needed to compute $|R| - 1$ minimum-cuts in graphs of size at most that of G.*

Finally, we present the following alternative proof of the last step of theorem 7.10 (that is, showing that we can choose r_1 and r_2 arbitrarily in GomoryHuAlg). As before, let $\delta(W)$ be an r_1, r_2 minimum cut, and assume that $r_1 \in W, r_2 \in V \setminus W$. Assume for simplicity that $r_1 \neq r'$ and $r_2 \neq r''$ (the other cases are similar). We claim that $\alpha_{G_1}(r_1, r') = \alpha_G(r_1, r') \geq \alpha_G(r_1, r_2)$. To see this, note that if $\alpha_{G_1}(r_1, r') < \alpha_G(r_1, r_2)$, there is an edge $uv \in E_{T_1}$ on the path from r_1 to r' that has weight less than $\alpha_G(r_1, r_2)$, which gives a smaller r_1, r_2 cut in G than W (since $v_1 \in C^1_{r'}$). For similar reasons, we see that $\alpha_G(r_2, r'') \geq \alpha_G(r_1, r_2)$.

Thus, by the triangle inequality we have

$$\alpha_G(r', r'') \geq \min\{\alpha_G(r', r_1), \alpha_G(r'', r_2), \alpha_G(r_1, r_2)\} \geq \alpha_G(r_1, r_2)$$

which completes the proof.

Gomory-Hu trees allow one to easily show some facts that are otherwise hard to prove directly. Some examples are the following.

Exercise 7.4. For any undirected graph there is a pair of nodes s, t and an s-t minimum cut consisting of a singleton node (either s or t). Such a pair is called a *pendant* pair.

Exercise 7.5. Let G be a graph such that $\deg(v) \geq k$ for all $v \in V$. Show that there is some pair s, t such that $\alpha_G(s, t) \geq k$.

Notice that the proof of the correctness of the algorithm relied only on the key lemma which in turn used only the symmetry and submodularity of the cut function. One can directly extend the proof to show the following theorem.

Theorem 7.12. *Let V be a ground set, and let $f : 2^V \to \mathbb{R}^+$ be a symmetric submodular function. Given s, t in V, define the minimum cut between s and t as*

$$\alpha_f(s, t) = \min_{W \subseteq V, |W \cap \{s,t\}|=1} f(W)$$

Then, there is a Gomory-Hu tree that represents α_f. That is, there is a tree $T = (V, E_T)$ and a capacity function $c : E_T \to \mathbb{R}^+$ such that $\alpha_f(s, t) = \alpha_T(s, t)$ for all s, t in V, and moreover, the minimum cut in T induces a minimum cut according to f for each s, t.

Exercise 7.6. Let V be a ground set, and let $f : 2^V \to \mathbb{R}^+$ be a symmetric submodular function. Suppose $R \subseteq V$. Define a a function $g : 2^R \to \mathbb{R}^+$ where $g(A) = \min_{A \subseteq S \subseteq V - A} f(S)$. In other words we are deriving a function on the ground set R via f. Prove that g is symmetric and submodular.

Exercise 7.7. A hypergraph $G = (V, \xi)$ consists of a finite vertex set V and a set of hyperedges ξ where each hyperedge $e \in \xi$ is a subset of V, that is, $e \subseteq V$. Graphs are hypergraphs where each hyperedge has cardinality two. Given a hypergraph $G = (V, \xi)$, define the cut function of G as: $f : 2^V \to \mathbb{R}^+$ as $f(W) = |\delta(W)|$, where $S \in \xi$ is in $\delta(W)$ iff $S \cap W$ and $S \setminus W$ are non-empty. Show that f is a non-negative, symmetric, submodular function.

Remark 7.2. Isolating cuts from [45] and other ideas have led to a spate of new results on graphs, submodular functions, hypergraphs, and related problems and faster algorithms for Gomory-Hu tree computation in both the exact and approximate setting.

Chapter 8

Perfect Matching and Matching Polytopes[1]

Let $G = (V, E)$ be a graph. For a set $E' \subseteq E$, let $\chi^{E'}$ denote the characteristic vector of E' in $\mathbb{R}^{|E|}$. We define two polytopes:

$$\mathcal{P}_{perfect\_matching}(G) = \textbf{convexhull}(\{\chi^M \mid M \text{ is a perfect matching in } G\})$$

$$\mathcal{P}_{matching}(G) = \textbf{convexhull}(\{\chi^M \mid M \text{ is a matching in } G\})$$

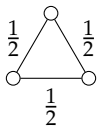
Edmonds gave a description of these polytopes. Recall that for bipartite graphs, $\mathcal{P}_{perfect\_matching}(G)$ is given by the following polytope.

$$\begin{aligned} x(\delta(v)) &= 1 \quad \forall v \in V \\ x(e) &\geq 0 \quad \forall e \in E \end{aligned}$$

And $\mathcal{P}_{matching}(G)$ is given by the following polytope.

$$\begin{aligned} x(\delta(v)) &\leq 1 \quad \forall v \in V \\ x(e) &\geq 0 \quad \forall e \in E \end{aligned}$$

We saw an example of a non-bipartite graph, namely a tringle, for which $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ is basic feasible solution for both of the preceding polytopes.



[1]Based on notes scribed by Vivek Srikumar from 2010.

Hence, (perfect) matching polytope for non-bipartite graphs are not captured by the simple constraints that work for bipartite graphs.

Theorem 8.1 (Edmonds). *$\mathcal{P}_{perfect\_matching}(G)$ is determined by the following set of inequalities.*

$$\begin{aligned} x(e) &\geq 0; && e \in E \\ x(\delta(v)) &= 1; && v \in V \\ x(\delta(U)) &\geq 1; && U \subseteq V, |U| \geq 3, |U| \text{ odd} \end{aligned}$$

Edmonds gave a proof via an algorithmic method. In particular, he gave a primal-dual algorithm for the minimum cost perfect matching problem, which, as a by product showed that for any cost vector c on the edges, there is a minimum cost perfect matching whose cost is equal the minimum value of cx subjecto to the above set of inequalities. This implies that the polytope is integral. We describe a short non-algorithmic proof that was given later [57] (Chapter 25).

Proof. Let $Q(G)$ denote the polytope described by the inequalities in the theorem statement. It is easy to verify that for each graph G, $\mathcal{P}_{perfect\_matching}(G) \subseteq Q(G)$. Suppose there is a graph G such that $Q(G) \not\subseteq \mathcal{P}_{perfect\_matching}(G)$. Among all such graphs, choose the one that minimizes $|V| + |E|$. Let G be this graph. In particular, there is a basic feasible solution (vertex) x of $Q(G)$ such that x is not in $\mathcal{P}_{perfect\_matching}(G)$.

We claim that $x(e) \in (0, 1)$; $\forall e \in E$. If $x(e) = 0$ for some e, then deleting e from G gives a smaller counter example. If $x(e) = 1$ for some e, then deleting e and its end points from G gives a smaller counter example.

We can assume that $|V|$ is even, for otherwise $Q(G) = \emptyset$ and $\mathcal{P}_{perfect\_matching}(G) = \emptyset$ as well (why?). Since $0 < x(e) < 1$ for each e and $x(\delta(v) = 1$ for all v, $deg(v) \geq 2$; $\forall v \in V$. Suppose $|E| = 2|V|$. Then $deg(v) = 2$; $\forall v \in V$ and therefore, G is a collection of vertex disjoint cycles. Then, either G has an odd cycle in its collection of cycles, in which case, $Q(G) = \emptyset = \mathcal{P}_{perfect\_matching}(G)$, or G is a collection of even cycles and, hence bipartite and $Q(G) = \mathcal{P}_{perfect\_matching}(G)$.

Thus $|E| > 2|V|$. Since x is a vertex of $Q(G)$, there are $|E|$ inequalities in the system that are tight and determine x. Therefore there is some odd set $U \subset V$ such that $x(\delta(U)) = 1$. Here $|U| \geq 3$. Note that $|V - U| > 1$ for if $V - U = \{v\}$ the inequality $x(\delta(U)) = 1$ is implied by $x(\delta(v)) = 1$. Let $G' = G/U$, where U is shrunk to a node, say u'. Define $G'' = G/\bar{U}$, where $\bar{U} = V - U$ is shrunk to a node u''; see Figure 8.1.

The vector x when restricted to G' induces $x' \in Q(G')$ and similarly x induces $x'' \in Q(G'')$. Since G' and G'' are smaller than G, we have that $Q(G') = \mathcal{P}_{perfect\_matching}(G')$ and $Q(G'') = \mathcal{P}_{perfect\_matching}(G'')$. Hence, x' can

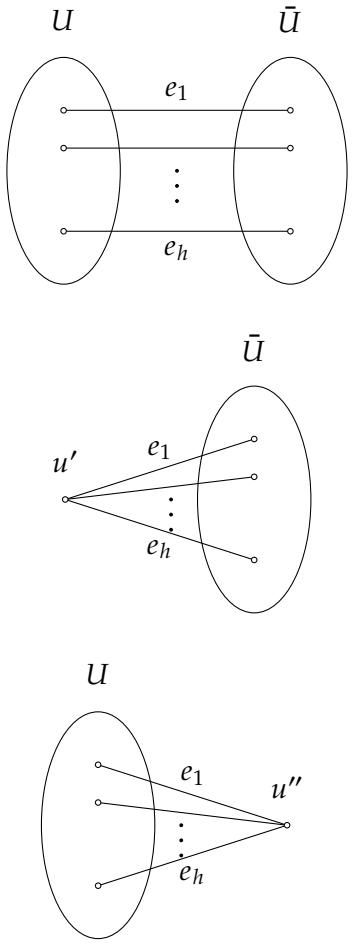


Figure 8.1: Graphs G, G' and G'' from top to bottom.

be written as a convex combination of perfect matchings in G' and x'' can be written as a convex combination of perfect matchings in G''. The vector x is rational since we chose it as a vertex of $Q(G)$, therefore, x', x'' are also rational; hence, $\exists$ integer k such that $x' = \frac{1}{k}\sum_{i=1}^{k} \chi^{M'_i}$, where $M'_1, M'_2, \cdots, M'_k$ are perfect matchings in G' and $x'' = \frac{1}{k}\sum_{i=1}^{k} \chi^{M''_i}$, where $M''_1, M''_2, \cdots, M''_k$ are perfect matchings in G''. (Note that k is the same in both expressions.)

Let $e_1, e_2, \cdots, e_h$ be edges in $\delta(U)$. Since $x'(\delta(u') = 1$ and u' is in every perfect matching, we have that e_j is in exactly $kx'(e_j) = kx(e_j)$ matchings $M'_1, \cdots, M'_k$. Similarly, e_j is in exactly $kx(e_j)$ matchings $M''_1, \cdots, M''_k$. Note that $\sum_{j=1}^{n} kx(e_j) = k$ and moreover, exactly one of $e_1, \cdots, e_h$ can be in M'_i and M''_i. We can, therefore,

assume (by renumbering if necessary) that M_i' and M_i'' share exactly one edge from $e_1, \cdots, e_h$. Then, $M_i = M_i' \cup M_i''$ is a perfect matching in G. Hence, $x = \frac{1}{k}\sum_{i=1}^{k} \chi^{M_i}$, which implies that $x \in \mathcal{P}_{perfect\_matching}(G)$, contradicting our assumption.

The proof can alternatively be viewed as giving an inductive proof to express any vertex x of $Q(G)$ as a convex combination of perfect matchings of G. ■

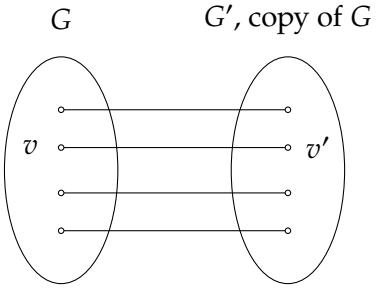
Now, we use the above theorem to derive the following:

Theorem 8.2. *$\mathcal{P}_{matching}(G)$ is determined by*

$$\begin{aligned} x(e) &\geq 0; && e \in E \\ x(\delta(v)) &\leq 1; && v \in V \\ x(E[U]) &\leq \frac{|U|-1}{2}; && U \subseteq V, |U| \text{ odd} \end{aligned}$$

Here $E[U]$ is the set of edges with both end points in U.

Proof. We use a reduction of weighted matching to weighted perfect matching: Given $G = (V, E)$, create a copy $G' = (V', E')$ of G. And let $\tilde{G}$ be the graph $(\tilde{V}, \tilde{E})$ defined as $\tilde{V} = V \cup V'$, $\tilde{E} = E \cup E' \cup \{(v, v') \mid v \in V\}$.



The following claim is easy to prove.

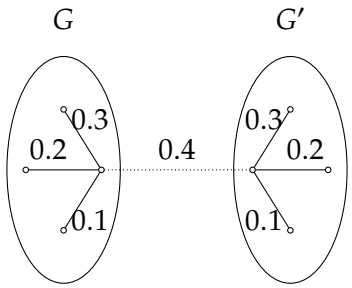
Claim 8.0.1. *Suppose M is a matching in G. Then there is a perfect matching $\tilde{M}$ in $\tilde{G}$ such that $\tilde{M} \cap E = M$. Suppose $\tilde{M}$ is a perfect matching in $\tilde{G}$. Then $\tilde{M} \cap E$ is a matching in G.*

Corollary 8.3. *The maximum weight matching problem is poly-time equivalent to maximum weight perfect matching problem.*

Proof. Given a graph G in which we wish to compute a maximum-weight matching we first remove edges with negative weights and then construct $\tilde{G}$ as above where we assign a zero weight to each edge vv' and retain the original edge weights in G and set edge-weights in G' to zero. ■

We can use the above idea to establish the theorem. Let x be feasible for the system of inequalities in the theorem. We show that x can be written a convex combination of matchings in G. It is clear that χ^M satisfies the inequalities for every matching M. From G, create $\tilde{G}$ as above and define a fractional solution $\tilde{x} : \tilde{E} \rightarrow \mathbb{R}^+$ as follows: first, we define $x' : E' \rightarrow \mathbb{R}^+$ as the copy of x on E. That is, $x'(e') = x(e)$, where e' is the copy of e. Then,

$$\tilde{x} = \begin{cases} x(e); \text{ if } e \in E \\ x'(e); \text{ if } e \in E' \\ 1 - x\left(\delta(v)\right); \text{ if } e = vv' \end{cases}$$

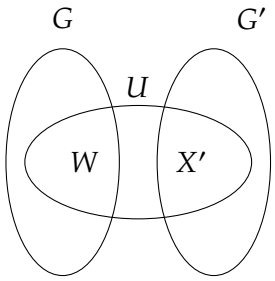


Claim 8.0.2. *$\tilde{x}$ belongs to $\mathcal{P}_{perfect\_matching}(\tilde{G})$.*

Assuming the claim, we see that $\tilde{x}$ can be written as a convex combination of perfect matchings in $\tilde{G}$. Each perfect matching in $\tilde{G}$ induces a matching in G and it is easy to verify that x can therefore be written as a convex combination of matchings in G.

It only remains to verify the claim. From the previous theorem, it suffices to show that

$$\tilde{x}\left(\tilde{\delta}(U)\right) \geq 1; \;\; \forall U \subseteq \tilde{V}, |U| \text{ odd}$$

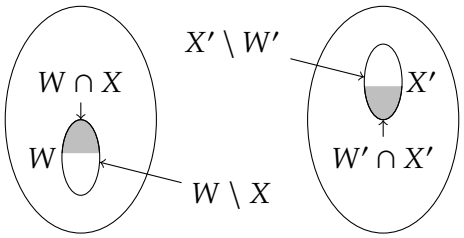


Let $U \subseteq \tilde{V}$ and $|U|$ odd. Let $W = U \cap V$ and $X' = U \cap V'$, where X' is the copy of $X \subseteq V$. First we consider the case that $X' = \emptyset$ and $|W|$ is odd. Then

$$\begin{aligned}
\tilde{x}\left(\tilde{\delta}(U)\right) &= \tilde{x}\left(\tilde{\delta}(W)\right) \\
&= \sum_{v \in W} \tilde{x}\left(\tilde{\delta}(v)\right) - 2\tilde{x}\left(E[W]\right) \\
&= |W| - 2x\left(E[W]\right) \\
&\geq |W| - 2\left(\frac{|W|-1}{2}\right) \\
&\geq 1
\end{aligned}$$

For the general case, we claim that $\tilde{x}\left(\tilde{\delta}(U)\right) \geq \tilde{x}\left(\tilde{\delta}(W \setminus X)\right) + \tilde{x}\left(\tilde{\delta}(X' \setminus W')\right)$. Without loss of generality, $W \setminus X$ is odd. Then $\tilde{x}\left(\tilde{\delta}(U)\right) \geq \tilde{x}\left(\tilde{\delta}(W \setminus X)\right) \geq 1$ from above.

The claim can be verified as follows:



Notice that only edges between W and X' are between $W \cap X$ and $X' \cap W'$. Let $A = W \cap X, A' = W' \cap X'$. Then

$$\begin{aligned}
\tilde{x}\left(\tilde{\delta}(U)\right) &= \tilde{x}\left(\tilde{\delta}(W \cup X')\right) \\
&= \tilde{x}\left(\tilde{\delta}(W \setminus X)\right) + \tilde{x}\left(\tilde{\delta}(X' \setminus W')\right) + \\
&\quad x\left(\delta(A)\right) - 2x((\delta\left(E[A, W \setminus X]\right)) + \\
&\quad x\left(\delta'(A')\right) - 2x\left(\delta'\left(E[A', X' \setminus W']\right)\right)
\end{aligned}$$

The claim follows from the observation that $x(\delta(A)) \geq x\left(E[A, W \setminus A]\right) + x\left(\delta(E[A, X \setminus W]\right)$. ■

Corollary 8.4. *$\mathcal{P}_{perfect\_matching}(G)$ is also determined by*

$$\begin{aligned}
x(e) &\geq 0; && e \in E \\
x\left(\delta(v)\right) &= 1; && v \in V \\
x\left(E[U]\right) &\leq \frac{|U|-1}{2}; && U \subseteq V, |U| \text{ odd}
\end{aligned}$$

We note that although the system in the above corollary and the earlier theorem both determine $\mathcal{P}_{perfect\_matching}(G)$, they are not identical.

8.1 Separation Oracle for Matching Polytope

The inequality systems that we saw for $\mathcal{P}_{perfect\_matching}(G)$ and $\mathcal{P}_{matching}(G)$ have an exponential number of inequalities. Therefore, we cannot use them directly to solve the optimization problems of interest, namely, the maximum weight matching problem or the minimum weight perfect matching problem. To use the Ellipsoid method, we need a polynomial time separation oracle for the polytopes. Edmonds gave efficient strongly polynomial time algorithms for optimizing over these polytopes via the primal-dual method. From the equivalence of optimization and separation (via the ellipsoid method), this implies that there are polynomial time separation oracles for these polytopes. However, the oracle obtained via the above approach is indirect and cumbersome. Padberg and Rao [1982] gave a simple and direct separation oracle. We discuss this for the system

$$\begin{aligned} x(e) &\geq 0; & e &\in E \\ x(\delta(v)) &= 1; & v &\in V \\ x(\delta(U)) &\geq 1; & |U| \text{ odd}, U &\subseteq V \end{aligned} \tag{8.1}$$

and it can be used to obtain a separation oracle for for the matching polytope via the reduction we discussed earlier.

Theorem 8.5. *There is a strongly polynomial time algorithm, that given $G = (V, E)$ and $x : E \rightarrow \mathbb{R}$ determines if x satisfies (8.1) or outputs an inequality from (8.1) that is violated by x.*

It is trivial to check the first two sets of inequalities. Therefore, we assume that $x \geq 0$ and $x(\delta(v)) = 1; \forall v \in V$. We can also assume that $|V|$ is even otherise the perfect matching polytope is empty since V itself is violated. Thus the question is whether there is a set $U \subset V, |U|$ odd, such that $x(\delta(U)) < 1$. It is sufficient to give an algorithm for the *minimum odd-cut problem*, which is the following: Given a capacitated graph $G = (V, E)$, find a cut $\delta(U)$ of minimum capacity among all sets U such that $|U|$ is odd. We claim that the following is a correct algorithm for the minimum odd-cut problem.

1. Compute a Gomory-Hu tree $T = (V, E_T)$ for G with edge capacities given by x.
2. Among the odd-cuts induced by the edges of T, output the one with the minimum capacity.

To see the correctness of the algorithm, let $\delta(U^*)$ be a minimum capacity odd cut in G. Then $\delta_T(U^*)$ is a set of edges in E_T.

Claim 8.1.1. *There is an edge $st \in \delta_T(U^*)$ such that $T - st$ has a component with an odd number of nodes.*

The proof of the preceding claim is left as an exercise. Assuming the claim, by the prperties of the Gomory-Hu tree, $T - st$ induces an odd cut in G of capacity equal to $\alpha_G(s,t)$ (recall that $\alpha_G(s,t)$ is the capacity of a minimum s-t cut in G). Since $\delta(U^*)$ separates s and t, the odd cut induced by $T - st$ has capacity at most $x(\delta(U^*))$.

Exercise 8.1. Given a graph $G = (V, E)$ and an even cardinality subset $S \subseteq V$ of vertices, the odd-S-cut problem is to find a cut $\delta(U)$ in G of minimum capacity such that $|U \cap S|$ is odd. Derive a poly-time algorithm via the Gomory-Hu tree for this problem. Note that we considered the case when $S = V$.

8.2 Edge Covers and Matchings

Given $G = (V, E)$ an edge cover is the subset $E' \subseteq E$ such that each node is covered by some edge in E'. This is the counterpart to vertex cover. Edge covers are closely related to matchings and hence optimization problems related to them are tractable, unlike the vertex cover problem whose minimization version is NP-Hard. Note that the Edge Cover problem can be viewed as a special case of the Set Cover problem where each set has size exactly two (the case with sets of size two or one can be reduced to the case with size exactly two). The Set Cover problem when each set has size exactly three is NP-Complete (we can view this as Edge Cover in a rank 3 hypergraph).

Theorem 8.6 (Gallai)**.** *Let $\rho(G)$ be the cardinality of a minimum edge cover in G. Then*

$$\nu(G) + \rho(G) = |V|$$

where $\nu(G)$ is the cardinality of a maximum matching in G.

Proof. Take any matching M in G. Then M covers $2|M|$ nodes, the end points of M. There are at most $|V| - 2|M|$ uncovered nodes. For each such uncovered node pick an arbitrary edge to cover it. This gives an edge cover of size $\leq |V| - 2|M| + |M| \leq |V| - |M|$. Hence $\rho(G) \leq |V| - \nu(G)$.

We now show that $\nu(G) + \rho(G) \geq |V|$. Let E' be any inclusion-wise minimal edge cover and let M be an inclusion-wise maximal matching in E'. If v is not incident to an edge of M then since it is covered by E' there is an edge $e_v \in E' \setminus M$ that covers v; since M is maximal the other end point of e_v is covered by M. This

implies that $2|M| + |E' \setminus M| \geq |V|$, that is $2|M| + |E'| - |M| \geq |V|$ and hence $|M| + |E'| \geq |V|$. If E' is a minimum edge cover then $|E'| = \rho(G)$ and $|M| \leq \nu(G)$, therefore, $\nu(G) + \rho(G) \geq |V|$. ■

The above proof gives an efficient algorithm to compute $\rho(G)$ and also a minimum cardinality edge cover via an algorithm for maximum cardinality matching.

Weighted case: Consider the minimum weight edge cover problem. Let $w(e)$ denote the weight of edge e. Note that weights can be negative. We can use minimum weight perfect matching to solve this problem via a reduction. Consider the graph $\tilde{G}$ that we saw earlier when reducing maximum weight matching to maximum weight perfect matching. $\tilde{G}$ consists of two copies of G, namely $G = (V, E)$ and $G' = (V', E')$ and we add a perfect matching between V and V'. We will use the same graph but with different weights. For each edge $e = uv$ of G we set its weight to be same as $w(e)$ in the original graph G; similarly if e' is a copy of e then its weight is also the same as that of e. Each edge vv' has its weight set $2\mu(v)$ where $\mu(v)$ is the weight of the least weight edge incident to v in G (note that if any v is a isolated vertex in G then there is no feasible edge cover in G). Computing an minimum weight edge cover in G corresponds to computing a minimum weight perfect matching in $\tilde{G}$.

Exercise 8.2. Complete the details of the preceding reduction and show that this leads to a polynomial-time algorithm for computing the minimum weight edge cover in a graph.

Polytope: The following set of inequalities determine the edge cover polytope (the convex hull of the characterstic vectors of edge covers in G).

$$\begin{aligned} x(\delta(V)) &\geq 1 && \forall v \in V \\ x(E[U] \cup \delta(U)) &\geq \tfrac{|U|+1}{2} && U \subseteq V; |U| odd \\ 0 \leq x(e) &\leq 1; && e \in E \end{aligned}$$

Exercise 8.3. Prove that the polytope above is the edge cover polytope and obtain a polynomial time separation oracle for it.

Chapter 9

Edmonds-Gallai Decomposition and Factor-Critical Graphs

This material is based on notes of Michel Goemans and also borrows from [57] (Chapter 24).

Recall the Tutte-Berge formula for the size of a maximum matching in a graph G.

Theorem 9.1 (Tutte-Berge)**.** *Given a graph G, the size of a maximum cardinality matching in G, denoted by $\nu(G)$, is given by:*

$$\nu(G) = \min_{U \subseteq V} \frac{1}{2}(|V| + |U| - o(G - U))$$

where $o(G-U)$ is the number of connected components in $G[V \setminus U]$ with odd cardinality.

We call a set U that achieves the minimum on the right hand side of the Tutte-Berge formula, a Tutte-Berge witness set. Such a set U gives some information on the set of maximum matchings in G. In particular we have the following.

- All nodes in U are covered in every maximum matching of G.
- If K is the vertex set of a component of $G - U$, then every maximum matching in G covers at least $\lfloor K/2 \rfloor$ nodes in K. In particular, every node in an even component is covered by every maximum matching.

A graph can have different Tutte-Berge witness sets as the example in Fig **??** shows. The witness set $U = \{v\}$ is more useful than $U = \emptyset$ since it gives more information on which of the vertices are in every maximum matching.

A natural question is whether each graph has a *canonical* Tutte-Berge witness set that gives us as much information as possible. The Edmonds-Gallai decomposition shows that there is such a canonical witness set. Before we describe the

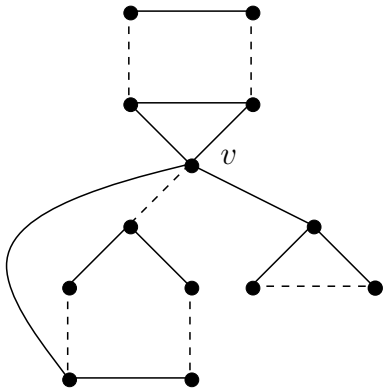


Figure 9.1: Graph G above has 13 nodes, and $\nu(G) = 6$. $U = \emptyset$ and $U = \{v\}$ are both Tutte-Berge witness sets.

theorem, we digress to describe some other settings with canonical witness sets. The reader can skip the next paragraph.

Let $D = (V, A)$ be a directed graph and $s, t \in V$. It is easy to see that s has no directed path to t iff there is a set $X \subseteq V$ such that $s \in X$, $t \notin X$ and $\delta^+(X) = \emptyset$, that is no arcs leave X. Among all such sets X, the set X^* defined as the set of all nodes reachable from s, is a canonical set. It is a simultaneous witness for all nodes that are not reachable from s. Moreover, most algorithms for checking reachability of t from s would output X^*. Similarly, consider the s-t maximum flow problem in a capacitated directed graph D. By the maxflow-mincut theorem, the maximum flow value F is equal to the capacity of a minimum capacity cut $\delta(X)$ that separates s from t. Again, there could be multiple minimum cuts. One can show that if $\delta(X)$ and $\delta(Y)$ are s-t minimum cuts (here X and Y contain s and do not contain t) then $\delta(X \cap Y)$ and $\delta(X \cup Y)$ are also minimum cuts (follows from submodularity of the cut function). From this, it follows that there exists a unique minimal minimum cut $\delta(X^*)$ and a unique maximal minimum cut $\delta(Y^*)$. We note that X^* is precisely the set of vertices reachable from s in the residual graph of *any* maximum flow; similarly $V \setminus Y^*$ is the set of nodes that can reach t in the residual graph of any maximum flow.

9.0.1 Factor-Critical Graphs

Suppose G has a perfect matching. Then every vertex in every matching and there is no information in a Tutte-Berge witness set and we may as well take $U = \emptyset$. It is therefore interesting to examine graphs that do *not* contain a perfect

matching. In graph theory it is common to consider criticality with respect to having a property.

Definition 9.2. *A graph $G = (V, E)$ is* factor-critical[1] *if G has no perfect matching but for each $v \in V$, $G - v$ has a perfect matching.*

Factor-critical graphs are connected and have an odd number of vertices. Two simple examples are (i) an odd cycle and (ii) a complete graph on an odd number of vertices.

Theorem 9.3. *A graph G is factor-critical if and only if for each node v there is a maximum matching that misses v.*

Proof. If G is factor-critical then $G - v$ has a perfect matching and hence a maximum matching in G. We saw the converse direction in the proof of the Tutte-Berge formula — it was shown that if each node v is missed by some maximum matching then G has a matching of size $(|V| - 1)/2$. ■

If U is a non-empty Tutte-Berge witness set for a graph G, then it follows that there are nodes in G that are covered in every maximum matching. If G is factor-critical then $U = \emptyset$ is the unique Tutte-Berge witness set for G for otherwise there would be a node that is in every maximum matching. In fact the converse is also true, but is not obvious. It is an easy consequence of the Edmonds-Gallai decomposition to be seen shortly. We give a useful fact about factor-critical graphs.

Proposition 9.0.1. *Let C be an odd cycle in G. If the graph G/C, obtained by shrinking C into a single vertex, is factor-critical then G is factor-critical.*

Proof. Proof sketch: Let c denote the vertex in G/C in place of the shrunken cycle C. Let v be an arbitrary node in $V(G)$. We need to show that $G - v$ has a perfect matching.

If $v \notin C$ then $G/C - v$ has a perfect matching M that matches c, say via edge cu. When we unshrink c into C, let w be the vertex in C that corresponds to the edge cu. We can extend M a perfect matching in $G - v$ by adding edges in the even length path $C - w$ to cover all the nodes in $C - w$.

If $v \in C$, consider a perfect matching M in $G/C - c$. It is again easy to extend M to a perfect matching in $G - v$ by considering $C - v$. ■

We will see later a structural characterization of factor-critical graphs via ear decompositions.

[1] In graph theory a k-factor of a graph G is a subgraph in which each vertex has degree exactly k. In particular 1-factor is a perfect matching. This is the reason for the terminology of factor-critical.

9.1 Edmonds-Gallai Decomposition

Theorem 9.4 (Edmonds-Gallai)**.** *Given a graph $G = (V, E)$, let*

$$\begin{aligned} D(G) &:= \{v \in V \mid \text{there exists a maximum matching that misses } v\} \\ A(G) &:= \{v \in V \mid v \text{ is a neighbor of } D(G) \text{ but } v \notin D(G)\} \\ C(G) &:= V \setminus (D(G) \cup A(G)). \end{aligned}$$

Then, the following hold.

1. *The set $U = A(G)$ is a Tutte-Berge witness set for G.*
2. *$C(G)$ is the union of the even components of $G - A(G)$.*
3. *$D(G)$ is the union of the odd components of $G - A(G)$.*
4. *Each odd component in $G - A(G)$ is factor-critical.*

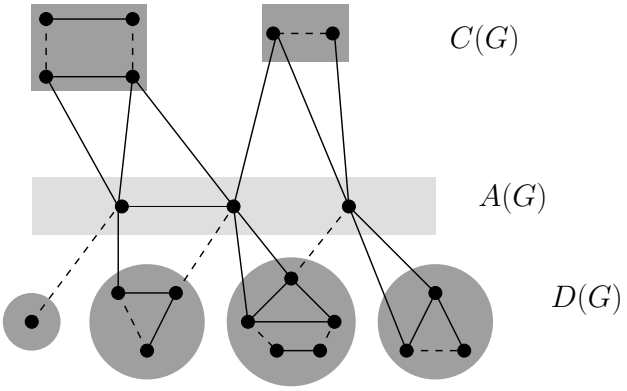


Figure 9.2: Edmonds-Gallai Decomposition

Corollary 9.5. *A graph G is factor-critical if and only if $U = \emptyset$ is the unique Tutte-Berge witness set for G.*

We prove the theorem in the rest of this section. We make use of the properties, and proof of correctness, of Edmonds algorithm for maximum cardinality matching that we discussed previously.

Let M be *any* maximum matching in G and let X be the set of M-exposed nodes. We define three sets of nodes with respect to M and X.

$$\begin{aligned} \textsc{Even}(G, M) &:= \{v \in V \mid \text{there is an even length } M\text{-alternating } X\text{-}v \text{ path}\} \\ \textsc{Odd}(G, M) &:= \{v \in V \mid \text{there is an } M\text{-alternating } X\text{-}v \text{ path}\} \setminus \textsc{Even} \\ \textsc{Free}(G, M) &:= \{v \in V \mid \text{there is no } M\text{-alternating } X\text{-}v \text{ path}\} \end{aligned}$$

Note that $v \in \textsc{Odd}(G, M)$ implies that there is an odd length M-alternating X-v path but no even length path. A node $v \in \textsc{Even}(G, M)$ may have both an even and odd length path; also $X \subseteq \textsc{Even}(G, M)$.

Lemma 9.1. *For* any *maximum matching M in G we have (i)* $\textsc{Even}(G, M) = D(G)$ *(ii)* $\textsc{Odd}(G, M) = A(G)$ *and (iii)* $\textsc{Free}(G, M) = C(G)$.

Proof. We prove the claims in order. If $v \in \textsc{Even}(G, M)$, let P be an even length M-alternating path from some $x \in X$ to v. Then, $M \Delta E(P)$ is another maximum matching in which v is exposed; hence, $v \in D(G)$. Conversely, if $v \in D(G)$ there is a maximum matching M_v that misses v. Then $M \Delta M_v$ gives an even length X-v M-alternating path implying that $v \in \textsc{Even}(G, M)$. Therefore, $\textsc{Even}(G, M) = D(G)$.

If $v \in \textsc{Odd}(G, M)$, let P be an X-v M-alternating path. Since $v \notin \textsc{Even}(G, M)$, P is of odd length and its last edge is uv where $u \in \textsc{Even}(G, M)$. Therefore v is a neighbor of $\textsc{Even}(G, M) = D(G)$ and $v \notin D(G)$ and hence $v \in A(G)$. Conversely, suppose $v \in A(G)$ and let $uv \in E$ where $u \in D(G) = \textsc{Even}(G, M)$. There is an M-alternating X-u path P of even length which ends in an edge $wu \in M$. If $v \in V(P)$ then clearly there is an X-v alternating path. Otherwise, $P + uv$ is an X-v alternating path ($wu \in M$, hence $uv \notin M$ unless $w = v$ but then $v \in V(P)$). Therefore $v \in \textsc{Odd}(G, M)$ since $v \notin D(G) = \textsc{Even}(G, M)$.

Finally, $C(G) = V \setminus (D(G) \cup A(G))$ and hence $\textsc{Free}(G, M) = C(G)$. ■

Lemma 9.2. *Let M be any maximum matching in G, then each node in $A(G) \cup C(G)$ is covered by M and moreover every node $v \in A(G)$ is matched to some node in $D(G)$.*

Proof. From Lemma 9.1, $X \subseteq D(G)$ where X is the set of M-exposed nodes. Hence each node in $A(G) \cup C(G)$ is covered by M.

Suppose $u \in A(G)$ and $uv \in M$. Since $u \in \textsc{Odd}(G, M)$, there is an odd length X-v alternating path P which ends in an edge $wu \notin M$. If v is not in P then $P + uv$ is an M-alternating X-v path and hence $v \in \textsc{Even}(G, M) = D(G)$. If v is in P, let Q be the prefix of P till v, then $Q + vu$ is an even length M-alternating X-u path which contradicts the fact that $u \in A(G)$. ■

Corollary 9.6. *Each component in $G[C(G)]$ is even and $|M \cap C(G)| = |C(G)|/2$.*

Proof. All nodes in $C(G)$ are covered by M. Since $A(G)$ separates $D(G)$ from $C(G)$, and $A(G)$ is matched only to $D(G)$ (by the above lemma), nodes in $C(G)$ are matched internally and hence the corollary follows. ■

The main technical lemma is the following.

Lemma 9.3. *Let M be a maximum matching in G and X be the M-exposed nodes. Each component H of $G[D(G)]$ satisfies the following properties:*

1. *Either* $|V(H) \cap X| = 1$ *and* $|M \cap \delta_G(V(H))| = 0$, *or* $|M \cap \delta_G(V(H))| = 1$.

2. *H is factor-critical.*

Assuming the above lemma, we finish the proof of the theorem. Since each component of $G[D(G)]$ is factor-critical, it is necessarily odd. Hence, from Corollary 9.6 and Lemma 9.3, we have that $G[C(G)]$ contains all the even components of $G - A(G)$ and $G[D(G)]$ contains all the odd components of $G - A(G)$. We only need to show that $A(G)$ is a Tutte-Berge witness. To see this, consider any maximum matching M and the M-exposed nodes X. We need to show $|M| = \frac{1}{2}(|V| + |A(G)| - o(G - A(G)))$. Since $|M| = \frac{1}{2}(|V| - |X|)$, this is equivalent to showing that $|X| + |A(G)| = o(G - A(G))$. From Lemma 9.2, M matches each node in $A(G)$ to a node in $D(G)$. From Lemma 9.3, each odd component in $G[D(G)]$ either has a node in X and no M-edge to $A(G)$ or has exactly one M-edge to $A(G)$. Hence $|X| + |A(G)| = o(G - A(G))$ since all the odd components in $G - A(G)$ are in $G[D(G)]$.

We need the following proposition before the proof of Lemma 9.3.

Proposition 9.1.1. *Let M be a maximum matching in G. If there is an edge $uv \in G$ such that $u, v \in \textsc{Even}(G, M)$, then there is an M-flower in G.*

Proof sketch. Let P and Q be even length M-alternating paths from X to u and v, respectively. If $uv \notin M$ then $P + uv + Q$ is an X-X alternating *walk* of odd length; since M is maximum, this walk has an M-flower. If $uv \in M$, then uv is the last edge of both P and Q and in this case $P - uv + Q$ is again an X-X alternating walk of odd length. ■

Proof of Lemma 9.3. We proceed by induction on $|V|$. Let M be a maximum matching in G and X be the M-exposed nodes. First, suppose $D(G)$ is a stable set (independent set). In this case, each component in $G[D(G)]$ is a singleton node and the lemma is trivially true.

If $G[D(G)]$ is not a stable set, by Proposition 9.1.1, there is an M-flower in G. Let B the M-blossom with the node b as the base of the stem. Recall that b has an even length M-alternating path from some node $x \in X$; by going around the odd cycle according to required parity, it can be seen that $B \subseteq \textsc{Even}(G, M) = D(G)$. Let $G' = G/B$ be the graph obtained by shrinking B. We identify the shrunken node with b. Recall from the proof of correctness of Edmonds algorithm that $M' = M/B$ is a maximum matching in G'. Moreover, the set of M'-exposed nodes in G' is also X (note that we identified the shrunken node with b, the base of the stem, which belong to X if the stem consists only of b). We claim the following with an informal proof.

Claim 9.1.1. *$D(G') = (D(G) \setminus B) \cup \{b\}$, and $A(G') = A(G)$ and $C(G') = C(G)$.*

Proof. We observed that X is the set of exposed nodes for both M and M'. We claim that $v \in \text{EVEN}(G', M')$ implies $v \in \text{EVEN}(G, M)$. Let P be an even length X-v M'-alternating path in G'. If it does not contain b then it is also an X-v even length M-alternating path in G. If P contains b, then one can obtain an even length X-v M-alternating path Q in G by expanding b into B and using the odd cycle B according to the desired parity. Conversely, let $v \in \text{EVEN}(G, M) \setminus B$ and let P be an X-v M-alternating path of even length in G. One can obtain an even length X-v M'-alternating path Q in G' as follows. If P does not intersect B then $Q = P$ suffices. Otherwise, we consider the first and last nodes of $P \cap B$ and shortcut P between them using the necessary parity by using the odd cycle B and the matching edges in there. Therefore, $D(G') = (D(G) \setminus B) \cup \{b\}$ and the other claims follow. ■

By induction, the components of $G' - A(G')$ satisfy the desired properties. Except for the component H_b that contains b, every other such component is also a component in $G - A(G)$. Therefore, it is not hard to see that it is sufficient to verify the statement for the component H in $G - A(G)$ that contains B which corresponds to H_b in $G' - A(G')$ that contains b. We note that X is also the set of M'-exposed nodes in G' and since $\delta_G(H) \cap M = \delta_{G'}(H_b) \cap M'$ (B is internally matched by M except possibly for b), the first desired property is easily verified.

It remains to verify that H is factor-critical. By induction, H_b is factor-critical. Since H_b is obtained by shrinking an odd cycle in H, Proposition 9.0.1 show that H is factor-critical. ■

Algorithmic aspect: Given G, its Edmonds-Gallai decomposition can be efficiently computed by noting that one only needs to determine $D(G)$. A node v is in $D(G)$ iff $\nu(G) = \nu(G - v)$ and hence one can use the maximum matching algorithm to determine this. However, as the above proof shows, one can compute $D(G)$ in the same time it takes to find $\nu(G)$ via the algorithm of Edmonds, which has an $O(n^3)$ implementation. The proof also shows that given a maximum matching M, $D(G)$ can be obtained in $O(n^2)$ time.

9.2 Ear Decompositions and Factor-Critical Graphs

A graph H is obtained by *adding an ear* to G if H is obtained by adding to G a path P that connects two not-necessarily distinct nodes u, v in G. The path P is called an ear. P is a *proper* ear if u, v are distinct. An ear is an *odd* (even) ear if the length of P is odd (even). A sequence of graph $G_0, G_1, \ldots, G_k = G$ is an ear decomposition for G starting with G_0 if for each $1 \leq i \leq k$, G_i is obtained from G_{i-1} by adding an ear. One defines, similarly, proper ear decomposition and odd ear decomposition by restricting the ears to be proper and odd respectively.

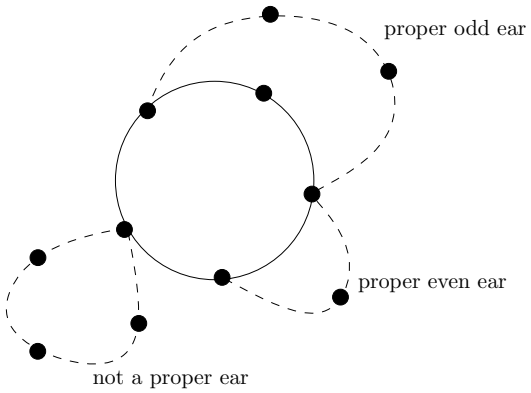


Figure 9.3: Variety of ears.

The following theorems are well-known and standard in graph theory.

Theorem 9.7 (Robbins, 1939)**.** *A graph G is* 2*-edge-connected if and only if it has an ear-decomposition starting with a cycle.*

Theorem 9.8 (Whitney, 1932)**.** *A graph G is* 2*-node-connected if and only if it has a proper ear-decomposition starting with a cycle.*

Factor-critical graphs have the following characterization.

Theorem 9.9 (Lovász, 1972)**.** *A graph G is factor-critical if and only if it has an odd ear decomposition starting with a single vertex.*

Proof. If G has an odd ear decomposition it is factor-critical by inductively using Proposition 9.0.1 and noting that an odd cycle is factor-critical for the base case.

We now prove the converse. G is necessarily connected. Let v be an arbitrary vertex and let M_v be a perfect matching in $G - v$. We iteratively build the ear decomposition starting with the empty graph v. At each step we maintain a (edge-induced) subgraph H of G such that H has an odd ear decomposition and no edge $uv \in M_v$ crosses H (that is, $|V(H) \cap \{u, v\}| \neq 1$). The process stops when $E(H) = E(G)$. Suppose $E(H) \neq E(G)$, then since G is connected, there is some edge $ab \in E(G)$ such that $a \in V(H)$ and $b \neq V(H)$. By the invariant, $ab \notin M_v$. Let M_b be a perfect matching in G that misses b. Then $M_b \Delta M_v$ contains an even length M_v-alternating path $Q := u_0 = b, u_1, \ldots, u_t = v$ starting at b and ending at v. Let j be the smallest index such that $u_j \in V(H)$ (j exists since $u_t = v$ belongs to $V(H)$); that is u_j is the first vertex in H that the path Q hits starting from b. Then, by the invariant, $u_{j-1}u_j \notin M_v$ and hence j is even. The path $a, b = u_0, u_1, \ldots, u_j$ is of odd length and is a valid ear to add to H while maintaining the invariant. This enlarges H and hence we eventually reach G and the process generates an odd ear decomposition. ■

One can extend the above proof to show that G is 2-node-connected and factor-critical iff it has an proper odd ear decomposition starting from an odd cycle.

From Proposition 9.0.1 and Theorem 9.9, one obtains the following.

Corollary 9.10. *G is factor-critical iff there is an odd cycle C in G such that G/C is factor-critical.*

Chapter 10

Primal-Dual Algorithms for Weighted Matching[1]

The primal-dual method is an important and standard technique in optimization. It finds several applications in combinatorial optimization. In this chapter we will consider primal-dual algorithms for weighted matching problems to illustrate the technique.

10.1 Primal-Dual Method for Linear Programs

We briefly discuss the high-level approach of primal-dual for solving LPs. The technique is easier to illustrate whe the primal LP is in the standard form

$$\min c^T x \mid Ax = b, x \geq 0\}.$$

Then the dual is

$$\max\{y^T b \mid y^T A \geq c^T\}.$$

The primal-dual algorithm is an iterative procedure that starts with a feasible dual solution y_0 and tries to find a feasible primal solution x_0 such that x_0 and y_0 satisfy the complementary slackness condition. If they do then we obtain optimality for both. Failing to find an x_0 satisfying complementary slackness will yield a new dual solution y_1 which is better than y_0 in terms of the objective function, that is, $y_1^T b > y_0^T b$. Thus, the dual solution value improves until the algorithm terminates.

