CMPUT 675: Topics in Algorithms and Combinatorial Optimization (Fall 2009)

Lecture 16-17: Matroid Intersection

Lecturer: Mohammad R. Salavatipour **Scriber:** Yining Wang
Date: Oct 25,27, 2009

1 Matroid Intersection

We have seen that greedy algorithm works well on matroids to find maximum weight independent sets. So any problem that can be formulated as optimizing over independent sets of a matroid (e.g. min spanning tree) can be solved using the greedy algorithm. However, not many problems fall into this category and so the greedy algorithm doesn't work. It turns out that many optimization problems can be formulated as finding a common independent set in the intersection of two matroids and for maximizing over such sets there is an algorithm that was first presented by Edmonds. Below, we describe a few problems that can be formulated as finding a common independent set between two matroids.

Example: [Bipartite Matching] Given a bipartite graph $G = (A \cup B, E)$, where $A = \{a_1, \cdots, a_n\}$ and $B = \{b_1, \cdots, b_m\}$, let $\mathcal{M}$ be the set of all matchings of G. As we saw, $(E, \mathcal{M})$ does not define a matroid (and therefore we cannot find the maximum matching using the greedy algorithm). However, $\mathcal{M}$ has the property that it is the intersection of two matroids; in fact it is the intersection of two partition matroids that we define below. For each vertex v, let $\delta(v)$ denote the edges incident to v, and we define two partition matroids as follows:

$$M_1 = (E, \mathcal{I}_1), \text{ where } E = \delta(a_1) \cup \delta(a_2) \cup \cdots \cup \delta(a_n)$$

and

$$M_2 = (E, \mathcal{I}_2), \text{ where } E = \delta(b_1) \cup \delta(b_2) \cup \cdots \cup \delta(b_m)$$

Then, $\forall \mathcal{F} \in \mathcal{I}_1 \cap \mathcal{I}_2$ has degree ≤ 1 on A (because of $\mathcal{I}_1$) and degree ≤ 1 on B (because of $\mathcal{I}_2$). So every $F \subseteq E$ is a matching if and only if it is an independent set in both M_1 and M_2. Thus a maximum matching in G is a a common independent set of largest size. ■

Example: [r-arborescence (r-branching)] Given a digraph $D = (V, A)$ and a root node $r \in V$, an r-arborescence is a rooted tree (rooted at r). That is, the indegree of each node (other than r) is 1 and the root r has indegree 0. Note that if we drop the directions on the edges, an r-arborescence is just a spanning tree of G. We can assume that r has in-degree zero in D since any such edge can be deleted from the graph as they don't belong to any r-arborescence. Here we show that an r-arborescence can be viewed as an independent set of in the intersection of two matroids. Let G be the underlying undirected graph of D obtained by disregarding the directions of the edges in D. Let $M_1 = (E, \mathcal{I}_1)$ be the graphic matroid, where E is the same as A ignoring directions of edges in A. Let $M_2 = (A, \mathcal{I}_2)$ be the partition matroid, where $A = \delta^{-1}(v_1) \cup \delta^{-1}(v_2) \cup \cdots \cup \delta^{-1}(v_n)$. It is easy to see that an r-arborescence is independent set of both matroids: the underlying graph must be a spanning tree (so it is a bases of M_1) and the set of edges must have indegree at most 1 for each $v \neq r$, so it is an independent set of M_2. Conversely, any common independent set has an underlying graph that is acyclic and has indegree at most 1 for each node. Although the greedy algorithm can find a MST in an undirected graph, we cannot use that to find minimum cost (or maximum cost) r-arborescence in a graph. ■

Example: [rainbow spanning trees] Suppose that we have colored the edges of a given graph $G = (V, E)$ with k colors. Let $E = E_1 \cup E_2 \cup \cdots \cup E_k$ be a partition of of edges into k color sets. A rainbow spanning tree is a spanning tree whose edges have all distinct colors. We can show that finding a rainbow spanning tree is equivalent to finding a maximum independent set common to two matroids. Let $M_1 = (E, \mathcal{I}_1)$ be a graphic matroid on G. Let $M_2 = (E, \mathcal{I}_2)$, where $\mathcal{I}_2 = \{E : |F \cap E_i| \leq 1, \forall 1 \leq i \leq k\}$, be a partition matroid. The a maximum common independent set of M_1 and M_2 is a rainbow spanning tree. ■

Example: [Forest in two graphs] Suppose we are given two graphs $G = (V, E)$ and $G' = (V', E')$ with $|E| = |E'|$; we assume the edges of the two graphs are labeled with indices from 1 to $m = |E|$. Our goal is to find a largest set I of indices $\mathcal{I} \subseteq \{1, \cdots, m\}$ such that a subgraph induced by these edges in each of G and G' is a forest. This problem too can be represented as finding a largest common independent set of two graphic matroids . ■