Recall that a primal-dual pair of solutions x, y satisfy complementary slackness if the following condition is true: $x_i > 0$ iff $y^T a_i = c_i$ for all i. Thus, given

[1]Notes for bipartite matching were scribed by Abner Guzmán-Rivera based on Nitish Korula's lecture. Notes for non-bipartite graphs were scribed by Rajhans Samdhani.

y_0 we see if there is a solution to the system

$$x' \geq 0, A'x' = b$$

where x' corresponds to those rows i where $y_0^T a_i = c_i$ which are the *tight constraints* in the dual. Note that we are now solving another LP! However, this LP is simpler in that it is only a feasiblity LP rather than an optimization LP. Suppose we find a feasible solution x'_0 to this LP. Then by padding x'_0 with 0's for the other rows will yield a feasible primal solution x_0 that satisfies complementary slackness with respect to y_0. Thus the algorithm can terminate.

Suppose there is no solution to $x' \geq 0, A'x' = b$. Then, by Farkas lemma there is z such that

$$z^T b > 0 \text{ and } z^T A \leq 0.$$

Let $y_1 = y_0 + \alpha z$ where α is the largest real number such that $y_1^T A \leq c^T$. Note that $\alpha > 0$. If $\alpha = \infty$ then the dual is unbounded and hence the primal is infeasible. Otherwise we see that $y_1^T b > y_0^T b$ and thus we obtain a better dual solution and we iterate.

Combinatorial settings: In combinatorial settings, especially in exact algorithms, the primal-dual algorithm converts a weighted problem to an unweighted problem. In the above we see that the LP we need to solve a feasibility LP while the original primal LP had an objective with cost/weight vector c^T. Further, in combinatorial settings we wish to find an *integer* solution. Thus, we work with an underyling integer polytope. The primal-dual iteration requires one to typically understand a combinatorial min-max relation that characterizes integer solutions. This is useful/necessary to maintain integrality of the primal (and often dual solutions), to find a way to improve the dual (as guaranteed by Farkas lemma), and to bound the total number of iterations and the running time.

10.2 Weighted Matching Problems

Definition 10.1 (Maximum Weight Matching)**.** *Given a graph $G = (V, E)$ and weight function $w : E \to \mathbb{R}$ find a matching of maximum weight where the weight of matching M is given by $w(M) = \sum_{e \in M} w(e)$.*

Note that in a maximum weight matching we can assume that $w(e) > 0$ for all e, otherwise we can ignore e.

Definition 10.2 (Minimum Weight/Cost Perfect Matching)**.** *Given a graph $G = (V, E)$ and weight function $w : E \to \mathbb{R} \cup \{\infty\}$, find a perfect matching M minimizing $w(M) = \sum_{e \in M} w(e)$.*

We could also assume that no edge weights are negative as we may add a large enough constant C to all weights. This is a useful assumption to make in some situations.

Remark 10.1. Although the two problems are poly-time equivalent in the exact algorithms settings, they behave differently in the approximate setting. Approximating maximum-weight matching is feasible in near-linear time [20] while the min-weight perfect matching requires one to solve the decision problem of checking whether there is a perfect matching which seems harder, especially in non-bipartite graphs.

Exercise 10.1. Describe a reduction from max-weight matching to min-weight perfect matching. Describe a reduction in the converse direction. Show that the reductions can be done so that if the original graph is bipartite then the new graph is also bipartite.

Reduction to Min-Cost Flow in Bipartite Graphs: In bipartite graphs the min-weight perfect matching problem can be reduced to min-cost flow problem. This is easy to see and follows the same type of reduction we use to reduce maximum matching to maximum flow. We have seen strongly polynomial-time algorithms for min-cost flow and in fact the successive shortest path algorithm for min-cost flow easily yields an $O(n(n + m \log n))$-time algorithm for min-cost perfect matching in bipartite graphs. Despite this we describe a primal-dual algorithm for bipartite graphs without explicitly referring to this reduction since it helps set up the ideas for non-bipartite case.

We will focus on the min-weight perfect matching problem since its formulation is more convenient for the primal-dual algorithm.

10.3 Minimum Weight Perfect Matching in Bipartite Graphs

We consider the easier case of bipartite graphs. We will assume $G = (A \cup B, E)$ where $|A| = |B|$. The following is an ILP formulation of the minimum weight perfect matching problem:

$$\begin{aligned} \min \sum_{(a,b)} w(a,b)x(a,b) \text{ subject to:} & \\ \sum_{b} x(a,b) &= 1 \qquad \forall a \in A \\ \sum_{a} x(a,b) &= 1 \qquad \forall b \in B \\ x(a,b) &\in \{0,1\} \quad \forall a \in A, b \in B \end{aligned} \tag{10.1}$$

Definition 10.3 (Primal)**.** *This is the LP relaxation of the above ILP:*

$$\begin{aligned} &\min \sum_{(a,b)} w(a,b)x(a,b) \text{ subject to:} \\ \sum_{b} x(a,b) &= 1 \quad \forall a \in A \\ \sum_{a} x(a,b) &= 1 \quad \forall b \in B \\ x(a,b) &\geq 0 \quad \forall a \in A, b \in B \end{aligned} \tag{10.2}$$

Recall that we saw, in an earlier lecture, a proof of the following theorem by noting that the constraint matrix of the polytope is totally unimodular.

Theorem 10.4. *Any extreme point of the polytope defined by the constraints in (10.2) is integral.*

We obtain a different proof of Theorem 10.4 via algorithms to find a minimum-weight perfect matching. Our algorithms are *primal-dual*; we will construct a feasible solution to the dual of LP (10.2) with value equal to the weight of the perfect matching output by the algorithm. By weak duality, this implies that the matching is optimal. More precisely, our algorithms will always maintain a feasible dual solution y, and will attempt to find a primal feasible solution (a perfect matching M) that satisfies complementary slackness.

(Dual) The following LP is the dual for (10.2):

$$\begin{aligned} &maximize \sum_{(a \in A} y(a) + \sum_{b \in B} y(b) \text{ subject to:} \\ y(a) + y(b) &\leq w(a,b) \quad \forall (a,b) \in E \end{aligned} \tag{10.3}$$

Given a dual-feasible solution y, we say that an edge $e = (a,b)$ is *tight* if $y(a) + y(b) = w(a,b)$. Let $\hat{y}$ be dual-feasible, and let M be a perfect matching in $G(V,E)$: Then,

$$\begin{aligned} w(M) = \sum_{(a,b)\in M} w(a,b) &\geq \sum_{(a,b)\in M} \hat{y}(a) + \hat{y}(b) \\ &= \sum_{a \in A} \hat{y}(a) \cdot (\delta(a) \cap M) + \sum_{b \in B} \hat{y}(b) \cdot (\delta(b) \cap M) \\ &= \sum_{a \in A} \hat{y}(a) + \sum_{b \in B} \hat{y}(b) \end{aligned}$$

where the first inequality follows from the feasibility of $\hat{y}$, and the final equality from the fact that M is a perfect matching. That is, any feasible primal

solution (a perfect matching M) has weight at least as large as the value of any feasible dual solution. (One could conclude this immediately from the principle of weak duality.) Note, though, that if M only uses edges which are *tight* under $\hat{y}$, we have equality holding throughout, and so by weak duality, M must be *optimal*. That is, given any dual feasible solution $\hat{y}$, if we can find a perfect matching M only using tight edges, M must be optimal. (Recall that this is the principle of *complementary slackness*.)

Our primal-dual algorithms apply these observations as follows: We begin with an arbitrary feasible dual solution y, and find a maximum-cardinality matching M that uses only tight edges. If M is perfect, we are done; if not, we *update* our dual solution. This process continues until we find an optimal solution.

We first give a simple algorithm (Algorithm 1 in the following page) exploiting these ideas to prove Theorem 10.4. The existence of set S in line 6 is a consequence

Algorithm 3 MinWeightPerfectMatching($G = (V, E), w$)

1: $y \leftarrow 0$
2: $E' \leftarrow$ set of tight edges
3: $M \leftarrow$ max cardinality matching for graph $G' = (V, E')$
4: **while** M is not a perfect matching **do**
5: let $G' = (V, E')$
6: let $S \subseteq A$ be such that $|S| > |N(S)|$
7: let $\epsilon = \min_{a \in S, b \in B \setminus N(S)} \{w(a, b) - y(a) - y(b)\}$
8: $\forall a \in S \; y(a) = y(a) + \epsilon$
9: $\forall b \in N(S) \; y(b) = y(b) - \epsilon$
10: update E', M
11: **end while**
12: **return** M

of Hall's theorem. Observe that the value of y increases at the end of every iteration. Also, the value of y remains feasible as tight edges remain tight and it is easy to verify that by the choice of ϵ the constraints for other edges are not violated.

Claim 10.3.1. *Algorithm 3 terminates if w is rational.*

Proof. Suppose all weights in w are integral. Then at every iteration ϵ is integral and furthermore $\epsilon \geq 1$. It follows that the number i of iterations is bounded by $i \leq \max w(a, b) \cdot |E|$. If weights are rational we may scale them appropriately so that all of them become integers. ∎

Proof. Proof of Theorem 10.4 The incidence vector of a perfect matching computed by Algorithm 3 is an extreme point of the polytope in (10.2). This vector is integral. Furthermore, by carefully choosing the cost function one can make any extreme point be the unique optimum solution to the primal linear program. ■

Note that Algorithm 3 does not necessarily terminate in strongly polynomial time; in the rest of this section, we describe a more efficient algorithm for the minimum-weight bipartite matching problem.

As before, Algorithm 4 always maintains a feasible dual y and attempts to find a close to primal feasible solution (matching M) that satisfies complementary slackness. One key difference from Algorithm 3 is that we now carefully use the maximum cardinality matching M as a guide in constructing the updated dual solution y; this allows us to argue that we can augment M efficiently. (In contrast, Algorithm 3 effectively "starts over" with a new matching M in each iteration.)

Algorithm 4 MinWeightPerfectMatchingPD($G = (V, E), w$)

$\forall b \in B\ y(b) \leftarrow 0$
$\forall a \in A\ y(a) \leftarrow \min_b \{w(a, b)\}$
$E' \leftarrow$ set of tight edges
$M \leftarrow$ max cardinality matching for graph $G' = (V, E')$
while M is not a perfect matching **do**
    let $E_{dir} \leftarrow \{e$ directed from A to $B \mid e \in E', e \notin M\} \bigcup$
                $\{e$ directed from B to $A \mid e \in E', e \in M\}$
    let $D = (V, E_{dir})$ ▷ D is a directed graph
    let $L \leftarrow \{v \mid v$ is reachable in D from an unmatched vertex in $A\}$
    let $\epsilon = \min_{a \in A \cap L, b \in B \setminus L} \{w(a, b) - y(a) - y(b)\}$
    $\forall a \in A \cap L\ y(a) = y(a) + \epsilon$
    $\forall b \in B \cap L\ y(b) = y(b) - \epsilon$
    update E', M
end while
return M

Claim 10.3.2. *At every iteration, $C = (A \setminus L) \cup (B \cap L)$ is a vertex cover for graph $G' = (V, E')$. Moreover, $|C| = |M|$.*

Proof. Assume C is not a vertex cover. Then there must be an edge $e = (a, b) \in E'$ with $a \in A \cap L$ and $b \in B \setminus L$. If e is directed from a to b, then since a is reachable from an unmatched vertex in A, so is b; this contradicts the fact that $b \in B \setminus L$. Therefore, e must be directed from b to a, and hence e is in the matching M.

As a itself is matched (using edge e) and $a \in L$, it must be reachable from an unmatched vertex of A. But the only incoming edge to a is (b, a) (this is the unique edge incident to a in the matching M), and hence b is reachable from this unmatched vertex of A; again, this contradicts the fact that $b \notin L$. To show the second part of the proof we show that $|C| \leq |M|$, since the reverse inequality is true for any matching and any vertex cover. The proof follows from the following observations:

1. No vertex in $A \setminus L$ is unmatched by the definition of L.
2. No vertex in $B \cap L$ is unmatched since this would imply the existence of an augmenting path (contradicting the maximality of M).
3. There is no edge $e = (a, b) \in M$ such that $a \in A \setminus L$ and $b \in B \cap L$. Otherwise, as this edge would be directed from b to a, a would be in L.

These remarks imply that every vertex in C is matched and moreover the corresponding edges of the matching are distinct. Hence $|C| \leq |M|$, and so C is an optimum vertex cover for $G'(V, E')$. ∎

At every iteration where the maximum cardinality matching M output is not perfect, the algorithm will use information from the optimum vertex cover C to update the dual solution and improve its value. By the proof of claim 10.3.2 there is no tight edge between $a \in A \cap L$ and $b \in B \setminus L$, which implies $\epsilon > 0$; it is easy to check that the updated dual solution is feasible. Moreover, the difference between the new dual solution and the old dual solution is:

$$\epsilon \cdot (|A \cap L| - |B \cap L|) = \epsilon \cdot (|A \cap L| + |A \setminus L| - |A \setminus L| - |B \cap L|) = \epsilon \cdot (\frac{|V|}{2} - |C|),$$

but $|C| = |M| < \frac{|V|}{2}$, since M is not perfect, which implies the value of the dual solution strictly increases. When the algorithm terminates, we obtain a perfect matching M and a dual feasible solution which satisfy complementary slackness.

Claim 10.3.3. *Algorithm (4) terminates in $O(|V|^2)$ iterations.*

Proof. We first observe that after any iteration, all edges in M are still tight: The only edges (a, b) that are tight at the beginning of an iteration but not at the end are those with $a \in A \cap L$ and $b \in B \setminus L$; from observation 3 in the proof of Claim 10.3.2, there are no edges in M of this form. Thus, after any iteration, the size of a maximum cardinality matching M in $G'(V, E')$ cannot decrease.

Say that an iteration is *successful* if the size of a maximum cardinality matching using the tight edges E' increases. Clearly, after at most $|V|/2$ successful iterations,

we have a perfect matching, and the algorithm terminates. We show that there are at most $|B| = |V|/2$ consecutive unsuccessful iterations between any pair of successful iterations. Hence, the total number of iterations is at most $\frac{|V|}{2} \cdot \frac{|V|}{2}$, which is $O(|V|^2)$.

To bound the number of consecutive unsuccessful iterations, we argue below that after an unsuccessful iteration, $|B \cap L|$ increases. Assume for now that this is true: After at most $|B|$ unsuccessful iterations, we have $B \cap L = B$. Once this occurs, every vertex of B (which must include at least one unmatched vertex) is reachable from an unmatched vertex of A, and so we can augment M to find a larger matching, which means that the current iteration is successful.

It remains only to prove that at every unsuccessful iteration, at least one more vertex in B must become reachable from an exposed vertex in A (i.e. $|B \cap L|$ increases). First note that no vertex of A or B becomes unreachable; the only way this could happen is if for some path P from an unmatched vertex $a \in A$ to vertex $v \in L$, an edge $e \in P$ that was previously tight is no longer tight. But the only edges that are no longer tight are between $A \setminus L$ and $B \cap L$, and by definition, no such path P visits a vertex in $A \setminus L$. To see that at least one new vertex of B becomes reachable, note that some edge $e = (a, b)$ with $a \in A \cap L$ and $b \in B \setminus L$ now has become tight by our choice of ϵ. As the edge (a, b) is directed from a to b, b is now reachable. ■

It is not hard to see that each iteration takes only $O(|V|^2)$ time, and hence the overall running time of the algorithm is $O(|V|^4)$. A more careful analysis would yield a tighter running time of $O(|V|^3)$.

10.4 Min Cost Perfect Matching in Non-Bipartite Graphs

We describe a strongly polynomial time algorithm for the minimum cost perfect matching problem in a **general** graph. Using a simple reduction discussed earlier, one can also obtain an algorithm for the maximum weight matching problem. We also note that when discussing perfect matching, without loss of generality, we can assume that all weights/costs are non-negative (why?).

The algorithm we describe is essentially due to Edmonds. The algorithm is *primal-dual* based on the following LP formulation and its dual.

Primal:

$$\begin{aligned} \min \sum_{e \in E} w(e)x(e) & \qquad \text{subject to:} \\ x(\delta(v)) &= 1 \quad \forall v \in V \\ x(\delta(U)) &\geq 1 \quad \forall U \subset V, |U| \geq 3, |U| odd \\ x(e) &\geq 0 \quad \forall e \in E \end{aligned} \tag{10.4}$$

Dual:

$$\begin{aligned} \max \sum_{U \subseteq V, |U| \geq 3, |U| odd} \pi(U) & \qquad \text{subject to:} \\ \sum_{U \subseteq V, |U| \geq 3, |U| odd, \delta(U) \ni e} \pi(U) &= w(e) \quad \forall e \in E \\ \pi(U) &\geq 0 \qquad \forall U \subseteq V, |U| \geq 3, |U| odd \end{aligned}$$

We note that non-negativity constraints on the dual variables are only for odd sets U that are not singletons (because the equations for the singleton sets are equalities). In certain descriptions of the algorithm the dual variables for the singleton sets are distinguished from those for odd sets of size ≥ 3, however we do not do that here.

Like other primal dual algorithms, we maintain a feasible dual solution π and an integral infeasible primal solution x and iteratively reduce the infeasiblity of x. Here x corresponds to a matching and we wish to drive it towards a perfect matching. In particular, we will also maintain the primal complementary slackness, that is,

$$x(e) > 0 \Rightarrow \sum_{U : e \in \delta(U)} \pi(U) = w(e)$$

(a primal variable being positive implies the corresponding dual constraint is tight). Thus, at the end, if we have a perfect matching in the primal, it is feasible and certifies its optimality.

The main question is how to update the dual and the primal. Also, we observe that the dual has an exponential number of variables and hence any polynomial time algorithm can only maintain an implicit representation for a subset of the variables.

10.4.1 Notation

Definition 10.5. *A family of sets $\mathcal{S} = \{S_1, S_2, \ldots, S_p\}$ is* laminar *if no two sets of $\mathcal{S}$ properly cross. In other words for any i, j exactly one of the following holds: (i) $S_i \cap S_j = \emptyset$ (ii) $S_i \subset S_j$ (iii) $S_j \subset S_i$.*

Remark 10.2. Laminar families are often represented by a rooted forest where there is a node for each set in the family and node u a child of node v if the set S_u corresponding to u is a maximal set contained in S_v. A laminar family over a ground set of size n can have at most $2n - 1$ sets.

The algorithm maintains a *laminar family* of odd subsets of V denoted by Ω. Ω always includes the singletons $\{v\}, v \in V$. It maintains the invariant that $\pi(U) = 0$ if $U \notin \Omega$, hence Ω is the implicit representation of the "interesting" dual variables. Note that $|\Omega| \leq 2|V|$.

Given G, Ω and $\pi : \Omega \to \mathbb{R}$ where π is dual feasible, we say an edge is π-*tight* (or *tight* when π is implicit) if $\sum_{U \in \Omega : e \in \delta(U)} \pi(U) = w(e)$.

Let E_π be the set of tight edges (with Ω, π implicit) and G_π be the graph induced by them. We obtain a new graph G' in which we contract each *maximal set* in Ω into a (pseudo) vertex. For a node $v \in G'$, let $S_v \in \Omega$ be the set of nodes of G contracted to v.

For each $U \in \Omega$, $|U| \geq 3$, consider the graph $G_\pi[U]$ and let H_U be the graph obtained from $G_\pi[U]$ by contracting each maximal *proper* subset $S \subset U$ where $S \in \Omega$. The algorithm also maintains the invariant that H_U has a Hamiltonian cycle B_U. The laminar family corresponds to a series of blossom shrinkings that arise in the algorithm to find a maximum cardinality matching that we saw previously.

10.4.2 Recap of Edmonds-Gallai Decomposition

We restate the Edmonds-Gallai decomposition theorem and make some observations which help us in proposing and analysing an algorithm for min-cost perfect matching. We also use the notation from this section in subsequent sections.

Theorem 10.6 (Edmonds-Gallai)**.** *Given a graph $G = (V, E)$, let*

$$\begin{aligned} D(G) &:= \{v \in V \mid \text{there exists a maximum matching that misses } v\} \\ A(G) &:= \{v \in V \mid v \text{ is a neighbor of } D(G) \text{ but } v \notin D(G)\} \\ C(G) &:= V \setminus (D(G) \cup A(G)). \end{aligned}$$

Then, the following hold.

1. *The set $U = A(G)$ is a Tutte-Berge witness set for G.*
2. *$C(G)$ is the union of the even components of $G - A(G)$.*
3. *$D(G)$ is the union of the odd components of $G - A(G)$.*
4. *Each component in $G - A(G)$ is factor-critical.*

Let M be a matching in G and X be the set of M-exposed nodes. We can partition V into $\textsc{Even}(G, M)$, $\textsc{Odd}(G, M)$ and $\textsc{Free}(G, M)$ where $\textsc{Even}(G, M)$ is the set of all nodes reachable from X with an even length M-alternating path, $\textsc{Odd}(G, M)$ are those that are reachable from X with only an odd length M-alternating path, and $\textsc{Free}(G, M)$ are the rest of the nodes. If there is an edge $uv \in E$ where $u, v \in \textsc{Even}(G, M)$ then there is an odd length X-X M-alternating walk which implies that there is an M-flower (and hence also a blossom) in G. Moreover, shrinking a blossom results in a new graph and new matching where the set of M-exposed nodes and the parity of the remaining nodes does not change — recall that we identify the shrunken node for a blossom B with the base b.

Finally, as we saw in the proof of the Edmonds-Gallai decomposition, if M is a maximum cardinality matching, then $\textsc{Even}(G, M) = D(G)$ and $\textsc{Odd}(G, M) = A(G)$ and $\textsc{Free}(G, M) = C(G)$. It also follows that if M is maximum then either there is an M-blossom or $D(G)$ consists of singleton nodes.

10.4.3 Algorithm

Following is the algorithm for min cost perfect matching using primal dual method.

Initialize: $\Omega = \{\{v\} \mid v \in V\}, \pi(U) = 0 \; \forall U$ with odd $|U|, M = \emptyset$, and $G' = G_\pi$.
while (M is not a perfect matching in G') **do**

1. $X \leftarrow M$-exposed nodes in G'.
2. Find $X - X, M-$alternating walk P in G'.
3. If P is an $M-$augmenting path then do
$$M \leftarrow M \Delta E(P)$$
continue.
4. If P has an $M-$blossom B, then do *shrinking* as:
$$U = \cup_{v \in B} S_v, \Omega \leftarrow \Omega \cup U, \pi(U) = 0, G' \leftarrow G'/B, M \leftarrow M/B$$
continue.
5. Else P is empty $\Rightarrow$ M is a maximum matching in G'. Compute $D(G')$, $A(G')$, and $C(G')$ as in *Edmonds-Gallai decomposition*. Let ϵ be the largest value

such that $\pi(S_v) = \pi(S_v) + \epsilon, \forall v \in D(G')$ and $\pi(S_v) = \pi(S_v) - \epsilon, \forall v \in A(G')$ maintains dual feasibility[2] of π in G.

If ϵ is unbounded then G has no perfect matching; **STOP**.

Else update as follows:

- Add newly tight edges to G_π. Remove any edges in G_π that become non-tight.
- For each $v \in A(G')$ with $|S_v| \geq 3$ and $\pi(S_v) = 0$ *deshrink* as
  - Remove S_v from Ω
  - Update G'
  - Extend M by a perfect matching in $B_{S_v} - \{v\}$.

end while
Extend M in G' to a perfect matching in G_π and output it.

10.4.4 Example

Consider the execution of this algorithm on the following graph:

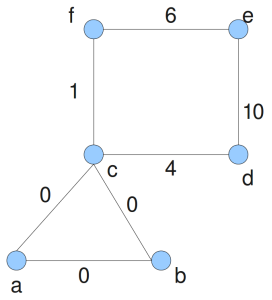


Figure 10.1: Original graph G

The execution is shown in figures 2 to 9. Red edges are the current edges in the matching; black edges are tight.

[2]Dual feasibility requires that $\pi(U) \geq 0$ for all odd $|U| \geq 3$ and for all $e \in E$, $\sum_{U:e\in\delta(U)} \pi(U) \leq w(e)$. Thus the choice of ϵ is defined by some new edges becoming tight as we increase π values for nodes in $D(G')$, or because for some $v \in A(G')$ with $|S_v| \geq 3$ we have $\pi(S_v)$ reaches 0 as we decrease π values for nodes in $A(G')$.

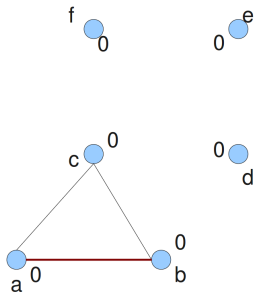


Figure 10.2: G' after iteration 1.

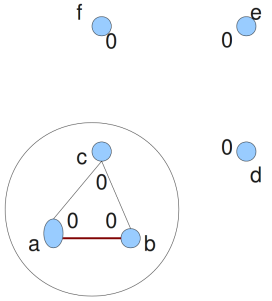


Figure 10.3: G' after iteration 2. Shrinking.

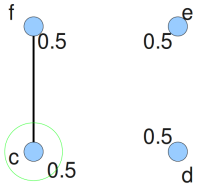


Figure 10.4: G' after iteration 3. Edge tight.

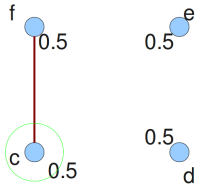


Figure 10.5: G' after iteration 4. Augment.

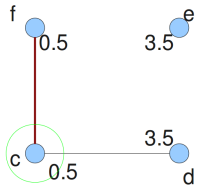


Figure 10.6: G' after iteration 5. Edge tight.

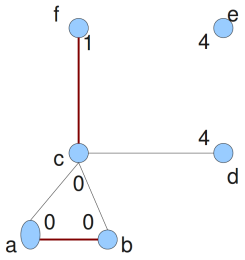


Figure 10.7: G' after iteration 6. Deshrink.

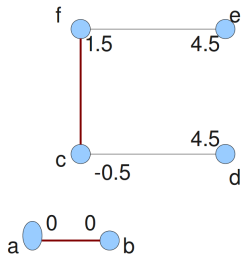


Figure 10.8: G' after iteration 7. Edge tight. Some edges become slack and hence disappear from G_π.

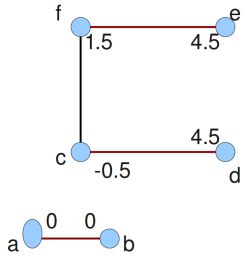


Figure 10.9: G_π after iteration 8. Augment. Maximum hence STOP.

10.4.5 Proof

Lemma 10.1. *The algorithm maintains the following invariants over the iterations*

- π *is dual feasible*
- Ω *is laminar*
- *for each* $U \in \Omega$, H_U *has a hamiltonian cycle* B_U.

Proof sketch. We need to check that each iteration maintains the given properties. We analyze all the considered cases and see that in each case, this property is preserved.

If M is augmented then Ω and π don't change.

If we shrink a blossom B in finding P then we add $U = \cup_{u \in B} S_v$ to Ω. This preserves laminarity since nodes in G' correspond to the maximal sets in Ω. Since we set $\pi(U) = 0$ no dual violation happens. Moreover, B is an odd cycle and hence H_U indeed contains a hamiltonian cycle for the new set U added to Ω.

For the final case we observe that we are not adding any sets to Ω and ϵ is chosen to ensure dual feasiblity. Deshrinking preserves laminarity. ■

Claim 10.4.1. *If M is a matching in G' then there is a matching N in G_π where number of N-exposed nodes is same as M-exposed nodes.*

Proof. We can recursively expand the nodes in G' and extend M using the fact that H_U has a Hamiltonian cycle for each $U \in \Omega$. ■

Corollary 10.7. *If M is a perfect matching in G' then it can be extended to perfect matching N in G_π.*

Claim 10.4.2. *If the algorithm terminates with a perfect matching then it is an optimal matching.*

Proof. Let π be the feasible dual solution at the end of the algorithm. If M is a perfect matching in G_π then, $\{x(e) = 1$ if $e \in M$ and $x(e) = 0$ otherwise$\}$, is a feasible primal solution and x and π satisfy complementary slackness conditions thus implying that both the solutions are optimal. ■

The above claims show that if the algorithm terminates then it outputs an optimum solution. Now we establish that the algorithm indeed terminates.

Lemma 10.2 (Main Lemma)**.** *The algorithm terminates in $O(|V|^2)$ iterations.*

Each iteration can be implemented in $O(m)$ time with minimal data structures, Thus we have the following theorem due to Edmonds.

Theorem 10.8 (Edmonds)**.** *There is a an $O(n^2 m)$ time algorithm for the min cost perfect matching problem.*

As a corollary we also obtain

Corollary 10.9. *The polytope $Q(G)$ described by the inequalities below:*

$$\begin{array}{rcll} x(\delta(v)) & = & 1 & \forall v \in V \\ x(\delta(U)) & \geq & 1 & \forall U \subset V, |U| \geq 3, |U| odd \\ x(e) & \geq & 0 & \forall e \in E \end{array} \tag{10.5}$$

is the convex hull of the perfect matchings in G.

Proof. The algorithm shows that for any given weights $w : E \rightarrow \mathcal{R}^+$, the linear program $\min\{w \cdot x \mid x \in Q(G)\}$ has an integral optimum solution whenever $Q(G) \neq \emptyset$. Since $w \geq 0$ can be assumed w.l.o.g. so $Q(G)$ is an integral polyhedron. ■

Remark 10.3. We observe that the dual update increases the value of the dual. To see this recall that if M is maximum matching in G' and is not perfect then $|D(G')| > |A(G')|$ since $A(G')$ is a Tutte-Berge witness set. The dual update increase the dual value for each vertex in $D(G')$ by ϵ and decreases the value for each vertex in $A(G')$ by ϵ. Hence the dual value increases. Note, however, that the dual constructed by the algorithm is not necessarily integer valued even if w is integer valued (recall the example we saw).

Now we finish the proof of the key lemma on termination. First we observe the following.

Proposition 10.4.1. *In any iteration, the number of M-exposed nodes in G' does not increase and all edges of M remain tight in G_π. Thus the matching size in G does not decrease.*

Proof. It is easy to see that steps (1) - (4) of the algorithm do not increase the number of M-exposed nodes in G'. The only non-trivial case is step (5) in which the dual value is changed. Now in this step, recall the Edmonds-Gallai decomposition and notice that $A(G')$ is matched only to $D(G')$. The dual update in this step leaves any edge uv between $D(G')$ and $A(G')$ tight and so all the M-edges remain tight in this step. ■

Lemma 10.3. *If a set U is added to Ω in some iteration ("shrinking") then it is removed from Ω ("deshrinking") only after the matching size in G has increased.*

Proof. When U is added to Ω, it corresponds to the blossom of an M-flower where M is the current matching in G'. Let v be the node in G' corresponding to U after it is shrunk. If X is the set of M-exposed nodes then there is an $X - v$, M-alternating even length path. If there is no matching augmentation then v continues to have an X-v, $M-$alternating even length path or v is swallowed by a larger set U' that is shrunk. In the former case, v cannot be in $A(G')$ and hence cannot be "deshrunk". In the latter case, U' is not deshrunk before a matching augmentation and hence U, which is inside U', cannot be deshrunk before a matching augmentation. ■

Claim 10.4.3. *Suppose iteration i has a matching augmentation and iteration $j > i$ is the next matching augmentation. Then between i and j, there are at most $|V|$ shrinkings and at most $|V|$ deshrinkings.*

Proof. Let Ω be the laminar family of shrunk sets at the end of iteration i. By the previous claim, before iteration j, we can only deshrink sets in Ω. Hence number of deshrinkings is $\leq |\Omega| - |V|$ since we cannot deshrink singletons. Thus the number of deshrinkings is $\leq |V|$.

Similarly number of of shrinkings is at most $|\Omega'| - |V|$ where Ω' is the laminar family just before iteration j. This gives an upper bound of $|V|$ on the number of shrinkings. ■

We now examine whan can happen in an iteration other than augmentation, shrinking, and deshrinking. In step (5), an edge $uv \in E \setminus E_\pi$ can become tight and join E_π. One of the following two cases must happen since dual values are increased for nodes in $D(G')$, decreased for nodes in $A(G')$, and unchanged for $C(G')$:

1. $u, v \in D(G')$.
2. $u \in D(G'), v \in C(G')$.

The following two claims take care of these cases.

Claim 10.4.4. *If edge uv becomes tight in an iteration and $u, v \in D(G')$ then the next iteration is either a shrinking iteration or an augmentation iteration.*

Proof. Let X be the set of M-exposed nodes in G'. If $u, v \in D(G')$ then $G' + uv$ creates an X-X M-alternating walk of odd length because $D(G')$ consists of all the vertices reachable by a walk of even length from X. This implies that in the next iteration we have a non-empty walk leading to an augmentation or shrinking. ■

Claim 10.4.5. *If $u \in D(G')$ and $v \in C(G')$ then in $G'' = G' + uv$, v is reachable from X by an M-alternating path and hence $C(G')$ decreases.*

With the above in place we can upper bound the total number of iterations. We have a total of $\frac{|V|}{2}$ augmentation iterations. Between consecutive augmentation iterations, there are at most $2|V|$ shrinking and deshrinking iterations. Each other iteration is an edge becoming tight. The number of Case 1 iterations (uv added to E_π where $u, v \in D(G')$) can be charged to shrinking iterations. Total number of Case 2 iterations (uv added to E_π where $u \in D(G'), v \in C(G')$) is at most $|V|$ since each such iteration increases by 1, the number of nodes reachable from X with an M-alternating path. No other iteration decreases the number of nodes reachable before a matching augmentation. Thus the number of iterations between augmentation is $O(|V|)$. Hence total number of iterations is $O(|V|^2)$.

Chapter 11

Total Dual Integrality and Cunningham-Marsh Theorem[1]

Recall that if A is TUM and b, c are integral vectors, then $\max\{cx : Ax \leq b\}$ and $\min\{yb : y \geq 0, yA = c\}$ are attained by integral vectors x and y whenever the optima exist and are finite. This gives rise to a variety of min-max results. For example we derived König's theorem on bipartite graphs. There are several examples where we have integral polyhedra defined by a system $Ax \leq b$ but A is not TUM; the polyhedron is integral only for some specific b. We may still ask for the following. Given an integral vector c, consider the maximization problem $\max\{cx : Ax \leq b\}$; is it the case that the dual minimization problem $\min\{yb : y \geq 0, yA = c\}$ has an integral optimal solution whenever a finite optimum exists?

This motivates the following definition:

Definition 11.1. *A rational system of inequalities $Ax \leq b$ is* totally dual integral (TDI) *if, for all integral c, $\min\{yb : y \geq 0, yA = c\}$ is attained by an integral vector y^* whenever the optimum exists and is finite.*

Remark 11.1. If A is TUM, $Ax \leq b$ is TDI for all b.

This definition was introduced by Edmonds and Giles [23] who derived the following theorem:

Theorem 11.2. *If $Ax \leq b$ is TDI and b is integral, then $\{x : Ax \leq b\}$ is an integral polyhedron.*

This is useful because $Ax \leq b$ may be TDI even if A is not TUM; in other words, this is a weaker sufficient condition for integrality of $\{x : Ax \leq b\}$ and

[1]Based on notes scribed by Bill Kinnersley in 2010.

moreover guarantees that the dual is integral whenever the primal objective vector is integral.

Proof sketch. Let $P = \{x : Ax \leq b\}$. Recall that we had previously shown that the following are equivalent:

(i) P is integral.

(ii) Every face of P contains an integer vector.

(iii) Every minimal face of P contains an integer vector.

(iv) $\max\{cx : x \in P\}$ is achieved by an integral vector whenever the optimum is finite.

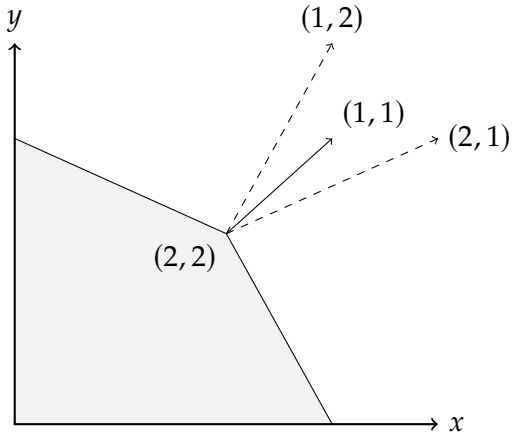
Edmonds and Giles proved two more equivalent conditions:

(v) Every rational supporting hyperplane of P contains an integer vector.

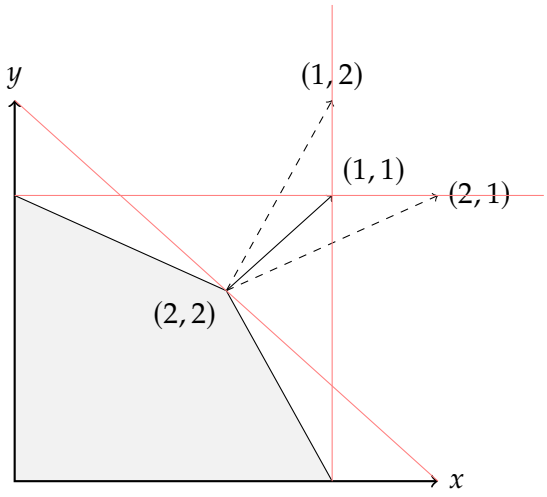
(vi) If c is integral, then $\max\{cx : x \in P\}$ is an integer whenever the optimum exists and is finite.

Condition (vi) implies the theorem as follows. If $Ax \leq b$ is TDI *and* b is integral, $\max\{cx : x \in P\}$ is an integer for all integral c whenever it is finite; this is because the dual optimum is achieved by an integral vector y^* (TDI property) and the objective function by^* is integral because b is integral. This implies that P is integral. ■

There is an important subtlety to the definition of total dual integrality: being TDI is a property of a system of inequalities, *not* a property of the corresponding polyhedron.



$x \geq 0, y \geq 0$
$x + 2y \leq 6$
$2x + y \leq 6$



$x \geq 0, y \geq 0$
$x + 2y \leq 6$
$2x + y \leq 6$
$x + y \leq 4$
$x \leq 3, y \leq 3$

We will illustrate this with an example from notes of Michel Goemans. Consider the system $Ax \leq b$ drawn above on the left. If we take the cost vector c to be $(1, 1)$, then the primal has an optimum at $(2, 2)$ with value 4. The tight constraints at this vertex have normal vectors $(2, 1)$ and $(1, 2)$ (these are rows of A). Therefore, in order for the dual $yA = c$ to have an integer solution, we must be able to express $(1, 1)$ as an *integer* combination of $(2, 1)$ and $(1, 2)$. Since this is impossible, $Ax \leq b$ is *not* TDI.

However, suppose we add more constraints to obtain the system $A'x \leq b'$ drawn above on the right. Note that this system corresponds to the same polyhedron as $Ax \leq b$. However, now we have an additional normal vector at $(2, 2)$, namely, $(1, 1)$. Thus $(1, 1)$ is now an integer combination of the normal vectors at $(2, 2)$. The system $A'x \leq b'$ is in fact TDI, even though it corresponds to the same polytope as the (non-TDI) system $Ax \leq b$.

The example demonstrates a necessary condition for a system to be TDI. We explain this in the general context. Consider the problem $\max\{cx : Ax \leq b\}$ with c integral, and assume it has a finite optimum β. Then it is achieved by some vector x^* in the face F defined by the intersection of $\{x : Ax \leq b\}$ with the hyperplane $cx = \beta$. For simplicity assume that the face F is an extreme point/vertex of the polyhedron and let $A'x^* = b'$ be the set of all inequalities in $Ax \leq b$ that are tight at x^*. The dual is $\min\{yb : y \geq 0, yA = c\}$. By LP duality theory, any dual optimum solution y corresponds to c being expressed a

non-negative combination of the row vectors of A', in other words c is in the cone of the row vectors of A'. If $Ax \leq b$ is TDI then we ask for an integral dual optimum solution; this requires that there is an integer solution to $yA' = c, y \geq 0$. This motivates the following definition.

Definition 11.3. *A set $\{a_1, \ldots, a_k\}$ of vectors in R^n is a* Hilbert basis *if every integral vector $x \in Cone(\{a_1, \ldots, a_k\})$ can be written as $x = \sum_{i=1}^{k} \mu_i a_i$, $\mu_i \geq 0$, $\mu_i \in \mathbf{Z}$ (that is, x is a non-negative integer combination of $a_1, \ldots, a_k$). If the a_i are themselves integral, we call $\{a_1, \ldots, a_k\}$ an* integral Hilbert basis.

The following theorem is not difficult to prove with the background that we have developed.

Theorem 11.4. *The rational system $Ax \leq b$ is TDI if and only if the following property is true for each face F of P; let $A'x = b'$ be the set of* all *inequalities in $Ax \leq b$ that are tight/active at F, then the rows vectors of A' form a Hilbert basis.*

Corollary 11.5. *If the system $Ax \leq b, \alpha x \leq \beta$ is TDI then $Ax \leq b, \alpha x = \beta$ is also TDI.*

The example above raises the question of whether one can take any rational system $Ax \leq b$ and make it TDI by adding sufficiently many redundant inequalities. Indeed that is possible, and is based on the following theorem.

Theorem 11.6. *Every rational polyhedral cone has a finite integral Hilbert basis.*

Theorem 11.7 (Giles-Pulleyblank)**.** *Any rational polyhedron P has a representation $Ax \leq b$ such that*

(i) $P = \{x : Ax \leq b\}$,

(ii) *A is integral, and*

(iii) *$Ax \leq b$ is TDI.*

Moreover, b is integral if and only if P is integral.

11.1 The Cunningham-Marsh Theorem

Suppose we have a graph $G = (V, E)$. Let $P_{odd}(V)$ denote the family of all odd subsets of V with size at least 3. Recall that in our study of matchings, we have examined three different systems of inequalities.

$$
\begin{array}{lrll}
 & x(\delta(v)) & = 1 & \forall v \in V \\
P_1: & x(\delta(U)) & \geq 1 & U \in P_{odd}(V) \\
 & x & \geq 0 &
\end{array}
$$

$$
\begin{array}{lrll}
 & x(\delta(v)) & \leq 1 & \forall v \in V \\
P_2: & x(E[U]) & \leq \lfloor \frac{1}{2}|U| \rfloor & U \in P_{odd}(V) \\
 & x & \geq 0 &
\end{array}
$$

$$
\begin{array}{lrll}
 & x(\delta(v)) & = 1 & \forall v \in V \\
P_3: & x(E[U]) & \leq \lfloor \frac{1}{2}|U| \rfloor & U \in P_{odd}(V) \\
 & x & \geq 0 &
\end{array}
$$

Here P_2 determines the matching polytope for G, while P_1 and P_3 determine the perfect matching polytope.

It is not hard to see that P_1 is not TDI. Consider K_4 with $w(e) = 1$ for every edge e. In this case, the unique optimal dual solution is $y_v = \frac{1}{2}$ for each vertex v.

On the other hand, P_2 and P_3 *are* TDI; this was proven by Cunningham and Marsh [19]. Consider the primal maximization and dual minimzation problems for P_2 below:

$$
\begin{array}{ll}
\max wx & \\
x(\delta(v)) \leq 1 & \forall v \in V \\
x(E[U]) \leq \lfloor \frac{1}{2}|U| \rfloor & \forall U \in P_{odd}(V) \\
x \geq 0 &
\end{array}
\qquad\qquad
\begin{array}{ll}
\min \sum_{v \in V} y_v + \sum_{U \in P_{odd}(V)} z_U \cdot \lfloor \frac{1}{2}|U| \rfloor & \\
y_a + y_b + \sum_{\substack{U \in P_{odd}(V) \\ a,b \in U}} z_U \geq w(ab) & \forall ab \in E \\
y \geq 0,\ z \geq 0 &
\end{array}
$$

By integrality of the matching polytope, the maximum value of the primal is the maximum weight of a matching under w; by duality, this equals the minimum value of the dual. The Cunningham-Marsh Theorem tells us that this minimum value is achieved by integral dual vectors y^*, z^* with the additional condition that the sets $\{U : z^*_U > 0\}$ form a laminar family.

Theorem 11.8 (Cunningham-Marsh)**.** *The system P_2 is TDI (as is P_3). More precisely, for every integral w, there exist integral vectors y and z that are dual feasible such that $\{U : z_U > 0\}$ is laminar and*

$$\sum_{v \in V} y_v + \sum_{U \in P_{odd}(V)} z_U \cdot \lfloor \frac{1}{2}|U| \rfloor = \nu(w)$$

where $\nu(w)$ is the maximum weight of a matching under w.

Exercise 11.1. Show that the Tutte-Berge Formula can be derived from the Cunningham-Marsh Theorem.

Cunningham and Marsh originally proved this theorem algorithmically, but we present a different proof from [57] (Chapter 25); the proof relies on the fact that P_2 is the matching polytope. A different proof is given in [**S**] that does not assume this and in fact derives that P_2 is the matching polytope as a consequence.

Proof. We will use induction on $|E| + w(E)$ (which is legal because w is integral). Note that if $w(e) \leq 0$ for some edge e, we may discard it; hence we may assume that $w(e) \geq 1$ for all $e \in E$.

Case I: Some vertex v belongs to every maximum-weight matching under w.
Define $w' : E \to \mathbf{Z}^+$ by

$$\begin{aligned} w'(e) &= w(e) - 1 \quad \text{if } e \in \delta(v) \\ w'(e) &= w(e) \qquad \text{if } e \notin \delta(v) \end{aligned}$$

Now induct on w'. Let y', z' be an integral optimal dual solution with respect to w' such that $\{U : z'_U > 0\}$ is laminar; the value of this solution is $\nu(w')$. Because v appears in every maximum-weight matching under w, $\nu(w') \leq \nu(w) - 1$; by definition of w', $\nu(w') \geq \nu(w) - 1$. Thus $\nu(w') = \nu(w) - 1$. Let y^* agree with y' everywhere except v, and let $y^*_v = y'_v + 1$. Let $z^* = z'$. Now y^*, z^* is a dual feasible solution with respect to w, the solution is optimal since it has weight $\nu(w') + 1 = \nu(w)$, and $\{U : z^*_U > 0\}$ is laminar since $z^* = z'$.

Case II: No vertex belongs to every maximum-weight matching under w.
Let y, z be a *fractional* optimal dual solution. Observe that $y = 0$, since $y_v > 0$ for some vertex v, together with complementary slackness, would imply that every optimal primal solution covers v, i.e. v belongs to every maximum-weight matching under w. Among all optimal dual solutions y, z (with $y = 0$) choose the one that maximizes $\sum_{U \in P_{odd}(V)} z_U \lfloor \frac{1}{2}|U| \rfloor^2$. To complete the proof, we just need to show that z is integral and $\{U : z_U > 0\}$ is laminar.

Suppose $\{U : z_U > 0\}$ is not laminar; choose $W, X \in P_{odd}(V)$ with $z_W > 0, z_X > 0$, and $W \cap X \neq \emptyset$. We claim that $|W \cap X|$ is odd. Choose $v \in W \cap X$, and let M be a maxmimum-weight matching under w that misses v. Since $z_W > 0$, by complementary slackness, M contains $\lfloor \frac{1}{2}|W| \rfloor$ edges inside W; thus v is the *only* vertex in W missed by M. Similarly, v is the only vertex in X missed by M. Thus M covers $W \cap X - \{v\}$ using only edges inside $W \cap X - \{v\}$, so $|W \cap X - \{v\}|$ is even, and so $|W \cap X|$ is odd. Let ϵ be the smaller of z_W and z_X; form a new dual solution by decreasing z_W and z_X by ϵ and increasing $z_{W \cap X}$ and $z_{W \cup X}$ by ϵ (this is an uncrossing step).

We claim that this change maintains dual feasibility and optimality. Clearly z_W and z_X are still nonnegative. If an edge e is contained in W and X, then

the sum in e's dual constraint loses 2ϵ from z_W and z_X, but gains 2ϵ from $z_{W\cap X}$ and $z_{W\cup X}$, and hence still holds. Likewise, if e is contained in W but not X (or vice-versa), the sum loses ϵ from z_W but gains ϵ from $z_{W\cup X}$. Thus these changes maintained dual feasibility and did not change the value of the solution, so we still have an optimal solution. However, we have increased $\sum_{U\in P_{odd}(V)} z_U \lfloor \frac{1}{2}|U|\rfloor^2$ (the reader should verify this), which contradicts the choice of z. Thus $\{U : z_U > 0\}$ is laminar.

Suppose instead that z is not integral. Choose a maximal $U \in P_{odd}(V)$ such that z_U is not an integer. Let $U_1, \ldots, U_k$ be maximal odd sets contained in U such that each $z_{U_i} > 0$. (Note that we may have $k = 0$.) By laminarity, $U_1, \ldots, U_k$ are disjoint. Let $\alpha = z_U - \lfloor z_U \rfloor$. Form a new dual solution by decreasing z_U by α and increasing each z_{U_i} by α.

We claim that the resulting solution is dual feasible. Clearly we still have $z_U \geq 0$, and no other dual variable was decreased. Thus we need only consider the edge constraints; moreover, the only constraints affected are those corresponding to edges contained within U. Let e be an edge contained in U. If e is contained in some U_i, then the sum in e's constraint loses α from z_U but gains α from z_{U_i}, so the sum does not change. On the other hand, suppose e is not contained in any U_i. By maximality of U and the U_i, U is the only set in P_{odd} containing e. Thus before we changed z_U we had $z_U \geq w(e)$; because $w(e)$ is integral, we must still have $z_U \geq w(e)$. Hence our new solution is dual feasible.

Since the U_i are disjoint, contained in U, and odd sets, $\lfloor \frac{1}{2}|U|\rfloor > \sum_{i=1}^{k} \lfloor \frac{1}{2}|U_i|\rfloor$. Thus our new solution has a smaller dual value than the old solution, which contradicts the optimality of z. It follows that z was integral, which completes the proof.

To show that the system P_3 is TDI, we use Corollary 11.5 and the fact that system P_2 is TDI. ■

Chapter 12

T-joins and Applications[1]

We borrow mainly from [18] (Chapter 5), and [57] (Chapter 29).

Edmonds was motivated to study T-joins by the Chinese postman problem which is related to the well-known Traveling Salesman Problem (TSP).

Problem 12.1. *Let $G = (V, E)$ be an undirected graph and $c : E \to \mathbb{R}^+$ be non-negative edge weights on the edges. A Chinese postman tour is a walk that starts at some arbitrary vertex and returns to it after traversing each edge of E. Note that an edge may be traversed more than once. The goal is to find a postman tour of minimum total edge cost.*

Proposition 12.0.1. *If G is Eulerian then the optimal postman tour is an Eulerian tour of G and has cost equal to $\sum_{e \in E} c(e)$.*

Thus the interesting case is when G is not Eulerian. Let $T \subseteq V$ be the nodes with odd degree in G.

Fact 12.1. *$|T|$ is even.*

Consider a postman tour and say it visits an edge $x(e)$ times, where $x(e) \geq 1$ is an integer. Then, it is easy to see that the multigraph induced by placing $x(e)$ copies of e is in fact Eulerian. Conversely if $x(e) \geq 1$ for each edge and $x(e) \in \mathbb{Z}^+$ such that the multigraph is Eulerian, then it induces a postman tour of cost $\sum_{e \in E} c(e)x(e)$.

We observe that if $x(e) > 2$ then reducing $x(e)$ by 2 maintains feasibility. Thus $x(e) \in \{1, 2\}$ for each e in any minimal solution. If we consider the graph induced by $'(e) = x(e) - 1$ we see that each node in T has odd degree and every other node has even degree. This motivates the definition of T-joins.

Definition 12.2. *Given a graph, $G = (V, E)$, and a set, $T \subseteq V$, a T-join is a subset $J \subseteq E$ such that in the graph (V, J), T is the set of nodes with odd degree.*

[1] Based on notes scribed by Ben Raichel in 2010.

Proposition 12.0.2. *There is a T-join in G iff $|K \cap T|$ is even for each connected component K of G. In particular, if G is connected then there exists a T-join iff $|T|$ is even.*

Proof. Necessity is clear. For sufficiency, assume G is connected, otherwise we can work with each connected component separately. Let $T = \{v_1, v_2, \ldots, v_{2k}\}$. Let P_i be an arbitrary path joining v_i and v_{i+k}. Then the union of the paths $P_1, P_2, \ldots, P_k$ induces a multigraph in which the nodes in T are the only ones with odd degree. Let $x(e)$ be the number of copies of e in the above union. Then $x'(e) = x(e) \ mod \ 2$, is the desired T-join. (Note that the pairing of the vertices was arbitrary and hence any pairing would work.) ■

We leave the proof of the following proposition as an exercise.

Proposition 12.0.3. *J is a T-join iff J is the union of edge disjoint cycles and $\frac{1}{2}|T|$ paths connecting disjoint pairs of nodes in T.*

12.1 Algorithms for Min-cost T-joins

Given $G = (V, E)$, $c : E \to \mathbb{R}$ and $T \subseteq V$, where $|T|$ even, we want to find the min-cost T-join. If all edge costs are non-negative then one can easily reduce the problem to a matching problem as follows. Assume without loss of generality that G is connected.

1. For each pair $u, v \in T$ let $w(uv)$ be the shortest path distance between u and v in G, with edge lengths given by c. Let P_{uv} be the shortest path between u and v.
2. Let H be the complete graph on T with edge weights $w(uv)$.
3. Compute a minimum weight perfect matching M in H.
4. Let $J = \{e \mid e \text{ occurs in an odd number of paths } P_{uv}, uv \in M\}$. Output J.

Theorem 12.3. *There is a strongly polynomial time algorithm to compute a min-cost T-join in a graph, $G = (V, E)$ with $c \geq 0$.*

Proof. To see the correctness of this algorithm first note that it creates a T-join since it will return a collection of $\frac{1}{2}|T|$ disjoint paths, which by Proposition 12.0.3 is a T-join (Note the fourth step in the algorithm is required to handle zero cost edges, and is not necessary if $c > 0$). It can be seen that this T-join is of min-cost since the matching is of min-cost (and since, ignoring zero cost edges, the matching returned must correspond to disjoint paths in G). ■

12.1.1 Negative costs

The interesting thing is that min-cost T-joins can be computed even when edge costs can be negative. This has several non-trivial applications. We reduce the general case to the non-negative cost case. Recall that if A, B are two sets then $A \Delta B = (A - B) \cup (B - A)$ is the symmetric difference between A and B.

Idea: Given $G = (V, E)$ and edge costs $c : E \to \mathbb{Z}$. Let $E^{-1} = \{e \in E \mid c(e) < 0$ be the set of edges with strictly negative costs. To minimize cost we would ideally like to include all edges in E^-. Suppose we consider the graph (V, E^-) and consider the odd-degree nodes T' in this graph; note that $|T'|$ is even. Suppose we get (very) lucky and $T = T'$. Then it is easy to see that E^- is the optimal T-join. However T may not be T'. What do we need to fix? The nodes in $T - T'$ have even degree in (V, E^-) and we need odd degree and $T' - T$ have odd degree while we need even degree for them. Thus we can fix this by a $(T \Delta T')$-join. What is the advantage of this approach? We already included all of E^- and hence we view the problem of computing $(T \Delta T')$-join as adding edges from $E \setminus E^-$ and *removing* some of the added edges from E^-. We can model the process of removing an edge $e \in E^-$ from our existing solution as adding edge e with cost $-c(e)$ which is now positive! Thus we can now compute a join in a graph with non-negative edge costs which we already know how to solve.

The formal algorithm is described below.

1. Let E^- be the set of edges with negative costs in G. Let T' be the set of odd-degree nodes in (V, E^-).
2. Let $d : E \to \mathbb{Z}_+$ where $d(e) = -c(e)$ if $e \in E^-$ and $d(e) = c(e)$ otherwise.
3. Compute a $(T \Delta T')$-join J'' in G with edge costs d (note that $d \geq 0$).
4. Output $J = J'' \Delta E^-$.

It is easy to see that the preceding algorithm runs in strongly polynomial time since we have essentially reduced the problem to non-negative case. We now prove the correctness of the preceding algorithm.

Fact 12.2. *If A,B are two subsets of U then $|A \Delta B|$ is even if and only if $|A|$ and $|B|$ have the same parity (both even or both odd).*

Proposition 12.1.1. *Let J be a T-join and J' be a T'-join then $J \Delta J'$ is a $(T \Delta T')$-join.*

Proof. Verify using the above fact that each $v \in T \Delta T'$ has odd degree and every other node has even degree in $J \Delta J'$.

Alternatively, consider the multigraph induced by $J \cup J'$. A node v has odd degree in this graph iff $v \in T \Delta T'$. $J \Delta J'$ is obtained by removing parallel edges

from $J \cup J'$ which does not affect the parity of the node degrees and hence $J \Delta J'$ is a $(T \Delta T')$-join. ■

Corollary 12.4. *Suppose J' is a T'-join and J'' is a $(T \Delta T')$-join then $J'' \Delta J'$ is a T-join.*

Proof. Note that $(T \Delta T') \Delta T' = T$ and hence the corollary is implied by the preceding proposition. ■

Theorem 12.5. *There is a strongly polynomial time algorithm for computing a min-cost T-join in a graph, even with negative costs on the edges.*

Proof. Consider the algorithm that we had described. We first observe that E' is a T'-join and moreover E' is a min-cost T'-join (why?). Since J'' is a $(T \Delta T')$-join we have $J = J'' \Delta E^-$ is a T-join by Claim 12.4. We now establish that the cost of J is optimal.

Consider *any* T-join X in G with costs c.

$$\begin{aligned} c(X) &= c(X - E^-) + c(X \cap E^-) \\ &= c(X - E^-) - c(E^- - X) + c(E^- - X) + c(X \cap E^-) \\ &= d(X - E^-) + d(E^- - X) + c(E^-) \\ &= d(X \Delta E^-) + c(E^-). \end{aligned}$$

Thus the cost of X is the d-cost of $(X \Delta E^-)$ plus a constant term $c(E^-)$. Moreover $X \Delta E^-$ is a $(T \Delta T')$-join in G. Thus finding a minimum cost T-join in G with respect to c is equivalent to finding a minimum cost $(T \Delta T')$-join in G with cost d. Since J'' is a min-cost $(T \Delta T')$-join in G with cost d we have $d(J'') \leq d(J^* \Delta E^-)$ where J^* is a min-cost T-join.

$$c(J) = d(J \Delta E^-) + c(E^-) = d(J'') + c(E') \leq d(J^* \Delta E^-) + c(E^-) = c(J^*).$$

Thus $c(J) \leq c(J^*)$ and J is a T-join. This proves the correctness of the algorithm. ■

12.1.2 Polyhedral aspects

The following set of inequalities can be shown to determine the characteristic vectors of the set of T-joins in a graph G. As one can see, the odd-set inequalities are a bit complicated to specify.

$$\begin{aligned} 0 \leq x(e) &\leq 1 \\ x(\delta(U) \setminus F) - x(F) &\geq 1 - |F| \qquad U \subseteq V,\ F \subseteq \delta(U),\ |U \cap T| + |F| \text{ is odd} \end{aligned}$$

Another useful polytope that comes up in applications of T-joins is the *dominant* of a the T-joint polytope. The dominant of a polyhedron $P \in \mathbb{R}^n$ is the set $\{y \in \mathbb{R}^n \mid \exists x \in P, x \leq y\}$. Alternatively it is the polyhedron $P + \mathbb{R}^n_+$ written as the Minkowski sum of P and the positive orthant. The dominant is also referred to as the up hull. The dominant of the T-join polytope has a particularly simple form.

$$\begin{aligned} x(e) &\geq 0 \\ x(\delta(U)) &\geq 1 \quad |U \cap T| \text{ is odd} \end{aligned}$$

A set U such that $|U \cap T|$ is odd is called a T-cut. T-cuts are closely related to T-joins and matchings. Recall that we saw the separation oracle for finding the minimum cost T-cut in a graph via the Gomory-Hu tree.

Properties of T-join polytope have found important applications in approximation algorithms for Metric-TSP (see [38] and references).

12.2 Applications

12.2.1 Chinese Postman

We saw earlier that a min-cost postman tour in G is the union of E and a T-join where T is the set of odd degree nodes in G. Hence we can compute a min-cost postman tour.

12.2.2 Shortest Paths and Negative lengths

In directed graphs the well known Bellman-Ford algorithm can be used to check whether a given directed graph, $D = (V, A)$, has negative length cycles in $O(mn)$ time. Moreover, if there is no negative length cycle then the shortest s-t path can be found in the same time. However, one cannot use directed graph algorithms for undirected graphs when there are negative lengths, since bi-directing an undirected edge creates a negative length cycle. However, we can use T-join techniques.

Claim 12.2.1. *An undirected graph, $G = (V, E)$, with $c : E \to \mathbb{R}$ has a negative length cycle iff an $\emptyset$-join has negative cost.*

Proof. An $\emptyset$-join is a subgraph of G in which all degree are even (Eulerian subgraph). A negative length cycle is an $\emptyset$-join of negative cost. Conversely, suppose there J is a negative cost $\emptyset$-join. We can decompose J into a collection of edge-disjoint cycles. One of these cycles must have negative cost if J has negative cost. ■

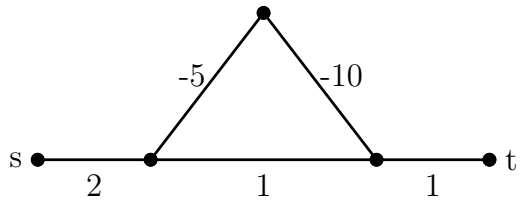


Figure 12.1: An example of a graph with a negative cost $\emptyset$-join

Claim 12.2.2. *If G has no negative length cycle then the min-cost $\{s,t\}$-join gives an s-t shortest path.*

Remark 12.1. It is important to first check for negative length cycles before finding an $\{s,t\}$-join since the shortest s-t path can have negative length even when there is no negative length cycle. For instance, in the example in Fig **??** the min-cost T-join consists of all edge other than the cost 1 edge in the triangle but the graph has a negative length cycle.

Theorem 12.6. *There is a strongly polynomial time algorithm that given an undirected graph, $G(V,E)$, with $c : E \to \mathbb{R}$, either outputs a negative length cycle or an s-t shortest path.*

Proof sketch. We first compute a min-cost $\emptyset$-join. From Claim 12.2.1, if the $\emptyset$-join has negative cost then we can produce a negative length cycle. Otherwise, we know there is no negative length cycle and from Claim 12.2.2 the min-cost $\{s,t\}$-join yields an s-t shortest path. In each case the T-join can be computed using the algorithm from the previous section. ■

12.2.3 Max-cut in planar graphs

Since one can compute min-cost T-joins with negative costs, one can compute max-cost T-joins as well. The max-cut problem is the following.

Problem 12.7. *Given an undirected graph $G = (V,E)$ with non-negative edge weights $w : E \to \mathbb{R}_+$, find a partition of V into $(S, S \setminus V)$ so as to maximize $w(\delta(S))$.*

Max-cut is NP-hard in general graphs, but Hadlock showed how T-joins can be used to solve it in polynomial time for planar graphs. A basic fact is that in planar graphs, cuts in G correspond to collections of edge disjoint cycles in the dual graph G^*. Thus to find a max-cut in G we compute a max $\emptyset$-join in G^* where the weight of an edge in G^* is the same as its corresponding edge in the primal.

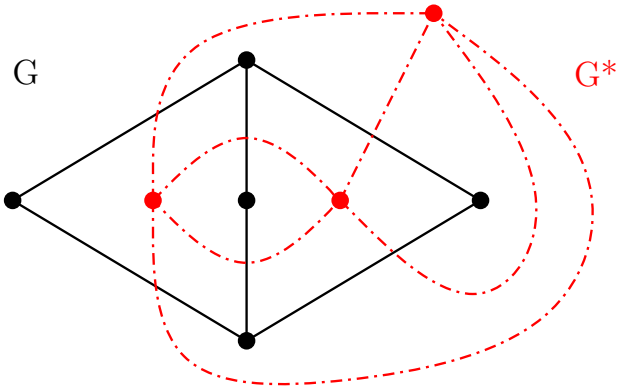


Figure 12.2: A planar graph, G, in black, and its dual, G^*, in dashed red.

12.2.4 Approximating Metric-TSP

TSP is a well-known NP-Hard problem. Given an undirected graph $G = (V, E)$ with non-negative edge costs $c : E \to \mathbb{Z}_+$, find a Hamiltonian cycle of minimum cost. TSP is inapproximable since even deciding whether G has a Hamilton cycle is NP-Complete. However, an important special case is Metric-TSP. Here we assume that G is a complete graph and c satisfies triangle inequality: $c(u, w) \leq c(u, v) + c(v, w)$ for all $u, v, w \in V$. Alternatively, we are interested in a shortest spanning tour rather than in a Hamilton cycle; that is, the tour should visit each vertex and return to the starting point and can visit a vertex multiple times. In this view what we seek is a minimum cost spanning connected Eulerian muligraph in G. This view allows us to retain the structure of G while the metric view requires us to compute the shortest path distances in G to make it complete and potentially loses the sparsity structure as well as topology of G. However, both view points are useful.

Metric-TSP is well-studied and one can find good approximation algorithms in various setting. A very simple 2-approximation is the following. Compute an MST $H = (V, E_H)$ of G. It is easy to see that $c(E_H) \leq \text{OPT}$ since any TSP tour contains a connected subgraph of G; here OPT is the cost of an optimum TSP tour in G. Once H is computed we can make two copies of each edge in H which makes it Eulerian and connected and the cost of $2H$ is $2c(E_H) \leq 2\,\text{OPT}$.

Christofides's heuristic from late 1970's obtained a 3/2-approximation by refining the above simple heuristic. Let H be an MST as before. Note that we wish to make H Eulerian. For this let T be the set of odd-degree nodes in H. Thus it suffices to compute a T-join J and add J to H to obtain an Eulerian graph. Christofides's main objservation is the following.

Lemma 12.1. *Let T be* any *even subset of nodes. Then there is a T-join in G whose cost is at most* OPT/2.

Proof sketch. Consider the metric-view of TSP. Let C be the optimum Hamilton cycle whose cost is OPT. Consider the vertices of T on C. Note that $|T|$ is even. One can partition C into two T-joins. If $v_1, v_2, \ldots, v_{2k-1}, v_{2k}$ are the vertices in T along cycle C we consider one T-join by considering the pairing $v_1v_2, v_3v_4, \ldots, v_{2k-1}v_{2k}$ and another T-join by considering the pairing $v_2v_3, v_4v_5, \ldots, v_{2k}v_1$. One of these T-joins has cost at most $c(C)/2$ since the edge set of C is partitioned between these two joins. ■

This immediately implies a 3/2 approximation since we can compute a min-cost T-join in G and the graph with edge-set $E_H \cup J$ is an Eulerian and connected spanning subgraph.

LP Relaxation and the 4/3-conjecture: There is a natural LP relaxation for TSP that was first considered by Dantzig, Fulkerson and Johnson. Held and Karp developed an iterative combinatorial algorithm that converges to the lower bound provided by the LP relaxation which is well-known and for this reason the LP relaxation is sometimes referred to as the Held-Karp LP even though it was developed earlier. Wolsey [69] showed that the integrality gap of this LP-relaxation is at most 3/2 by mimicking the analysis of Christofides's heuristic with respect to the lower bound provided by the LP relaxation. For this one needs to understand the polyhedral aspects of spanning trees, T-joints and the TSP LP. There is a class of examples which show a lower bound of 4/3 on the integrality gap. A well-known open problem in the literature is to determine the worst-case integrality gap of the LP relaxation. It is conjectured that it is 4/3.

Randomized variants of Christofides and progress: The first important progress in improving the 3/2-approximation for Metric-TSP in general instances (improved results are known for various special cases) came about from the work of Oveis-Gharan, Saberi and Singh [27] which itself derived its inspiration from a work on the ATSP problem (assymetric TSP problem which is on directed graphs) [4]. This approach is based on adapting the Christofides heuristic by picking a random spanning tree according to the LP solution and then augmenting with a T-join. The key to the analysis is understanding the expected cost of the T-join and this requires rather sophisticated techniques. It is only very recently that the 3/2 approximation and integrality gap have been improved for all instances (by a tiny but fixed ϵ) in the work of Karlin, Klein and Oveis-Gharan [38]; the paper has pointers to the considerable amount of literature on this topic.

Chapter 13

Matroids[1]

13.1 Introduction to Matroids

Matroids (formally introduced by Whitney in 1935) are combinatorial structures that capture the abstract properties of linear independence defined for vector spaces and borrow from graph theory. The power and utitlity of matroids is because of the numerous settings in which they arise naturally (and sometimes surprisingly), and their connection to several areas of mathematics and optimization.

Definition 13.1. *A matroid $\mathcal{M}$ is a tuple $(S, \mathcal{I})$, where S is a finite ground set and $\mathcal{I} \subseteq 2^S$ (the power set of S) is a collection of* independent sets *which satisfy the following properties.*

1. *$\mathcal{I}$ is nonempty, in particular, $\emptyset \in \mathcal{I}$.*
2. *$\mathcal{I}$ is downward closed, that is, if $X \in \mathcal{I}$ and $Y \subseteq X$, then $Y \in \mathcal{I}$.*
3. *If $X, Y \in \mathcal{I}$, and $|X| < |Y|$, then $\exists y \in Y \backslash X$ such that $X + y \in \mathcal{I}$.*

Exercise 13.1. Show that the third property in Definition 13.1 can be replaced by the following: if $X, Y \in \mathcal{I}$, $|Y| = |X| + 1$, then $\exists y \in Y \backslash X$ such that $X + y \in \mathcal{I}$.

Example 13.1 (Vector Matroid (Linear Matroid)). Let M be a $m \times n$ matrix with entries in some field $\mathbb{F}$ and v_i be the i^{th} column of M, viewed as a vector in the vector space $\mathbb{F}^m$. Let $S = \{1, 2, \ldots, n\}$ and $\mathcal{I} = \{I : I \subseteq S, \{v_i\}_{i \in I}$ are linearly independent$\}$ (under the usual definition of linear independence in linear algebra). Then $\mathcal{M} = (S, \mathcal{I})$ is a matroid. To see this, notice that properties 1 and 2 of Definition 13.1 are trivially satisfied. To show property

[1]Based on notes scribed by Vineet Abhishek, Sreeram Kannan, and Alina Ene in 2010.

3, suppose $X, Y \in \mathcal{I}$ and $|X| < |Y|$. If there is no $y \in Y \backslash X$ such that $X + y \in \mathcal{I}$, then Y is in the span of $\{v_x\}_{x \in X}$. Hence, $|Y| \leq |X|$ which is a contradiction.

Example 13.2 (Graphic Matroid). Let $\mathcal{G} = (V, E)$ be an undirected multi-graph (loops allowed). Let $\mathcal{I} = \{I : I \subseteq E,$ I induces a forest in $\mathcal{G}\}$. Then $\mathcal{M} = (E, \mathcal{I})$ is a matroid. Again, the first two properties of Definition 13.1 are easy to verify. To show property 3, suppose $X, Y \in \mathcal{I}$ such that $|X| < |Y|$. Both X and Y induce forests in $\mathcal{G}$. Let $V_1, V_2, \ldots, V_{k(X)}$ be the vertex sets of the connected components in $G[X]$ ($\mathcal{G}$ restricted to the edge set X). Here, $k(X)$ denotes the number of connected components in $G[X]$. Each connected component is a tree. Hence, if there is an edge $y \in Y$ that connects two different components of $\mathcal{G}[X]$ then $\mathcal{G}[X + y]$ is again a forest and we are done. If not, then every edge $y \in Y$ have its both ends in the same component of $\mathcal{G}[X]$. Thus, the number of connected components in $\mathcal{G}[Y]$, denoted by $k(Y)$, is at least $k(X)$. Thus, $|X| = |V| - k(X) \geq |V| - k(Y) = |Y|$, which is a contradiction.

Graphic matroids are also referred to as the Cycle matroid.

Example 13.3 (Uniform Matroid). Let $\mathcal{M} = (S, \mathcal{I})$, where S is any finite nonempty set, and $\mathcal{I} = \{I : I \subseteq S, |I| \leq k\}$ for some positive integer k. Then $\mathcal{M}$ is a matroid.

Example 13.4 (Partition Matroid). Let $S_1, S_2, \ldots, S_h$ be a partition of S and $k_1, k_2, \ldots, k_h$ be positive integers. Let $\mathcal{I} = \{I : I \subseteq S,\ |I \cap S_i| \leq k_i$ for all $1 \leq i \leq h\}$. Then $\mathcal{M} = (S, \mathcal{I})$ is a matroid.

Example 13.5 (Laminar Matroid). Let $\mathcal{F}$ be a laminar family on S (i.e., if $X, Y \in \mathcal{F}$ then $X, Y \subseteq S$; and either $X \cap Y = \emptyset$, or $X \subseteq Y$, or $Y \subseteq X$) such that each $x \in S$ is in some set $X \in \mathcal{F}$. For each $X \in \mathcal{F}$, let $k(X)$ be a positive integer associated with it. Let $\mathcal{I} = \{I : I \subseteq S,\ |I \cap X| \leq k(X)\ \forall X \in \mathcal{F}\}$. Then $\mathcal{M} = (S, \mathcal{I})$ is a matroid. Notice that laminar matroids generalize partition matroids, which in turn generalize uniform matroids.

Exercise 13.2. Verify Example 13.5.

Example 13.6 (Transversal Matroid). Let $\mathcal{G} = (V, E)$ be a bipartite graph with bipartition V_1 and V_2. let $\mathcal{I} = \{I : I \subseteq V_1, \exists$ a matching M in $\mathcal{G}$ that covers $I\}$. Then $\mathcal{M} = (V_1, \mathcal{I})$ is a matroid.

Example 13.7 (Matching Matroid). Let $\mathcal{G} = (V, E)$ be an undirected graph. Let $\mathcal{I} = \{I : I \subseteq V, \exists$ a matching M in $\mathcal{G}$ that covers $I\}$. Then $\mathcal{M} = (V, \mathcal{I})$ is a matroid.

Exercise 13.3. Verify Examples 13.6 and 13.7.

13.1.1 Representation of Matroids

As we discussed, matroids were defined by Whitney to abstractly define independence. A vector matroid (also called linear matroid) is defined by a collection

of n vectors from a vector space over a finite field where independence in the matroid is defined via linear independence in the vector space. A matroid $\mathcal{M} = (S, \mathcal{I})$ is *representable* over a field $\mathbb{F}$ if there exist a vector matroid $(S', \mathcal{I}')$ over $\mathbb{F}$ and a bijection $f : S \to S'$ such that $f(A) \in \mathcal{I}'$ iff $A \in \mathcal{I}$. Such a matroid is $\mathbb{F}$-representable.

We note that all the matroids we discussed so far are representable. We show that graphic matroid can be represented over *any* field. Given a graph $G = (V, E)$ let $|V| = n$ and $|E| = m$. Note that the ground set of the the graphic matroid is the set of edges E. We stick to the usual graph notation even though we have been using n to represent the size of the ground set for a matroid. Let $V = \{1, 2, \ldots, n\}$. We arbitrarily orient the edges of G to obtain a directed graph G' where each $e \in E$ is now an oriented arc. Suppose $e = (i, j)$ where we think of e as oriented from i to j. We create a n-dimensional vector v_e for e where $v_e(i) = -1$, $v_e(j) = 1$ and $v_e(k) = 0$ if $k \notin \{i, j\}$. Note that 1 is the multiplicative identity in $\mathbb{F}$ and hence this works over any field. We leave it as an exercise to verify that if a set of edges form a forest iff their corresponding vectors are linearly independent.

A matroid is *binary* if it can be represented over the two element binary field $\{0, 1\}$. A matroid is *regular* if it can be represented over any field. We just saw that graphic matroids are regular. An important connection of matroid theory to TU matrices is the following. Call a matroid $\mathcal{M}$ *unimodular* if it can be presented by a TU matrix A over the fields of real numbers. Then one can show that a matroid is unimodular iff it is regular.

A natural question is whether every matroid is representable over some field. The answer is no which makes matroids interesting as a concept that is motivated by linear independence but is not subsumbed by it. Such matroids are called non-representable and the smallest one is the Vámost matroid over 8 elements. Algebraic matroids generalize linear matroids and there are non-algebraic matroids such as the Vámos matroid.

13.1.2 Base, Circuit, Rank, Span and Flat

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

Definition 13.2. *A set $X \subseteq S$ such that $X \notin \mathcal{I}$ is called a* dependent set *of $\mathcal{M}$.*

Definition 13.3. *A* loop *is an element $x \in S$ such that $\{x\}$ is dependent.*

Notice that a loop cannot appear in any sets in $\mathcal{I}$ and can be effectively removed from S.

Definition 13.4. *A* base *is an inclusion wise maximal set in $\mathcal{I}$.*

Proposition 13.1.1. *If B and $\widehat{B}$ are bases of $\mathcal{M}$ then $|B| = |\widehat{B}|$.*

Proof. If $|B| < |\widehat{B}|$ then from Definition 13.1, $\exists x \in \widehat{B} \setminus B$ such that $B + x \in \mathcal{I}$, contradicting the maximality of B. ∎

Notice that the notion of base here is similar to that of a *basis* in linear algebra.

Lemma 13.1. *Let B and $\widehat{B}$ be bases of $\mathcal{M}$ and $x \in \widehat{B} \setminus B$. Then $\exists y \in B \setminus \widehat{B}$ such that $\widehat{B} - x + y$ is a base of $\mathcal{M}$.*

Proof. Since $\widehat{B} - x \in \mathcal{I}$ and $|\widehat{B} - x| < |B|$, $\exists y \in B \setminus \widehat{B}$ such that $\widehat{B} - x + y \in \mathcal{I}$. Then $|\widehat{B} - x + y| = |B|$, implying that $\widehat{B} - x + y$ is a base of $\mathcal{M}$. ∎

Definition 13.5. *Let $\mathcal{M} = (S, \mathcal{I})$ be a matroid. Given $\widehat{S} \subseteq S$, let $\widehat{\mathcal{I}} = \{I : I \subseteq \widehat{S}, I \in \mathcal{I}\}$. Then $\widehat{\mathcal{M}} = (\widehat{S}, \widehat{\mathcal{I}})$ is also a matroid and is referred to as the* restriction of $\mathcal{M}$ to $\widehat{S}$.

Definition 13.6. *Given $\mathcal{M} = (S, \mathcal{I})$ and $\widehat{S} \subseteq S$, $\widehat{B}$ is a* base for $\widehat{S}$ *if $\widehat{B}$ is a base of $\widehat{\mathcal{M}}$, where $\widehat{\mathcal{M}}$ is a restriction of $\mathcal{M}$ to $\widehat{S}$.*

Proposition 13.1.2. *Given $\mathcal{M} = (S, \mathcal{I})$, let $B \subseteq X$ be a base for X. Then for any $Y \supseteq X$, there exist a base $\widehat{B}$ for Y that contains B.*

Proof. Notice that B is independent in the restriction of $\mathcal{M}$ to Y (henceforth *independent in Y*). Let $\widehat{B}$ be the maximal independent set in Y that contains B. Since all maximal independent sets have same size, $\widehat{B}$ is a base of Y. ∎

Definition 13.7. *Given $\mathcal{M} = (S, \mathcal{I})$, a circuit is a minimal dependent set (i.e., an inclusion wise minimal set in $2^S \setminus \mathcal{I}$). Thus, if C is a circuit then $\forall x \in C$, $C - x \in \mathcal{I}$.*

The definition of a circuit is related to graph theory in the following sense: if $\mathcal{M}$ is the graphic matroid of a graph $\mathcal{G}$, then the circuits of $\mathcal{M}$ are the cycles of $\mathcal{G}$. Single element circuits of a matroid are loops; if $\mathcal{M}$ is a graphic matroid of a graph $\mathcal{G}$, then the set of loops of $\mathcal{M}$ is precisely the set of loops of $\mathcal{G}$.

Lemma 13.2. *Let C_1 and C_2 be two circuits such that $C_1 \neq C_2$ and $x \in C_1 \cap C_2$. Then for every $x_1 \in C_1 \setminus C_2$ there is a circuit C such that $x_1 \in C$ and $C \subseteq C_1 \cup C_2 - x$. In particular, $C_1 \cup C_2 - x$ contains a circuit.*

Proof. Notice that $C_1 \setminus C_2$ is nonempty (and so is $C_2 \setminus C_1$), otherwise, $C_1 \subseteq C_2$. Since $C_1 \neq C_2$, C_1 is a strict subset of C_2, contradicting the minimality of C_2.

Let $C_1 \cup C_2 - x$ contain no circuits. Then $B = C_1 \cup C_2 - x$ is independent, and hence, a base for $C_1 \cup C_2$ (since it is maximal). Also, $|B| = |C_1 \cup C_2| - 1$. Since $C_1 \cap C_2$ is an independent set (otherwise C_1, C_2 are not minimal dependent sets), we can find a base $\widehat{B}$ for $C_1 \cup C_2$ that contains $C_1 \cap C_2$. Then $|\widehat{B}| = |B| = |C_1 \cup C_2| - 1$.

Since $C_1 \setminus C_2$ and $C_2 \setminus C_1$ are both non-empty, this is possible only if either $C_1 \subseteq \widehat{B}$ or $C_2 \subseteq \widehat{B}$, contradicting that $\widehat{B}$ is a base. Hence, $C_1 \cup C_2 - x$ must contain a circuit.

Now let $x_1 \in C_1 \setminus C_2$. Let B_1 be a base for $C_1 \cup C_2$ that contains $C_1 - x_1$, and B_2 be a base for $C_1 \cup C_2$ that contains $C_2 - x$. Clearly, $x_1 \notin B_1$ and $x \notin B_2$. If $x_1 \notin B_2$ then $B_2 + x_1$ must have a circuit and we are done. If $x_1 \in B_2$, then from Lemma 13.1, there exists $\widehat{x} \in B_1 \setminus B_2$ such that $\widehat{B} = B_2 - x_1 + \widehat{x}$ is a base for $C_1 \cup C_2$. Notice that $\widehat{x} \neq x$, otherwise $C_2 \subseteq \widehat{B}$. Thus, $x_1 \notin \widehat{B}$ and $\widehat{B} + x_1$ contains the circuit satisfying the condition of Lemma 13.2. ■

Corollary 13.8. *Let $\mathcal{M} = (S, \mathcal{I})$ be a matroid. If $X \in \mathcal{I}$ and $y \notin X$ then either $X + y \in \mathcal{I}$ or there is a unique circuit C in $X + y$. Moreover, for each $\widehat{y} \in C$, $X + y - \widehat{y} \in \mathcal{I}$.*

Proof. If $X + y \notin \mathcal{I}$, then it must contain a circuit C_1. Assume there is another circuit $C_2 \subseteq X + y$, and $C_1 \neq C_2$. Since $X \in \mathcal{I}$, both C_1 and C_2 must contain y. From Lemma 13.2, $C_1 \cup C_2 - y$ contains a circuit. But this is a contradiction since $C_1 \cup C_2 - y \subseteq X$. Hence, $X + y$ contains a unique circuit, call it C. Now, if for some $\widehat{y} \in C$, $X + y - \widehat{y} \notin \mathcal{I}$, then $X + y - \widehat{y}$ is dependent and contains a circuit $\widehat{C}$. However, $\widehat{C} \neq C$ since $\widehat{y} \notin \widehat{C}$, contradicting that C is unique. ■

Corollary 13.9. *If B and $\widehat{B}$ are bases. Let $\widehat{x} \in \widehat{B} \setminus B$, then $\exists x \in B \setminus \widehat{B}$ such that $B + \widehat{x} - x$ is a base.*

Proof. Follows from Corollary 13.8. ■

Definition 13.10. *Let $\mathcal{M} = (S, \mathcal{I})$ be a matroid. The* rank function*, denoted by $r_{\mathcal{M}}$, of $\mathcal{M}$ is $r_{\mathcal{M}} : 2^S \mapsto \mathbb{Z}_+$, where for $X \subseteq S$, $r_{\mathcal{M}}(X)$ is the size of a maximum independent set contained in X.*

Note that the above definition assigns a unique number to each set X since all maximal independent sets contained in X have the same cardinality.

Proposition 13.1.3. *Given a matroid $\mathcal{M} = (S, \mathcal{I})$, the rank function $r_{\mathcal{M}}$ has the following properties:*

1. *$0 \leq r_{\mathcal{M}}(X) \leq |X|$ for all $X \subseteq S$.*
2. *$r_{\mathcal{M}}$ is submodular; i.e., for any $X, Y \subseteq S$, $r_{\mathcal{M}}(X \cup Y) + r_{\mathcal{M}}(X \cap Y) \leq r_{\mathcal{M}}(X) + r_{\mathcal{M}}(Y)$.*

Proof. Property 1 is by the definition of $r_{\mathcal{M}}$. To show the second property, we use the equivalent definition of submodularity; i.e., we show that if $X \subseteq Y$ and $z \in S$, then $r_{\mathcal{M}}(X + z) - r_{\mathcal{M}}(X) \geq r_{\mathcal{M}}(Y + z) - r_{\mathcal{M}}(Y)$. First notice that for any

$X \subseteq S$ and $z \in S$, $r_{\mathcal{M}}(X + z) \leq r_{\mathcal{M}}(X) + 1$. Thus, we only need to show that if $r_{\mathcal{M}}(Y + z) - r_{\mathcal{M}}(Y) = 1$ then $r_{\mathcal{M}}(X + z) - r_{\mathcal{M}}(X) = 1$ for any $X \subseteq Y$.

If $r_{\mathcal{M}}(Y + z) - r_{\mathcal{M}}(Y) = 1$, then every base B of $Y + z$ contains z. Let $\widehat{B}$ be a base of X. Since $X \subseteq Y + z$, from Proposition 13.1.2, there exists a base $\bar{B}$ of $Y + z$ such that $\bar{B} \supseteq \widehat{B}$. Then $\widehat{B} + z$ is independent, implying $r_{\mathcal{M}}(X + z) - r_{\mathcal{M}}(X) = 1$ as $\widehat{B} + z$ is a base in $X + z$. ■

Definition 13.11. *Let $\mathcal{M} = (S, \mathcal{I})$ be a matroid. For any $X \subseteq S$, the* span *of X, denoted by $span_M(X)$, is defined as $span_M(X) = \{y : y \in S, r_{\mathcal{M}}(X + y) = r_{\mathcal{M}}(X)\}$. A set $X \subseteq S$ is* spanning *if $span_M(X) = S$.*

Exercise 13.4. Prove the following properties about the span function $\text{span}_M : 2^S \to 2^S$.

- If $T, U \subseteq S$ and $U \subseteq \text{span}_M(T)$ then $\text{span}_M(U) \subseteq \text{span}_M(T)$.
- If $T \subseteq S$, $t \in S \setminus T$ and $s \in \text{span}_M(T + t) \setminus \text{span}_M(T)$ then $t \in \text{span}_M(T + s)$.

Definition 13.12. *Let $\mathcal{M} = (S, \mathcal{I})$ be a matroid. A subset $X \subseteq S$ is a* flat *of $\mathcal{M}$ iff $span_M(X) = X$.*

Exercise 13.5. Prove the following properties about flats.

- If F_1 and F_2 are flats then $F_1 \cap F_2$ is a flat.
- If F is a flat and $t \in S \setminus F$ and F' is a smallest flat containing $F + t$ then there is no flat F'' with $F \subset F'' \subset F'$.

Exercise 13.6. Show that a set $I \subseteq S$ is independent in a matroid $\mathcal{M}$ iff $\forall y \in I$, there exists a flat F such that $I - y \subseteq F$ and $y \notin F$.

Remark 13.1. We showed basic properties of bases, circuits, rank, span and flats of a matroid. One can show that a matroid can alternatively be specified by defining its bases or circuits or rank or span or flats that satisfy these properties. We refer the reader to [57].

13.1.3 Operations on a Matroid

Definition 13.13. *A matroid $\mathcal{M} = (S, \mathcal{I})$ is defined as connected if $r_{\mathcal{M}}(U) + r_{\mathcal{M}}(S \setminus U) > r_{\mathcal{M}}(S)$ for each $U \subseteq S, U \neq \emptyset$. Equivalently, for each $s, t \in S, s \neq t$, there is a circuit containing both s, t.*

Dual

Given a matroid $\mathcal{M} = (S, \mathcal{I})$ its dual matroid $\mathcal{M}^* = (S, \mathcal{I}^*)$ is defined as follows:

$$\mathcal{I}^* = \{I \in S \mid S \setminus I \text{ is spanning in } \mathcal{M}, \text{ i.e., } r_{\mathcal{M}}(S\backslash I) = r_{\mathcal{M}}(S)\}.$$

Exercise 13.7. Verify that $(S, \mathcal{I}^*)$ is indeed a matroid.

The following facts are easy to prove:

1. $\mathcal{M}^{**} = \mathcal{M}$.
2. B is a base of $\mathcal{M}^*$ iff $S\backslash B$ is a base of $\mathcal{M}$.
3. $r_{\mathcal{M}^*}(U) = |U| + r_{\mathcal{M}}(S\backslash U) - r_{\mathcal{M}}(S)$

Remark 13.2. Having an independence or rank oracle for $\mathcal{M}$ implies one has it for $\mathcal{M}^*$ too.

Exercise 13.8. Prove that $\mathcal{M}$ is connected if and only if $\mathcal{M}^*$ is.

Deletion

Definition 13.14. *Given a matroid $\mathcal{M} = (S, \mathcal{I})$ and $e \in S$, deleting e from $\mathcal{M}$ generates a new matroid*

$$\mathcal{M}' = \mathcal{M}\backslash e = (S - e, \mathcal{I}'), \tag{13.1}$$

where $\mathcal{I}' = \{I - e | I \in \mathcal{I}\}$. For $Z \subseteq S$, the matroid $\mathcal{M}\backslash Z$ is obtained similarly by restricting the matroid $\mathcal{M}$ to $S\backslash Z$.

Contraction

Definition 13.15. *Given a matroid $\mathcal{M} = (S, \mathcal{I})$ and $e \in S$, the contraction of the matroid with respect to e is defined as $\mathcal{M}/e = (\mathcal{M}^*\backslash e)^*$, i.e., it is obtained by deleting e in the dual and taking its dual. Similarly for a set $Z \subseteq S$, we can similarly define $\mathcal{M}/Z = (\mathcal{M}^*\backslash Z)^*$*

One can also define contraction in a different way which is perhaps more natural. It is instructive to consider it from a graph theoretic perspective. If e is a loop, $\mathcal{M}/e = \mathcal{M}\backslash e$, else $\mathcal{M}/e = (S - e, \mathcal{I}')$ where

$$\mathcal{I}' = \{I \in S - e | I + e \in \mathcal{I}\}.$$

For the case of contracting a subset Z, we can take a base $X \subseteq Z$ and $M/Z = (S\backslash Z, \mathcal{I}')$, where

$$\mathcal{I}' = \{I \in S\backslash Z | I \cup Z \in \mathcal{I}\}.$$

Also

$$r_{\mathcal{M}/Z}(X) = r_{\mathcal{M}}(X \cup Z) - r_{\mathcal{M}}(Z)$$

Exercise 13.9. Show that

$$\mathcal{M}/\{e_1, e_2\} = (\mathcal{M}/e_1)/e_2 = (\mathcal{M}/e_2)/e_1. \tag{13.2}$$

Minor

Similar to graph minors, matroid minors can be defined, and they play an important role in characterizing the type of matroids.

Definition 13.16. *A matroid $\mathcal{M}'$ is a minor of a matroid $\mathcal{M}$ if $\mathcal{M}'$ is obtained from $\mathcal{M}$ by a sequence of contractions and deletions.*

13.2 Maximum Weight Independent Set in a Matroid

Matroids have some important algorithmic properties, the simplest one being that the problem of determining the maximum weight independent set in a matroid can be solved using a greedy algorithm. The maximum weight independent set problem is stated as follows: Given $\mathcal{M} = (S, \mathcal{I})$ and $w : S \to R$, output

$$\arg\max_{I \in \mathcal{I}} w(I). \tag{13.3}$$

13.2.1 Greedy Algorithm

The greedy algorithm can be stated as follows:

1. Discard all $e \in S$ where $w(e) \leq 0$ or e is a loop.
2. Let $S = \{e_1, ..., e_n\}$ such that $w(e_1) \geq w(e_2) ... \geq w(e_n)$.
3. $X \leftarrow \emptyset$.
4. For $i = 1$ to n do
   if $(X + e_i \in \mathcal{I})$ then $X \leftarrow X + e_i$.
5. Output X.

The above algorithm had to specifically take care of loops and edges with non-negative weights. An equivalent algorithm is the following.

1. Let $S = \{e_1, ..., e_n\}$ such that $w(e_1) \geq w(e_2) ... \geq w(e_n)$.
2. $X \leftarrow \emptyset$.
3. For $i = 1$ to n, do
   if $(X + e_i \in \mathcal{I})$ and $w(X + e_i) \geq w(X)$, then $X \leftarrow X + e_i$.

4. Output X.

Theorem 13.17. *The greedy algorithm outputs an optimum solution to the maximum weight independent set problem.*

Proof. Without loss of generality, assume $w(e) > 0, \forall e \in S$ and that there are no loops.

Claim 13.2.1. *There exists an optimum solution that contains e_1.*

Assuming this claim is true, we can use induction to show that greedy algorithm has to yield an optimum solution. This is because the greedy algorithm is recursively finding an optimum solution in the matroid $\mathcal{M}'$ which is obtained by contracting e_1; formally $\mathcal{M}' = (S - e, \mathcal{I}')$ where $\mathcal{I}' = \{I \in S - e \mid I + e_1 \in \mathcal{I}\}$.

To prove the claim, let I^* be an optimum solution. If $e_1 \in I^*$, we are done, else, we can see that $I^* + e_1$ is not independent, otherwise $w(I^* + e_1) > w(I^*)$ contradicting optimality of I^*. Thus $I^* + e_1$ contains a circuit, and hence, from Corollary 13.8, $\exists e \in I^*$ such that $I^* - e + e_1 \in \mathcal{I}$. $w(I^* - e + e_1) \geq w(I^*)$ since $w(e_1)$ has the largest weight among all the elements in the set. Thus there is an optimum solution $I^* - e + e_1$ that contains e_1. ■

Remark 13.3. If all weights are non-negative then it is easy to see that the greedy algorithm outputs a base of $\mathcal{M}$. We can adapt the greedy algorithm to solve the maximum weight base problem by making all weights non-negative by adding a large constant to each of the weights. Thus max-weight base problem, and equivalently min-cost base problem can be solved (by taking the weights to be the negative of the costs).

Remark 13.4. Kruskal's algorithm for finding the maximum weight spanning tree can be interpreted as a special case of the greedy algorithm for matroids when applied to the graphic matroid corresponding to the graph.

Oracles for a Matroid

Since the set of all independence sets could be exponential in $|S|$, it is infeasible to use this representation. Instead we resort to one of the two oracles in order to efficiently solve optimization problems:

- An independence oracle that given $A \subseteq S$, returns whether $A \in \mathcal{I}$ or not.
- A rank oracle that given $A \subseteq S$, returns $r_{\mathcal{M}}(A)$.

These two oracles are equivalent in the sense that one can be recovered from the other in polynomial time.

13.3 Matroid Polytope

Edmonds utilized the Greedy algorithm in proving the following theorem:

Theorem 13.18. *The following polytope is the convex hull of the characteristic vectors of the independent sets of a matroid $\mathcal{M} = (S, \mathcal{I})$ with rank function $r_{\mathcal{M}} : 2^S \to \mathcal{Z}_+$,*

$$\begin{aligned} x(A) &\leq r_{\mathcal{M}}(A) \quad \forall A \subseteq S, \\ x(A) &\geq 0. \end{aligned}$$

Also, the system of inequalities described above is TDI.

Proof. We will show that the above system of inequalities is TDI (Totally Dual Integral), which will in turn imply that the polytope is integral since $r_{\mathcal{M}}(.)$ is integer valued.

Let us consider the primal and dual linear programs for some integral weight vector $w : S \to \mathcal{Z}$. We will show that the solution picked by Greedy algorithm is the optimal solution for primal by producing a dual solution that attains the same value. Alternately we could show that the dual solution and the primal solution picked by Greedy satisfy complementary slackness.

$$\begin{aligned} \text{Primal: } \max & \sum_{e \in S} w(e)x(e) \\ & x(A) \leq r(A), \quad \forall\, A \subseteq S \\ & x \geq 0 \end{aligned}$$

$$\begin{aligned} \text{Dual: } \min & \sum_{A \subseteq S} r(A)y(A) \\ & \sum_{A : e \in A} y(A) \geq w(e), \quad \forall e \in S \\ & y \geq 0 \end{aligned}$$

Let $S = \{e_1, ..., e_n\}$ such that $w(e_1) \geq w(e_2) ... \geq w(e_n) \geq 0$, since setting $w(e_i) = 0$ whenever $w(e_i) < 0$ does not alter the solution to the primal or dual. Define $A_j = \{e_1, ..., e_j\}$ with $A_0 = \emptyset$. It is easy to see that $r(A_j) = r(A_{j-1}) + 1$ iff e_j is picked by Greedy. Consider the following dual solution

$$\begin{aligned} y(A_j) &= w(e_j) - w(e_{j+1}), j < n \\ &= w(e_n), j = n \\ y(A) &= 0, \text{ if } A \neq A_j \text{ for some } j \end{aligned}$$

Claim 13.3.1. *y is dual feasible.*

Clearly, $y \geq 0$ since $w(e_j) \leq w(e_{j-1})$ for all $j \leq n$. Consider any i.

$$\begin{aligned}
\sum_{A: e_i \in A} y(A) &= \sum_{j \geq i} y(A_i) \\
&= \left(\sum_{j=i}^{n-1} (w(e_j) - w(e_{j+1}) \right) + y(A_n) \\
&= \left(\sum_{j=i}^{n-1} (w(e_j) - w(e_{j+1}) \right) + w(e_n) \\
&= w(e_i).
\end{aligned}$$

Define $I = \{i \mid e_i \text{ is picked by Greedy}\}$. As we noted earlier, $i \in I \iff r(A_i) = r(A_{i-1}) + 1$.

Claim 13.3.2.

$$\sum_{i \in I} w(e_i) = \sum_{A \subseteq S} r(A) y(A)$$

Proof.

$$\begin{aligned}
\sum_{i \in I} w(e_i) &= \sum_{i \in I} w(e_i)(r(A_i) - r(A_{i-1})) \\
&= \sum_{j=1}^{n} w(e_j)(r(A_j) - r(A_{j-1})) \\
&= w(e_n) y(A_n) + \sum_{j=1}^{n-1} (w(e_j) - w(e_{j+1})) r(A_j) \\
&= \sum_{j=1}^{n} y(A_j) r(A_j) \\
&= \sum_{A \subseteq S} r(A) y(A).
\end{aligned}$$

■

Thus y is dual optimal, and the solution produced by Greedy is primal optimal. Further, we have an integral dual optimal solution whenever w is integral, and therefore the system is TDI. ■

Corollary 13.19. *The base polytope of $\mathcal{M} = (S, I)$, i.e., the convex hull of the bases of $\mathcal{M}$ is determined by*

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

13.3.1 Spanning Set Polytope

Another polytope associated with a matroid is the spanning set polytope, which is the convex hull of the incidence vectors of all spanning sets.

Theorem 13.20. *The spanning set polytope of a matroid $\mathcal{M} = (S, I)$ with rank function $r_{\mathcal{M}}$ is determined by*

$$\begin{aligned} 0 \leq x(e) \leq 1, \quad & \forall e \in S \\ x(U) \geq r_{\mathcal{M}}(S) - r_{\mathcal{M}}(S \backslash U), \quad & \forall U \subseteq S. \end{aligned}$$

Proof. A given set $A \subseteq S$ is spanning in $\mathcal{M}$ iff $S \backslash A$ is independent in $\mathcal{M}^*$. Thus $x \in \mathcal{P}_{\text{spanning}}(\mathcal{M})$ iff $1 - x \in \mathcal{P}_{\text{independence}}(\mathcal{M}^*)$. Now, by the relation between the ranks of dual matroids,

$$r_{\mathcal{M}^*}(U) = |U| + r_{\mathcal{M}}(S \backslash U) - r_{\mathcal{M}}(S).$$

Thus $1 - x \in \mathcal{P}_{\text{independence}}(\mathcal{M}^*)$ iff

$$\begin{aligned} 1 - x &\geq 0, \\ |U| - x(U) &\leq r_{\mathcal{M}^*}(U) = |U| + r_{\mathcal{M}}(S \backslash U) - r_{\mathcal{M}}(S), \end{aligned}$$

which matches the statement of the theorem. ∎

13.3.2 Separation Oracle

We have now determined that the independence polytope of a matroid is given by the linear conditions $x \geq 0$ and $x(U) \leq r_{\mathcal{M}}(U), U \subseteq S$. The greedy algorithm allows us to optimize over the polytope and by the equivalence between optimization and separation, there is a polynomial time separation oracle for the polytope. It is instructive to consider it explicitly.

For the separation problem, given a test vector $z : S \to \mathbb{R}$, we need to find out if $z \in \mathcal{P}_{\text{independence}}(\mathcal{M})$. We can easily test for non-negativity. To test the second condition, it is sufficient to check that $\min_{A \subseteq S}(r_{\mathcal{M}}(A) - z(A)) \geq 0$. In fact any violated inequality in the linear system can be found by constructing the set

$$U = \arg\min_{A \subseteq S}(r_{\mathcal{M}}(A) - z(A)).$$

Define $f : 2^S \rightarrow \mathbb{R}$, where $f(A) = r_{\mathcal{M}}(A) - z(A)$. f is a submodular set function since $r(\cdot)$ is the submodular rank function and $-z(\cdot)$ is modular. Thus if we can minimize an arbitrary submodular function specified by a value oracle, we can use the same for separating over a matroid polytope. However, there is a more efficient algorithm for separating over the independence polytopes given by Cunningham. See [57] for details.

13.3.3 Primal proof for Matroid Polytope

For a matroid $M = (S, \mathcal{I})$ the following system of inequalities determines the convex hull of the independent sets of M (i.e., sets in $\mathcal{I}$):

$$\begin{aligned} x(U) &\leq r_M(U) && U \subseteq S \\ x(e) &\geq 0 && e \in S \end{aligned}$$

where $r_M(\cdot)$ is the rank function of M. The first proof was based on a dual fitting technique via the Greedy algorithm for a maximum weight independent set problem.

We give a different primal proof that is built on uncrossing. This is based on [40].

Theorem 13.21. *Let x be an extreme point of the polytope*

$$(*) \begin{cases} x(U) \leq r_M(U) & U \subseteq S \\ x(e) \geq 0 & e \in S \end{cases}$$

Then there is some $e \in S$ such that $x(e) \in \{0, 1\}$.

The following corollary follows by induction from Theorem 13.21. We leave a formal proof as an exercise.

Corollary 13.22. *The system of inequalities $(*)$ determine the independent set polytope of $M = (S, \mathcal{I})$.*

Now we turn our attention to the proof of Theorem 13.21.

Proof. Proof of Theorem 13.21 Let x be an extreme solution for the polytope $(*)$. Suppose that M has a loop e. Since $r_M(\{e\}) = 0$, it follows that $x(e) = 0$ and we are done. Therefore we may assume that M does not have any loops and thus the polytope $(*)$ is full dimensional[2]. Now suppose that $x(e) \in (0, 1)$ for all

[2]A polytope is full dimensional if it has an interior point, i.e., a point x that does not satisfy any of the constraints with equality. Consider x such that, for all e, $x(e) = \epsilon$ for some $0 < \epsilon < 1/|S|$. Clearly, $x(e) > 0$ for any e. For any set U, we have $x(U) = \epsilon|U| < 1$. If M does not have any loops, $r_M(U) \geq 1$ for all sets U. Thus M is full-dimensional if there are no loops.

elements $e \in S$. Let n denote the number of elements in S. Let

$$\mathcal{F} = \{U \mid U \subseteq S, x(U) = r_M(U)\}$$

Differently said, $\mathcal{F}$ is the set of all sets whose constraints are tight at x (i.e., sets whose constraints are satisfied with equality by the solution x).

Before proceeding with the proof, we note that the submodularity for the rank function $r_M(\cdot)$ implies that $\mathcal{F}$ has the following "uncrossing" property.

Lemma 13.3. *If $A, B \in \mathcal{F}$ then $A \cap B$ and $A \cup B$ are in $\mathcal{F}$.*

Proof. Let A and B be two sets in $\mathcal{F}$; thus $x(A) = r_M(A)$ and $x(B) = r_M(B)$. It follows from the submodularity of the rank function that

$$x(A) + x(B) = r_M(A) + r_M(B) \geq r_M(A \cap B) + r_M(A \cup B)$$

Additionally,

$$x(A) + x(B) = x(A \cap B) + x(A \cup B)$$

Therefore $x(A \cap B) + x(A \cup B) \geq r_M(A \cap B) + r_M(A \cup B)$. Since $x(A \cap B) \leq r_M(A \cap B)$ and $x(A \cup B) \leq r_M(A \cup B)$, it follows that $x(A \cap B) = r_M(A \cap B)$ and $x(A \cup B) = r_M(A \cup B)$. Thus $A \cap B$ and $A \cup B$ are also in $\mathcal{F}$. ■

Let $\chi(U)$ denote the characteristic vector of U. Since x is a vertex solution, $(*)$ is full dimensional, and $x(e) \neq 0$ for all e, there is a collection $\{U_1, U_2, \ldots, U_n\}$ of n sets such that x satisfies the constraint corresponding to each U_i with equality (i.e., $x(U_i) = r_M(U_i)$ for $1 \leq i \leq n$) and the vectors $\chi(U_1), \ldots, \chi(U_n)$ are linearly independent. Therefore the set $\{\chi(U) \mid U \in \mathcal{F}\}$ has n linearly independent vectors.

For a set $\mathcal{A} \subseteq 2^S$, let $span(\mathcal{A})$ denote $span(\{\chi(U) \mid U \in \mathcal{A}\})$, where $\chi(U)$ is the characteristic vector of U.

Lemma 13.4. *There exists a laminar family $\mathcal{C} \subseteq \mathcal{F}$ such that $span(\mathcal{C}) = span(\mathcal{F})$. Moreover, $\mathcal{C}$ is a **chain**, i.e., for any two sets $A, B \in \mathcal{C}$, either $A \subseteq B$ or $B \subseteq A$.*

Assuming Lemma 13.4, we can complete the proof of Theorem 13.21 as follows. Let $\mathcal{C}$ be the chain guaranteed by Lemma 13.4. Since $span(\mathcal{C}) = span(\mathcal{F})$, there exists a chain $\mathcal{C}' \subseteq \mathcal{C}$ such that $|\mathcal{C}'| = n$ and x is the unique solution to the system

$$x(U) = r_M(U) \qquad U \in \mathcal{C}'$$

Let $\mathcal{C}' = \{A_1, A_2, \ldots, A_n\}$; wlog, $A_1 \subset A_2 \subset \cdots \subset A_n$. Let $A_0 = \emptyset$. Suppose that there exists an i such that $|A_i \setminus A_{i-1}| = 1$, and let $e \in A_i \setminus A_{i-1}$. Now we claim that we must have $x(e) \in \{0, 1\}$. To see why this is true, note that we have

$$x(e) = x(A_i) - x(A_{i-1}) = r_M(A_i) - r_M(A_{i-1})$$

Since $r_M(A_i) - r_M(A_{i-1})$ is an integer and $r_M(A_{i-1}) \leq r_M(A_i) \leq r_M(A_{i-1}) + 1$, it follows that $r_M(A_i) - r_M(A_{i-1}) \in \{0, 1\}$. But this contradicts the fact that $x(e) \in (0, 1)$. Therefore we may assume that $|A_i \setminus A_{i-1}| \geq 2$. But then $|S| \geq 2n$, which is a contradiction.

Finally, we turn our attention to the proof of Lemma 13.4.

Proof of Lemma 13.4. Let C be a chain in $\mathcal{F}$ that is maximal with respect to inclusion (i.e., C is not a proper subset of any chain in $\mathcal{F}$). We claim that $span(C) = span(\mathcal{F})$. Suppose not and let $A \in \mathcal{F}$ be such that $\chi(A) \in span(\mathcal{F}) \setminus span(C)$. If there are several such sets A, we choose one that minimizes the number of sets in C that it *properly* intersects[3].

Now suppose that A does not *properly* intersect any set in C. Clearly, $C + A$ is not a chain, since this contradicts the maximality of C. Therefore there exist $B, B' \in C$ such that B is the minimal set in C that contains A and B' is the maximal set in C that is contained in B. By Lemma 13.3, $A \cup B'$ is in $\mathcal{F}$. If $A \cup B'$ is a proper subset of B, $C + (A \cup B')$ is a chain, which contradicts the maximality of C. Therefore we must have $A \cup B' = B$. Since A and B' are disjoint, we have $\chi(A) + \chi(B') = \chi(B)$ and thus $\chi(A)$ is in the span of $\chi(B)$ and $\chi(B')$, which contradicts the fact that $\chi(A) \notin span(C)$.

Therefore we may assume that A properly intersects a set B in C. By Lemma 13.3, $A \cup B$ and $A \cap B$ are in $\mathcal{F}$.

Proposition 13.3.1. *Each of $A \cup B$, $A \cap B$ properly intersects fewer sets in C than A.*

Assuming Proposition 13.3.1, we can complete the proof as follows. It follows from our choice of A that $A \cup B$ and $A \cap B$ are both in $span(C)$. Since $\chi(A) + \chi(B) = \chi(A \cup B) + \chi(A \cap B)$, it follows that $\chi(A)$ is in $span(C)$ as well, which is a contradiction. Therefore it suffices to prove Proposition 13.3.1.

Proof of Proposition 13.3.1. Since each of $A \cup B$, $A \cap B$ does not properly intersect B, it suffices to show that if a set $B' \in C$ properly intersects $A \cup B$ (or $A \cap B$) then it properly intersects A as well.

Let $B' \in C$ be a set that properly intersects $A \cup B$. Since B and B' are both in C, it follows that one of B, B' is a subset of the other. If B' is a subset of B, B' is contained in $A \cup B$ (and thus does not properly intersect $A \cup B$). Therefore B must be a proper subset of B'. Clearly, B' intersects A (since $A \cap B$ is nonempty). If B' does not properly intersect A, it follows that one of A, B' is a subset of the other. If $A \subseteq B'$, it follows that $A \cup B \subseteq B'$, which is a contradiction. Therefore we must have $B \subset B' \subseteq A$, which is a contradiction as well. Thus B' properly intersects A.

[3]Two sets X and Y properly intersect if $X \cap Y, X - Y, Y - X$ are all non-empty.

Let $B' \in C$ be a set that properly intersects $A \cap B$. Clearly, B' intersects A and thus it suffices to show that $B' \setminus A$ is nonempty. As before, one of B, B' is a subset of the other. Clearly, B' must be a subset of B (since otherwise $A \cap B \subseteq B \subseteq B'$). Now suppose that $B' \subseteq A$. Since B' is a subset of B, it follows that $B' \subseteq A \cap B$, which is a contradiction. Therefore $B' \setminus A$ is non-empty, as desired. ■

13.4 Facets and Edges of Matroid Polytopes

Recall that the following system of inequalities determines the matroid polytope.

$$(*) \begin{cases} x(U) \leq r_M(U) & U \subseteq S \\ x(e) \geq 0 & e \in S \end{cases}$$

Throughout this section, we assume that the matroid has no loops and thus the polytope is full dimensional.

It is useful to know which inequalities in the above system are redundant. As we will see shortly, for certain matroids, the removal of redundant inequalities gives us a system with only polynomially many constraints.

Recall that a *flat* is a subset $U \subseteq S$ such that $U = span(U)$. Consider a set U that is *not* a flat. Since $r_M(U) = r_M(span(U))$ and $U \subset span(U)$, any solution x that satisfies the constraint

$$x(span(U)) \leq r_M(span(U))$$

also satisfies the inequality

$$x(U) \leq r_M(U)$$

Therefore we can replace the system $(*)$ by

$$(**) \begin{cases} x(F) \leq r_M(F) & F \subseteq S,\ F \text{ is a flat} \\ x(e) \geq 0 & e \in S \end{cases}$$

Definition 13.23. *A flat F is separable if there exist flats F_1, F_2 such that F_1 and F_2 partition F and*

$$r_M(F_1) + r_M(F_2) = r_M(F)$$

If F is a separable flat, any solution x that satisfies the constraints

$$x(F_1) \leq r_M(F_1)$$

$$x(F_2) \leq r_M(F_2)$$

also satisfies the constraint

$$x(F) \leq r_M(F)$$

since $x(F) = x(F_1) + x(F_2)$ and $r_M(F) = r_M(F_1) + r_M(F_2)$. Therefore we can remove the constraint $x(F) \leq r_M(F)$ from $(**)$. Perhaps surprisingly, the resulting system does not have any redundant constraints. The interested reader can consult Chapter 40 in [57] for a proof.

Theorem 13.24. *The system of inequalities*

$$\begin{aligned} x(F) &\leq r_M(F) \qquad F \subseteq S,\ F \text{ is an inseparable flat} \\ x(e) &\geq 0 \qquad\qquad e \in S \end{aligned}$$

is a minimal system for the independent set polytope of a loopless matroid M.

As an example, consider the uniform matroid. The independent set polytope for the uniform matroid is determined by the following constraints:

$$\sum_{e \in S} x(e) \leq k$$

$$x(e) \geq 0 \qquad e \in S$$

Similarly, the independent set polytope for the partition matroid induced by the partition $S_1, \ldots, S_h$ of S and integers $k_1, \ldots, k_h$ is determined by the following constraints:

$$\sum_{e \in S_i} x(e) \leq k_i \qquad 1 \leq i \leq k$$

$$x(e) \geq 0 \qquad e \in S$$

Finally, consider the graphic matroid induced by a graph $G = (V, E)$. The base polytope of a graphic matroid corresponds to the the spanning tree polytope, which is determined by the following constraints:

$$\begin{aligned} x(E[U]) &\leq |U| - 1 \qquad U \subseteq V \\ x(E) &= |V| - 1 \\ x(e) &\geq 0 \qquad\qquad e \in E \end{aligned}$$

where $E[U]$ is the set of edges inside the vertex set $U \subseteq V$.

Definition 13.25. *Two vertices x, x' of a polyhedron P are* adjacent *if they are contained in a face F of P of dimension one, i.e., a line.*

Theorem 13.26. *Let $M = (S, \mathcal{I})$ be a loopless matroid. Let $I, J \in \mathcal{I}$, $I \neq J$. Then $\chi(I)$ and $\chi(J)$ are adjacent vertices of the independent set polytope of M if and only if $|I \triangle J| = 1$ or $|I \setminus J| = |J \setminus I| = 1$ and $r_M(I) = r_M(J) = |I| = |J|$.*

The interested reader can consult [57] for a proof.

13.5 Further Base Exchange Properties

We saw earlier the following base exchange lemma.

Lemma 13.5. *Let B and B' be two bases of a matroid M, and let y be an element of $B' \setminus B$. Then*

1. *there exists $x \in B \setminus B'$ such that $B' - y + x$ is a base*
2. *there exists $x \in B \setminus B'$ such that $B + y - x$ is a base*

We will prove a stronger base exchange theorem below and derive some corollaries that will be useful in matroid intersection and union.

Theorem 13.27 (Strong Base Exchange Theorem)**.** *Let B, B' be two bases of a matroid M. Then for any $x \in B \setminus B'$ there exists an $y \in B' \setminus B$ such that $B - x + y$ and $B' - y + x$ are both bases.*

Proof. Let x be any element in $B \setminus B'$. Since B' is a base, $B' + x$ has a unique circuit C. Moreover, for any $y \in C - x$ we have $B' - y + x$ is a base. However, we want to find a y such that $B - x + y$ is also a base and for this we need to choose a $y \in C - x$ carefully — not all $y \in C - x$ are suitable (consider the case of two spanning trees in a graph as an example).

Let $A = (B \cup C) - x$. Since $x \in span(C - x)$ it follows that $span(A) = span(B) = S$ and hence A contains a base. Let B'' be a base from A that contains $B - x$. We have $B'' = B - x + y$, for some $y \in C - x$. This is the desired y. ■

In fact, Theorem 13.27 holds when B, B' are independent sets of the same size instead of bases.

Corollary 13.28. *Let I, J be two independent sets of a matroid $M = (S, \mathcal{I})$ such that $|I| = |J|$. Then for any $x \in I \setminus J$ there exists an $y \in J \setminus I$ such that $I - x + y$ and $J - y + x$ are both independent sets.*

Proof. Let $k = |I| = |J|$. Let $M' = (S, \mathcal{I}')$, where

$$\mathcal{I}' = \{I \mid I \in \mathcal{I} \text{ and } |I| \leq k\}$$

It is straightforward to verify that M' is a matroid as well. Additionally, since every independent set in M' has size at most k, I and J are bases in M'. It follows from Theorem 13.27 that for any $x \in I \setminus J$ there exists an $y \in J \setminus I$ such that $I - x + y$ and $J - y + x$ are both bases in M', and thus independent sets in M. ■

Let $M = (S, \mathcal{I})$ be a matroid, and let $I \in \mathcal{I}$. We define a directed bipartite graph $D_M(I)$ as follows. The graph $D_M(I)$ has vertex set S; more precisely, its bipartition is $(I, S \setminus I)$. There is an edge from $y \in I$ to $z \in S \setminus I$ iff $I - y + z$ is an independent set.

Lemma 13.6. *Let $M = (S, \mathcal{I})$ be a matroid, and let I, J be two independent sets in M such that $|I| = |J|$. Then $D_M(I)$ has a perfect matching on $I \triangle J$[4].*

Proof. We will prove the lemma using induction on $|I \triangle J|$. If $|I \triangle J| = 0$, the lemma is trivially true. Therefore we may assume that $|I \triangle J| \geq 1$. It follows from Corollary 13.28 that there exists an $y \in I$ and $z \in J$ such that $I' = I - y + z$ and $J' = J + y - z$ are independent sets. Note that $|I' \triangle J'| < |I \triangle J|$ and $|I'| = |J'|$. It follows by induction that $D_M(I)$ has a perfect matching N on $I' \triangle J'$. Then $N \cup \{(y, z)\}$ is a perfect matching on $I \triangle J$. ∎

Lemma 13.7. *Let $M = (S, \mathcal{I})$ be a matroid. Let I be an independent set in M, and let J be a subset of S such that $|I| = |J|$. If $D_M(I)$ has a* unique *perfect matching on $I \triangle J$ then J is an independent set.*

Before proving the lemma, we note the following useful property of unique perfect matchings.

Proposition 13.5.1. *Let $G = (X, Y, E)$ be a bipartite graph such that G has a* unique *perfect matching N. Then we can label the vertices of X as $x_1, \ldots, x_t$, and we can label the vertices of Y as $y_1, \ldots, y_t$ such that*

$$N = \{(x_1, y_1), \ldots, (x_t, y_t)\}$$

and $(x_i, y_j) \notin E$ for all i, j such that $i < j$.

Proof. We start by noting that there is an edge $xy \in N$ such that one of x, y has degree one. We construct a trail[5] by alternately taking an edge in N and an edge not in N, until either we cannot extend the trail or we reach a previously visited vertex. Now suppose that the trail has a cycle C. Since G is bipartite, C has even length. Thus we can construct a perfect matching from N by removing the edges of C that are in N and adding the edges of C that are not in N, which contradicts the fact that G has a unique perfect matching. Therefore we may assume that the trail is a path. If the last edge of the trail is not in N, we can extend the trail by taking the edge of N incident to the last vertex. Therefore the last edge must be in N. Then the last vertex on the trail has degree one, since

[4] A perfect matching on a set U is a matching such that S is the set of endpoints of the edges in the matching.

[5] A trail is a walk in which all edges are distinct.

otherwise we could extend the trail using one of the edges incident to it that are not in N. It follows that the last edge of the trail is the desired edge.

Now let xy be an edge in N such that one of its endpoints has degree one in G. Suppose that x has degree one. We let $x_1 = x$, $y_1 = y$, and we remove x and y to get a graph G'. Since $N - xy$ is the unique perfect matching in G', it follows by induction that we can label the vertices of G' such that

$$N - xy = \{(x_2, y_2), \ldots, (x_t, y_t)\}$$

such that (x_i, y_j) is not an edge in G', for all $2 \leq i < j \leq t$. Since x_1 has degree one in G, we are done. Therefore we may assume that y has degree one. We let $x_t = x$, $y_t = y$, and we remove x and y to get a graph G'. As before, it follows by induction that we can label the vertices of G' such that

$$N - xy = \{(x_1, y_1), \ldots, (x_{t-1}, y_{t-1})\}$$

such that (x_i, y_j) is not an edge in G', for all $1 \leq i < j \leq t-1$. Since y_t has degree one in G, we are done. ■

Proof of Lemma 13.7. Let G denote the (undirected) subgraph of $D_M(I)$ induced by $I \triangle J$, and let N denote the unique perfect matching in G. Since G is a bipartite graph, it follows from Proposition 13.5.1 that we can label the vertices of $I \setminus J$ as $y_1, \ldots, y_t$, and we can label the vertices of $J \setminus I$ as $z_1, \ldots, z_t$ such that

$$N = \{(y_1, z_1), \ldots, (y_t, z_t)\}$$

and $(y_i, z_j) \notin E(G)$, for all $1 \leq i < j \leq t$.

Now suppose that J is not independent, and let C be a circuit in J. Let i be the smallest index such that $z_i \in C$. Consider any element z_j in $C - z_i$. Since $j > i$, it follows that $(y_i, z_j) \notin D_M(I)$. Therefore any element z in $C - z_i$ is in $span_M(I - y_i)$, since for any $z \in C - z_i$, either z is in $I \cap J$ or $z = z_j$ for some j. Hence $C - z_i$ is a subset of $span(I - y_i)$. Since C is a circuit,

$$C \subseteq span(C - z_i) \subseteq span(I - y_i)$$

Thus $z_i \in span(I - y_i)$, which contradicts the fact that $I - y_i + z_i$ is independent. ■

Corollary 13.29. *Let $M = (S, \mathcal{I})$ be a matroid, and let $I \in \mathcal{I}$. Let J be a subset of S with the following properties:*

1. $|I| = |J|$
2. $r_M(I \cup J) = |I|$
3. *$D_M(I)$ has a unique perfect matching on $I \triangle J$*

Let e be any element not in $I \cup J$ such that $I + e \in \mathcal{I}$. Then $J + e \in \mathcal{I}$.

Proof. It follows from Lemma 13.7 that J is independent. Since $r_M(I \cup J) = |I|$, both I and J are maximal independent sets in $I \cup J$. Thus $I \subseteq span(J)$ and $J \subseteq span(I)$. Since $I + e$ is independent, $e \notin span(I)$. As we have seen in Lecture 14, since $J \subseteq span(I)$, it follows that $span(J) \subseteq span(I)$. Therefore $e \notin span(J)$ and thus $J + e$ is independent. ■

Chapter 14

Matroid Intersection[1]

One of several major contributions of Edmonds to combinatorial optimization is algorithms and polyhedral theorems for matroid intersection, and more generally polymatroid intersection.

From an optimization point of view, the matroid intersection problem is the following: Let $M_1 = (S, \mathcal{I}_1)$ and $M_2 = (S, \mathcal{I}_2)$ be two matroids on the same ground set S. Then $\mathcal{I}_1 \cap \mathcal{I}_2$ is the collection of all sets that are independent in both matroids.

One can ask the following algorithmic questions:

1. Is there a common base in the two matroids? That is, is there $\mathcal{I} \in \mathcal{B}_1 \cap \mathcal{B}_2$ where $\mathcal{B}_1$ and $\mathcal{B}_2$ are the bases of M_1 and M_2.
2. Output a maximum cardinality set in $\mathcal{I}_1 \cap \mathcal{I}_2$.
3. Given $w : S \to \Re$, output a maximum weight set in $\mathcal{I}_1 \cap \mathcal{I}_2$. Or output a maximum weight common base, if it exists.

Remark 14.1. It is easy to see that the intersection of two matroids, i.e., $(S, \mathcal{I}_1 \cap \mathcal{I}_2)$, is not necessarily a matroid.

Exercise 14.1. If $M_1 = (S, \mathcal{I}_1)$ is a matroid and $M_2 = (S, \mathcal{I}_2)$ is the uniform matroid, then $M_3 = (S, \mathcal{I}_1 \cap \mathcal{I}_2)$ is a matroid.

As one can imagine, matroid intersection can capture several additional optimization problems beyond matroids. We give some canonical and illustrative examples.

[1]Based on notes scribed by Jason Sauppe in 2010.

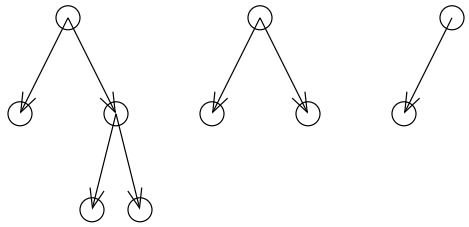


Figure 14.1: Example of a branching

Example 14.1. Bipartite Matching Let $G = (V, E)$ be a bipartite graph with bipartition $A \cup B$. Let $M_1 = (E, \mathcal{I}_1)$ and $M_2 = (E, \mathcal{I}_2)$ be two partition matroids on E, where

$$\mathcal{I}_1 = \{E' \subseteq E \mid |\delta(v) \cap E'| \leq 1,\ v \in A\}$$
$$\mathcal{I}_2 = \{E' \subseteq E \mid |\delta(v) \cap E'| \leq 1,\ v \in B\}.$$

Then it is easy to see that $I \in \mathcal{I}_1 \cap \mathcal{I}_2$ if and only if I induces a matching in G. Thus bipartite matching problems are special cases of matroid intersection problems.

Example 14.2. Branchings and Arborescences Let $D = (V, A)$ be a directed graph. A *branching* in D is a set of edges $A' \subseteq A$ such that the in-degree of each node is at most one and the edges in A form a forest. (An example is shown in Figure 14.1.) An *arborescence* rooted at a node $r \in V$ is a directed out-tree such that r has a path to each node $v \in V$. Thus an arborescence is a branching in which r is the only node with in-degree 0.

Consider two matroids $M_1 = (A, \mathcal{I}_1)$ and $M_2 = (A, \mathcal{I}_2)$ where $M_1 = (A, \mathcal{I}_1)$ is a partition matroid:

$$\mathcal{I}_1 = \{A' \subseteq A \mid |\delta^-(v) \cap A'| \leq 1,\ v \in V\}$$

and M_2 is a graphic matroid on $G = (V, A^u)$ obtained by making an undirected graph on V by removing directions from arcs in A with:

$$\mathcal{I}_2 = \{A' \subseteq A \mid A' \text{ induces a forest in } G^u\}$$

It is easy to see that $\mathcal{I}_1 \cap \mathcal{I}_2$ is the set of all branchings, and a common basis corresponds to arborescences.

Example 14.3. Colorful Spanning Trees Let $G = (V, E)$ where edges in E are colored with k colors. That is, $E = E_1 \uplus E_2 \uplus \ldots \uplus E_k$. Suppose we are given integers $h_1, h_2, \ldots, h_k$ and wish to find a spanning tree that has at most h_i edges of color i (i.e., from E_i). Observe that this can be phrased as a matroid intersection problem: it is the combination of a spanning tree matroid and a partition matroid.

14.1 Min-max Theorem for Maximum Cardinality Independent Set

We now state a min-max theorem for the size of the maximum cardinality set in the intersection of two matroids.

Theorem 14.1. *Let $M_1 = (S, \mathcal{I}_1)$ and $M_2 = (S, \mathcal{I}_2)$ be two matroids with rank functions r_1 and r_2. Then the size of the maximum cardinality set in $\mathcal{I}_1 \cap \mathcal{I}_2$ is given by:*

$$\min_{U \subseteq S} r_1(U) + r_2(S \setminus U)$$

Proof. Let $I \in \mathcal{I}_1 \cap \mathcal{I}_2$. Take any set $U \subseteq S$. Then

$$I = |I \cap U| + |I \setminus U| \leq r_1(U) + r_2(S \setminus U)$$

since $I \cap U \in \mathcal{I}_1$ and $I \setminus U \in \mathcal{I}_2$. ■

We prove the difficult direction algorithmically. That is, we describe an algorithm for the maximum cardinality set in $\mathcal{I}_1 \cap \mathcal{I}_2$ that, as a byproduct, proves the other direction.

The algorithm is an "augmenting" path type algorithm inspired by bipartite matching and matroid base exchange properties that we discussed earlier. Given $I \in \mathcal{I}_1 \cap \mathcal{I}_2$, the algorithm outputs a $J \in \mathcal{I}_1 \cap \mathcal{I}_2$ such that $|J| = |I| + 1$, or certifies correctly that I is a maximum cardinality set in $\mathcal{I}_1 \cap \mathcal{I}_2$ by exhibiting a set $U \subseteq S$ such that $|I| = r_1(U) + r_2(S \setminus U)$.

Recall that for a matroid $M = (S, \mathcal{I})$ and $I \in \mathcal{I}$, we defined a directed graph $D_M(I) = (S, A(I))$ where

$$A(I) = \{(y, z) \mid y \in I,\ z \in S \setminus I,\ I - y + z \in \mathcal{I}\}$$

as a graph that captures exchanges for I.

Now we have two matroids M_1 and M_2 and $I \in \mathcal{I}_1 \cap \mathcal{I}_2$ and we wish to augment I to another set $J \in \mathcal{I}_1 \cap \mathcal{I}_2$ if possible. For this purpose we define a graph $D_{M_1,M_2}(I) = (S, A(I))$ where

$$\begin{aligned} A(I) = &\{(y, z) \mid y \in I,\ z \in S \setminus I,\ I - y + z \in \mathcal{I}_1\} \\ &\cup \{(z', y') \mid z' \in S \setminus I,\ y' \in I,\ I - y' + z' \in \mathcal{I}_2\} \end{aligned}$$

In other words, $D_{M_1,M_2}(I)$ is the union of $D_{M_1}(I)$ and the reverse of $D_{M_2}(I)$. In this sense there is asymmetry in M_1 and M_2. (An example is shown in Figure 14.2.)

$$\begin{aligned} (y, z) \in A(I) &\Rightarrow I - y + z \in \mathcal{I}_1 \\ (z', y') \in A(I) &\Rightarrow I - y' + z' \in \mathcal{I}_2 \end{aligned}$$

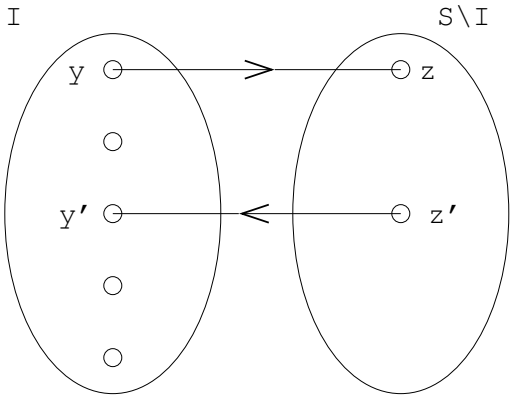


Figure 14.2: Exchange Graph $D_{M_1,M_2}(I)$

Let $X_1 = \{z \in S \setminus I \mid I + z \in \mathcal{I}_1\}$ and $X_2 = \{z \in S \setminus I \mid I + z \in \mathcal{I}_2\}$, and let P be a *shortest* path from X_1 to X_2 in $D_{M_1,M_2}(I)$. Note that the shortest path could consist of a single $z \in X_1 \cap X_2$. There may not be any path P between X_1 and X_2.

Lemma 14.1. *If there is no $X_1 - X_2$ path in $D_{M_1,M_2}(I)$, then I is a maximum cardinality set in $\mathcal{I}_1 \cap \mathcal{I}_2$.*

Proof. Note that if X_1 or X_2 are empty then I is a base in one of M_1 or M_2 and hence a max cardinality set in $\mathcal{I}_1 \cap \mathcal{I}_2$. So assume $X_1 \neq \emptyset$ and $X_2 \neq \emptyset$. Let U be the set of nodes that can reach X_2 in $D_{M_1,M_2}(I)$. No $X_1 - X_2$ path implies that $X_1 \cap U = \emptyset$, $X_2 \subseteq U$, and $\delta^-(U) = \emptyset$ (i.e., no arcs enter U). Then we have the following:

Claim 14.1.1. $r_1(U) \leq |I \cap U|$

Proof. If $r_1(U) > |I \cap U|$, then $\exists z \in U \setminus (I \cap U)$ such that $(I \cap U) + z \in \mathcal{I}_1$ with $I + z \notin \mathcal{I}_1$. If $I + z \in \mathcal{I}_1$, then $z \in X_1$ and $X_1 \cap U \neq \emptyset$, contradicting the fact that there is no $X_1 - X_2$ path. Since $(I \cap U) + z \in \mathcal{I}_1$ but $I + z \notin \mathcal{I}_1$, there must exist a $y \in I \setminus U$ such that $I - y + z \in \mathcal{I}_1$. But then $(y, z) \in A(I)$, contradicting the fact that $\delta^-(U) = \emptyset$ (shown in Figure 14.3).

■

Claim 14.1.2. $r_2(S \setminus U) \leq |I \setminus U|$ *(The proof is similar to the previous proof.)*

Thus $|I| = |I \cap U| + |I \setminus U| \geq r_1(U) + r_2(S \setminus U)$, which establishes that $|I| = r_1(U) + r_2(S \setminus U)$. Therefore, I is a max cardinality set in $\mathcal{I}_1 \cap \mathcal{I}_2$. ■

Lemma 14.2. *If P is a shortest $X_1 - X_2$ path in $D_{M_1,M_2}(I)$, then $I' = I \Delta V(P)$ is in $\mathcal{I}_1 \cap \mathcal{I}_2$.*

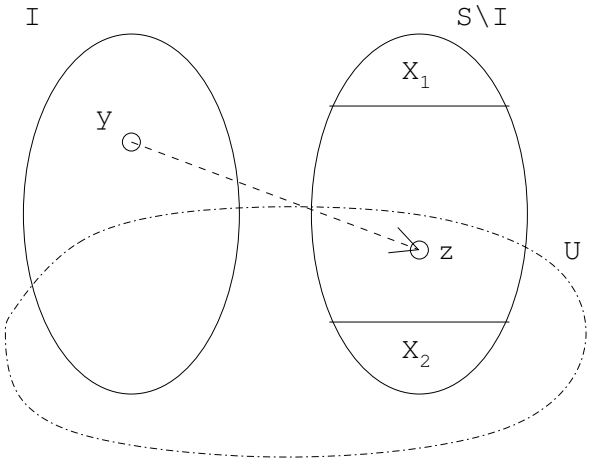


Figure 14.3: Exchange Graph with a (y, z) arc entering U

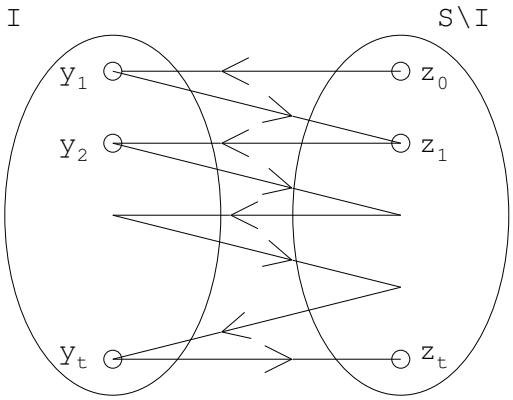


Figure 14.4: A path P in $D_{M_1,M_2}(I)$

Proof. Recall the following lemma from the previous lecture which we will use here:

Lemma 14.3. *Let $M = (S, \mathcal{I})$ be a matroid. Let $I \in \mathcal{I}$ and $J \subseteq S$ such that $|I| = |J|$. If there is a* unique *perfect matching on $I \Delta J$ in $A(I)$, then $J \in \mathcal{I}$.*

Let $P = z_0, y_1, z_1, \ldots, y_t, z_t$ (shown in Figure 14.4) be a shortest path from X_1 to X_2. Let $J = \{z_1, \ldots, z_t\} \cup (I \setminus \{y_1, \ldots, y_t\})$. Then $J \subseteq S$, $|J| = |I|$, and the arcs from $\{y_1, \ldots, y_t\}$ to $\{z_1, \ldots, z_t\}$ form a unique perfect matching from $I \setminus J$ to $J \setminus I$ (otherwise P has a short cut and is not a shortest path). Then by Lemma 14.3, $J \in \mathcal{I}_1$.

Also, $z_i \notin X_1$ for $i \geq 1$, otherwise P would not be the shortest possible

$X_1 - X_2$ path. This implies that $z_i + I \notin \mathcal{I}_1$, which implies that $r_1(I \cup J) = r_1(I) = r_1(J) = |I| = |J|$. Then since $I + z_0 \in \mathcal{I}_1$, it follows that $J + z_0 \in \mathcal{I}_1$ (i.e., $I' = (I \setminus \{y_1, \ldots, y_t\}) \cup \{z_0, z_1, \ldots, z_t\} \in \mathcal{I}_1$).

By symmetry, $I' \in \mathcal{I}_2$. This implies that $I' \in \mathcal{I}_1 \cap \mathcal{I}_2$. ■

Theorem 14.2. *There is a polynomial time algorithm to find a maximum cardinality set in the intersection of two matroids.*

Algorithm 5 computes a maximum cardinality independent set in the intersection of two matroids M_1 and M_2 in polynomial time.

Algorithm 5 Algorithm for Maximum Cardinality Independent Set in Intersection of Two Matroids

```
1:  procedure maxIndepSet(M_1 = (S, I_1), M_2 = (S, I_2))
2:      I ← ∅
3:      repeat
4:          Construct D_{M_1,M_2}(I)
5:          X_1 ← {z ∈ S \ I | I + z ∈ I_1}
6:          X_2 ← {z ∈ S \ I | I + z ∈ I_2}
7:          Let P be a shortest X_1 − X_2 path in D_{M_1,M_2}(I)
8:          if P is not empty then
9:              I ← IΔV(P)              ▷ I' = (I \ {y_1, ..., y_t}) ∪ {z_0, z_1, ..., z_t}
10:         end if                      ▷ Else P is empty and I is maximal
11:     until I is maximal
12: end procedure
```

14.2 Weighted Matroid Intersection

We saw an algorithm for finding a maximum cardinality set in the intersection of two matroids. The algorithm generalized in a straightforward fashion to the weighted case. The correctness is more complicated and we will not discuss it here. See [57].

The algorithm for the weighted case is also an augmenting path algorithm. Recall the cardinality algorithm 5: The weighted case differs only in finding P. Let $w : S \to \mathfrak{R}^+$ be the weights. Then in computing P we assign weights to each vertex $x \in D_{M_1,M_2}(I)$ as $w(x)$ if $x \in I$ and $-w(x)$ to $x \notin I$. The desired path P should now be a minimum length path according to the weights; further, P should have the smallest number of arcs among all minimum length paths.

Theorem 14.3. *There is a strongly polynomial time combinatorial algorithm for weighted matroid intersection.*

14.3 Matroid Intersection Polytope

Edmonds proved the following theorem about the matroid intersection polytope:

Theorem 14.4. *Let $M_1 = (S, \mathcal{I}_1)$ and $M_2 = (S, \mathcal{I}_2)$ be two matroids on S. Then the convex hull of the characteristic vectors of sets in $\mathcal{I}_1 \cap \mathcal{I}_2$ is determined by the following set of inequalities:*

$$\begin{aligned} x &\geq 0 \\ x(U) &\leq r_1(U) \qquad \forall\, U \subseteq S \\ x(U) &\leq r_2(U) \qquad \forall\, U \subseteq S \end{aligned}$$

where r_1 and r_2 are the rank functions of M_1 and M_2, respectively. Moreover, the system of inequalities is TDI. In other words,

$$P_{\textit{common indep. set}}(M_1, M_2) = P_{\textit{indep. set}}(M_1) \cap P_{\textit{indep. set}}(M_2).$$

Proof. Consider the primal-dual pair

$$\begin{aligned} \max \sum_{e \in S} w(e)x(e) & \\ \text{subject to} \quad x(U) &\leq r_1(U) \qquad \forall\, U \subseteq S \\ x(U) &\leq r_2(U) \qquad \forall\, U \subseteq S \\ x &\geq 0 \end{aligned}$$

$$\begin{aligned} \min \sum_{U \subseteq S} \Big(r_1(U)y_1(U) + r_2(U)y_2(U) \Big) & \\ \text{subject to} \quad \sum_{\substack{U \subseteq S \\ U \ni e}} \Big(y_1(U) + y_2(U) \Big) &\geq w(e) \qquad \forall\, e \in S \\ y_1 &\geq 0 \\ y_2 &\geq 0 \end{aligned}$$

We will prove that the dual has an integral optimum solution whenever w is integral. We can assume that $w(e) \geq 0$ for each e without loss of generality.

Lemma 14.4. *There exists an optimum solution y_1^*, y_2^* to the dual such that*

$$\begin{aligned} \mathcal{F}_1 &= \{U \subseteq S \mid y_1^*(U) > 0\} \\ \mathcal{F}_2 &= \{U \subseteq S \mid y_2^*(U) > 0\} \end{aligned}$$

are chains.

Proof. Choose an optimum y_1^*, y_2^* with $\mathcal{F}_1 = \{U \subseteq S \mid y_1^*(U) > 0\}$ and $\mathcal{F}_2 = \{U \subseteq S \mid y_2^*(U) > 0\}$ such that the number of proper intersections plus the number of disjoint sets in $\mathcal{F}_1$ and $\mathcal{F}_2$ is minimal.

Then for $A, B \in \mathcal{F}_1$, if A and B properly intersect or are disjoint, we can increase $y_1^*(A \cap B)$ and $y_1^*(A \cup B)$ by ϵ and decrease $y_1^*(A)$ and $y_1^*(B)$ by ϵ to create a new dual solution. This new solution is still dual feasible since

$$\chi(A \cup B) + \chi(A \cap B) = \chi(A) + \chi(B).$$

and the dual objective value changes by

$$-\epsilon\Big(r_1(A) + r_1(B)\Big) + \epsilon\Big(r_1(A \cup B) + r_1(A \cap B)\Big).$$

By the submodularity of r_1, this is ≤ 0. If this value is < 0, then this contradicts the optimality of the original solution y_1^*, y_2^*. On the other hand, if this value equals 0, then we have a new optimum solution for the dual. Increasing ϵ by the largest amount without violating non-negativity of the y_1 values will ensure that $y_1^*(A)$ or $y_1^*(B)$ becomes 0. The new dual solution has a smaller number of proper intersections plus disjoint sets in $\mathcal{F}_1, \mathcal{F}_2$, contradicting the choice of y_1^*, y_2^*. This follows similarly for $A, B \in \mathcal{F}_2$.

Another way to do the above argument is to choose among all optimum dual solutions y_1, y_2 the one that minimizes $\sum_{U \subseteq S}(y_1(U) + y_2(U))|U||S \setminus U|$. One can show that uncrossing as above strictly reduces this value while maintaining the optimality of the solution. ■

The following very useful lemma was shown by Edmonds.

Lemma 14.5. *Let S be a set and $\mathcal{F}_1$ and $\mathcal{F}_2$ be two laminar families on S. Let $\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2$ and let A be the $S \times \mathcal{F}$ incidence matrix. Then A is the transpose of a network matrix and hence is a TUM.*

Proof sketch. Each laminar family is naturally associated with a rooted forest where each set X of the laminar family becomes a node v_X and the parent of v_X is v_Y iff Y is the inclusion-wise minimal set in the family that contains X and is not X. The roots of the forest are the maximal sets in the laminar family. Given $\mathcal{F}_1$ and $\mathcal{F}_2$ we create the rooted forests, one for each of them, and create a single directed tree T as follows. We add a new root vertex r and connect the roots of both forests to r. We direct all edges in the forest for $\mathcal{F}_1$ towards r and all edges in the forest for $\mathcal{F}_2$ away from r. This creates the directed tree T for the network matrix. We will now create the directed graph for the network matrix — note that the directed graph will have have the same vertex set as the directed tree T we created. Now consider any element $e \in S$. If e does not belong to any set in $\mathcal{F}_1$ or $\mathcal{F}_2$ then we can ignore it since its row in the matrix A is an all 0 row. First

assume that e belongs to at least one set in $\mathcal{F}_1$ and at least one set in $\mathcal{F}_2$. Let X be the minimal set in $\mathcal{F}_1$ that contains e and let Y be the minimal set in $\mathcal{F}_2$ that contains e. We add the arc (v_X, v_Y) to the directed graph. One can see that the directed path in T from v_X to v_Y goes from v_X to r in the forest corresponding to $\mathcal{F}_1$ and then from r to v_Y in the forest corresponding to the $\mathcal{F}_2$ and the alignment of the arcs implies that all corresponding entries in the network matrix will be 0 or 1 (we do not get any -1 entries). If e belongs to only a set in $\mathcal{F}_1$, say X, then we add the arc (v_X, r) to the directed graph. Similarly if it belongs only to a set Y in $\mathcal{F}_2$ then we add the arc (r, v_Y) to the directed graph. See Fig 14.5.

We leave it as an exercise to verify that the resulting network matrix is precisely the incidence matrix A that we seek. ■

Let y_1^*, y_2^* be an optimum dual solution such that $\mathcal{F}_1$ and $\mathcal{F}_2$ are chains. Let $\mathcal{F} \subseteq \mathcal{F}_1 \cup \mathcal{F}_2$. Thus there is an optimum dual solution when restricted to sets $U \subseteq \mathcal{F}$. Hence an optimum solution to the following system is an optimum solution to the original dual.

$$\min \sum_{U \in \mathcal{F}} \left(r_1(U) y_1(U) + r_2(U) y_2(U) \right)$$

$$\sum_{\substack{U \in \mathcal{F} \\ U \ni e}} \Big(y_1(U) + y_2(U) \Big) \geq w(e) \qquad \forall\, e \in S$$

$$y_1, y_2 \geq 0$$

Then by Lemma 14.5, the constraint matrix for the above system corresponds to a TUM matrix. This implies that there is an integral solution y_1, y_2 for integral w. From this we can conclude that the dual LP has an integral optimum solution whenever w is integral, and therefore the system of inequalities for the matroid intersection polytope is TDI. ■

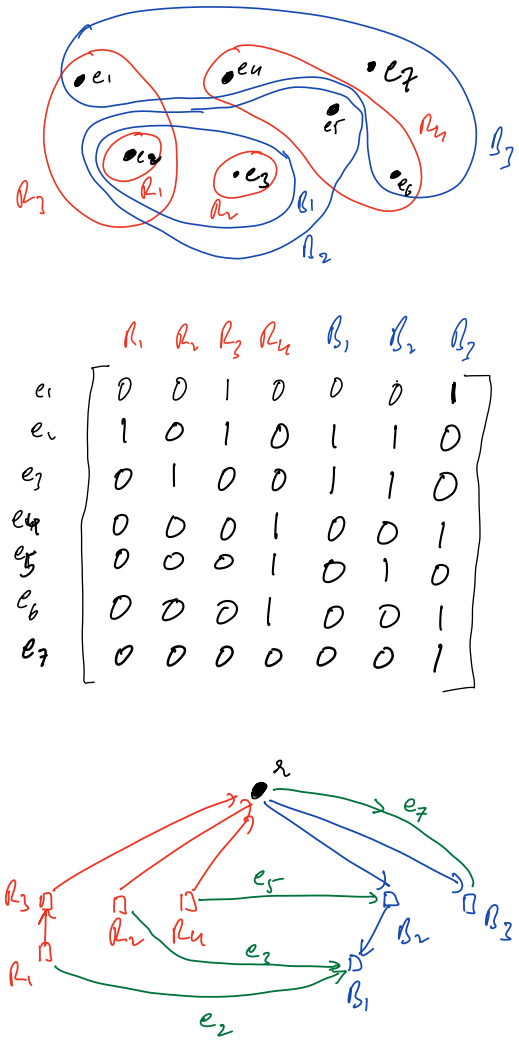


Figure 14.5: Proof of Lemma 14.5. Two laminar families over a ground set of seven elements and the corresponding incidence matrix. Third figure shows the construction to prove that the incidence matrix is a network matrix. The red and blue edges correspond to the two laminar families that form the directed tree T. The green edges, one per element of S, form the directed graph for the network matrix.

Chapter 15

Matroid Union[1]

Matroid union and matroid intersection are closely related in the sense that one can be derived from the other. However they are from different perspectives and have different applications.

15.1 Motivation

To motivate matroid union theorem we state a well known theorem of Tutte and Nash-Williams on packing disjoint spanning trees in graphs.

Theorem 15.1 (Nash-Williams and Tutte)**.** *An undirected multi-graph $G = (V, E)$ contains k edge-disjoint spanning trees iff for every partition P of V into ℓ sets, $V_1, V_2, \ldots, V_\ell$, the number of edges crossing the partition P is at least $k(\ell - 1)$.*

It is easy to see that the condition is necessary; if $T_1, \ldots, T_k$ are the edge-disjoint spanning trees then each T_i has to contain at least $\ell - 1$ edges across the partition P to connect them. A useful corollary of the above was observed by Gusfield. It is an easy exercise to derive this from the above theorem.

Corollary 15.2. *If a multi-graph $G = (V, E)$ is $2k$-edge-connected then G contains k edge-disjoint spanning trees.*

Nash-Williams proved a related theorem on covering the edge-set of a graph by forests.

Theorem 15.3 (Nash-Williams)**.** *Let $G = (V, E)$ be an undirected multi-graph. Then E can be partitioned into k forests iff for each set $U \subseteq V$,*

$$|E[U]| \leq k(|U| - 1). \tag{15.1}$$

[1]Based on notes scribed by Quan Geng in 2010.

Again, necessity is easy to see; any forest can contain at most $|U| - 1$ edges from $E[U]$. The above two theorems were first shown via graph theoretica arguments but turn out to be special cases of the matroid union theorem, and hence are properly viewed as matroidal results.

15.2 A Lemma of Nash-Williams

We start with a basic result of Nash-Williams that gives a clean proof of the matroid union theorem to follow.

Theorem 15.4 (Nash-Williams)**.** *Let $\mathcal{M}' = (S', \mathcal{I}')$ be a maroid with rank function r'. Let $f : S' \to S$ be a function mapping S' to S. Let $\mathcal{M} = (S, \mathcal{I})$, where $\mathcal{I} = \{f(I') \mid I' \in \mathcal{I}'\}$. Then $\mathcal{M}$ is a matroid with rank function r, where*

$$r(U) = \min_{T \subseteq U}(|U \setminus T| + r'(f^{-1}(T))). \tag{15.2}$$

Before we proceed with the proof we interpret the theorem. The mapping f when viewed from the side of S' partitions S': each part corresponds to $f^{-1}(u)$ for some $u \in S$. Thus f can be viewed as starting with a partition of S' and naming each part by an element of S.

Proof. We start with a simple observation. Suppose $A \in \mathcal{I}$. Then there is an $A' \in \mathcal{I}'$ such that $f(A') = A$. Suppose we choose A' minimal such that $f(A') = A$; then it follows that for all $u \in A$, $A' \cap f^{-1}(u)$ is a singleton since $\mathcal{I}'$ is downclosed. This also implies that $|A'| = |A|$.

We verify the three axioms to prove that $\mathcal{M}$ is a matroid.

1. $f(\emptyset) = \emptyset$ and hence $\emptyset \in \mathcal{I}$.

2. Say $A \in \mathcal{I}$ and $B \subseteq A$. Then

$$\begin{aligned} A \in \mathcal{I} &\Rightarrow \exists A' \in \mathcal{I}', \text{ s.t. } f(A') = A \\ &\Rightarrow \forall u \in A, f^{-1}(u) \cap A' \neq \emptyset. \end{aligned}$$

   Let $B' = \{u' \in A' \mid f(u') \in B\}$, then $B = f(B')$ and since $B' \subseteq A', B' \in \mathcal{I}'$ and hence $B \in \mathcal{I}$.

3. Say $A, B \in \mathcal{I}$ and $|B| > |A|$. Let A' be minimal s.t. $f(A') = A$. Similarly let B' be minimal s.t. $f(B') = B$. As we noted before, due to minimality, we have $|A'| = |A|$ and $|B'| = |B|$. Since $|B| > |A|$ it follows that $|B'| > |A'|$. This implies that there is $e' \in B' \setminus A'$ such that $A' + e' \in \mathcal{I}'$. Suppose $f(e') \in B \setminus A$ then $f(A' + e') = A + f(e')$ and we are done. However it may be the case that $f(e') \in A$. If this happens then there is $e'' \in A' - B'$ such that $f(e'') = f(e')$.

However, if this happens, we can consider $A'' = A + e' - e''$. Notice that $f(A'') = A$ and $A'' \in \mathcal{I}'$. We also observe that $|A'' \cap B'| > |A' \cap B'|$. Thus, to make the argument work we choose A', B' minimal such that $f(A') = A$ and $f(B') = B$ and among all such A', B' the pair that maximize $|A' \cap B'|$. It then follows, from the preceding argument, that for any $e' \in B' \setminus A'$, $f(e') \in B \setminus A$.

Therefore $\mathcal{M}$ is a matroid.

We now derive the rank formula for $\mathcal{M}$. Although one can derive it from elementary methods, it is easy to obtain it from the matroid intersection theorem. Recall that if $\mathcal{M}_1 = (N, \mathcal{I}_1)$ and $\mathcal{M}_2 = (N, \mathcal{I}_2)$ are two matroids on N, then the max cardinality of a common independent set in $\mathcal{I}_1 \wedge \mathcal{I}_2$ is given by

$$\min_{X \subseteq N} r_1(X) + r_2(N \setminus X).$$

Now consider $U \subseteq S$. Let $U' = f^{-1}(U)$. We observe that $A \subseteq U$ is independent in $\mathcal{I}$ iff there is an $A' \subseteq f^{-1}(U)$ such that $|A'| = |A|$, $f(A') = A$ and A' is independent in $\mathcal{I}'$.

Define a matroid $\mathcal{M}'' = (S', \mathcal{I}'')$, where

$$\mathcal{I}'' = \{I \subseteq f^{-1}(U) \mid |I \cap f^{-1}(U)| \leq 1, u \in U\}.$$

Note that $\mathcal{M}''$ is a partition matroid. Let r'' be the rank of $\mathcal{M}''$. We leave the following claim as an exercise.

Claim 15.2.1. *$r(U)$ is the size of a maximum cardinality independent set in $\mathcal{M}' \wedge \mathcal{M}''$.*

Therefore, by the matroid intersection theorem we have that

$$r(U) = \min_{T \subseteq U'} (r'(T) + r''(U' \setminus T)) = \min_{T \subseteq U} (r'(f^{-1}(T)) + |U \setminus T|),$$

using the fact that $\mathcal{M}''$ is a partition matroid. We leave it to the reader to verify the second equality in the above. ■

Remark 15.1. The proof of the preceding lemma shows that the rank function of $\mathcal{M}$ can be efficiently evaluated via an efficient algorithm for matroid intersection (and oracle access to r'). Having an algorithm for evaluating r allows us to optimize over $\mathcal{M}$ via the greedy algorithm.

15.3 Matroid Union Theorem and Applications

We now formally define the notion of matroid union and the theorem formulated by Edmonds.

Let $\mathcal{M}_1 = (S_1, \mathcal{I}_1), \ldots, \mathcal{M}_k = (S_k, \mathcal{I}_k)$ be matroids. Define

$$\mathcal{M} = \mathcal{M}_1 \vee \mathcal{M}_2 \vee \cdots \vee \mathcal{M}_k = (S_1 \cup S_2 \cup \cdots \cup S_k, \mathcal{I}),$$

where

$$\mathcal{I} = \mathcal{I}_1 \vee \mathcal{I}_2 \vee \cdots \vee \mathcal{I}_k := \{I_1 \cup I_2 \cup \cdots \cup I_k \mid I_i \in \mathcal{I}_i, 1 \leq i \leq k\}.$$

Theorem 15.5 (Matroid Union)**.** *Let $\mathcal{M}_1 = (S_1, \mathcal{I}_1), \ldots, \mathcal{M}_k = (S_k, \mathcal{I}_k)$ be matroids. Then*

$$\mathcal{M} = \mathcal{M}_1 \vee \mathcal{M}_2 \vee \cdots \vee \mathcal{M}_k \tag{15.3}$$

is a matroid. The rank function of $\mathcal{M}$ is given by r, where

$$r(U) = \min_{T \subseteq U}(|U \backslash T| + r_1(T \cap S_1) + \cdots + r_k(T \cap S_k)). \tag{15.4}$$

Remark 15.2. Note that the interesting case is when $S_1, S_2, \ldots, S_k$ are not necessarily disjoint. If they are then the theorem is straight forward. In fact even $k = 2$ the fact that $\mathcal{M}_1 \vee \mathcal{M}_2$ is a matroid is not obvious. The reader may want to try the obvious proof strategy and see why it does not quite work while it is easy to see it when S_1, S_2 are disjoint.

Remark 15.3. The preceding theorem is also referred to as the matroid *partition* theorem for the following reason. A set $U \in S$ is independent in $\mathcal{M}$ iff U can be partitioned into $U_1, \ldots, U_k$, such that for $1 \leq i \leq k$, U_i is independent in $\mathcal{I}_i$; note that U_i are allowed to be $\emptyset$.

Proof. Let $S'_1, \ldots, S'_k$ be copies of $S_1, \ldots, S_k$, such that

$$S'_i \cap S'_j = \emptyset, i \neq j.$$

Let $\mathcal{M}'_i = (S'_i, \mathcal{I}'_i)$, where $\mathcal{I}'_i$ corresponds to $\mathcal{I}_i$. Let $S' = S'_1 \uplus S'_2 \uplus \cdots \uplus S'_k$ and define $\mathcal{M}' = (S', \mathcal{I}')$, where

$$\mathcal{I}' = \{I'_1 \cup I'_2 \cup \cdots \cup I'_k \mid I'_i \in \mathcal{I}_i\}.$$

It is easy to verify that $\mathcal{M}'$ is a matroid since it is disjoint union of matroids. Moreover, it is also easy to see that the rank function of $\mathcal{M}'$ is the following: $r'(T') = \sum_{i=1}^{k} r'_i(T' \cap S'_i)$ for any $T' \subseteq S'$.

Now define $f : S' \to S$ where $S = S_1 \cup S_2 \cup \cdots \cup S_k$, and $f(s') = s$ if s' is the copy of s. Then $\mathcal{M}$ is obtained from $\mathcal{M}'$ by f, and hence by Theorem 15.4,

$\mathcal{M}$ is a matroid. Recall that the rank function of $\mathcal{M}$, via the same theorem, is given by $r(U) = \min_{T \subseteq U} |U \setminus T| + r'(f^{-1}(T))$. Let $T' = f^{-1}(T)$. We have seen that $r'(T') = \sum_{i=1}^{k} r'_i(S'_i \cap T')$. We leave it as an exercise to verify that $\sum_{i=1}^{k} r'_i(S'_i \cap T') = \sum_{i=1}^{k} r_i(S_i \cap T)$. This gives the desired formula for the rank function of $\mathcal{M}$. ■

Remark 15.4. The proof of the preceding theorem shows that the rank function of $\mathcal{M}$ can be efficiently evaluated via an efficient algorithm for matroid intersection and oracle access to $r_1, r_2, \ldots, r_k$. Having an algorithm for evaluating r allows us to optimize over $\mathcal{M}$ via the greedy algorithm.

We state a useful corollary.

Corollary 15.6. *Let $\mathcal{M} = (S, \mathcal{I})$ be a matroid and k be an integer. Then the maximum rank of the union of k independent sets of $\mathcal{M}$ is equal to*

$$\min_{U \subseteq S}(|S \setminus U| + k \cdot r(U)). \tag{15.5}$$

Proof. Take $\mathcal{M}'$ to be union of $\mathcal{M}_1 \vee \mathcal{M}_2 \vee \cdots \vee \mathcal{M}_k$, where $\mathcal{M}_i = \mathcal{M}$. Then the union of k independent sets in $\mathcal{M}$ is an independent set in $\mathcal{M}'$. Thus we are asking for the maximum possible rank in $\mathcal{M}'$. S achieves the maximum rank and by the previous theorem

$$\begin{aligned} r'(S) &= \min_{U \subseteq S}(|S \setminus U| + k \cdot r(S \cap U)) && (15.6)\\ &= \min_{U \subseteq S}(|S \setminus U| + k \cdot r(U)). && (15.7) \end{aligned}$$

■

We now easily derive two important theorems that were first stated by Edmonds.

Theorem 15.7 (Matroid base covering theorem)**.** *Let $\mathcal{M} = (S, \mathcal{I})$ be a matroid. Then S can be covered by k independent sets iff*

$$|U| \leq k \cdot r(U), \forall U \subseteq S. \tag{15.8}$$

Proof. S can be covered by k independent sets iff the rank of S in the union of $\mathcal{M}_1 \vee \mathcal{M}_2 \vee \cdots \vee \mathcal{M}_k$, where $\mathcal{M}_i = \mathcal{M}$, is equal to $|S|$. By Corollary 15.6, this is equivalent to

$$\begin{aligned} &|S \setminus U| + k \cdot r(U) \geq |S|, \forall U \subseteq S \\ &\Rightarrow k \cdot r(U) \geq |U|, \forall U \subseteq S. \end{aligned}$$

■

Exercise 15.1. Derive Nash-Williams forest-cover theorem (Theorem 15.3) as a corollary.

Now we derive the matroid base packing theorem, also formulated by Edmonds.

Theorem 15.8 (Matroid Base Packing Theorem)**.** *Let $\mathcal{M} = (S, \mathcal{I})$ be a matroid. Then there are k disjoint bases in $\mathcal{M}$ iff*

$$k(r(S) - r(U)) \leq |S \setminus U|, \forall U \subseteq S. \tag{15.9}$$

Proof. To see necessity, consider any set $U \subseteq S$. Any base B has the property that $r(B) = r(S)$. And $r(B \cap U) \leq r(U)$. Thus

$$B \cap (S \setminus U) \geq r(S) - r(U).$$

Therefore if there are k disjoint bases then each of these bases requires $r(S) - r(U)$ distinct elements from $S \setminus U$, and hence

$$k(r(S) - r(U)) \leq |S \setminus U|.$$

For sufficiency, we take the k-fold union of $\mathcal{M}$ and there are k disjoint bases if $r'(S)$ in the union matroid $\mathcal{M}'$ satisfies the equation

$$r'(S) = k \cdot r(S)$$

in other words,

$$\min_{U \subseteq S} |S \setminus U| + k \cdot r(U) = k \cdot r(S)$$
$$\Rightarrow |S \setminus U| + k \cdot r(U) \geq k \cdot r(S)|$$

■

Exercise 15.2. Derive Nash-Williams-Tutte theorem on packing spanning trees (Theorem 15.1) as a corollary.

15.4 Algorithmic and Polyhedral Aspects

Let $\mathcal{M} = \mathcal{M}_1 \vee \mathcal{M}_2 \vee \cdots \vee \mathcal{M}_k$. Algorithmic results for $\mathcal{M}$ follow from an independence oracle or rank oracle for $\mathcal{M}$. Recall that a set $I \in \mathcal{I}$ is independent in $\mathcal{M}$ iff I an be partitioned into $I_1, I_2, \ldots, I_k$ such that for $1 \leq i \leq k$, I_i is independent in $\mathcal{I}_i$. Note that this is non-trivial to solve.

Theorem 15.9. *Given rank functions $r_1, \ldots, r_k$ for $\mathcal{M}_1, \ldots, \mathcal{M}_k$, as polynomial time oracles, there is a polynomial time algorithm to implement the rank function oracle r for $\mathcal{M} = \mathcal{M}_1 \vee \mathcal{M}_2 \vee \cdots \vee \mathcal{M}_k$.*

We sketch the proof of the above theorem. Recall the construction in Theorem 15.5 that showed $\mathcal{M}$ is a matroid. We first constructed an intermediate matroid $\mathcal{M}'$ by taking copies of $\mathcal{M}_1, \ldots, \mathcal{M}_k$ and then applied Theorem 15.4 to map $\mathcal{M}'$ to $\mathcal{M}$.

For the matroid $\mathcal{M}'$, one easily obtains an algorithm to implement r' from $r_1, \ldots, r_k$, i.e.

$$r'(U) = \sum_{i=1}^{k} r_i(U \cap S'_i).$$

Recall that we obtained the rank function r for $\mathcal{M}$ from r' for $\mathcal{M}'$ using matroid intersection (see proof of Theorem 15.4). Thus, one can verify that an algorithm for matroid intersection implies an algorithm for r using algorithms for $r_1, \ldots, r_k$. There is also a direct algorithm that avoids using the matroid intersection algorithm — see [57] for details.

Polyhedrally, the base covering and packing theorems imply and are implied by the following

Theorem 15.10. *Given a matroid $\mathcal{M} = (S, \mathcal{I})$, the independent set polytope and base polytope of $\mathcal{M}$ have the integer decomposition property.*

Exercise 15.3. Prove the above theorem using Theorem 15.7 and 15.8.

Capacitated case and algorithmic aspects of packing and covering: The matroid union algorithm allows us to obtain algorithmic versions of the matroid base covering and base packing theorems. As a consequence, for example, there is a polynomial time algorithm that given a multi-graph $G = (V, E)$, outputs the maximum number of edge-disjoint spanning trees in G. It is also possible to solve the capacitated version of the problems in polynomial time. More precisely, let $\mathcal{M} = (S, \mathcal{I})$ and let $c : S \to \mathcal{Z}_+$ be integer capacities on the elements of S. The capacitated version of the base packing theorem is to ask for the maximum number of bases such that no element $e \in S$ is in more than $c(e)$ bases. Similarly, for the base covering theorem, one seeks a minimum number of independent sets such that each element e is in at least $c(e)$ independent sets. The capacitated case be handled by making $c(e)$ copies of each element e, however, this would give only a pseudo-polynomial time algorithm.

Assuming we have a polynomial time rank oracle for $\mathcal{M}$, the following capaciatated problems can be solved in polynomial time. To solve the capacitated versions, one needs polyhedral methods; see [57] for more details.

1. fractional packing of bases, i.e., let $\mathcal{B}$ denote the set of bases of $\mathcal{M}$,

$$\max_{B \in \mathcal{B}} \lambda_B$$

$$\sum_{B \ni e} \lambda_B \leq c(e), \forall e \in S$$

$$\lambda_B \geq 0$$

2. integer packing of bases, same as above but λ_B are restricted to be integer.

3. fractional covering by independent sets, i.e.

$$\min_{I \in \mathcal{I}} \lambda_I$$

$$\sum_{I \ni e} \lambda_I \geq c(e), \forall e \in S$$

$$\lambda \geq 0$$

4. integer covering by independent sets, same as above but λ_I are constrained to be integer.

Matroid Intersection from Matroid Union: We have seen that the matroid union algorithm follows from an algorithm for matroid intersection. The converse can also be shown. To see this, let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two matroids on the same ground set S. Then, one can find the maximum cardinality common independent set in $\mathcal{M}_1 \wedge \mathcal{M}_2$ be considering $\mathcal{M}_1 \vee \mathcal{M}_2^*$ where $\mathcal{M}_2^*$ is the dual of $\mathcal{M}_2$.

Chapter 16

Spanning Trees and Arborescences[1]

16.1 Spanning Trees

Let $G = (V, E)$ be an undirected graph and let $c : E \to R$ be an edge-cost function. Efficient polynomial time algorithms for computing a minimum cost spanning tree (MST) are standard. Spanning trees in G are bases in the associated graphic matroid and Kruskal's algorithm for MST is the essentially the greedy algorithm for computing a minimum cost base in a matroid. From polyhedral results on matroids we obtain corresponding results for spanning trees.

The spanning tree polytope of $G = (V, E)$ is the polytope formed by the convex hull of the characteristic vectors of spanning trees of G, and is determined by the following inequalities. We have a variable $x(e)$ for each $e \in E$ and for a set $U \subseteq V$, $E[U]$ is the set of edges with both end points in U.

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

$$x(E[U]) \leq |U| - 1 \quad U \subseteq V$$

$$x \geq 0$$

If we drop the constraint $x(E) = n - 1$, then we obtain the convex hull of the characterstic vectors of forests in G, called the forest polytope of G; note that forests are the independent sets in the graphic matroid of G. Note that the set of constraints in the above system do not include all the constraints that we would include if we view the spanning tree as a matroid polytope: we would have

[1]Based on notes scribed by Jing Gao in 2010. Mohit Singh pointed out some errors in the previous version of the notes.

inequalities of the form $x(S) \leq \text{rank}(S)$ for each $S \subseteq E$. However, we argued in Section 13.4 that we can drop some redundant constraints.

A natural cut-based formulation for spanning trees is the following:

$$x(\delta(U)) \geq 1 \quad \forall \emptyset \subset U \subset S$$

$$x \geq 0$$

It is easy to check that every spanning tree satisfies the above constraints, but the following example shows that the constraints do not determine the spanning tree polytope. Take G to the n-cycle C_n and set $x(e) = \frac{1}{2}$ on each edge; satisfies the cut-constraints but cannot be written as a convex combination of spanning trees. In fact, it does not even satisfy the constraint that $x(E) = n - 1$.

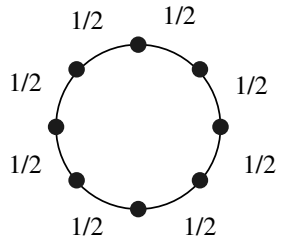


Figure 16.1: Cut LP is not integral for spanning tree.

Exercise 16.1. Show that even if we add the constraint $x(E) = n - 1$ to the cut-constraints, it still does not determine the spanning tree polytope.

We have seen Tutte-Nash-Williams Theorem on maximum number of edge-disjoint spanning trees. Matroid union theorem gives polynomial-time algorithms to find a maximum number of edge-disjoint spanning trees in a given graph. We have also seen Nash-Williams forest cover theorem and again matroid union algorithm can be used to obtain the minimum number of forests that cover E.

16.2 Branchings and Arborescences

Let $D = (V, A)$ be a directed graph. Recall that a branching is a set of edges $A' \subseteq A$ such that

1. $\delta^{-1}_{A'}(v) \leq 1 \quad \forall v \in V$, i.e., at most one edge in A' enters any node v;
2. A' when viewed as undirected edges induces a forest on V.

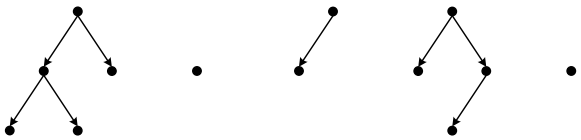


Figure 16.2: Example of a Branching

An arborescence is a branching that has in-degree 1 for all nodes except one, called the root. An arborescence has a directed path from the root to each node $v \in V$.

Proposition 16.2.1. *Let $D = (V, A)$ be a directed graph and let $r \in V$. If r can reach every node $v \in V$, then there is an arborescence in D rooted at r. If G is strongly connected, then for every $v \in V$, there is an arborescence rooted at v.*

Branchings and Matroid Intersection: We saw earlier that branchings in a directed graph $D = (V, A)$ can be viewed as the common independent sets in the intersection of two matroids on A. Let $M_1 = (A, \mathcal{I}_1)$ where $\mathcal{I}_1 = \{A' \subseteq A \mid |A' \cap \delta^{-1}(v)| \leq 1 \quad \forall v \in V\}$. M_1 is a partition matroid. $M_2 = (A, \mathcal{I}_2)$ where $\mathcal{I}_2 = \{A' \subseteq A \mid A'$ when viewed as undirected edges induces a forest on $V\}$. M_2 is a graphic matroid. Thus, one easily sees that $\mathcal{I}_1 \cap \mathcal{I}_2$ is precisely the set of branchings. Moreover, for a fixed r, if we modify M_1 such that $M_1 = \{A' \subseteq A \mid |A' \cap \delta^{-1}(v)| \leq 1 \quad \forall v \in V \setminus \{r\}$ and $|A' \cap \delta^{-1}(r)| = 0\}$, then the set of arborescences rooted at r are precisely the common *bases* of $\mathcal{I}_1$ and $\mathcal{I}_2$.

Using matroid intersection results, one can solve the following problems in polynomial time:

- given $D = (V, A)$ and $w : A \to R$, find a maximum weight branching;
- given D and $c : A \to R$, find a min-cost arborescence rooted at r;
- given D and $c : A \to R$, find a max-cost arborescence rooted at r;

Polyhedral results also follow from matroid intersection. However, one can obtain direct and simple algorithms, and also polyhedral results, for arborescences. We explain them below. We first address algorithms for the optimization problems discussed above.

Combinatorial Algorithms: Let $D = (V, A)$ and $c : A \to R_+$ be a non-negative cost function on the the arcs. We wish to find a min-cost arborescence rooted at given node $r \in V$. We observe that a greedy algorithm similar to Prim's algorithm for computing an MST does not work.

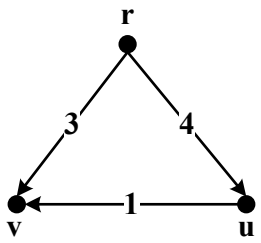


Figure 16.3: An Example

For the above example, greedy method will pick (r, v) and then has to pick (r, u) for total cost of 7. However, optimal arborescence is $\{(r, u), (u, v)\}$ of cost 5.

Algorithm for min-cost r-aroborescence:

1. Let $A_0 = \{a \in A \mid c(a) = 0\}$ be the set of zero-cost arcs. If there is an r-arborescence in A_0 (of cost 0), output it as the min-cost arborescence.

2. Else, let $S_1, \ldots, S_k$ be the vertex sets of the strong connected components of $D[A_0]$. Let $\alpha_i = \min_{a \in \delta^{-1}(S_i)} c(a)$, that is α_i is the cost of the min-cost edge entering S_i.

   for $i = 1$ to k do

   for each $a \in \delta^{-1}(S_i)$

   $c'(a) = c(a) - \alpha_i$

3. Recursively compute a min-cost arborescence in $D = (V, A)$ with edge-cost function c'. Output the solution of recursive call.

First, we argue that the algorithm terminates in polynomial time. If A_0 does not contain an r-arborescence then at least one S_i has all incoming edges of strictly positive cost (why?) and hence in step 2, at least one additional arc has its cost reduced to zero; thus the size of A_0 increases and hence at most $O(m)$ recursive calls suffice. In fact, if we shrink each S_i to a single vertex, one can show that the number of vertices reduces by at least one and hence $O(n)$ recursive calls suffice. Since strong connected components can be found in $O(m)$ time, this leads to an $O(mn)$ running time.

Now we argue correctness. It is easy to see that step 1 is correct. To argue correctness of step 3, we have the following lemma.

Lemma 16.1. *Let $S_1, \ldots, S_k$ be vertex sets of the strong connected components of $D[A_0]$. Then there exists a min-cost arborescence A^* in D s.t. $|\delta^{-1}(S_i) \cap A^*| = 1$.*

Proof. Say A^* is an optimal arborescence and $|A^* \cap \delta^{-1}(S_i)| \geq 2$. Let $a = \arg\min_{a' \in A^* \cap \delta^{-1}(S_i)} c(a')$ be the least cost arc entering S_i in A^*. Then let $A' = (A^* \setminus \delta^{-1}(S_i)) \cup \{a\} \cup (A_0 \cap A[S_i])$; A' is the set of arcs obtained by adding to A^* all zero-cost arcs inside S_i ($A_0 \cap A[S_i]$) and removing all arcs from A^* that enter S_i other than the least cost arc a defined above. It is easy to check that A' contains an r-arborescence and moreover its cost is no more than that of A_0. Further, $|\delta^{-1}(S_i) \cap A'| = 1$ and $|\delta^{-1}(S_j) \cap A'| = |\delta^{-1}(S_j) \cap A^*|$ for all $j \neq i$. Repeating the above process for each S_i gives the desired claim. ■

This leads to the following theorem.

Theorem 16.1. *There is an $O(nm)$ time algorithm to compute a min-cost r-arborescence in a directed graph with n nodes and m edges.*

There is an $O(m + n \log n)$-time algorithm to find a min-cost r-arboresence problem [26]. There is also an algorithm that runs in $O(m \log \log n)$ time for integer weights [52]. Whether there is an $O(m)$-time deterministic or randomized algorithm is an open problem.

Max-weight Arborescences and Branchings. Since any arborescence has exactly $n-1$ edges, one can solve the max-weight arborescence by negating weights, adding a large positive number to make weights positive and then computing a min-weight arborescence.

One can use the max-weight arborescence algorithm to compute a max-weight branching. Note that given $w : A \to R$, we can assume $w(a) \geq 0 \; \forall a$ by removing all arcs with negative weights. We note that a max-weight branching may not be maximal even when weights are positive; this is unlike the case of matroids (in particular, a max-weight forest is a spanning tree if all weights are non-negative and the input graph is connected). To solve the max weight branching problem, we add a new vertex r and connect it to each $v \in V$ with an arc (r, v) of weight 0. Now we find a max-weight arborescence rooted at r. We leave the correctness of this algorithm as an easy exercise.

16.2.1 Polyhedral Aspects

One can obtain polyhedral descriptions for branchings and arborescences via matroid intersection. However, some natural and direct descriptions exist.

Let $\mathcal{P}_{r-arborescence}(D) = convexhull\{\chi(B) \mid B \text{ is a } r\text{-arborescence in } D\}$.

Theorem 16.2. *$\mathcal{P}_{r-arborescence}(D)$ is determined by*

$$x(a) \geq 0 \quad a \in A$$
$$x(\delta^{-1}(v)) = 1 \quad v \in V \setminus \{r\}$$
$$x(\delta^{-1}(U)) \geq 1 \quad U \subseteq V \setminus \{r\}$$

One can prove the preceding in several different ways. We give one proof below and suggest others in exercises and remarks.

Proof. We give an iterated rounding based proof to show that the following set of inequalities

$$0 \leq x(a) \leq 1 \quad a \in A$$

$$x(\delta^{-1}(U)) \geq 1 \quad U \subseteq V \setminus \{r\}$$

is the convex hull of the characteristic vectors of arc sets that contain an r-arborescence. One can easily adapt the proof to show the theorem statement. Let x be any basic feasible solution to the above system. We claim that $\exists a \in A$ s.t. $x(a) = 0$ or $x(a) = 1$. In either case, we obtain the desired proof by induction on $|A|$. If $x(a) = 0$ we consider $D[A \setminus \{a\}]$, if $x(a) = 1$, we shrink the end points of a into a single vertex and consider the resulting graph.

We now prove that $\exists a \in A$ s.t. $x(a) \in \{0, 1\}$. Assume not, then $x(a) \in (0, 1)$ $\forall a \in A$. Let $\mathcal{F} = \{U \in V \setminus \{r\} \mid x(\delta^{-1}(U)) = 1\}$ be the collection of tight sets.

Claim 16.2.1. *Let $X, Y \in \mathcal{F}$ such that $X \cup Y \neq V$. Then $X \cap Y, X \cup Y \in \mathcal{F}$.*

One can prove the preceding via submodularity of the cut function.

Claim 16.2.2. *Let $\mathcal{L}$ be a maximal laminar family in $\mathcal{F}$. Then span($\{\chi(U) \mid U \in \mathcal{L}\}$)=span($\{\chi(U) \mid U \in \mathcal{F}\}$).*

The above claims are based on uncrossing arguments that we have seen in several contexts. We leave the formal proofs as an exercise.

Since $\mathcal{L}$ is a laminar family on $V \setminus \{r\}$, we have

$$|\mathcal{L}| \leq 2(|V| - 1) - 1 \leq 2|V| - 3$$

Since $x(\delta^{-1}(v)) \geq 1$ for each $v \in V \setminus \{r\}$ and $x(a) \in (0, 1)$ for all $a \in A$,

$$|\delta^{-1}(v) \cap A| \geq 2 \quad \forall v \in V \setminus \{r\}$$

This implies that $|A| \geq 2|V| - 2$. However, x is a basic feasible solution and $\mathcal{L}$ determines x, and thus $|\mathcal{L}| = |A|$, a contradiction. ∎

Remark 16.1. In fact, one can show that the system of inequalities is TDI and this implies a min-max result as well. See [57] for more details. Note that the arborescence polytope can be derived as a special case of the matroid intersection polytope. Although the inequalities look a little different, as in the spanning tree polytope, one can show that these set of inequalities also capture the same polytope. However, one has to be more careful about the TDI property.

16.3 Arc-Disjoint Arborescences

A beautiful theorem of Edmonds is the following.

Theorem 16.3. *Let $D = (V, A)$ be a digraph and let $r \in V$. D has k arc-disjoint r-arborescences iff for each $v \in V \setminus \{r\}$ there are k arc-disjoint paths from r to v.*

Proof. If D has k arc-disjoint r-arborescences then clearly for each $v \in V \setminus \{r\}$, there are k arc-disjoint $r \rightarrow v$ paths in D, one in each of the arboresences.

We prove the converse via a proof given by Lovász using induction on k (proof adapted from [39]). Let $\mathcal{C} = \{U \subset V \mid r \in U\}$ be the collection of all proper subsets of V that contain r. Note that the condition that there are k-arc-disjoint paths from r to each v is equivalent to, by Menger's theorem,

$$|\delta^+(U)| \geq k \quad \forall U \in \mathcal{C}.$$

The idea is to start with the above condition and find an r-arborescence A_1 s.t.

$$|\delta^+(U) \setminus A_1| \geq k - 1 \qquad \forall U \in \mathcal{C}.$$

Then, by induction, we will be done. We obtain A_1 by growing an r-arborescence from r as follows. We start with $A_1 = \emptyset$ and $S = \{r\}$; S is the set of vertices reachable from r via arcs in the current set of arcs A_1. We maintain the property that $|\delta^+(U) \setminus A_1| \geq k - 1 \; \forall U \subset V, r \in U$. If we reach $S = V$, we are done.

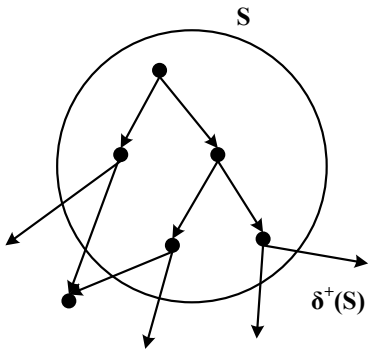


Figure 16.4: S and $\delta^+(S)$

If $S \neq V$, we wish to find an arc in $\delta^+(S)$ to add to A_1 and grow S. Call a set $X \subset V$ *critical/dangerous* if

- $X \in \mathcal{C}$ and
- $|\delta^+(X) \setminus A_1| = k - 1$, and

- $X \cup S \neq V$

Claim 16.3.1. *Suppose there are no critical sets. Then any arc $a \in \delta^+(S)$ can be used to augment A_1.*

To see the claim, suppose we add $a = (u, v) \in \delta^+(S)$ to A_1 and we violate the invariant; that is there is some $U \subset V, r \in U$ such that $|\delta_{G-(A_1+a)}(U)| < k-1$. This implies that $(u, v) \in \delta_G(U)$ and $|\delta_{G-A_1}(U)| = k-1$. Moreover, since $(u, v) \in \delta^+(S)$ and $(u, v) \in \delta^+(U)$ we have $v \notin S \cup U$ (hence $S \cup U \neq V$). Therefore U is critical.

If X is critical, then we cannot pick any unused arcs from $\delta^+(X)$ to grow A_1. The goal is to show that there always exists an arc $a \in \delta^+(S)$ such that a does not cross any critical set. We claim the following uncrossing property for critical sets.

Claim 16.3.2. *Let $G' = (V, A \setminus A_1)$ Suppose X, Y are critical and $X \cup Y \neq V$, then $|\delta_{G'}(X \cap Y)| = |\delta_{G'}(X \cup Y)| = k - 1$. Moreover, if $X \cup Y \cup S \neq V$ then $X \cap Y$ and $X \cup Y$ are critical.*

Proof. We have, by submodularity of the cut function $|\delta^+_G()|$,

$$|\delta_{G'}(X)| + |\delta_{G'}(Y)| \geq |\delta_{G'}(X \cup Y)| + |\delta_{G'}(X \cap Y)|.$$

Since $r \in X \cap Y$ and $r \in X \cup Y$ and $X \cup Y \neq V$, we have that $|\delta_{G'}(X \cap Y)| \geq k-1$ and $|\delta_{G'}(X \cup Y)| \geq k - 1$. Since X, Y are critical, $|\delta^+_{G'}(X)| = k - 1$ and $|\delta_{G'}(Y)| = k - 1$. This implies,

$$(k - 1) + (k - 1) \geq |\delta_{G'}(X \cap Y)| + |\delta_{G'}(X \cup Y)| \geq (k - 1) + (k - 1)$$

and hence $|\delta_{G'}(X \cap Y)| = |\delta_{G'}(X \cup Y)| = k - 1$. If $X \cup Y \cup S \neq V$ then $X \cup Y$ and $X \cap Y$ satisfy the definition of a critical set. ■

Let X be a inclusion-wise *maximal* critical set.

Claim 16.3.3. *There exists an arc $(u, v) \in A \setminus A_1$ such that $u \in S \setminus X$ and $v \in V \setminus (S \cup X)$.*

Proof. See Fig 16.6. Note that $A_1 \cap \delta^+(S) = \emptyset$ since S, by defintion, is a set of reachable nodes in A_1. Since X is a critical set we have $S \cup X \neq V$, and hence $|\delta^+(S \cup X)| \geq k$ (by assumption on G) and $|\delta^+_{G-A_1}(X)| = k - 1$ (since X is critical), we have an arc as desired. See Figure 16.5. ■

Now let $A'_1 = A_1 + (u, v)$. The claim is that for all $U \in C$, $|\delta^+(U) \setminus A'_1| \geq k - 1$. Suppose not. Then let Y be such that $|\delta^+(Y) \setminus A'_1| < k - 1$ but this implies that $|\delta^+(Y) \setminus A_1| = k - 1$, that is Y is critical and $(u, v) \in \delta^+(Y)$. But consider Y, X both critical and $Y \cup X \neq V$ since $v \notin Y$, $v \notin X$. Moreover, since $v \notin S$, we have $X \cup Y \cup S \neq V$. Therefore, $X \cup Y$ is critical, which contradicts maximality of X. ■

We note that the above theorem shows the integer decomposition property for the arboresence polytope discussed earlier. The proof can be converted into a polynomial time algorithm to find a maximum number of arc-disjoint r-arborescences in a given digraph. First we let k be the $\min_{v \in V \setminus \{r\}} \lambda_D(r, v)$ where $\lambda_D(r, v)$ is the arc-connectivity between r and v. The theorem guarantees k r-aroboresences. In the above proof, the main issue is to find in each iteration an arc to augment A_1 with. We note that given an arc a, we can check if $A_1 + a$ satisfies the invariant by checking the min-cut value from r to each node v, in the graph $D' = D[A \setminus (A_1 + a)]$. The proof guarantees the existence of an arc a that can be used to augment A_1 and hence one of the m arcs in D will succeed. It is easy to see that this leads to a polynomial time algorithm. There is also a polynomial time algorithm for the capacitated case. See [57] for details.

Edmonds derived the arc-disjoint r-arborescences theorem from a more general theorem on disjoint branchings. We refer the reader to [57].

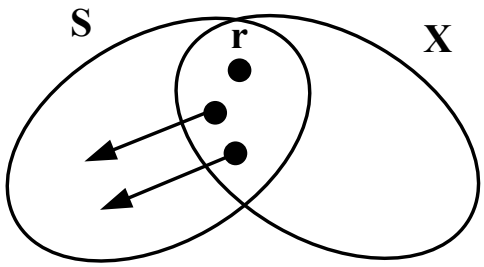


Figure 16.5: All arcs from $A_1 \cap \delta^+(X)$ go from X to S

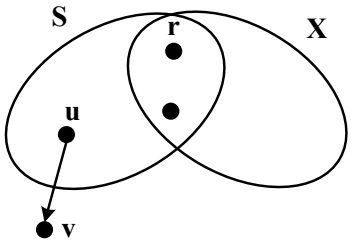


Figure 16.6: Proof of Claim 16.3.3.

Chapter 17

Submodular Set Functions and Polymatroids[1]

Submodularity plays an important role in combinatorial optimization. Given a finite ground set S, a *set function* $f : 2^S \to \mathbb{R}$ is *submodular*[2] if

$$f(A) + f(B) \geq f(A \cap B) + f(A \cup B) \quad \forall A, B \subseteq S \tag{17.1}$$

which can be rewritten as

$$f(A) - f(A \cap B) \geq f(A \cup B) - f(B) \quad \forall A, B \subseteq S. \tag{17.2}$$

A different looking definition is the following:

$$f(A + e) - f(A) \geq f(B + e) - f(B) \quad \forall A \subseteq B \text{ and } e \in S \setminus B. \tag{17.3}$$

A seemingly restricted version of the preceding definition is the following:

$$f(A+e_1)-f(A) \geq f(A+e_2+e_1)-f(A+e_1+e_2) \quad \forall A \subseteq S \text{ and distinct } e_1, e_2 \in S \setminus A. \tag{17.4}$$

Definition 17.1. *Given a real-valued set function $f : 2^S \to \mathbb{R}$ the* marginal value *of $e \in S$ to a set A, denoted by $f(e \mid A)$, is $f(A + e) - f(A)$. The marginal value of a set $X \subseteq S$ to another set A, denoted by $f(X \mid A)$, is defined as $f(X \cup A) - f(A)$.*

We see that (17.3) defines submodularity via the so called *diminishing marginal value/utility* property.

[1] Based on notes scribed by Bolin Ding in 2010.

[2] A function $f : 2^S \to \mathbb{R}$ is modular iff $f(A \cup B) + f(A \cap B) = f(A) + f(B)$. f is modular iff there exists a weight function $w : S \to \mathbb{R}$ such that $f(A) = \sum_{e \in A} w(e)$.

Exercise 17.1. Prove that definitions (17.3) and (17.4) are equivalent. That is, any set function that satisfies one property satisfies the other.

Exercise 17.2. Suppose f satisfies (17.3). Then prove that for all $A \subset B$ and $X \subset S \setminus B$, $f(X \mid A) \geq f(X \mid B)$. Use this to argue that (17.3) implies (17.2) and hence (17.1).

Exercise 17.3. Prove that definitions (17.3) and (17.4) are equivalent. That is, any set function that satisfies one property satisfies the other.

Exercise 17.4. Prove that (17.2) implies (17.3).

The preceding exercises show the equivalence of the definitions of submodularity.

Additional properties of submodular set functions are useful to keep in mind when considering specific settings and applications. A set function $f : 2^S \to \mathbb{R}$ is *non-negative* if $f(A) \geq 0 \; \forall A \subseteq S$. f is *symmetric* if $f(A) = f(S \setminus A) \; \forall A \subseteq S$. f is *monotone* (*non-decreasing*) if $f(A) \leq f(B) \; \forall A \subseteq B$. f is *integer-valued* if $f(A) \in \mathbb{Z} \; \forall A \subseteq S$. f is *normalized* if $f(\emptyset) = \emptyset$.

17.1 Examples of submodular set functions

Cut functions in graphs and hypergraphs. Given an undirected graph $G = (V, E)$ and a a non-negative edge capacities $c : E \to \mathbb{R}_+$, the *cut function* $f : 2^V \to \mathbb{R}_+$ is defined as $f(U) = c(\delta(U))$, that is, the sum of capacities of edges between U and $V \setminus U$. f is submodular. It is non-negative, symmetric, but not monotone.

In an undirected hypergraph $G = (V, \mathcal{E})$ with capacity function $c : \mathcal{E} \to \mathbb{R}_+$, the *cut function* is defined as $f(U) = c(\delta_{\mathcal{E}}(U))$, where $\delta_{\mathcal{E}}(U) = \{e \in \mathcal{E} \mid e \cap U \neq \emptyset \text{ and } e \cap (S \setminus U) \neq \emptyset\}$. This is also submodular, symmetric and not necessarily monotone.

In a directed graph $D = (V, A)$ with capacity function $c : A \to \mathbb{R}_+$, the *cut function* is defined as $f(U) = c(\delta_{\text{out}}(U))$, where $\delta_{\text{out}}(U)$ is the set of arcs leaving U. This function is submodular and not necessarily symmetric or monotone.

Exercise 17.5. Prove that if the edge capacities are allowed to be negative then the cut function need not be submodular.

Matroid rank function. Let $M = (S, \mathcal{I})$ be a matroid. Then the rank function $r_M : 2^S \to \mathbb{Z}_+$ is submodular (also non-negative, integer-valued, and monotone). In particular a non-negative integer-valued monotone submodular function which has the property that $f(e) \leq 1$ for all e is the rank function of a matroid.

Matroid intersection. Let $M_1 = (S, \mathcal{I}_1)$ and $M_2 = (S, \mathcal{I}_2)$ be two matroids. Then the function f given by $f(U) = r_{M_1}(U) + r_{M_2}(S \setminus U)$, for $U \subseteq S$, is submodular, non-negative and integer-valued. By the matroid intersection theorem, the minimum value of f is equal to the maximum cardinality of a common independent set in the two matroids.

Coverage in set system. Let $T_1, T_2, \ldots, T_n$ be subsets of a finite set T. Let $S = [n] = \{1, 2, \ldots, n\}$ be the ground set. The *coverage function* $f : 2^S \to \mathbb{R}_+$ is defined as $f(A) = |\cup_{i \in A} T_i|$.

A generalization is obtained by introducing the weights $w : T \to \mathbb{R}_+$ of elements in T, and defining the weighted coverage $f(A) = w(\cup_{i \in A} T_i)$.

Another generalization is to introduce a submodular and monotone weight-function $g : 2^T \to \mathbb{R}_+$ of subsets of T. Then the function f is defined as $f(A) = g(\cup_{i \in A} T_i)$.

All the three versions of f here are submodular, non-negative, and monotone.

Flows to a sink. Let $D = (V, A)$ be a directed graph with an arc-capacity function $c : A \to \mathbb{R}_+$. Let a vertex $t \in V$ be the *sink*. Consider a subset $S \subseteq V \setminus \{t\}$ of vertices. Define a function $f : 2^S \to \mathbb{R}_+$ as $f(U) =$ max flow from U to t in the directed graph D with edge capacities c, for a set of 'sources' U. Then f is submodular, non-negative and monotone.

Max element. Let S be a finite set and let $w : S \to \mathbb{R}$. Define a function $f : 2^S \to \mathbb{R}$ as $f(U) = \max\{w(u) \mid u \in U\}$ for nonempty $U \subseteq S$, and $f(\emptyset) = \min\{w(u) \mid u \in S\}$. Then f is submodular and monotone.

Entropy and Mutual information. Let $X_1, X_2, \ldots, X_n$ be random variables over some underlying probability space, and $S = \{1, 2, \ldots, n\}$. For $A \subseteq S$, define $X_A = \{X_i \mid i \in A\}$ to be the set of random variables with indices in A. Then $f(A) = H(X_A)$, where $H(\cdot)$ is the entropy function, is submodular (also non-negative and monotone). Also, $f(A) = I(X_A; X_{S \setminus A})$, where $I(\cdot;\cdot)$ is the mutual information of two random variables, is submodular.

Exercise 17.6. Prove the submodularity of the functions introduced in this subsection.

17.1.1 Unconstrained Submodular Set Function Optimization

Two fundamental discrete optimization problems related to submodular functions are described below. We will assume that submodular set function $f : 2^S \to \mathbb{R}$ is provided as a value oracle, that is, given a set $A \subseteq S$ the oracle returns $f(A)$. In some settings it is also necessary to assume that f is rational (or integer) valued.

Submodular set function minimization: Given f as a value oracle, output $\min_{A \subseteq S} f(A)$. The search problem is to find a set A that achieves the minimum.

A fundamental theorem in combinatorial optimization with many applications is the following.

Theorem 17.2. *There is a strongly-polynomial time algorithm that solves the submodular function minimization problem in the value oracle model.*

Exercise 17.7. Show how the minimum s-t cut in a capacitated directed graph can be cast as a special case of submodular set function minimization.

Exercise 17.8. Suppose $\mathcal{M} = (S, \mathcal{I})$ is a matroid with rank function r. Consider the matroid polytope described as the set of vectors $\{x \in \mathbb{R}^S \mid x \geq 0, x(U) \leq r(U), U \subseteq S\}$. The separation problem for the matroid polytope is: given $z \in \mathbb{R}^S$, is z in the polytope and if not, output a hyperplane separating z from the polytope. Show how this can be reduced to submodular set function minimization.

Submodular set function maximization: Given f as a value oracle, output $\max_{A \subseteq S} f(A)$. The search problem is to find a set A that achieves the maximum.

In contrast to the minimization problem we have the following hardness result.

Claim 17.1.1. *The submodular function maximization problem is NP-Complete even for the special case of Max-Cut where the input in a graph $G = (V, E)$ and the goal is to find* $\max_{A \subseteq V} |\delta(A)|$.

Even though the maximization problem is NP-Hard, a wealth of results are known about approximation algorithms — see [6].

17.2 Polymatroids

Edmonds wrote a seminal paper on submodular functions [] which he viewed from a polyhedral viewpoint that he developed initially for matroids. Recall that the independence polytope of a matroid $\mathcal{M} = (S, \mathcal{I})$ is given by the system $\{x \in \mathbb{R}^S \mid x \geq 0, x(U) \leq r(U) \quad \forall U \subseteq S\}$ where $r : 2^S \to \mathbb{Z}_+$ is the rank function of $\mathcal{M}$.

17.2.1 Digression on connection to matroids

Submodular set functions can be negative and non-monotone. However, every submodular set function $f : 2^S \to \mathbb{R}$ can be expressed as the sum of a monotone, normalized, non-negative submodular function $g : 2^S \to \mathbb{R}$ and a modular function $h : 2^S \to \mathbb{R}$. A useful way to see it as via the following lemma.

Lemma 17.1. *Let $f : 2^S \to \mathbb{R}$ be a normalized submodular set function. For each $e \in S$ let $w(e) = \max\{0, -f(e \mid S - e)\}$. Let $g : 2^S \to \mathbb{R}$ where $g(A) = f(A) + w(A)$ for each $A \subseteq S$. Then g is a normalized monotone submodular set function.*

Proof. It is easy to see that g is normalized submodular. To verify that g is monotone it suffices to prove that $g(e \mid A) = g(A + e) - g(A) \geq 0$ for each $A \subset S$ and $e \in S \setminus A$.

$$g(e \mid A) = f(e \mid A) + w(e) \geq f(e \mid A) - f(e \mid S - e) \geq 0$$

where the second inequality follows via submodularity. ■

Example 17.1. Given graph $G = (V, E)$ let $f(A) = |\delta(A)|$ for $A \subseteq V$, the cut function. Then f is submodular but not monotone. Let $w(v) = \delta(v)$ the degree of v. Then $f + w$ is monotone submodular.

Since a matroid rank function is monotone and integer valued it is natural to wonder if every integer-valued monotone submodular set function can be thought of as a matroid rank function. Indeed this is true via a construction of Helgason. To develop some intution consider a monotone submodular function $f : 2^S \to \mathbb{Z}_+$. In general $f(e)$ can be some integer larger than one while a matroid rank function r satisfies $r(e) \in \{0, 1\}$. Thus, if we want to construct a matroidal representation of f it is natural to associate $f(e)$ new elements for each e. We formalize the construction. Given S we construct a new set X where $X = \uplus_e X_e$ where X_e is a set of $f(e)$ elements (X_e and $X_{e'}$ for $e \neq e'$ are disjoint sets). We define a set function r over ground set X as follows. For $U \subseteq X$ let

$$r(U) = \min_{T \subseteq E} (f(T) + |U \setminus \bigcup_{e \in T} X_e|).$$

One can then prove the following which establishes a connection between f and the rank function of a matroid.

Lemma 17.2. *(i) r is the rank function of a matroid over X. (ii) For any $T \subseteq E$, $f(T) = r(\cup_{e \in T} X_e)$.*

Exercise 17.9. Prove the preceding lemma.

Edmonds define two polyhedra associated with a set function f on S:

$$P_f = \{x \in \mathbb{R}^S \mid x(U) \leq f(U) \; \forall U \subseteq S, \; x \geq \mathbf{0}\}$$

$$EP_f = \{x \in \mathbb{R}^S \mid x(U) \leq f(U) \; \forall U \subseteq S\}.$$

If f is a submodular function, then P_f is called the *polymatroid associated with* f, and EP_f the *extended polymatroid associated with* f. Note that EP_f drops the

non-negativity constraints. A polyhedron is called an (extended) polymatroid if it is the (extended) polymatroid associated with some submodular function. Since $0 \leq x_s \leq f(\{s\})$ for each $s \in S$, a polymatroid is bounded, and hence is a polytope. On the other hand EP_f is unbounded if it is non-empty since $x \in EP_f$ implies that $x - y \in EP_f$ for all $y \geq 0$.

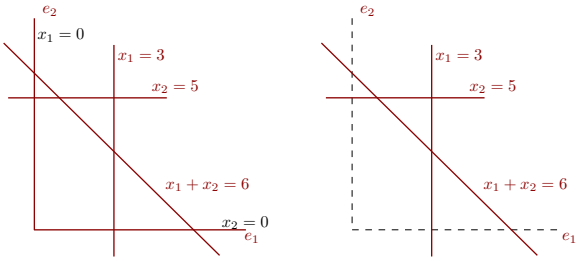


Figure 17.1: Example of a polymatroid with two elements: $f(\emptyset) = 0, f(e_1) = 3, f(e_2) = 5, f(e_1, e_2) = 6$. The extended polymatroid, shown in the second figure, does not have the non-negativity constraints.

Claim 17.2.1. *P_f is non-empty iff $f \geq \mathbf{0}$, and EP_f is non-empty iff $f(\emptyset) \geq 0$.*

Remark 17.1. Suppose $f : 2^S \to \mathbb{R}$ is a normalized submodular function. Let $a \in \mathbb{R}^n$ be weights on the elements which induces a modular function on S. Consider $g = f + a$ which is submodular. Then $EP_g = EP_f + a = \{x + a \mid x \in EP_f\}$, in other words the polyhedron for g is a translation of the one for f.

Remark 17.2. Suppose f is the rank function of a matroid $\mathcal{M}$, then P_f is the independent set polytope of $\mathcal{M}$.

A vector x in EP_f (or in P_f) is called a *base vector* of EP_f (or of P_f) if $x(S) = f(S)$. A *base vector* of f is a base vector of EP_f. The set of all base vectors of f is called the *base polytope* of EP_f or of f. It is a face of EP_f and denoted by B_f:

$$B_f = \{x \in \mathbb{R}^S \mid x(U) \leq f(U)\ \forall U \subseteq S,\ x(S) = f(S)\}.$$

B_f is a polytope. To see that we first observe that $x_e \leq f(\{e\})$ for each $e \in S$. Second, since $x(S) = f(S)$ we have $x_e = f(S) - x(S - e)$ but $x(S - e) \leq f(S - e)$ and hence $x_e \geq f(S) - f(S - e)$. Thus for each e we have $f(S) - f(S - e) \leq x_e \leq f(e)$.

The following claim is about the set of tight constraints in the extended polymatroid associated with a submodular function f.

Claim 17.2.2. *Let $f : 2^S \to \mathbb{R}$ be a submodular set function. For $x \in EP_f$, define $\mathcal{F}_x = \{U \subseteq S \mid x(U) = f(U)\}$ (tight constraints). Then $\mathcal{F}_x$ is closed under taking unions and intersections.*

Proof. Consider any two sets $U, V \in \mathcal{F}_x$, we have

$$f(U \cup V) \geq x(U \cup V) = x(U) + x(V) - x(U \cap V) \geq f(U) + f(V) - f(U \cap V) \geq f(U \cup V).$$

Therefore, $x(U \cup V) = f(U \cup V)$ and $x(U \cap V) = f(U \cap V)$. ∎

17.3 Greedy for optimizing over a polymatroid

Let $f : 2^S \to \mathbb{R}$ be a submodular function and assume it is given as a value oracle. Also given a weight vector $w : S \to \mathbb{R}_+$, we consider the problem of maximizing $w \cdot x$ over EP_f.

$$\begin{aligned} &\max w \cdot x \\ &x \in EP_f. \end{aligned} \tag{17.5}$$

Edmonds showed that the greedy algorithm for matroids can be generalized to this setting.

We can assume without loss of generality that $w \geq \mathbf{0}$, because otherwise, the maximum value is unbounded (why?). We we can assume that $f(\emptyset) = 0$: if $f(\emptyset) < 0$, $EP_f = \emptyset$; and if $f(\emptyset) > 0$, setting $f(\emptyset) = 0$ does not violate the submodularity.

Greedy algorithm. Consider the following greedy algorithm:

1. Order $S = \{s_1, s_2, \ldots, s_n\}$ such that $w(s_1) \geq \ldots \geq w(s_n)$. Let $A_i = \{s_1, \ldots, s_i\}$ for $1 \leq i \leq n$ and let $A_0 = \emptyset$.
2. Let $x'(s_i) = f(A_i) - f(A_{i-1})$, for $1 \leq i \leq n$.
3. Output x'.

Why is the above called a greedy algorithm? It corresponds the following algorithm.

1. Order $S = \{s_1, s_2, \ldots, s_n\}$ such that $w(s_1) \geq \ldots \geq w(s_n)$.
2. $x' = -\infty$
3. for $i = 1$ to n do
   increase x'_i as much as possible without violating feasibility of x'
4. Ouput x'

Why are the two algorithms the same? We can estbalish that the second algorithm sets x'_i to $f(A_i) - f(A_{i-1})$ by induction on i. For the base case we see that there is a constraint $x(s_1) \leq f(s_1)$. Moreoever for all $i > 1$ we have $x'(s_i) = -\infty$ and hence the other constraints of the form $x(U) \leq f(U)$ where $s_1 \in U$ do not constrain $x'(s_1)$. Thus the maximum value that we can increase

$x'(s_1)$ to is $f(s_1)$. Given that $x'(s_1) = f(s_1)$, when increasing $x'(s_2)$, greedy has two constraints to worry about. $x(s_1) + x(s_2) \leq f(s_1, s_2)$ and $x(s_2) \leq f(s_2)$. Note that $x'(s_1) = f(s_1)$ after first iteration. By submodularity $f(s_1, s_2) - f(s_1) \leq f(s_2)$, and hence the maximum we can set $x'(s_2)$ is $f(s_1, s_2) - f(s_1)$. Thus, inductively we can prove that $x'(s_i) = f(A_i) - f(A_{i-1})$ is the greedy choice for i. We leave the formal proof as an easy exercise.

Note that the greedy algorithm is a strongly polynomial-time algorithm and calls the value oracle $O(n)$ times.

Lemma 17.3. *The output x' of the Greedy algorithm is a feasible solution when $f(\emptyset) = 0$.*

Proof. We need to prove that for all $U \subseteq S$, $x'(U) \leq f(U)$. If $U = \emptyset$ it follows from $f(\emptyset) = 0$. Suppose $U - \{s_{i_1}, e_{i_2}, \ldots, s_{i_k}\}$ for some $k \geq 1$ and $i_1 < i_2 \ldots < i_k$. For $j = 1$ to k let $B_j = \{s_{i_1}, s_{i_2}, \ldots, s_{i_j}\}$ and let $B_0 = \emptyset$. Then

$$f(U) = \sum_{j=1}^{k} (f(B_j) - f(B_{j-1})) = \sum_{j=1}^{k} f(s_{i_j} \mid B_{j-1}) \geq \sum_{j=1}^{k} f(s_{i_j} \mid A_{i_j-1}) = \sum_{j=1}^{k} x'_{i_j} = x'(U).$$

In the above we used submodularity of f to say that $f(s_{i_j} \mod B_{j-1}) \geq f(s_{i_j} \mid A_{i_j j-1})$ since $B_{j-1} \subseteq A_{i_j-1}$. ∎

To show that the greedy algorithm above yields an optimum solution we consider the dual LP.

$$\begin{aligned} \min \sum_{U \subseteq S} y(U) f(U) \\ \sum_{U \ni s_i} y(U) = w(s_i) \\ y \geq \mathbf{0}. \end{aligned} \tag{17.6}$$

Define the dual solution: $y'(A_n) = y'(S) = w(s_n)$, $y'(A_i) = w(s_i) - w(s_{i+1})$ for $1 \leq i \leq n-1$, and $y'(U) = 0$ for all other $U \subseteq S$.

The following two claims are easy to verify.

Claim 17.3.1. *y' is dual feasible.*

Claim 17.3.2. $\sum_{i=1}^{n} w(s_i) x'(s_i) = \sum_{i=1}^{n} f(A_i) y(A_i) = \sum_{U \subseteq S} f(U) y'(U)$.

Thus, we see that x' and y' are optimum primal and dual solutions. Moreover, y' is integral if w is integral.

We thus obtain the following.

Theorem 17.3. *If $f : 2^S \to \mathbb{R}$ is a submodular function with $f(\emptyset) = 0$, the greedy algorithm (computing x') gives an optimum solution to* (17.5)*. Moreover, the system of inequalities $\{x \in \mathbb{R}^S \mid x(U) \leq f(U),\ \forall U \subseteq S\}$ is totally dual integral (TDI).*

Exercise 17.10. Prove that the system $\{x \in \mathbb{R}^S \mid x(U) \leq f(U),\ \forall U \subseteq S\}$ is box-TDI.

Now consider the case of P_f. Note that P_f is non-empty iff $f \geq \mathbf{0}$. We note that if f is monotone and non-negative, then the solution x' produced by the greedy algorithm satisfies $x \geq \mathbf{0}$ and hence is feasible for P_f. So we obtain:

Corollary 17.4. *Suppose f is a non-negative monotone submodular function on S with $f(\emptyset) = 0$. Let $w : S \to \mathbb{R}_+$, then the greedy algorithm gives an optimum solution x' to $\max\{w \cdot x \mid x \in P_f\}$. Moreover, the system of inequalities $\{x \in \mathbb{R}^S_+ \mid x(U) \leq f(U),\ \forall U \subseteq S\}$ is box-TDI.*

Therefore, from Theorem 17.3 and Corollary 17.4, for any integer-valued submodular function f, EP_f is an integer polyhedron, and if in addition f is non-negative and monotone, P_f is also an integer polyhedron.

One-to-one correspondence between f and EP_f. Theorem 17.3 also implies f can be recovered from EP_f. In other words, for any extended polymatroid P, there is a unique submodular function f satisfying $f(\emptyset) = 0$, with which P is associated with (that is, $EP_f = P$), since:

Claim 17.3.3. *Let f be a submodular function on S with $f(\emptyset) = 0$. Then $f(U) = \max\{x(U) \mid x \in EP_f\}$ for each $U \subseteq S$.*

Proof. Let $\alpha = \max\{x(U) \mid x \in EP_f\}$. $\alpha \leq f(U)$, because $x \in EP_f$. To prove $\alpha \geq f(U)$, in (17.5), define $w(s_i) = 1$ iff $s_i \in U$ and $w(s_i) = 0$ otherwise. Consider the greedy algorithm producing x'. We saw that $x'(s_i) = f(A_i) - f(A_{i-1}$. We have $\sum_i w(s_i)x'(s_i) = f(U)$ (why?). By the optimality of the greedy algorithm we see that $\alpha = f(U)$. ∎

There is a similar one-to-one correspondence between non-empty polymatroids and non-negative monotone submodular functions f with $f(\emptyset) = 0$. We can also show that, for any such function f, $f(U) = \max\{x(U) \mid x \in P_f\}$ for each $U \subseteq S$.

17.4 Operations on Submodular Functions

Sums: It is useful to consider a few basic operations on submodular functions. First, suppose $f : 2^S \to \mathbb{R}$ and $g : 2^S \to \mathbb{R}$ are two submodular set functions over the same ground set S. Then $f + g$ is submodular (why?). If $\alpha > 0$ is

non-negative number than αf which is the function that assigns value $\alpha f(A)$ to each set $A \subseteq S$, is submodular. Thus non-negative combinations of submodular functions are submodular. Note that negating a submodular function leads to a supermodular function. A modular function h is both supermodular and submodular and $-h$ is modular iff h is modular. Thus, if f is submodular and g is modular then $f + g$ and $f - g$ are both submodular.

Restriction and contraction: Given $f : 2^S \to \mathbb{R}$ and $S' \subset S$ we can restrict f to a subset $S' \subset S$ in the natural way. We obtain $g : 2^{S'} \to \mathbb{R}$ where $g(A) = f(A)$. Clearly g is submodular. This can be thought of deleting $S \setminus S'$. Similarly, given $X \subset S$, one can contract f to X. More formally we consider $g : 2^{S \setminus X} \to \mathbb{R}$ where $g(A) = f(X \cup A)$. It is not difficult to show that g is submodular.

Truncation: Given a submodular set function f on S and a vector $a \in \mathbb{R}^S$, define the set function $f|a$ as

$$(f|a)(U) = \min_{T \subseteq U} (f(T) + a(U \setminus T)).$$

Claim 17.4.1. *If f is a submodular set function on S, $f|a$ is also submodular.*

Proof. Let $g = f|a$ for the simplicity of notation. Fix $X, Y \subseteq S$. Let $X' \subseteq X$ such that $g(X) = f(X') + a(X \setminus X')$, and $Y' \subseteq Y$ such that $g(Y) = f(Y') + a(Y \setminus Y')$. Then, from the definition of g,

$$g(X \cap Y) + g(X \cup Y) \leq \left(f(X' \cap Y') + a((X \cap Y) \setminus (X' \cap Y'))\right) + \left(f(X' \cup Y') + a((X \cup Y) \setminus (X' \cup Y'))\right).$$

From the submodularity of f,

$$f(X' \cap Y') + f(X' \cup Y') \leq f(X') + f(Y').$$

And from the modularity of a,

$$\begin{aligned} a((X \cap Y) \setminus (X' \cap Y')) + a((X \cup Y) \setminus (X' \cup Y')) &= a(X \cap Y) + a(X \cup Y) - a(X' \cap Y') - a(X' \cup Y') \\ &= a(X) + a(Y) - a(X') - a(Y'). \end{aligned}$$

Therefore, we have

$$g(X \cap Y) + g(X \cup Y) \leq f(X') + f(Y') + a(X \setminus X') + a(Y \setminus Y') = g(X) + g(Y).$$

■

What is $EP_{f|a}$ and $P_{f|a}$? We have the following claim.

Claim 17.4.2. *If f is a submodular set function on S and $f(\emptyset) = 0$, $EP_{f|a} = \{x \in EP_f \mid x \leq a\}$ and $P_{f|a} = \{x \in P_f \mid x \leq a\}$.*

Proof. For any $x \in EP_{f|a}$ and any $U \subseteq S$, we have that $x(U) \leq (f|a)(U) \leq f(U) + a(U \setminus U) = f(U)$ implying $x \in EP_f$, and that $x(U) \leq (f|a)(U) \leq f(\emptyset) + a(U \setminus \emptyset) = a(U)$, implying $x \leq a$.

For any $x \in EP_f$ with $x \leq a$ and any $U \subseteq S$, suppose that $(f|a)(U) = f(T) + a(U \setminus T)$. Then we have, $x(U) = x(T) + x(U \setminus T) \leq f(T) + a(U \setminus T) = (f|a)(U)$, implying $x \in EP_{f|a}$.

The proof of $P_{f|a} = \{x \in P_f \mid x \leq a\}$ is similar. ■

A special case of the above claim is that when $a = \mathbf{0}$, then $(f|\mathbf{0})(U) = \min_{T \subseteq U} f(T)$ and $EP_{f|\mathbf{0}} = \{x \in EP_f \mid x \leq \mathbf{0}\}$.

17.5 Submodular Function Minimization via Ellipsoid

Let $f : 2^S \to \mathbb{R}$ be a submodular function and assume it is given as a value oracle, that is, when given $U \subseteq S$, the oracle returns $f(U)$. Our goal is to find $\min_{U \subseteq S} f(U)$. We describe an algorithm based on the equivalence of optimization and separation (the ellipsoid-based method).

We can assume $f(\emptyset) = 0$ (by resetting $f(U) \leftarrow f(U) - f(\emptyset)$ for all $U \subseteq S$). With the greedy algorithm introduced in Section 17.3, we can optimize over EP_f in polynomial time (Theorem 17.3). So the separation problem for EP_f is solvable in polynomial time, hence also the separation problem for $P = EP_f \cap \{x \mid x \leq \mathbf{0}\}$, and therefore also the optimization problem for P.

Fact 17.1. *There is a polynomial-time algorithm to separate over P, and hence to optimize over P.*

Claim 17.5.1. *If $f(\emptyset) = 0$, $\max\{x(S) \mid x \in P\} = \min_{U \subseteq S} f(U)$, where $P = EP_f \cap \{x \mid x \leq \mathbf{0}\}$.*

Proof. Define $g = f|\mathbf{0}$, and then we have $g(S) = \min_{U \subseteq S} f(U)$. Since g is submodular (from Claim 17.4.1) and $P = EP_g$ (from Claim 17.4.2), thus from Claim 17.3.3, $g(S) = \max\{x(S) \mid x \in P\}$. Therefore, we have $\max\{x(S) \mid x \in P\} = \min_{U \subseteq S} f(U)$. ■

Fact 17.1 and Claim 17.5.1 imply that we can compute the value of $\min_{U \subseteq S} f(U)$ in polynomial time. We still need an algorithm to find $U^* \subseteq S$ s.t. $f(U^*) = \min_{U \subseteq S} f(U)$.

Theorem 17.5. *There is a polynomial-time algorithm to minimize a submodular function f given by a value oracle.*

Proof. To complete the proof, we present an algorithm to find $U^* \subseteq S$ such that $f(U^*) = \min_{U \subseteq S} f(U)$. Let $\alpha = \min_{U \subseteq S} f(U)$ which can be computed as described above.

- If $f(S) = \alpha$ then output S
- Else find an element $s \in S$ such that the minimum value of f over all subsets of $S \setminus \{s\}$ is equal to α, which implies that there exists an $U^* \subseteq S$ with $f(U^*) = \alpha$ and $s \notin U^*$. This can be done by trying each possible s and checking for the minimum value of f on a subset of $S - s$. Recurse on $S - s$ and f restricted to $S - s$.

It is easy to see that the algorithm finds a minimizer in polynomial number of calls to an algorithm to find the minimum value. ■

Remark 17.3. The Ellipsoid based method can be shown to run in strongly polynomial time given a value oracle for f.

Combinatorial algorithms: A question of interest is whether there is a polynomial time "combinatorial" algorithm for this problem. Although there is no clear-cut and formal definition of a combinatorial algorithm, typically it is an algorithm whose operations have some combinatorial meaning in the underlying structure of the problem. Cunningham [**Cunningham**] gave a pseudo-polynomial time algorithm for this problem in 1985. It is only in 2000 that Schrijver [**Schrijver**] and independently Iwata, Fleischer and Fujishige gave polynomial time combinatorial algorithms for SFM. There have been several papers that followed these two; we mention the algorithm(s) of Iwata and Orlin [**IwataO**] that have perhaps the shortest proofs. All the algorithms follow the basic outline of Cunningham's approach which was orginally developed by him for the special case of SFM that arises in the separation oracle for the matroid polytope.

See articles on this subject by Fleischer [25], McCormick [50] and Toshev [63] in addition to the details in Schrijver's book [57].

17.6 Submodularity on Restricted Families of Sets

So far we have seen submodular functions on a ground set S. That is $f : 2^S \to R$ and $\forall A, B \subseteq S$,

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

In several applications, one needs to work with restricted families of subsets. Given a finite set S, a family of sets $\mathcal{C} \subseteq 2^S$ is

- a *lattice family* if $\forall A, B \in \mathcal{C}$, $A \cap B \in \mathcal{C}$ and $A \cup B \in \mathcal{C}$.
- an *intersecting family* if $\forall A, B \in \mathcal{C}$ and $A \cap B \neq \emptyset$, we have $A \cap B \in \mathcal{C}$ and $A \cup B \in \mathcal{C}$.

- a *crossing family* if $A, B \in \mathcal{C}$ and $A \cap B \neq \emptyset$ and $A \cup B \neq S$, we have $A \cap B \in \mathcal{C}$ and $A \cup B \in \mathcal{C}$.

For each of the above families, a function f is submodular on the family if

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

whenever $A \cap B$, $A \cup B$ are guaranteed to be in family for A, B. Function f is called intersection submodular and crossing submodular if $\mathcal{C}$ is intersecting and crossing family respectively.

We give some examples of interesting families that arise from directed graphs. Let $D = (V, A)$ be a directed graph.

Example 17.2. $\mathcal{C} = 2^V \setminus \{\emptyset, V\}$ is a crossing family.

Example 17.3. Fix $s, t \in V$, $\mathcal{C} = \{U \mid s \in U, t \notin U\}$ is lattice, intersecting, and crossing family.

Example 17.4. $\mathcal{C} = \{U \subset V \mid U$ induces a directed cut i.e. $\delta^+(U) = \emptyset$ and $\emptyset \subset U \subset V\}$ is a crossing family.

For the above example, we sketch the proof that $\mathcal{C}$ is a crossing family. If $A, B \in \mathcal{C}$ and $A \cap B \neq \emptyset$ and $A \cup B \neq V$, then by submodularity of δ^+, $|\delta^+(A \cup B)| + |\delta^+(A \cap B)| \leq |\delta^+(A)| + |\delta^+(B)|$. Therefore we have $\delta^+(A \cup B) = \emptyset$ and $\delta^+(A \cap B) = \emptyset$ and more over $A \cap B$ and $A \cup B$ are non-empty. Hence they both belong to $\mathcal{C}$ as desired.

Various polyhedra associates with submodular functions and the above special families are known to be well-behaved.

For lattice families the system

$$x(U) \leq f(U), U \in \mathcal{C}$$

is box-TDI. Also, the following system is also box-TDI

$$\begin{aligned} x(U) &\leq f_1(U), U \in \mathcal{C}_1 \\ x(U) &\leq f_2(U), U \in \mathcal{C}_2 \end{aligned}$$

where $\mathcal{C}_1$ and $\mathcal{C}_2$ are lattice families and f_1 and f_2 are submodular on $\mathcal{C}_\infty$ and $\mathcal{C}_2$ respectively. The above facts also hold for intersecting families and intersecting submodular functions.

For crossing family $\mathcal{C}$, the system

$$x(U) \leq f(U)$$

is not necessarily TDI. However, the system

$$\begin{aligned} x(U) &\leq f(U), U \in \mathcal{C} \\ x(S) &= k \end{aligned}$$

where $k \in R$ is box-TDI. Also, the system

$$\begin{aligned} x(U) &\leq f_1(U), U \in \mathcal{C}_1 \\ x(U) &\leq f_2(U), U \in \mathcal{C}_2 \\ x(S) &= k \end{aligned}$$

is box-TDI for crossing families $\mathcal{C}_1$ and $\mathcal{C}_2$ with f_1 and f_2 crossing supermodular on $\mathcal{C}_1$ and $\mathcal{C}_2$ respectively.

Although the polyhedra are well-behaved, the separation problem for them is not easy since one needs to solve submodular function minimization over a restricted family $\mathcal{C}$. It does not suffice to have a value oracle for f on sets in $\mathcal{C}$; one needs additional information on the representation of $\mathcal{C}$. We refer the reader to [57] for more details.

Chapter 18

Continuous Extensions of Submodular Set Functions

Let $f : 2^S \to \mathbb{R}$ be a submodular set function. We discuss a connection between submodular functions and convexity that was shown by Lovász [48]. We first discuss the notion of a continuous extension of set function, and a generic way to obtain a convex and concave extension for any set function.

Continuous extensions: Let $f : 2^S \to \mathbb{R}$ be real-valued set function. Let $n = |S|$. Without loss of generality we can assume $S = \{1, 2, \ldots, n\}$ and interpret f as providing values to the n-dimensional boolean hypercube $\{0,1\}^n$. A function $g : [0,1]^N \to \mathbb{R}$ is an *extension* of f from $\{0,1\}^N$ to $[0,1]^N$ iff for each set $S \subseteq N$, $f(S) = g(1_S)$. Continuous extensions provide several useful tools to understand set functions. A natural way to define an extension g is to define it via an interpolation scheme. That is, a fractional point $x \in [0,1]^N$ is expressed as a convex combination $x = \sum_S \lambda_S 1_S$, and then $g(x)$ is set to $\sum_S \lambda_S f(S)$. The interpolation for x can be usefully interpreted as choosing a probability distribution $D(x)$ on 2^S such that the marginal values of the distribution are precisely given by x; the value of $g(x)$ is the expected value of $f(R)$ where R is a random set drawn from $D(x)$. What are some desiderata for a continuous extension?

- g should be continuous
- g should inherit some structural properties of f
- For computational purposes, an extension g should have the property that, given x, $g(x)$ can be evaluated efficiently by having access to a value oracle for f.

- For optimization purposes g should be convex or concave depending on whether we wish to minimize g or maximize g.

18.1 The convex and concave closure

For optimization purposes it is helpful to consider extensions of a set function f where the resulting extension is convex (or concave). A canonical construction called the convex closure f^- is obtained from f via the following definition:

$$f^-(x) = \min\{\sum_{A \subseteq S} \alpha_A f(A) : \sum_A \alpha_A = 1, \alpha_A \geq 0 \quad \& \quad \forall j; \sum_{A: j \in A} \alpha_A = x_j\}.$$

We observe that $f^-(x)$ is the optimum solution to a linear program in the variables $\alpha_S, S \subseteq N$. The linear program, for a given x, chooses the distribution $D(x)$ on 2^N that minimizes the expected value of the function f. We use $D^-(x)$ to denote this distribution.

Lemma 18.1. *The function f^- is convex over $[0,1]^N$.*

Proof. Consider x, x' and let $y = \lambda x + (1-\lambda)x'$ where $\lambda \in (0,1)$. Let α, α' be optimum solutions to the linear program defining f^- at x and x' respectively. We observe that $\beta = \lambda\alpha + (1-\lambda)\alpha'$ is a feasible solution to the linear program for $f^-(y)$ whose value is $\lambda f(x) + (1-\lambda)f(x')$. Since the value of $f(y)$ can only be lower, $f(y) \leq \lambda f(x) + (1-\lambda)f(x')$, establishing convexity of f^-. ■

The concave closure f^+ of f is defined very similarly, the only difference is that we maximize over all distributions whose marginals equal x. Formally:

$$f^+(x) = \max\{\sum_{A \subseteq S} \alpha_A f(A) : \sum_A \alpha_A = 1, \alpha_A \geq 0 \quad \& \quad \forall j; \sum_{A: j \in A} \alpha_A = x_j\}.$$

Lemma 18.2. *The function f^+ is concave over $[0,1]^N$.*

One can find a set S minimizing f by minimizing f^- over $[0,1]^N$. Similarly, for finding a set S maximizing f, it suffices to maximize f^+ over $[0,1]^N$. This can be seen from the following lemma whose proof we leave as an exercise.

Lemma 18.3. *Let x be a minimizer of f^- over $[0,1]^N$. Then for any set A in the support of $D^-(x)$ we have $f(A) = f^-(x)$. Similarly, if x is a maximizer of f^+ over $[0,1]^N$ then for any set A in the support of $D^+(x)$ we have $f(S) = f^+(x)$.*

Note that the above holds for any set function f. Thus, unconstrained minimization of f can be reduced to minimizing the continuous convex function f^-. Similarly, maximizing f can be reduced to maximizing the concave function f^+. The catch is that we need (at least) the ability to efficiently evaluate the extension at a given x assuming oracle access to f. Interestingly, when f is a submodular function, the extension f^- can be evaluated in polynomial time for any given x. We will see a proof of this via the equivalence of f^- and the Lovász-extension. This, perhaps, is a high-level explanation as to why there is a polynomial-time algorithm for unconstrained submodular set function minimization. In contrast, there are explicit non-negative submodular functions f for which evaluating f^+ is NP-Hard. This is not surprising given that unconstrained submodular set function maximization is NP-Hard even for non-negative functions.

18.2 The Lovász extension and convexity for submodular set functions

The Lovász-extension $\hat{f}$ of a set function f is defined as follows. Let $x \in [0,1]^S$. Let $i_1, i_2, \ldots, i_n$ be a permutation of $\{1, 2, \ldots, n\}$ such that $x_{i_1} \geq x_{i_2} \geq \ldots \geq x_{i_n}$. For ease of notation we define $x_{i_{n+1}}$ to be 0. For $1 \leq j \leq n$ let $A_j = \{i_1, i_2, \ldots, i_j\}$. Thus, $A_1 = \{i_1\}$, $A_2 = \{i_1, i_2\}$, and $A_n = \{i_1, i_2, \ldots, i_n\}$. We let $A_0 = \emptyset$. Then,

$$\hat{f}(x) = (1 - x_{i_1}) f(\emptyset) + \sum_{j=1}^{n} (x_{i_j} - x_{i_{j+1}}) f(A_j) \tag{18.1}$$

As an example, if $S = \{1, 2, 3, 4, 5\}$ and $x = (0.75, 0.3, 0.2, 0.3, 0)$ then

$$\hat{f}(x) = 0.25 \cdot f(\emptyset) + 0.45 \cdot f(\{1\}) + 0.1 \cdot f(\{1, 2, 4\}) + 0.2 \cdot f(\{1, 2, 3, 4, 5\})$$

Thus, the distribution $D_L(x)$ that defines $\hat{f}(x)$ has as its support the sets $\emptyset, A_1, \ldots, A_n$ and these sets form a chain.

One can rewrite the expression in (18.1) as follows:

$$\hat{f}(x) = \sum_{j=1}^{n} x_{i_j} (f(A_j) - f(A_{j-1}). \tag{18.2}$$

One can see that the preceding form implies the following via the analysis of the Greedy algorithm for optimizing of the a polymatroid.

Lemma 18.4. *If f is submodular then $\hat{f}(x) = \max\{xy \mid y \in EP_f\}$.*

Recall that we saw that if f is submodular then $f(U) = \max\{y(U) \mid y \in EP_f\}$. Thus the interpretation of the extension via (18.2) is natural generalization of this.

Another useful and important view of the extension is via a probabilistic definition.

$$\hat{f}(x) = \mathbf{E}_{\theta\in[0,1]}\left[f(x^\theta)\right] = \int_0^1 f(x^\theta)d\theta \tag{18.3}$$

where $x^\theta \in \{0,1\}^n$ for a given vector $x \in [0,1]^n$ is defined as: $x_i^\theta = 1$ if $x_i \geq \theta$ and $x_i^\theta = 0$ otherwise. Here θ is chosen uniformly at random from the interval $[0,1]$. This randomized interpretation is useful when using the extension in constrained optimization and approximation algorithms — see [].

Exercise 18.1. Prove that the definition of the extension via expectation is equivalent to the first definition.

It is easy to see that for any x and any set function f, $f^-(x) \leq \hat{f}(x)$ since $f^-(x)$ finds the best distribution to minimize the expected value while the distribution $D_L(x)$ is a specific distribution that is in fact oblivious to the underlying function. For submodular functions f^- and $\hat{f}$ coincide. We first give a direct proof of the following.

Theorem 18.1 (Lovász)**.** *A set function $f : 2^S \to \mathbb{R}$ with $f(\emptyset) = 0$ is submodular iff $\hat{f}$ is convex.*

Proof. Suppose f is submodular. Let $x, x' \in [0,1]^n$ and $t \in [0,1]$ and let $z = tx + (1-t)x'$. To show that $\hat{f}$ is convex we need to show that $\hat{f}(z) \leq \hat{f}(tx) + \hat{f}((1-t)x')$. We use Lemma 18.4. Let $y^* \in EP_f$ be such that $\hat{f}(z) = z \cdot y^* = tx \cdot y^* + (1-t)x' \cdot y^*$. Then $\hat{f}(tx) \geq tx \cdot y^*$ and $\hat{f}((1-t)x') \geq (1-t)x' \cdot y^*$ (why?), and we have the desired claim.

Now suppose $\hat{f}$ is convex. Let $A, B \subseteq S$. From the definition of $\hat{f}$ we note that $\hat{f}((\chi(A) + \chi(B))/2) = \hat{f}(\chi(A \cup B)/2) + \hat{f}(\chi(A \cap B)/2)$ (the only reason to divide by 2 is to ensure that we stay in $[0,1]^n$). On the other hand, by convexity of $\hat{f}$, $\hat{f}((\chi(A) + \chi(B))/2) \leq \hat{f}(\chi(A)/2) + \hat{f}(\chi(B)/2)$. Putting together these two facts, we have $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$, and hence f is submodular. ■

Corollary 18.2. *If f is submodular then $\min_{U\subseteq S} f(S) = \min_{x\in[0,1]^n} \hat{f}(x)$.*

Proof. Clearly $\min_{x\in[0,1]^n} \hat{f}(x) \leq \min_{U\subseteq S} f(S)$. To see the converse, let $x^* \in [0,1]^n$ achieve the minimum of $\min_{x\in[0,1]^n} \hat{f}(x)$. Then one of the sets in the convex combination of x^* in the definition of the extension achieves a value equal to $\hat{f}(x^*)$. ■

The above shows that submodular function minimization can be reduced to convex optimization problem in a natural fashion. One advantage of an extension as above is that one can use it as a relaxation in optimization problems involving submodular functions and additional constraints. For example we may want to solve $\min_{U \subseteq S} f(S)$ subject to U satisfying some additional constraints that could perhaps be modeled as $x(S) \in P$ for some convex set P. Then we could solve $\min\{\hat{f}(x) \mid x \in P\}$ as a relaxation and round the solution in some fashion. There are several examples of this in the literature.

Theorem 18.3. *For a submodular set function f, the Lovász-extension $\hat{f}$ and the convex closure f^- coincide.*

We briefly sketch two proofs of the equivalence of $\hat{f}$ and f^- for submodular f. Fix a point x and let $D^-(x)$ be the distribution corresponding to the value $f^-(x)$. We claim that the support of $D^-(x)$ can be chosen to be a chain. To see this, suppose A, B are in the support and neither $A \subset B$ nor $B \subset A$. Since f is submodular we have $f(A) + f(B) \geq f(A \cap B) + f(A \cup B)$ and hence we can uncross and replace A, B by $A \cap B$ and $A \cup B$ so that the expected value does not decrease. Repeated application of this uncrossing operation gives the desired claim. There is a unique chain distribution whose marginal values are x which is precisely the one that defines $\hat{f}(x)$.

Here is another proof. We already have $f^-(x) \leq \hat{f}(x)$. We will show the reverse inequality which will prove that the two quantities are equal. Let R be a random set chosen according to the distribution $D^-(x)$; we have $f^-(x) = \mathbf{E}[f(R)]$. As in the definition of the Lovász-extension, let $i_1, i_2, \ldots, i_n$ be a permutation of $\{1, 2, \ldots, n\}$ such that $x_{i_1} \geq x_{i_2} \geq \ldots \geq x_{i_n}$ and let $A_j = \{i_1, \ldots, i_j\}$. Let Y_j be

the indicator variable for i_j to be present in R; we have $\mathbf{E}[Y_j] = x_{i_j}$.

$$\begin{aligned}
\mathbf{E}[f(R)] &= \mathbf{E}\left[f(\emptyset) + \sum_{j=1}^{n} Y_j \cdot f_{i_j}(A_{j-1} \cap R)\right] \\
&\geq \mathbf{E}\left[f(\emptyset) + \sum_{j=1}^{n} Y_j \cdot f_{i_j}(A_{j-1})\right] \\
&= f(\emptyset) + \sum_{j=1}^{n} \mathbf{E}[Y_j] \cdot f_{i_j}(A_{j-1}) \\
&= f(\emptyset) + \sum_{j=1}^{n} x_{i_j}(f(A_j) - f(A_{j-1})) \\
&= (1 - x_{i_1})f(\emptyset) + \sum_{j=1}^{n}(x_{i_j} - x_{i_{j+1}})f(A_j) \\
&= \hat{f}(x).
\end{aligned}$$

Submodularity is used in the second step.

18.3 Submodular set function maximization and the Multilinear extension

There are several applications for submodular set function maximization in the unconstrained and in the constrained settings. However, as we saw previously, submodular set function maximization even in the unconstrained setting is NP-Hard; a canonical example is to find the maximum cut in a given graph. Therefore the focus has been on polynomial-time approximation algorithms. Approximation algorithms based on greedy and local search methods were dominant for these problems based on the early work of Cornuejols, Fisher, Nemhauser, Wolsey and others. An approach based on continuous extensions was introduced in [9]. Since maximization is NP-Hard we cannot hope to use a concave extesion such as f^+, at least directly. The key idea was to introduce the multilinear extension.

The multilinear extension F for a set function f can be defined algebraically as:

$$F(x) = \sum_{A \subseteq N} f(A) \prod_{i \in A} x_i \prod_{j \in S \setminus A} (1 - x_j).$$

Alternatively,

$$F(x) = \mathbf{E}_{R \sim x}[f(R)]$$

where R is a random set obtained by sampling each $i \in S$ independently with probability x_i. In other words the distribution $D_F(x)$ defining $F(x)$ is the product distribution.

We observe that $F(x)$ is defined via a formula that is of exponential size, hence it is not feasible to directly evaluate it. However, one can use random sampling to estimate $F(x)$ arbitrarily well — take many random sets $R_1, R_2, \ldots, R_h$ where each R_i is sampled from x and estimate $F(x)$ as $\frac{1}{h}\sum_{\ell=1}^{h} f(R_i)$. Via concentration properties and related standard tools, it is possible to obtain an accurate estimate of $F(x)$ in most algorithmic applications of interest. One can also estimate the gradient $\nabla F(x)$ in this way.

The multilinear extension F of a submodular set function is neither convex nor concave. Despite this, the extension has useful mathematical properties and these can be exploited algorithmically. Here we list a few properties and state a few theorems, and refer the reader to other articles for more details.

Structural properties of F: The algebraic formulation allows us to prove the following properties.

Lemma 18.5. *Let $F : [0,1]^S \to \mathbb{R}$ be the multilinear extension of a set function $f : 2^S \to \mathbb{R}$.*

- *If f is submodular, then $\frac{\partial^2 F}{\partial x_i \partial x_j} \leq 0$ for all $i, j \in S$, everywhere in $[0,1]^S$.*
- *If f is non-decreasing, then $\frac{\partial F}{\partial x_i} \geq 0$ for all $i \in S$, everywhere in $[0,1]^S$.*

Corollary 18.4. *Let $F : [0,1]^S \to \mathbb{R}$ be the multilinear extension of a set function $f : 2^N \to \mathbb{R}$. Then*

- *If f is non-decreasing, then F is non-decreasing along any line of direction $d \geq 0$.*
- *If f is submodular, then F is concave along any line of direction $d \geq 0$.*
- *If f is submodular, then F is convex along any line of direction $d = e_i - e_j$ for $i, j \in S$.*

Algorithmic aspects: We note that the structural properties of F did not rely on non-negativity. On the other hand most of the algorithmic results on submodular set function maximization are primarily for non-negative functions (in particular normalized non-negative functions). Since F is not concave we settle for relative approximation results. Consider a polytope $P \subseteq [0,1]^S$ which represents a relaxation of constraints. One would seek to solve the mathematical

programming relaxation $\max_{x \in P} F(x)$. We say that P is a *solvable* polytope if there is an efficient algorithm to do linear optimization over P, that is $\max wx, x \in P$ admits an efficient algorithm.

Theorem 18.5 ([8, 24]). *Let F be the mutilinear relaxation of a non-negative submodular set function $f : 2^S \to \mathbb{R}_+$ and let $P \subseteq [0,1]^S$ be a solvable polytope. Then there is an efficient randomized algorithm that outputs a $(1 - 1/e)$-approximation with high probability for the problem $\max_{x \in P} F(x)$ when f is monotone. For non-negative F there is $1/e$-approximation.*

The second aspect of using the multilinear relaxation is to *round* a fractional solution. One structural result specific to matroids is the following.

Theorem 18.6 ([8]). *Let F be the mutilinear relaxation of a submodular set function $f : 2^S \to \mathbb{R}$. Let $\mathcal{M} = (S, \mathcal{I})$ be a matroid and let P be the matroid independence polytope. Then, given any fractional point $x \in P$ there is an efficient randomzed algorithm that output an independent set $I \in \mathcal{I}$ such that $\mathbf{E}[f(I)] = F(x)$. In other words there is no loss with resepct to F in rounding the fractional solution in a matroid polytope.*

For other constraints a framework based on *contention resolution schemes* has been the main technique. We refer the reader to [6, 16].

Bibliographic Remarks: Lovasz's influential paper on the connection of convexity to submodularity is [48]. There have been several recent applications of the Lovasz-extension in mathematical programming based algorithms for problems involving submodular and supermodular functions — we refer the reader to some of these [11–14, 28, 35, 55, 56]. Dughmi's survey [22] discusses continuous extensions and most of this chapter is based on it. The mutilinear relaxation for submodular set function maximization was introduced in [9] and was inspired by previous work of Ageev and Sviridenko [2]. See [6] for a survey on approximation algorithms for recent progress on approximation algorithms for submodular set function maximization including several that are based on the multilinear extensions. Jan Vondrak's thesis [66] discusses and proves several properties of the multilinear relaxation that we mentioned.

Chapter 19

Two Theorems Related to Directed Graphs[1]

We describe two well known theorems in combinatorial optimization related to directed graphs. We prove the theorems using submodular flows later.

19.1 Nash-Williams Graph Orientation Theorem

Definition 19.1. *Let $G = (V, E)$ be an undirected graph. For $u, v \in V$, we denote by $\lambda_G(u, v)$ the edge-connectivity between u and v in G, that is, the maximum number of edge-disjoint paths between u and v. Similarly for a directed graph $D = (V, A)$, $\lambda_D(u, v)$ is the maximum number of arc-disjoint paths from u to v.*

Note that for an undirected graph G, $\lambda_G(u, v) = \lambda_G(v, u)$ but it may not be the case that $\lambda_D(u, v) = \lambda_D(v, u)$ in a directed graph D.

Definition 19.2. *G is k-edge-connected if $\lambda_G(u, v) \geq k \ \forall u, v \in V$. Similarly, D is k-arc-connected if $\lambda_D(u, v) \geq k \ \forall u, v \in V$.*

Proposition 19.1.1. *G is k-edge-connected iff $|\delta(S)| \geq k \ \forall S \subset V$. D is k-arc-connected iff $|\delta^+(S)| \geq k \ \forall S \subset V$.*

Proof. By Menger's theorem. ∎

Definition 19.3. *$D = (V, A)$ is an orientation of $G = (V, E)$ if D is obtained from G by orienting each edge $uv \in E$ as an arc (u, v) or (v, u).*

Theorem 19.4 (Robbins 1939)**.** *G can be oriented to obtain a strongly-connected directed graph iff G is 2-edge-connected.*

[1] Based on notes scribed by Zhenhui Li in 2010.

Proof. "⇒" Suppose $D = (V, A)$ is a strongly connected graph obtained as an orientation of $G = (V, E)$. Then, since $\forall S \subset V$, $|\delta^+_D(S)| \geq 1$ and $|\delta^-_D(S)| \geq 1$, we have $|\delta_G(S)| \geq 2$. Therefore, G is 2-edge-connected.

"⇐" Let G be a 2-edge-connected graph. Then G has an ear-decomposition. In other words, G is either a cycle C or G is obtained from a 2-edge-connected graph G' by adding an ear P (a path) connecting two not-necessarily distinct vertices $u, v \in V$.

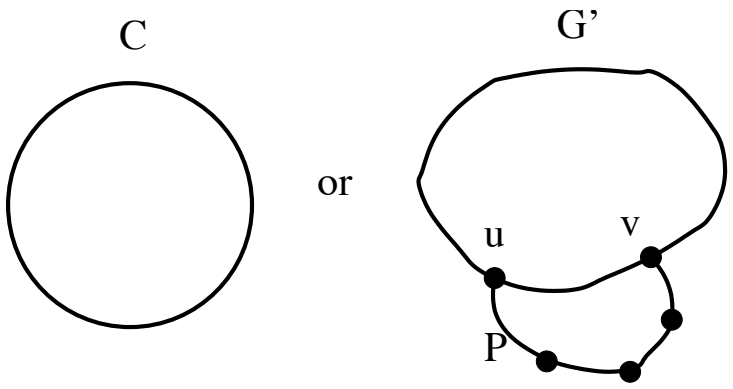


Figure 19.1: G is either a cycle C or is G' plus an ear P.

If $G = C$, orient it to obtain a directed cycle which is strongly-connected. Otherwise, inductively, G' has an orientation that is strongly-connected. Extend the orientation of G' to G by orienting P from u to v (or v to u). It is easy to check that this orientation results in strongly-connected graph. ■

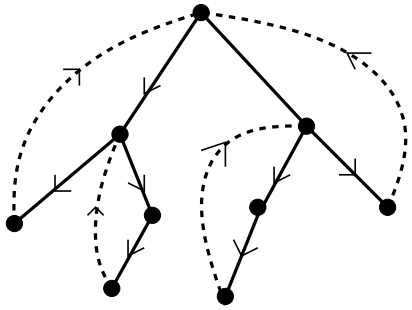
An alternative proof is as follows. Do a depth-first-search (DFS) of G starting at some node r. One obtains a DFS tree T. Orient all edges of T away from r to obtain an arborescence. Every other edge is a back-edge, that is if $uv \in E(G) \backslash E(T)$, then, either u is the ancestor of v in T or v is an ancestor of u in T. Orient uv from the descendant to the ancestor. We leave it as an exercise to argue that this is a strongly-connected orientation of G iff G is 2-edge-connected. Note that this is an easy linear time algorithm to obtain the orientation.

Nash-Williams proved the following generalization.

Theorem 19.5 (Nash-Williams)**.** *If G is $2k$-edge-connected, then it has an orientation that is k-arc-connected.*

In fact, he proved the following deep result, of which the above is a corollary.

Theorem 19.6 (Nash-Williams)**.** *G has an orientation D in which $\lambda_D(u, v) \geq \lfloor \lambda_G(u, v)/2 \rfloor$ for all $u, v \in V$.*



dashed edges are back edges

Figure 19.2: Orientation of a 2-edge-connected graph via a DFS tree.

The proof of the above theorem is difficult — see [57]. Frank showed that Theorem **??** can be derived from submodular flows and we will see this later.

19.2 Directed Cuts and Lucchesi-Younger Theorem

Definition 19.7. *Let $D = (V, A)$ be a directed graph. We say that $C \subset A$ is a directed cut if $\exists S \subset V$ such that $\delta^+(S) = \emptyset$ and $C = \delta^-(S)$.*

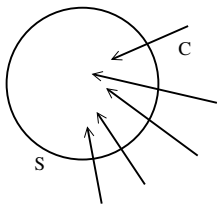


Figure 19.3: A directed cut $C = \delta^-(S)$.

If D has a directed cut then D is *not* strongly-connected.

Definition 19.8. *A* dijoin *(also called a directed cut cover) in $D = (V, A)$ is a set of arcs in A that intersect each directed cut of D.*

It is not difficult to see that the following are equivalent:

- $B \subseteq A$ is a dijoin.
- shrinking each arc in B results in a strongly-connected graph.
- adding all reverse arcs of B to D results in a strongly-connected graph.

Given $B \subseteq A$, it is therefore, easy to check if B is a dijoin; simply add the reverse arcs of B to D and check if the resulting digraph is strongly connected or not.

Definition 19.9. *A digraph D is weakly-connected if the underlying undirected graph is connected.*

Theorem 19.10 (Lucchesi-Younger)**.** *Let $D = (V, A)$ be a weakly-connected digraph. Then the minimum size of a dijoin is equal to the maximum number of disjoint directed cuts.*

A dijoin intersects every directed cut so its size is at least the the maximum number of disjoint directe cuts. The above theorem is yet another example of a min-max result. We will prove this later using submodular flows. One can derive easily a weighted version of the theorem.

Corollary 19.11. *Let $D = (V, A)$ be a digraph with $\ell : A \to \mathbb{Z}_+$. Then the minimum length of a dijoin is equal to the maximum number of directed cuts such that each arc a is in at most $\ell(a)$ of them (in other words a maximum packing of directed cuts in ℓ).*

Proof. If $\ell(a) = 0$, contract it. Otherwise replace a by a path of length $\ell(a)$. Now apply the Lucchesi-Younger theorem to the modified graph. ■

As one expects, a min-max result also leads to a polynomial time algorithm to compute a minimum weight dijoin and a maximum packing of directed cuts.

We describe an implication of Lucchesi-Younger theorem.

Definition 19.12. *Given a directed graph $D = (V, A)$, $A' \subseteq A$ is called a feedback arc set if $D[A \setminus A']$ is acyclic, that is, A' intersects each directed cycle of D.*

Computing a minimum cardinality feedback arc set is NP-hard. Now suppose D is a plane directed graph (i.e., a directed graph that is embedded in the plane). Then one defines its dual graph D^* as follows. For each arc (w, x) of D, we have a dual arc $(y, z) \in D^*$ that crosses (w, x) from "left" to "right". See example below.

Proposition 19.2.1. *The directed cycles of D correspond to directed cuts in D^* and vice versa.*

Thus, a feedback arc set of D corresponds to a dijoin in D^*. Via Lucchesi-Younger theorem, we have the following corollary.

Corollary 19.13. *For a planar directed graph, the minimum size of a feedback arc set is equal to the maximum number of arc-disjoint directed cycles.*

Using the algorithm to compute a minimum weight dijoin, we can compute a minimum weight feedback arc set of a planar digraph in polynomial time.

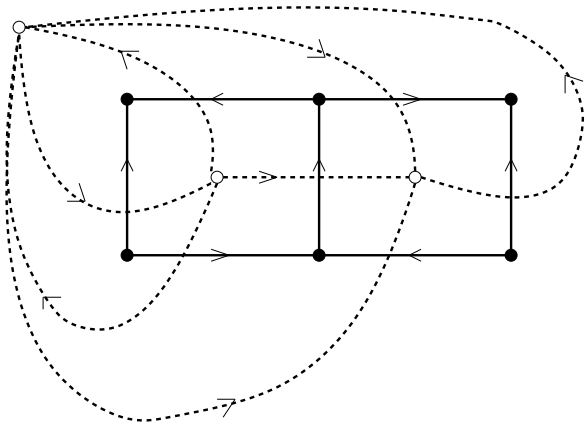


Figure 19.4: A planar digraph and its dual.

Packing dijoins: Woodall conjectured the following, which is still open.

Conjecture 19.14 (Woodall)**.** *For every directed graph, the minimum size of a directed cut equals to the maximum number of disjoint dijoins.*

Some special cases of the preceding conjecture have been solved but the conjecture is still open even in planar digraphs. See [1, 57, 60] for known results, progress and some relaxations of the conjecture that are also open.

Chapter 20

Polymatroid Intersection[1]

Recall the definition of total dual integrality of a system of inequalities.

Definition 20.1. *A rational system of inequalities $Ax \leq b$ is TDI if for all integral c, $\min\{yb \mid y \geq 0, yA = c\}$ is attained by an integral vector y^* whenever the optimum exists and is finite.*

Definition 20.2. *A rational system of inequalities $Ax \leq b$ is box-TDI if the system $d \leq x \leq c$, $Ax \leq b$ is TDI for each $d, c \in \mathcal{R}^n$.*

In particular, we have the following. If $Ax \leq b$ is box-TDI, then the polyhedron $\{x \mid Ax \leq b, d \leq \ell \leq u\}$ is an integer polyhedron whenever b, ℓ, u are integer vectors.

Recall that if $f : 2^S \rightarrow \mathcal{R}$ is a submodular function, EP_f is the extended polymatroid defined as

$$\{x \in \mathcal{R}^S | x(U) \leq f(U), U \subseteq S\}$$

We showed that the system of inequalities $x(U) \leq f(U), U \subseteq S$ is TDI. In fact, one can show that the system is also box-TDI. Polymatroids generalize matroids. One can also consider polymatroid intersection which generalizes matroid intersection.

Let f_1, f_2 be two submodular functions on S. Then the polyhedron $EP_{f_1} \cap EP_{f_2}$ described by

$$\begin{aligned} x(U) &\leq f_1(U) \qquad U \subseteq S \\ x(U) &\leq f_2(U) \qquad U \subseteq S \end{aligned}$$

is an integer polyhedron whenever f_1 and f_2 are integer valued. We sketch a proof of the following theorem.

[1]Based on notes scribed by Zhenhui Li in 2010.

Theorem 20.3 (Edmonds)**.** *Let f_1, f_2 be two submodular set functions on the ground set S. The system of inequalities*

$$x(U) \leq f_1(U) \qquad U \subseteq S$$
$$x(U) \leq f_2(U) \qquad U \subseteq S$$

is box-TDI.

Proof. (Sketch) The proof is similar to that of matroid intersection. Consider primal-dual pair below

$$\max wx$$
$$x(U) \leq f_1(U) \qquad U \subseteq S$$
$$x(U) \leq f_2(U) \qquad U \subseteq S$$
$$\ell \leq x \leq u$$

$$\min \textstyle\sum_{U \subseteq S}(f_1(U)y_1(U) + f_2(U)y_2(U)) + \sum_{a \in S} u(a)z_1(a) - \sum_{a \in S} \ell(a)z_2(a)$$
$$\textstyle\sum_{a \in U}(y_1(U) + y_2(U)) + z_1(a) - z_2(a) = w(a), a \in S$$
$$y \geq 0, z_1, z_2 \geq 0$$

Claim 20.0.1. *There exists an optimal dual solution such that $\mathcal{F}_1 = \{U \mid y_1(U) > 0\}$ and $\mathcal{F}_2 = \{U \mid y_2(U) > 0\}$ are chains.*

The proof of the above claim is similar to that in matroid intersection. Consider $\mathcal{F}_1 = \{U \mid y_1(U) > 0\}$. If it is not a chain, there exist $A, B \in \mathcal{F}_1$ such that $A \not\subset B$ and $B \not\subset A$. We change y_1 by adding ϵ to $y_1(A \cup B)$ and $y_1(A \cap B)$ and subtracting ϵ from $y_1(A)$ and $y_1(B)$. One observes that the feasibility of the solution is maintained and that the objective function can only decrease since f_1 is submodular. Thus, we can uncross repeatedly to ensure that $\mathcal{F}_1$ is a chain, similarly $\mathcal{F}_2$.

Let y_1, y_2, z_1, z_2 be an optimal dual solution such that $\mathcal{F}_1$ and $\mathcal{F}_2$ are chains. Consider $\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2$ and the $S \times \mathcal{F}$ incidence matrix M. As we saw earlier in the proof for matroid intersection, M is TUM. We then have y_1, y_2, z_1, z_2 are determined by a system $\begin{bmatrix} y_1 & y_2 & z_1 & z_2 \end{bmatrix}\begin{bmatrix} M & I & -I \end{bmatrix} = w$, where w is integer and M is TUM. Since $\begin{bmatrix} M & I & -I \end{bmatrix}$ is TUM, there exists integer optimum solution. ■

Note that, one can separate over $EP_{f_1} \cap EP_{f_2}$ via submodular function minimization and hence one can optimize $EP_{f_1} \cap EP_{f_2}$ in polynomial time via the ellipsoid method. Strongly polynomial time algorithm can also be derived. See [57] for details.

Chapter 21

Submodular Flows and Applications[1]

Network flows are a fundamental object and tool in combinatorial optimization. We have also seen submodular functions and their role in matroids, matroid intersection, polymatroids and polymatroid intersection. Edmonds and Giles developed the framework of submodular flows to find a common generalization of network flow and polymatroid intersection; they were inspired by the Lucceshi-Younger theorem. A seemingly different model was independently studied by Hassin [33] and Lawler and Martel [41]. These models can be shown to be equivalent (see [57]). However, depending on the application, one or the other model is more convenient and intuitive. We mainly discuss the Edmonds-Giles model in this chapter and briefly discuss the Hassin and Lawler-Martel model in a later section.

Definition 21.1 (Crossing Family)**.** *Let $D = (V, A)$ be a directed graph and let $\mathcal{C} \subseteq 2^V$ be a family of subsets of V. $\mathcal{C}$ is called a **crossing family** if: $A, B \in \mathcal{C}$, $A \cap B \neq \emptyset$, $A \cup B \neq V \Rightarrow A \cap B \in \mathcal{C}$ and $A \cup B \in \mathcal{C}$.*

Definition 21.2 (Crossing Submodular)**.** *Let $\mathcal{C}$ be a crossing family. A function $f : \mathcal{C} \to \mathbb{R}$ is called **crossing submodular** on $\mathcal{C}$ if it satisfies: $A, B \in \mathcal{C}$, $A \cap B \neq \emptyset$, $A \cup B \neq V \Rightarrow f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$.*

Definition 21.3 (Submodular Flow)**.** *Let $D = (V, A)$ be a digraph, $\mathcal{C}$ be a crossing family, and f be a crossing submodular function on $\mathcal{C}$. A vector $x \in \mathbb{R}^A$ is called a **submodular flow** if*

$$x(\delta^-(U)) - x(\delta^+(U)) \leq f(U), \qquad \forall U \in \mathcal{C} \tag{21.1}$$

[1]Based on notes scribed by Peiziang Zhao from 2010.

Theorem 21.4 (Edmonds-Giles, **1977**). *The system of inequalities shown in Equation 21.1 is box-TDI where $\mathscr{C}$ is a crossing family on V and f is crossing submodular on $\mathscr{C}$.*

Proof. We consider the primal-dual pair of LPs below where $w : A \to \mathbb{Z}_+$ and ℓ, u are integer vectors.

$$\max \sum wx$$
$$\text{s.t.} \qquad x(\delta^-(U)) - x(\delta^+(U)) \leq f(U) \qquad U \in \mathscr{C}$$
$$\ell \leq x \leq u$$

and

$$\min \sum_{U \in \mathscr{C}} f(U)y(U) + \sum_{a \in A} u(a)z_1(a) - \sum_{a \in A} \ell(a)z_2(a)$$
$$\text{s.t.} \sum_{U : U \in \mathscr{C}, a \in \delta^-(U)} y(U) - \sum_{U : U \in \mathscr{C}, a \in \delta^+(U)} y(U) + z_1(a) - z_2(a) = w(a) \qquad a \in A$$
$$y, z_1, z_2 \geq 0$$

A family of sets $\mathcal{F} \subseteq 2^V$ is cross-free if for all $A, B \in \mathcal{F}$ the following holds:

$$A \subseteq B \text{ or } B \subseteq A \text{ or } A \cap B = \emptyset \text{ or } A \cup B = V.$$

Claim 21.0.1. *There exists an optimum solution y, z_1, z_2 such that $\mathcal{F} = \{U \in \mathscr{C} \mid y(U) > 0\}$ is cross-free.*

Proof. Suppose $\mathcal{F}$ is not cross-free. Then let $A, B \in \mathcal{F}$, such that $y(A) > 0$ and $y(B) > 0$ and $A \cap B \neq \emptyset$ and $A \cup B \neq V$. Then add $\epsilon > 0$ to $y(A \cup B)$, $y(A \cap B)$ and subtract $\epsilon > 0$ from $y(A)$ and $y(B)$. By submodularity of f, the objective function increases or remains same. We claim that alterting y in this fashion maintains dual feasibility; we leave this as an exercise.

By repeated uncrossing we can make $\mathcal{F}$ cross-free. Formally one needs to consider a potential function. For example, among all optimal solutions pick one that minimizes

$$\sum_{U \in \mathscr{C}} y(U)|U||V \setminus U|$$

■

Theorem 21.5. *Let $\mathcal{F}$ be a cross-free family on 2^V. Let M be an $|A| \times |\mathcal{F}|$ matrix where*

$$M_{a,U} = \begin{cases} 1 & \text{if } a \in \delta^-(U) \\ -1 & \text{if } a \in \delta^+(U) \\ 0 & \text{otherwise} \end{cases}$$

Then M is TUM.

The proof of the above theorem proceeds by showing that M is a network matrix. See Schrijver Theorem 13.21 for details [57].

By the above one sees that the non-negative components of y, z_1, z_2 are determined by $[M, I, -I]$ and integer vector w where M is TUM. From this we infers that there exists an integer optimum solution to the dual. ■

Corollary 21.6. *The polyhedron P determined by*

$$x(\delta^-(U)) - x(\delta^+(U)) \leq f(U) \qquad U \in \mathscr{C}$$

$$l \leq x \leq u$$

is an integer polyhedron whenever f is integer valued and l, u are integer vectors.

One can show that optimality on P can be done in strongly polynomial time if one has a value oracle for f. This can be done via a reduction to polymatroid intersection. We refer to Schrijver, Chapter 60 for more details [57].

21.1 Applications

Submodular flows are a very general framework as they combine graphs and submodular functions. We describe some applications below.

21.1.1 Circulations

Given a directed graph $D = (V, A)$, $x : A \to \mathbb{R}$ is a circulation if

$$x(\delta^-(v)) - x(\delta^+(v)) = 0, \qquad \forall v \in V$$

This can be modeled as a special case of submodular flow by setting $\mathscr{C} = \{\{v\} \mid v \in V\}$ and $f = 0$. We get the inequalities

$$x(\delta^-(v)) - x(\delta^+(v)) \leq 0, \qquad v \in V.$$

One can check that the above inequalities imply that for any $\emptyset \subset U \subset V$ the inequality $x(\delta^-(U)) - x(\delta^+(U)) \leq 0$ holds by adding up the inequalities for each $v \in U$. Combining this with the inequality $x(\delta^-(V \setminus U)) - x(\delta^+(V \setminus U)) \leq 0$, we have $x(\delta^-(U)) - x(\delta^+(U)) = 0$ for all $\emptyset \subset U \subset V$, and in particular for each $v \in V$. The box-TDI result of submodular flow implies the basic results on circulations and flows including Hoffman's circulation theorem and the max-flow min-cut theorem.

21.1.2 Polymatroid Intersection

We saw earlier that the system

$$x(U) \leq f_1(U) \qquad U \subseteq S$$
$$x(U) \leq f_2(U) \qquad U \subseteq S$$

is box-TDI whenever f_1, f_2 are submodular functions on S. We can derive this from submodular flows as follows. Define S' and S'' are copies of S. Let $V = S' \biguplus S''$ and define $\mathscr{C} = \{U' \mid U \subseteq S\} \cup \{S' \cup U'' \mid U \subseteq S\}$, where U' and U'' denote the sets of copies of elements of U in S' and S''.

Claim 21.1.1. *$\mathscr{C}$ is a crossing family.*

Exercise 21.1. Prove Claim 21.1.1.

We further define $f : \mathscr{C} \to \mathbb{R}_+$ by

$$f(U') = f_1(U) \qquad U \subseteq S$$
$$f(V \backslash U'') = f_2(U) \qquad U \subseteq S$$
$$f(S') = \min\{f_1(S), f_2(S)\}$$

Claim 21.1.2. *f is crossing submodular on $\mathscr{C}$.*

Exercise 21.2. Prove Claim 21.1.2.

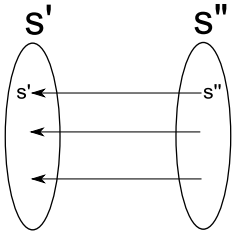


Figure 21.1: A Directed Graph Defined on S

Now define $G = (V, A)$ where $A = \{(s'', s') | s \in S\}$, as shown in Figure 21.1. The submodular flow polyhedron is

$$x(\delta^-(Z)) - x(\delta^+(Z)) \leq f(Z) \qquad Z \in \mathscr{C}$$

If $Z = U'$ where $U \subseteq S$, then we get $x(U) \leq f_1(U)$. And if $Z = V \backslash U''$, as shown in Figure 21.2, then we get $x(U) \leq f_2(U), U \subseteq S$. Thus, we recover the polymatroid intersection constraints. Since the submodular flow constraint inequalities are box-TDI, it implies that the polymatroid intersection constraints are also box-TDI.

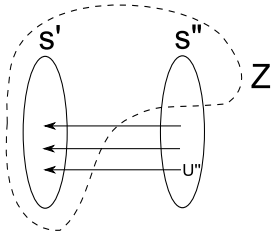


Figure 21.2: $Z = V \setminus U''$

21.1.3 Nash-Williams Graph Orientation Theorem

We discussed graph orientation in a previous chapter.

Definition 21.7. *$D = (V, A)$ is an orientation of $G = (V, E)$ if D is obtained from G by orienting each edge $uv \in E$ as an arc (u, v) or (v, u).*

Theorem 21.8 (Robbins 1939)**.** *G can be oriented to obtain a strongly-connected directed graph iff G is 2-edge-connected.*

Nash-Williams proved the following theorem.

Theorem 21.9 (Nash-Williams)**.** *G has an orientation D in which $\lambda_D(u, v) \geq \lfloor \lambda_G(u, v)/2 \rfloor$ for all $u, v \in V$.*

The proof of the preceding theorem is difficult — see [57]. An important corollary is the following theorem.

Theorem 21.10 (Nash-Williams)**.** *If G is $2k$-edge-connected, then it has an orientation that is k-arc-connected.*

Frank showed that Theorem 21.10 can be derived from submodular flows — we follow his argument. Consider an arbitrary orientation D of G. Now if D is k-arc-connected we are done. Otherwise we consider the problem of reversing the orientation of some arcs of D such that the resulting graph is k-arc-connected. We set it up as follows.

Let $D = (V, A)$, define a variable $x(a), a \in A$ where $x(a) = 1$ if we reverse the orientation of a. Otherwise $x(a) = 0$. For a set $U \subset V$ we want k arcs coming in after applying the switch of orientation, i.e., we want

$$x(\delta^-(U)) - x(\delta^+(U)) \leq |\delta^-(U)| - k \qquad \forall\, \emptyset \subset U \subset V$$

Note that $\mathcal{C} = \{U \mid \emptyset \subset U \subset V\}$ is a crossing family and $f(U) = |\delta^-_D(U)| - k$ is crossing submodular. Hence by Edmonds-Giles theorem, the polyhedron

determined by the inequalities

$$\begin{aligned} x(\delta^-(U)) - x(\delta^+(U)) &\leq |\delta^-(U)| - k \qquad \emptyset \subset U \subset V \\ x(a) &\in [0,1] \qquad a \in A \end{aligned}$$

is an integer polyhedron. Moreover, the polyhedron is non-empty since $x(a) = 1/2, \forall a \in A$ satisfies all the constraints. To see this, let $\emptyset \subset U \subset V$, and let $h = |\delta^-_D(U)|$ and $\ell = |\delta^+_D(U)|$, then we have $h + \ell \geq 2k$ since G is $2k$-edge-connected. Then by setting $x(a) = 1/2, \forall a \in A$, for U we need

$$\frac{h}{2} - \frac{\ell}{2} \leq h - k \Rightarrow \frac{h+l}{2} \geq k$$

which is true. Thus there is an integer vector x in the polyhedron for D if G is $2k$-edge-connected. By reversing the arcs $A' = \{a \mid x(a) = 1\}$ in D we obtain a k-arc-connected orientation of G.

21.1.4 Lucchesi-Younger theorem

Theorem 21.11 (Lucchesi-Younger)**.** *In any weakly-connected digraph, the size of the minimum cardinality dijoin equals the maximum number of disjoint directed cuts.*

Proof. Let $D = (V, A)$ be a directed graph and let $\mathscr{C} = \{U \mid \emptyset \subset U \subset V, |\delta^+(U)| = 0\}$, i.e., $U \in \mathscr{C}$ iff U induces a directed cut. We had seen that $\mathscr{C}$ is a crossing family. Let $f : \mathscr{C} \to \mathbb{R}$ be $f(U) = -1, \forall U \in \mathscr{C}$, clearly f is crossing submodular. Then by Edmonds-Giles theorem the following set of inequalities is TDI.

$$\begin{aligned} x(\delta^-(U)) - x(\delta^+(U)) &\leq -1 \qquad U \in \mathscr{C} \\ x &\leq 0 \end{aligned}$$

We note that $\delta^+(U) = \emptyset$ for each $U \in \mathscr{C}$. We can rewrite the above polyhedron as the one below by replacing $-x$ by x.

$$\begin{aligned} x(\delta^-(U)) &\geq 1 \qquad U \in \mathscr{C} \\ x &\geq 0 \end{aligned}$$

Note that the above is a “natural" LP relaxation for finding a set of arcs that cover all directed cuts. The above polyhedron is integral, and hence

$$\begin{aligned} \min \sum_{a \in A} x(a) & \\ x(\delta^-(U)) &\geq 1 \qquad U \in \mathscr{C} \\ x &\geq 0 \end{aligned}$$

gives the size of a minimum cardinality dijoin.

Consider the dual

$$\max \sum_{U \in \mathscr{C}} y(U)$$

$$\text{s.t.} \quad \sum_{U: a \in \delta^-(U), U \in \mathscr{C}} y(U) \leq 1 \qquad a \in A$$

$$y \geq 0$$

The dual is an integer polyhedron since the primal inequality system is TDI and the objective function is an integer vector. It is easy to see that the optimum value of the dual is a maximum packing of arc-disjoint directed cuts. Therefore by strong duality we obtain the Lucchesi-Younger theorem. ■

21.2 The polymatroidal network flow model

Here we discuss a different model of submodular flows that was suggested independently by Hassin and by Lawler and Martel. We mainly follow the Lawler-Martel model and notation. The original motivation for this model came from certain scheduling applications in the work of Martel. Here we describe a somewhat round about way based on the experience of this author who came to the polymatroid network flow model via some more recent applications [13].

Consider the standard network flow model with a directed graph $G = (V, A)$, source and sink nodes $s, t \in V$ and arc capacities $c : A \to \mathbb{R}_+$. Recall that a flow is a vector $x : A \to \mathbb{R}$ that satisfies two properties:

- flow conservation at each node $v \in V - \{s, t\}$: $x(\delta^-(v)) - x(\delta^+(v)) = 0$
- capacity constraints $0 \leq x(a) \leq c(a)$ for all $a \in A$.

In this model the capacities of each edge are independent. It is interesting to consider settings in which the capacity of arcs are jointly constrained. Let $f : A \to \mathbb{R}_+$ be a non-negative monotone set function over the arcs. Then we can ask for the following constraints on flow on the arcs:

$$x(B) \leq f(B) \quad \forall B \subseteq A.$$

Note that the standard capacity constraints are induced by the modular set function $f : 2^A \to \mathbb{R}_+$ where $f(B) = \sum_{a \in B} c(a)$. What kind of set functions can we consider? It is natural to consider monotone submodular set functions. The next question is whether we can solve the maximum flow problem under the constraint. Indeed we can solve the maximum flow LP under the capacity constraint $x(B) \leq f(B)$ since the separation oracle for this constraint can be

implemented by submodular function minimization. Although the max-flow in this very general setting can indeed by solved in polynomial time (assuming a value oracle access to f), the model is too general to allow for important properties that hold in the standard network flow model, namely the maxflow-mincut duality theorem, and integrality of flows when capacities are integral.

Remark 21.1. Note that the Edmonds-Giles model considers a submodular function f on the vertices which controls the total total capacity of the cuts induced by vertex subsets.

A local model: Instead of a global function f that jointly controls the capacity of all the arcs we will now focus on a local model that is more restriced in the joint capacity constraints that it imposes on the arcs. In a polymatroidal network, each node $v \in V$ has two associated polymatroids ρ_v^- and ρ_v^+ with ground sets as $\delta^-(v)$ and $\delta^+(v)$ respectively. These functions constrain the joint capacity on the edges incident to v as follows. If $S \subseteq \delta^-(v)$, then $\rho_v^-(S)$ upper-bounds the total capacity of the edges in S; similarly, if $S \subseteq \delta^+(v)$, then $\rho_v^+(S)$ upper-bounds the total capacity of the edges in S. We assume that the functions $\rho_v^-(\cdot), \rho_v^+(\cdot), \quad v \in V$, are provided via value oracles. The maximum s-t flow in this polymatroidal network flow model can be written as the following linear program.

$$\begin{aligned}
\max \sum_{a \in \delta^+(s)} x(a) - \sum_{a \in \delta^-(s)} x(a) & \\
\sum_{a \in \delta^+(v)} x(a) - \sum_{a \in \delta^-(v)} x(a) &= 0 \quad \forall v \in V - \{s,t\} \\
x(S) &\leq \rho^-(S) \quad \forall S \subseteq \delta^-(v), v \in V \\
x(S) &\leq \rho^+(S) \quad \forall S \subseteq \delta^+(v), v \in V \\
x(a) &\geq 0 \quad \forall a \in A
\end{aligned}$$

As before we can solve the above LP since the separation oracle for the capacity constraints reduces to submodular function minimization. We note, however, that there are combinatorial algorithms that avoid the Ellipsoid method.

Unlike the global one we saw before the polymatroidal network flow model has several nice properties.

- The system of inequalities defining the flow is box-TDI.
- There is a notion of a min-cut and maxflow is equal to mincut
- There is an integral max flow when the polymadroids at each vertex are integer valued.

- The model is equivalent to that of Edmonds and Giles

The polymatroidal network flow model generalizes polymatroid intersection in an easy fashion and this is more transparent than it is in the Edmonds and Giles model (in this author's view). Consider two polymatroids f_1 and f_2 over a ground set N. Recall that in polymatroid intersection we are interested in vectors $x \in \mathbb{R}^S, x \geq 0$ that satisfy the constraints $x(S) \leq f_1(S) \quad \forall S \subseteq N$ and $x(S) \leq f_2(S) \quad \forall S \subseteq N$. Consider a network $G = (V, A)$ consisting of two nodes s, t which are connected by $|N|$ parallel arcs: each arc a corresponds to an element $e \in N$. Let ρ_s^+ correspond to f_1 and ρ_t^- correspond to f_2. Then it is easy to see that the set of feasible flows in the constructed polymatroidal network is precisely the set of vectors that are feasible for the polymatroid intersection polyhedron.

The definition of the cut cost: Given a directed graph $G = (V, A)$ and a set of arcs $F \subseteq E$ we say that the ordered node pair (s, t) is separated by F if there is no path from s to t in the graph $G[E \setminus F]$. A minimal cut will be of the form $\delta^+(S)$ for some $S \subset V$ with $s \in S, t \notin S$. It is convenient to label $V - S$ as the set T and hence F is the set of arcs from S to T. In the standard network model the cost of a cut defined by a set of arcs F is simply $\sum_{a \in F} c(a)$ where $c(a)$ is the cost of e (capacity in the primal flow network) . In polymatroid networks the cost of F is defined in a more involved fashion. Each arc (u, v) in F is assigned to either u or v; we say that an assignment of edges to nodes $g : F \to V$ is *valid* if it satisfies this restriction. A valid assignment partitions F into sets $\{g^{-1}(v) \mid v \in V\}$ where $g^{-1}(v)$ (the pre-image of v) is the set of arcs from F assigned to v by g. For a given valid assignment g of F the cost of the cut $\nu_g(F)$ is defined as

$$\nu_g(F) := \sum_v \left(\rho_v^-(\delta^-(v) \cap g^{-1}(v)) + \rho_v^+(\delta^+(v) \cap g^{-1}(v))\right).$$

The cost of the cut F is now defined as the minimum $\nu_g(F)$ over all valid assignments g. Clearly this seems like an involved computation. Indeed, it is. Given a partition (S, T) of V with $F = \delta^+(S)$ one can compute $\min_g \nu_g(F)$ via polymatroid intersection. One may notice that we can S into a single node and T into a single node and combine the polymatroids on each side into a single polymatroid. We leave the formal details as an exercise to the reader.

The interesting fact is that the maximum s-t flow a polymatroid network is equal to the minimum s-t cut cost as defined above. We refer the reader [42] for more discussion on how polymatroid optimization problems can be seen via flow network problems. See [13] for multicommodity flow generalizations of the polymatroidal network flow model and connections to flow-cut gaps in both directed and undirected graphs. The same paper has a proof of the flow-cut equivalence which is new and is perhaps more easy to understand than classical

proofs. Schrijver [57] has proofs of the equivalence of the Edmonds-Giles model and the polymatroidal network flow model.

Chapter 22

Multiflows[1]

The maxflow-mincut theorem of Ford and Fulkerson generalizes Menger's theorem and is a fundamental result in combinatorial optimization with many applications.

Theorem 22.1. *In a digraph $D = (V, A)$ with arc capacity function $c : A \to \mathbb{R}_+$, the maximum s-t flow value is equal to the minimum s-t capacity cut value. Moreover, if c is integer valued, then there is an integer valued maximum flow.*

In particular, the maximum number of s-t arc-disjoint paths in a digraph is equal to the minimum number of arcs whose removal disconnects s from t (Menger's theorem). When applied to undirected graphs we obtain the edge-disjoint and node-disjoint path version of Menger's Thoerem.

In many applications in networks we are interested in multiflows, also referred to as multi-commodity flows. $s - t$ flows are also referred to as single-commodity flows.

A *multiflow instance* in a directed graph consists of a directed "supply" graph $D = (V, A)$ with non-negative arc capacities $c : A \to \mathbb{R}_+$ and a demand graph $H = (T, R)$ with $T \subseteq V$ called terminals, and non-negative demand requirements $d : R \to \mathbb{R}_+$. The arcs in R are referred to as nets. The demand graph can also be specified as a set of ordered pairs $(s_1, t_1), \ldots, (s_k, t_k)$ with $d_i \in \mathbb{R}_+$ denoting the demand for (s_i, t_i). This is referred to as the k-commodity flow problem.

A *multiflow instance* in an undirected graph consists of an undirected supply graph $G = (V, E)$ and an undirected demand graph $H = (T, R)$. The demand graph can be specified by a collection of unordered pairs $s_1t_1, \ldots, s_kt_k$.

Given a multiflow instance in a directed graph $D = (V, A)$ with demand graph $H = (T, R)$, a *multiflow* is a collection of flows, $f_r, r \in R$ where f_r is an s_r-t_r

[1]Based on notes scribed by Su Lu and Dong Ye from 2010.

flow and $r = (s_r, t_r)$. A multiflow satisfies the capacity constraints of the supply graph if for each arc $a \in A$,

$$\sum_{r \in R} f_r(a) \leq c(a). \tag{22.1}$$

The multiflow satisfies the demands if for each $r = (s_r, t_r) \in R$, the f_r flow from s_r to t_r is at least $d(r)$.

For undirected graphs we need a bit more care. We say that $f : E \to \mathbb{R}_+$ is an $s - t$ flow if there is an orientation $D = (V, A)$ of $G = (V, E)$ such that $f' : A \to \mathbb{R}_+$ defined by the orientation and $f : E \to \mathbb{R}_+$ is an $s - t$ flow in D. Thus f_r, $r \in R$ where $f_r : E \to \mathbb{R}_+$ is a multiflow if each f_r is an s_r-t_r flow. It satisfies the capacity constraints if $\forall e \in E$,

$$\sum_{r \in R} f_r(e) \leq c(e). \tag{22.2}$$

We say a multiflow is *integral* if each of the flows is integer valued; that is $f_r(a)$ is an integer for each arc a and each $r \in R$. Similarly half-integral (i.e., each flow on an arc is an integer multiple of 1/2).

Proposition 22.0.1. *Given a multiflow instance in a directed graph, there is a polynomial time algorithm to check if there exists a multiflow that satisfies the capacities of the supply graph and the demand requirements of the demand graph.*

Proof. Can be solved by expressing the problem as a linear program. Variables $f_r(a)$ $r \in R$, $a \in A$. Write standard flow conservation constraints that ensures $f_r : A \to \mathbb{R}_+$ is a flow for each r (flow conservation at each node other than the source and destination of r). We add the following set of constraints to ensure capacity constraints of the supply graph are respected.

$$\sum_{r \in R} f_r(a) \leq c(a) \quad a \in A. \tag{22.3}$$

Finally, we add constraints that the value of f_r (leaving the source of r) should be at least $d(r)$. ■

Proposition 22.0.2. *Given an undirected multiflow instance, there is a polynomial time algorithm to check if there is a feasible multiflow that satisfies the supply graph capacities and the demand requirements.*

Proof. We reduce it to the directed flow case as follows. Given $G = (V, E)$ obtain a digraph $D = (V, A)$ by dividing each edge e into two arcs $\overrightarrow{e}$ and $\overleftarrow{e}$. Now we have variable $f_r(a)$, $a \in A$, $r \in R$, and write constraints that ensure that

$f_r : A \to \mathbb{R}_+$ is a flow of value $d(r)$ from s_r to t_r where $r = s_r t_r$. The capacity constraint ensures that the total flow on both $\overrightarrow{e}$ and $\overleftarrow{e}$ is at most $c(e)$, i.e.,

$$\sum_{r \in R} (f_r(\overrightarrow{e}) + f_r(\overleftarrow{e})) \leq c(e),\ e \in E. \tag{22.4}$$

■

LP duality gives the following useful necessary and sufficient condition; it is some times referred to as the Japanese theorem.

Theorem 22.2. *A multiflow instance in directed graph is feasible iff*

$$\sum_{i=1}^{k} d_i \ell(s_i, t_i) \leq \sum_{a \in A} c(a)\ell(a) \tag{22.5}$$

for all length functions $\ell : A \to \mathbb{R}_+$.

Here $\ell(s_i, t_i)$ is the shortest path distance from s_i to t_i with arc lengths $\ell(a)$. For undirected graph the characterization is similar

$$\sum_{i=1}^{k} d_i \ell(s_i, t_i) \leq \sum_{e \in E} c(e)\ell(e) \tag{22.6}$$

for all $\ell : E \to \mathbb{R}_+$.

Proof. Consider the path formulation we prove it for undirected graphs. Let P_i be the set of $s_i t_i$ path in G. Let $f_i : P_i \to \mathbb{R}_+$ be an assignment of flow values to paths in P_i. We want feasibility of

$$\sum_{p \in P_i} f_i(p) \geq d_i \quad i = 1 \text{ to } k \tag{22.7}$$

$$\sum_{i=1}^{k} \sum_{p \in P_i : e \in p} f_i(p) \leq c(e) \quad e \in E \tag{22.8}$$

$$f_i(p) \geq 0, \quad p \in P_i, \quad 1 \leq i \leq k. \tag{22.9}$$

We apply Farkas lemma. Recal that $Ax \leq b,\ x \geq 0$ has a solution iff $yb \geq 0$ for each row vector $y \geq 0$ with $yA \geq 0$. We leave it as an exercise to derive the statement from Farkas lemma applied to the above system of inequalities. ■

It is useful to interpret the necessity of the condition. Suppose for some $\ell : E \to \mathbb{R}_+$

$$\sum_{i=1}^{k} d_i \ell(s_i, t_i) > \sum_{e \in E} c(e)\ell(e) \tag{22.10}$$

we show that there is no feasible multiflow. For simplicity assume ℓ is integer valued. Then replace an edge $e = (u, v)$ with length $\ell(e)$ by a path between u and v with $\ell(e)$ edges and place capacity $c(e)$ on each edge. Suppose there is a feasible flow. For each demand pair (s_i, t_i), each flow path length is of length at least $\ell(s_i, t_i)$. Implies that the total capacity used up by flow for (s_i, t_i) is $\geq d_i \ell(s_i, t_i)$. But total capacity available is $\sum_e c(e)\ell(e)$ (after expansion). Hence if $\sum_{i=1}^{k} d_i \ell(s_i, t_i) > \sum_{e \in E} c(e)\ell(e)$, there cannot be a feasible multiflow.

To show that a multiflow instance is not feasible it is sufficient to give an appropriate arc length function that violates the necessary condition above.

22.1 Integer Multiflow and Disjoint Paths

When all capacities are 1 and all demands are 1 the problem of checking if there exists an integer multiflow is the same as asking if there exist arc-disjoint path (edge-disjoint path if graph is undirected) connecting the demand pairs.

The *edge-disjoint paths problem (EDP)* is the following decision problem: given supply graph $D = (V, A)$ (or $G = (V, E)$) and a demand graph $H = (T, R)$, are there arc/edge-disjoint paths connecting the pairs in R?

Theorem 22.3 (Fortune-Hopcroft-Wyllie 1980)**.** *EDP in directed graphs is NP-complete even for two pairs.*

Theorem 22.4. *EDP in undirected graphs is NP-complete when $|R|$ is part of the input, even when $|R|$ consists of three sets of parallel edges.*

A deep, difficult and fundamental result of Robertson and Seymour is that EDP in undirected graphs is polynomial-time solvable when $|R|$ is fixed. In fact they prove that the vertex-disjoint path problem (the pairs need to be connected by vertex disjoint paths) is also tractable.

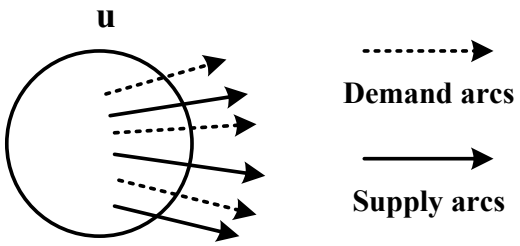
Theorem 22.5 (Robertson-Seymour)**.** *The vertex-disjoint path problem is polynomial-time solvable if the number of demand pairs is a fixed constant.*

The above theorem relies on the work of Robertson and Seymour on graph minors.

22.2 Cut Condition, Sparsest Cut, and Flow-Cut Gaps

A necessary condition for the existence of a feasible multiflow for a given instance is the so called cut-condition. In directed graphs it is

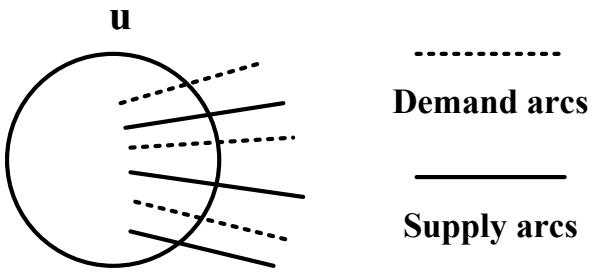
$$c(\delta_D^+(U)) \geq d(\delta_H^+(U)) \quad \forall U \subset V \tag{22.11}$$



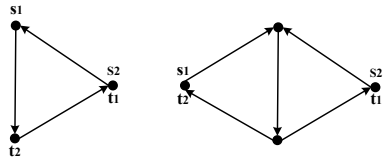
where $c(\delta_D^+(U))$ is capacity of all arcs leaving U, and $d(\delta_H^+(U))$ is the demand of all demand arcs leaving U. It is easy to see that this condition is necessary. Formally one sees that this condition is necessary by considering the length function $\ell : A \to \mathbb{R}_+$ where $\ell(a) = 1$ if $a \in \delta_D^+(U)$ and $\ell(a) = 0$.

For undirected graphs the cut condition states

$$c(\delta_G(U)) \geq d(\delta_H(U)) \quad \forall U \subset V \tag{22.12}$$

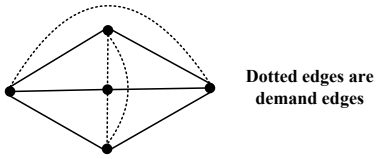


Cut condition is not sufficient in general. Consider the following examples in directed graphs with two demand pairs.



Cut condition is true for each case but no feasible multiflows exists as can be seen by considering the length function $\ell(a) = 1$ for each arc a.

For undirected graphs the following example is well known with three demand pairs. Supply graph is $K_{2,3}$, a series-parallel graph. Again, cut-condition is satisfied but $\ell(e) = 1$ for each e shows no feasible multiflow exists.



22.3 When is cut condition sufficient?

Given that the cut condition is not sufficient for feasible flow it is natural to consider cases where it is indeed sufficient. First consider directed graphs. Suppose we have demand pairs of the form $(s, t_1), (s, t_2), \ldots, (s, t_k)$, i.e., all pairs share a common source. Then it is easy to see that cut condition implies feasible multiflow by reduction to the single-commodity flow case by connecting $t_1, t_2, \ldots, t_k$ to a common sink t. Similarly if the demand pairs are of the form $(s_1, t), (s_2, t), \ldots, (s_k, t)$ with a common source.

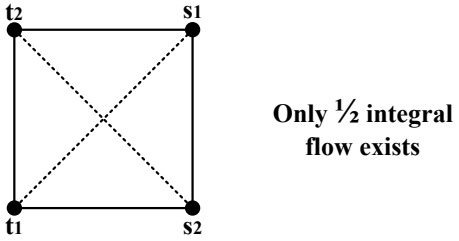
It turns out that these are the only interesting cases for which cut condition suffices. See Theorem 70.3 in Schrijver Book.

For undirected graphs several non-trivial and interesting cases where the cut-condition is sufficient are known. We list a few below:

- Hu's 2-commodity theorem shows that if there are only two pairs s_1t_1 and s_2t_2 then cut-condition is sufficient.
- Okamura-Seymour theorem states that if G is a planar graph and T is vertex set of a single face then cut condition is sufficient. Note that the theorem implies that when the supply graph is a capacitated ring, the cut condition is sufficient for routing.

- Okamura's theorem generalizes Okamura-Seymour Theorem. If G is planar and there are two faces F_1 and F_2 such that each $st \in R$ has both s, t on one of these two faces then cut condition is sufficient.
- Seymour's Theorem shows that if $G + H$ is planar then cut condition is sufficient.

For all of the above cases one has the following stronger result. If $G + H$ is Eulerian then the flow is guaranteed to be integral. To see that the Eulerian condition is necessary for integral flow in each of the above cases, consider the example below where the only feasible multiflow is a half-integral.



Note that if the supply graph is a tree then cut condition implies feasible routing, and moreover, integer capacities and integer demands imply integer routing exists. If the $G + H$ is outerplanar then this property still holds [53].

22.4 Okamura-Seymour Theorem

Theorem 22.6. *Let $G = (V, E)$ be a plane graph and let $H = (T, R)$ be a demand graph where T is the set of vertices of a single face of G. Then if G satisfies the cut condition for H and $G + H$ is eulerian, there is an integral multiflow for H in G.*

The proof is via induction on $2|E| - |R|$. Note that if G satisfies the cut condition for H, then $|R| \leq |E|$ (why?).

There are several "standard" induction steps and observations that are used in this and other proofs and we go over them one by one. For this purpose we assume G, H satisfy the conditions of the theorem and is a counter example with $2|E(G)| - |R|$ minimal**[Lex]**.

Claim 22.4.1. *No demand edge r is parallel to a supply edge e.*

Proof. If r is parallel to e then $G - e$, $H - r$ satisfy the conditions of the theorem and by induction $H - r$ has an integral multiflow in $G - e$. We can route r via e. Thus H has an integral multiflow in G. ■

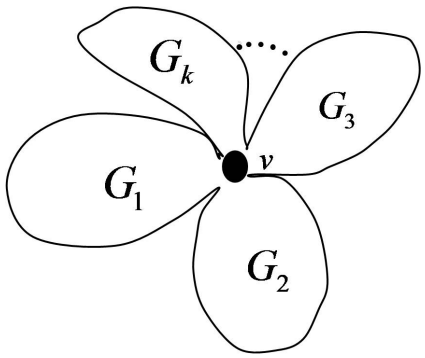
Definition 22.7. *A set $S \subset V$ is said to be tight if $|\delta_G(S)| = |\delta_H(S)|$.*

Claim 22.4.2. *For every demand edge $r \in R$ there is a tight cut S s.t. $r \in \delta_H(S)$.*

Proof. If r is not in any tight set, then adding two copies of r to H maintains cut condition and the Eulerian condition. By induction (note that the induction is on $2|E| - |R|$) the new instance is routable. ■

Claim 22.4.3. *G is 2-node connected.*

Proof. Suppose not and let v be a cut vertex of G. Let $G_1, G_2, \ldots, G_k$ be the graphs obtained by combining v with the components of $G - v$.



Suppose there is a demand edge $r = (s, t)$ s.t. $s \neq v, t \neq v$ and $s \in G_i$ and $t \in G_j, i \neq j$. Then we can replace (s, t) by two edges (s, v) and (v, t). The claim is that this new instance satisfies the cut condition - we leave the formal proof as an exercise. Clearly Euler condition is maintained. The new instance is routable by induction which implies that the original instance is also routable.

If no such demand edge exits then all demand edges have both end points in G_i for some i. Then let H_i be the demand graph induced on G_i. We can verify that each G_i, H_i satisfy the cut condition and the Euler condition. By induction each H_i is routable in G_i which implies that H is routable in G. ■

Definition 22.8. *A set $\emptyset \subset S \subset V$ is central if $G[S]$ and $G[V \setminus S]$ are connected.*

Lemma 22.1. *Let G be a connected graph. Then G, H satisfy the cut condition if and only if the cut condition is satisfied for each central set S.*

Proof. Clearly, if G, H satisfy the cut condition for all sets then it is satisfied for the central sets. Suppose the cut condition is satisfied for all central sets but there is some non-central set S' such that $|\delta_G(S)| < |\delta_H(S)|$. Choose S' to be minimal among all such sets. We obtain a contradiction as follows. Let $S_1, S_2, \ldots, S_k$ be

the connected components in $G \setminus \delta_G(S')$; since S' is not central, $k \geq 3$. Moreover each S_i is completely contained in S' or in $V \setminus S'$. We claim that some S_j violates the cut-condition, whose proof we leave as an exercise. Moreover, by minimality in the choice of S', S_j is central, contradicting the assumption. ■

One can prove the following corollary by a similar argument.

Corollary 22.9. *Let G, H satisfy the cut condition. If S' is a tight set and S' is not central, then there is some connected component S contained in S' or in $V \setminus S'$ such that S is a tight central set.*

Uncrossing:

Lemma 22.2. *Let G, H satisfy cut-condition, Let A, B be two tight sets such that $A \cap B \neq \emptyset$ and $A \cup B \neq V$. If $|\delta_H(A)| + |\delta_H(B)| \leq |\delta_H(A \cap B)| + |\delta_H(A \cup B)|$, then $A \cap B$ and $A \cup B$ are tight. If $|\delta_H(A)| + |\delta_H(B)| \leq |\delta_H(A - B)| + |\delta_H(B - A)|$, then $A - B$ and $A - B$ are tight.*

Proof. By submodularity and symmetry of the cut function $|\delta_G| : 2^V \to \mathbb{R}_+$, we have

$$|\delta_G(A)| + |\delta_G(B)| \geq |\delta_G(A \cap B)| + |\delta_G(A \cup B)|$$

and also

$$\delta_G(A)| + |\delta_G(B)| \geq |\delta_G(A - B)| + |\delta_G(B - A)|.$$

Now if

$$|\delta_H(A)| + |\delta_H(B)| \leq |\delta_H(A \cap B)| + |\delta_H(A \cup B)|$$

then we have

$$|\delta_G(A \cap B)| + |\delta_G(A \cup B)| \geq |\delta_H(A \cap B)| + |\delta_H(A \cup B)| \geq |\delta_H(A)| + |\delta_H(B)| = |\delta_G(A)| + |\delta_G(B)|$$

where the first inequality follows from the cut-condition, the second from our assumption and the third from the tightness of A and B. It follows that

$$|\delta_G(A \cap B)| = |\delta_H(A \cap B)|$$

and

$$|\delta_G(A \cup B)| = |\delta_H(A \cup B)|.$$

The other claim is similar. ■

Corollary 22.10. *If A, B are tight sets and $\delta_H(A - B, B - A) = \emptyset$ then $A \cap B$ and $A \cup B$ are tight.*

Proof. We note that

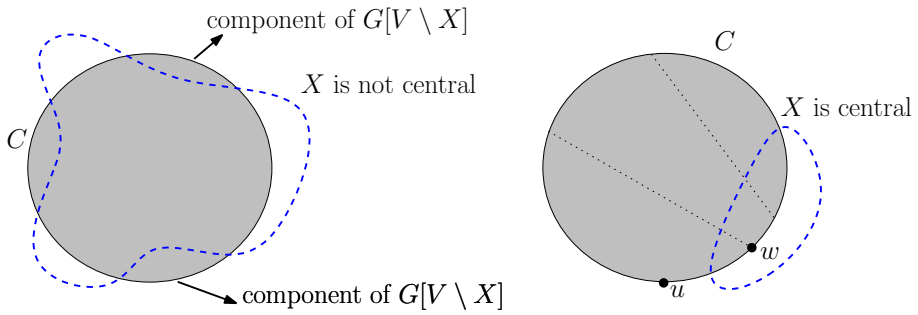
$$|\delta_H(A)| + |\delta_H(B)| = |\delta_H(A \cap B)| + |\delta_H(A \cup B)| + 2|\delta_H(A - B, B - A)|.$$

Thus, if $\delta_H(A - B, B - A) = \emptyset$ we have $|\delta_H(A)| + |\delta_H(B)| = |\delta_H(A \cap B)| + |\delta_H(A \cup B)|$ and we apply the previous lemma. ■

Proof. Now we come to the proof of the Okamura-Seymour theorem. Recall that G, H is a counter example with $2|E| - |R|$ minimal. Then we have established that:

1. G is 2-connected.
2. every demand edge is in a tight cut.
3. no supply edge is parallel to a demand edge.

Without loss of generality we assume that the all the demands are incident to the outer/unbounded face of G. Since G is 2-connected the outer face is a cycle C. Let $X \subset V$ be a tight set; a tight set exists since each demand edge is in some tight set. Then if $X \cap C$ is not a contiguous segment, X is not a central set as can be seen informally by the picture below; $G[V \setminus X]$ would have two or more connected components.



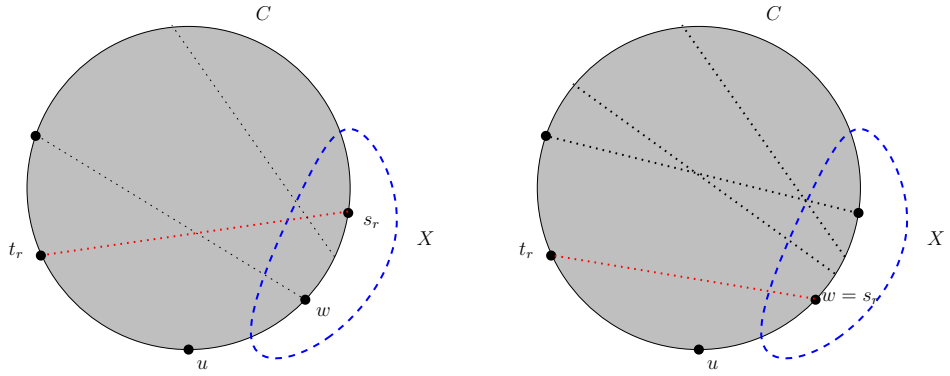
From Corollary 22.9 we can assume the existence of a tight set X such that $X \cap C$ is a contiguous segment. Choose such a tight set with $X \cap C$ minimal.

Let uw be one of the two edges of the cycle C that crosses X; let $w \in X$ and $u \notin X$. Since X is tight, $\delta_R(X) \neq \emptyset$. For each $r \in \delta_R(X)$, let s_r, t_r be the endpoints of r with $s_r \in X \cap C$ and $t_r \notin X \cap C$. Choose $r \in \delta_R(X)$ such that t_r is closest (in distance along the cycle C) to u in $C - X$. Note that r is not parallel to uw. So if $s_r = w$ then $t_r \neq u$ and if $t_r = u$ then $s_r \neq w$. Let $v \in \{u, w\} \setminus \{s_r, t_r\}$, v exists by above; for simplicity choose $v = w$ if $s_r \neq w$.

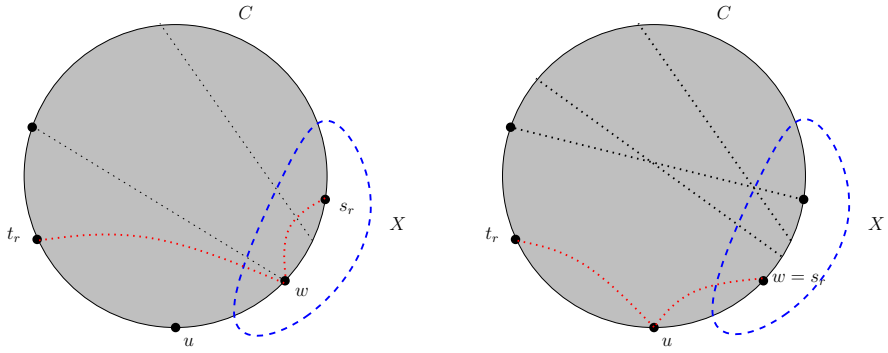
Let $R' = (R \setminus \{s_r, t_r\}) \cup \{s_r v, v t_r\}$. That is, we replace the demand edge $s_r t_r$ by two new demand edges $s_r v$ and $v t_r$ as shown in the figure.

Claim 22.4.4. *G satisfies cut condition for R' and $E + R'$ induces an Eulerian graph.*

Assuming claim, we are done because $2|E| - |R'| < 2|E| - |R|$ and by induction R' has an integral multiflow in G, and R has an integer multiflow if R' does.



In the picture on the left $v = w$ and on the right $v = u$.



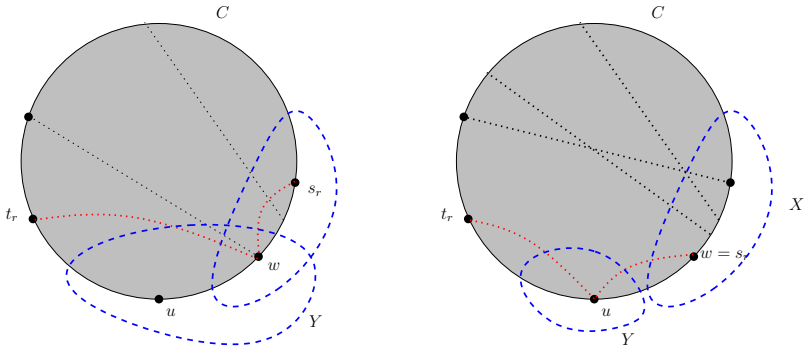
Replacing $s_r t_r$ by new demands $s_r v$ and $v t_r$.

Trivial to see $E + R'$ induces an Eulerian graph. Suppose G does not satisfy the cut condition for the demand set R'. Let Y be a cut that violates the cut condition for R'. For this to happen Y must be a tight set for R in G; this is the reason why replacing $s_r t_r$ by $s_r v$ and $v t_r$ violates the cut condition for Y for R'. By complementing Y if necessary we can assume that $v \in Y, s_r, t_r \notin Y$. Further, by Corollary 22.9, we can assume Y is central and hence $Y \cap C$ is a contiguous segment of C.

By choice of r there is no demand r' between $Y - X$ and $X - Y$. If there was, then $t_{r'}$ would be closer to u than t_r. We have X, Y tight and

$$\delta_R[X - Y, Y - X] = \emptyset.$$

We consider two cases. First, suppose $X \cap Y \neq \emptyset$ (this is guaranteed if $v = w$). Then from Corollary 22.10, $X \cap Y$ and $X \cup Y$ are tight since $X \cap Y \neq \emptyset$ by assumption and $X \cup Y \neq V$ (since $t_r \in V \setminus (X \cup Y)$). $X - Y \neq \emptyset$, since $s_r \in X - Y$. Since $X \cap Y$ is a tight set and $X \cap Y \neq X$, it contradicts the choice of X as the tight



Tight set Y in the two cases.

set with $X \cap C$ minimal. If $X \cap Y = \emptyset$ then $v = u$ and $u \in Y$; again $X \cup Y \neq V$. Note that the edge uw joins X and Y. In this case we claim that $X \cup Y$ does not satisfy the cut condition which is a contradiction. To see this note that

$$|\delta_G(X \cup Y)| \leq |\delta_G(X)| + |\delta_G(Y)| - 2$$

since uw connects X to Y. However,

$$|\delta_H(X \cup Y)| = |\delta_H(X)| + |\delta_H(Y)| = |\delta_G(X)| + |\delta_G(Y)|$$

where the first inequality follows since $X \cap Y = \emptyset$ and there are no demand edges between $X - Y$ and $Y - X$. The second inequality follows from the tightness of X and Y. ■

22.5 Sparse Cuts, Concurrent Multicomodity Flow and Flow-Cut Gaps

In traditional combinatorial optimization, the focus has been on understanding and characterizing those cases where cut condition implies existence of fractional/integral multiflow. However, as we saw, even in very restrictive settings, cut condition is not sufficient. A theoretical CS/algorithms perspective has been to quantify the "gap" between flow and cut. More precisely, suppose G satisfies the cut condition for H. Is it true that there is a feasible multiflow in G that routes λd_i for each pair $s_i t_i$ where λ is some constant in $(0, 1)$?

There are two reasons for considering the above. First, it is a mathematically interesting question. Second, and this was the initial motivation from a computer science/algorithmic point of view, is to obtain approximation algorithms for finding "sparse" cuts in graphs; these have many applications in science and engineering. The following is known.

Theorem 22.11. *Given a multiflow instance, it is co-NP complete to check if the cut-condition is satisfied for the instance.*

Definition 22.12. *Given a multiflow instance the maximum concurrent flow for the given instance is the maximum $\lambda \geq 0$ such that there is a feasible multiflow if all demand values are multiplied by λ.*

Proposition 22.5.1. *There is a polynomial time algorithm that, given a multiflow instance, computes the maximum concurrent flow.*

Proof. Write a linear program:

$$\begin{aligned} &\max \lambda \\ &\text{flow for each} \quad s_i t_i \geq \lambda d_i \end{aligned}$$

Flow satisfies capacity constraints. We leave the details to the reader. ■

Definition 22.13. *Given a multiflow instance on G, H, the sparsity of a cut $U \subset V$ is*

$$sparsity(U) := \frac{c\left(\delta_G(U)\right)}{d\left(\delta_H(U)\right)}.$$

A sparsest cut is $U \subset V$ such that $sparsity(U) \leq sparsity(U')$ for all $U' \subset V$. We refer to $\min_{U \subset V} sparsity(U)$ as the min-sparsity of the given multiflow instance.

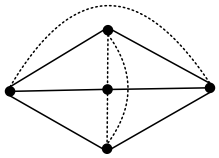
Observation 22.14. *(G, H) satisfies the cut condition implies sparsity(U) ≥ 1 for all $\mathcal{U} \subset V$.*

Proposition 22.5.2. *In many multiflow instance, if λ^* is the max concurrent flow then*

$$\lambda^* \leq sparsity(U), \forall U \subset V.$$

The ratio $\frac{min-sparity}{\lambda^*}$ is the flow cut gap for the given instance.

For example, in the instance shown in the figure with capacities and demands



equal to 1, the flow-cut gap is $\frac{4}{3}$. Min-sparsity for the above instance is 1 while $\lambda^* = \frac{3}{4}$. In general, we are interested in quantifying flow-cut gaps for classes of instances rather than a particular instance.

In the sequel, we think of G and H as "topological" graphs in that they are not endowed with capacities and demands. A multiflow instance on G, H is defined by $c : E \to \mathbb{R}_+$ and $d : R \to \mathbb{R}_+$. Note that by setting $c(e) = 0$ or $d(r) = 0$, we can "eliminate" some edges. We define $\alpha(G, H)$, the flow-cut gap for G, H, as the supremum over all instances on G, H defined by capacities $c : E \to \mathbb{R}_+$ and $d : R \to \mathbb{R}_+$. We can then define for a graph G:

$$\alpha(G) = \sup_{H=(T,R),T\subseteq V} \alpha(G, H).$$

Some results that we mentioned on the sufficiency of cut condition for feasible flow can be restated as follows: $\alpha(G, H) = 1$ if $|R| = 2$ (Hu's theorem), $\alpha(G, H) = 1$ if G is planar and T is the vertex set of a face of G (Okamura-Seymour theorem), and so on. What can we say about $\alpha(G)$ for an arbitrary graph?

Theorem 22.15 (Linial-London-Rabinovich [46], Aumann-Rabani [5])**.** *$\alpha(G) = O(\log n)$ where $n = |V|$ and in particular $\alpha(G, H) = O(\log |R|)$ i.e. the flow-cut gap is $O(\log k)$ for k-commodity flow. Moreover there exist graphs G, H for which $\alpha(G, H) = \Omega(\log |R|)$, in particular there exist graphs G for which $\alpha(G) = \Omega(\log n)$.*

Gupta *et al* [31] made the following conjecture called now the GNRS conjecture.

Conjecture 22.16. *$\alpha(G) = O(1)$ if G is a planar graph.*

In fact their conjecture is stronger and applies to any proper minor-closed family of graphs.

We do know that planar graphs admit a better worst-case flow-cut gap than general graphs due a result of Satish Rao [54].

Theorem 22.17 (Rao [54])**.** *$\alpha(G) = O(\sqrt{\log n})$ for a planar graph G.*

For the special case of series-parallel graphs we know that $\alpha(G) \leq 2$ [10], and moreoever this bound is tight in the worst case [15, 43].

Bibliography

[1] Ahmad Abdi, Gérard Cornuéjols, and Michael Zlatin. "On packing dijoins in digraphs and weighted digraphs". In: *arXiv preprint arXiv:2202.00392* (2022).

[2] Alexander A Ageev and Maxim I Sviridenko. "Pipage rounding: A new method of constructing algorithms with proven performance guarantee". In: *Journal of Combinatorial Optimization* 8.3 (2004), pp. 307–328.

[3] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. "Network flows". In: (1988).

[4] Arash Asadpour, Michel X Goemans, Aleksander Mądry, Shayan Oveis Gharan, and Amin Saberi. "An O (log n/log log n)-approximation algorithm for the asymmetric traveling salesman problem". In: *Operations Research* 65.4 (2017), pp. 1043–1061.

[5] Yonatan Aumann and Yuval Rabani. "An O (log k) approximate min-cut max-flow theorem and approximation algorithm". In: *SIAM Journal on Computing* 27.1 (1998), pp. 291–301.

[6] Niv Buchbinder and Moran Feldman. "Handbook of Approximation Algorithms and Metaheuristics". In: ed. by Teofilo Gonzalez. Second edition. Chapman and Hall/CRC, 2018. Chap. Submodular Functions Maximization Problems.

[7] Andrei A Bulatov. "Constraint satisfaction problems: complexity and algorithms". In: *ACM SIGLOG News* 5.4 (2018), pp. 4–24.

[8] Gruia Calinescu, Chandra Chekuri, Martin Pal, and Jan Vondrák. "Maximizing a monotone submodular function subject to a matroid constraint". In: *SIAM Journal on Computing* 40.6 (2011), pp. 1740–1766.

[9] Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. "Maximizing a submodular set function subject to a matroid constraint". In: *International Conference on Integer Programming and Combinatorial Optimization*. Springer. 2007, pp. 182–196.

[10] Amit Chakrabarti, Alexander Jaffe, James R Lee, and Justin Vincent. "Embeddings of topological graphs: Lossy invariants, linearization, and 2-sums". In: *2008 49th Annual IEEE Symposium on Foundations of Computer Science*. IEEE. 2008, pp. 761–770.

[11] Chandra Chekuri and Alina Ene. "Approximation algorithms for submodular multiway partition". In: *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*. IEEE. 2011, pp. 807–816.

[12] Chandra Chekuri and Alina Ene. "Submodular cost allocation problem and applications". In: *International Colloquium on Automata, Languages, and Programming*. Springer. 2011, pp. 354–366.

[13] Chandra Chekuri, Sreeram Kannan, Adnan Raja, and Pramod Viswanath. "Multicommodity flows and cuts in polymatroidal networks". In: *SIAM Journal on Computing* 44.4 (2015), pp. 912–943.

[14] Chandra Chekuri, Kent Quanrud, and Manuel R Torres. "Densest Subgraph: Supermodularity, Iterative Peeling, and Flow". In: *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM. 2022, pp. 1531–1555.

[15] Chandra Chekuri, F Bruce Shepherd, and Christophe Weibel. "Flow-cut gaps for integer and fractional multiflows". In: *Journal of Combinatorial Theory, Series B* 103.2 (2013), pp. 248–273.

[16] Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. "Submodular function maximization via the multilinear relaxation and contention resolution schemes". In: *SIAM Journal on Computing* 43.6 (2014), pp. 1831–1879.

[17] Hubie Chen. "A rendezvous of logic, complexity, and algebra". In: *ACM Computing Surveys (CSUR)* 42.1 (2009), pp. 1–32.

[18] W.J. Cook, W.H. Cunningham, W.R. Pulleyblank, and A. Schrijver. *Combinatorial Optimization*. Wiley Series in Discrete Mathematics and Optimization. Wiley, 1998.

[19] William H Cunningham and AB Marsh. "A primal algorithm for optimum matching". In: *Polyhedral Combinatorics*. Springer, 1978, pp. 50–72.

[20] Ran Duan and Seth Pettie. "Linear-time approximation for maximum weight matching". In: *Journal of the ACM (JACM)* 61.1 (2014), pp. 1–23.

[21] Ran Duan, Seth Pettie, and Hsin-Hao Su. "Scaling algorithms for weighted matching in general graphs". In: *ACM Transactions on Algorithms (TALG)* 14.1 (2018), pp. 1–35.

[22] Shaddin Dughmi. "Submodular functions: Extensions, distributions, and algorithms. a survey". In: *arXiv preprint arXiv:0912.0322* (2009).

[23] Jack Edmonds and Rick Giles. "Total dual integrality of linear inequality systems". In: *Progress in combinatorial optimization*. Elsevier, 1984, pp. 117–129.

[24] Moran Feldman, Joseph Naor, and Roy Schwartz. "A unified continuous greedy algorithm for submodular maximization". In: *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*. IEEE. 2011, pp. 570–579.

[25] Lisa Fleischer. *Recent Progress in Submodular Function Minimization.* OPTIMA: Mathematical Programming Society Newsletter. Available online at `http://www.mathprog.org/Optima-Issues/optima64.pdf`. Sept. 2000.

[26] Harold N Gabow, Zvi Galil, Thomas Spencer, and Robert E Tarjan. "Efficient algorithms for finding minimum spanning trees in undirected and directed graphs". In: *Combinatorica* 6.2 (1986), pp. 109–122.

[27] Shayan Oveis Gharan, Amin Saberi, and Mohit Singh. "A randomized rounding approach to the traveling salesman problem". In: *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*. IEEE. 2011, pp. 550–559.

[28] Gagan Goel, Chinmay Karande, Pushkar Tripathi, and Lei Wang. "Approximability of combinatorial problems with multi-agent submodular cost functions". In: *2009 50th Annual IEEE Symposium on Foundations of Computer Science*. IEEE. 2009, pp. 755–764.

[29] Andrew V Goldberg and Robert E Tarjan. "Finding minimum-cost circulations by canceling negative cycles". In: *Journal of the ACM (JACM)* 36.4 (1989), pp. 873–886.

[30] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*. Vol. 2. Springer Science & Business Media, 2012.

[31] Anupam Gupta, Ilan Newman, Yuri Rabinovich, and Alistair Sinclair. "Cuts, trees and l/sub 1/-embeddings of graphs". In: *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*. IEEE. 1999, pp. 399–408.

[32] David Hartvigsen. "Compact representations of cuts". In: *SIAM Journal on Discrete Mathematics* 14.1 (2001), pp. 49–66.

[33] REFAEL HASSIN. "ON NETWORK FLOWS." PhD thesis. Yale University, 1978.

[34] Refael Hassin. "Solution bases of multiterminal cut problems". In: *Mathematics of operations research* 13.4 (1988), pp. 535–542.

[35] Satoru Iwata and Kiyohito Nagano. "Submodular function minimization under covering constraints". In: *2009 50th Annual IEEE Symposium on Foundations of Computer Science*. IEEE. 2009, pp. 671–680.

[36] David R Karger. "Minimum cuts in near-linear time". In: *Journal of the ACM (JACM)* 47.1 (2000), pp. 46–76.

[37] David Ron Karger. "Random sampling in graph optimization problems". PhD thesis. stanford university, 1995.

[38] Anna Karlin, Nathan Klein, and Shayan Oveis Gharan. "A (Slightly) Improved Bound on the Integrality Gap of the Subtour LP for TSP". In: *CoRR* abs/2105.10043 (2021). arXiv: 2105.10043. URL: https://arxiv.org/abs/2105.10043.

[39] Bernhard H Korte, Jens Vygen, B Korte, and J Vygen. *Combinatorial optimization*. Vol. 1. Springer, 2011.

[40] Lap Chi Lau, Ramamoorthi Ravi, and Mohit Singh. *Iterative methods in combinatorial optimization*. Vol. 46. Cambridge University Press, 2011.

[41] Eugene L Lawler and Charles U Martel. "Computing maximal "polymatroidal" network flows". In: *Mathematics of Operations Research* 7.3 (1982), pp. 334–347.

[42] Eugene L Lawler and Charles U Martel. "Flow network formulations of polymatroid optimization problems". In: *North-Holland Mathematics Studies*. Vol. 66. Elsevier, 1982, pp. 189–200.

[43] James R Lee and Prasad Raghavendra. "Coarse differentiation and multi-flows in planar graphs". In: *Discrete & Computational Geometry* 43.2 (2010), pp. 346–362.

[44] Hendrik W Lenstra Jr. "Integer programming with a fixed number of variables". In: *Mathematics of operations research* 8.4 (1983), pp. 538–548.

[45] Jason Li and Debmalya Panigrahi. "Deterministic Min-cut in Poly-logarithmic Max-flows". In: *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2020, pp. 85–92.

[46] Nathan Linial, Eran London, and Yuri Rabinovich. "The geometry of graphs and some of its algorithmic applications". In: *Combinatorica* 15.2 (1995), pp. 215–245.

[47] László Lovász. *An algorithmic theory of numbers, graphs and convexity*. SIAM, 1986.

[48] László Lovász. "Submodular functions and convexity". In: *Mathematical programming the state of the art*. Springer, 1983, pp. 235–257.

[49] Jiri Matousek and Bernd Gärtner. *Understanding and using linear programming*. Springer Science & Business Media, 2007.

[50] S Thomas McCormick. "Submodular function minimization". In: *Handbooks in operations research and management science* 12 (2005), pp. 321–391.

[51] Andrew McGregor and Bruce Shepherd. "Island hopping and path colouring with applications to WDM network design". In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. 2007, pp. 864–873.

[52] Ran Mendelson, Robert E Tarjan, Mikkel Thorup, and Uri Zwick. "Melding priority queues". In: *ACM Transactions on Algorithms (TALG)* 2.4 (2006), pp. 535–556.

[53] Guyslain Naves, F Bruce Shepherd, and Henry Xia. "Maximum weight disjoint paths in outerplanar graphs via single-tree cut approximators". In: *Mathematical Programming* (2022), pp. 1–19.

[54] Satish Rao. "Small distortion and volume preserving embeddings for planar and Euclidean metrics". In: *Proceedings of the fifteenth annual symposium on Computational geometry*. 1999, pp. 300–306.

[55] Richard Santiago and F Bruce Shepherd. "Multi-agent submodular optimization". In: *arXiv preprint arXiv:1803.03767* (2018).

[56] Richard Santiago and F Bruce Shepherd. "Multivariate submodular optimization". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 5599–5609.

[57] Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*. Vol. 24. Springer Science & Business Media, 2003.

[58] Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[59] F. Bruce Shepherd. "Single-Sink Multicommodity Flow with Side Constraints". In: *Research Trends in Combinatorial Optimization, Bonn Workshop on Combinatorial Optimization, November 3-7, 2008, Bonn, Germany*. Ed. by William J. Cook, László Lovász, and Jens Vygen. Springer, 2008, pp. 429–450. DOI: 10.1007/978-3-540-76796-1_20. URL: https://doi.org/10.1007/978-3-540-76796-1%5C_20.

[60] FB Shepherd and A Vetta. "Visualizing, finding and packing dijoins". In: *Graph Theory and Combinatorial Optimization*. Springer, 2005, pp. 219–254.

[61] Éva Tardos. "A strongly polynomial algorithm to solve combinatorial linear programs". In: *Operations Research* 34.2 (1986), pp. 250–256.

[62] Éva Tardos. "A strongly polynomial minimum cost circulation algorithm". In: *Combinatorica* 5.3 (1985), pp. 247–255.

[63] Alexander Toshev. *Submodular Function Minimization*. Preliminary exam report, University of Pennsylvania. Available at `https://drive.google.com/file/d/1Wkv6uH0BSXCGHQUwSVEJ9LKUfLuyzpY0/view?usp=sharing`. 2010.

[64] Vijay V Vazirani. *Approximation algorithms*. Springer Science & Business Media, 2013.

[65] Vijay V. Vazirani. *A Proof of the MV Matching Algorithm*. 2020. arXiv: `2012.03582 [cs.DS]`.

[66] Jan Vondrák. "Submodularity in combinatorial optimization". In: (2007).

[67] David P Williamson. *Network flow algorithms*. Cambridge University Press, 2019.

[68] Peter Winkler and Lisa Zhang. "Wavelength assignment and generalized interval graph coloring". In: *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*. 2003, pp. 830–831.

[69] Laurence A Wolsey. "Heuristic analysis, linear programming and branch and bound". In: *Combinatorial Optimization II*. Springer, 1980, pp. 121–134.