Given two matroids $M_1 = (E, \mathcal{I}_1)$ with rank function r_1, and $M_2 = (E, \mathcal{I}_2)$ with rank function r_2. Let $S \in \mathcal{I}_1 \cap \mathcal{I}_2$ and $U \subseteq E$. We claim that

$$|S| \leq r_1(U) + r_2(E - U).$$

To see this, observe that $|S \cap U|$ is an independent set of both $\mathcal{I}_1$ and $\mathcal{I}_2$, in particular, $\mathcal{I}_1$, and $|S \cap (E - U)|$ is an independent set of both $\mathcal{I}_1$ and $\mathcal{I}_2$, in particular, $\mathcal{I}_2$. Therefore,

$$|S| = |S \cap U| + |S \cap (E - U)| \leq r_1(U) + r_2(E - U)$$

In particular, we have

$$\max_{S \in \mathcal{I}_1 \cap \mathcal{I}_2} \leq \min_{U \subseteq E} \{r_1(U) + r_2(E - U)\}.$$

It can be proved that in fact equality holds and this was proved by Edmonds:

Theorem 1.1 (Edmonds) *For matroids $M_1 = (E, \mathcal{I}_1$ and $M_2 = (E, \mathcal{I}_2)$ with rank functions r_1 and r_2, respectively:*

$$\max_{S \in \mathcal{I}_1 \cap \mathcal{I}_2} \leq \min_{U \subseteq E} \{r_1(U) + r_2(E - U)\}.$$

Given $M_1 = (E, \mathcal{I}_1)$ and $I \subseteq \mathcal{I}_1$, we define a bipartite directed graph $D_{M_1}(I)$ as follows:

$$D_{M_1}(I) = (E, A),$$

where $\forall y \in I$ and $x \in E - I$, if $I - y + x \in \mathcal{I}_1$, then add (y, x) to A. If we also have another matroid $M_2 = (E, \mathcal{I}_2)$, we define digraph $D_{M_1, M_2} = (E, A)$ as follows:

- $(y, x) \in A$ if $y \in I$ and $x \in E - I$, $I - y + x \in \mathcal{I}_1$;
- $(x, y) \in A$ if $y \in I$ and $x \in E - I$, $I - y + x \in \mathcal{I}_2$.

This is the union of $D_{M_1}(I)$ with reversed edge of $D_{M_2}(I)$. Let $X_1 = \{x \notin I | I + x \in \mathcal{I}_1\}$ and $X_2 = \{x \notin I | I + x \in \mathcal{I}_2\}$.

It can be proved that repeated application of the following algorithm finds a largest common independent set of the two matroids M_1 and M_2 and also gives a proof of Edmonds' matroid intersection theorem:

Matroid Intersection Algorithm

Input: given M_1, M_2, and I
Output: find $I' \in \mathcal{I}_1 \cap \mathcal{I}_2$ such that $|I'| > |I|$
find a shortest path p from X_1 to X_2
$I' \leftarrow I \Delta V(p)_J = (I - J) \cup (J - I)$
return I'

We are going to skip the proof of this algorithm and matroid intersection theorem. The algorithm can be generalized to solve the maximum weight common independent set of two matroids for the case that each element has a weight. Therefore, any problem that can be formulated as finding a maximum weight common independent set of two matroids can be solved in polynomial time. Edmonds proved this theorem using LP and duality. He characterized all inequalities defining the matroid intersection polytope, which is the convex-hull of independent sets common to two matroids (over the same ground set).

Unlike intersection of two matroids, finding a largest common independent sets of three or more matroids is NP-hard.

Theorem 1.2 *Given three matroids, $M_1 = (S, \mathcal{I}_1)$, $M2 = (S, \mathcal{I}_2)$, and $M3 = (S, \mathcal{I}_3)$, finding the largest common independent set of those three is NP-hard.*

Proof: Reduction from *Ham-Path*. Given a digraph $D = (V, A)$ with $s, t \in V$, the goal is to find a directed Hamiltonian s-t-path. We show this problem can be formulated as the largest common independent set of three matroids: $M_1 = (E, \mathcal{I}_1)$ is a graphic matroid where each independent set of $\mathcal{I}_1$ is a forest, $M_2 = (E, \mathcal{I}_2)$ is a partition matroid where $\mathcal{I}_2 = \{F \subseteq E : |\delta^-(v) \cap F| \leq 1 \quad v \neq s\}$, i.e. indegree is at most 1 for each node $v \neq s$, and $M_2 = (E, \mathcal{I}_3)$ is a partition matroid with $\mathcal{I}_3 = \{F \subseteq E : |\delta^+(v) \cap F| \leq 1 \quad v \neq t\}$, i.e. outdegree of each node is most 1 except for t.

■

2 Uncrossing technique, and Iterative Rounding and relaxation

So far we have seen different proofs for showing that certain polytopes (e.g. matching polytope) is integral. In this section (and in the future lectures) we introduce new techniques to analyse basic feasible solutions of other LP's. We use a different technique, called uncrossing and then show that a certain LP for minimum cost spanning tree is integral. We then introduce a different technique called iterative relaxation which combined with iterative rounding can be used to derive very good approximation algorithms for some NP-hard optimization problems.

Consider a given graph $G = (V, E)$ with $w : E \leftarrow \mathbb{R}^{\geq 0}$. We already know that the minimum spanning tree problem can be solved easily using greedy algorithms in polynomial time (it is also a special case of a matroid optimization). Here we consider the LP formulations for this problem. A natural LP relaxation for this problem is the following:

$$\begin{array}{rl} \min & \sum w_e x_e \\ \text{s.t.} & x(\delta(s)) \geq 1, \forall S \subset V \\ & x_e \geq 0. \end{array}$$

Clearly, any spanning tree satisfies all the constraints. Although this LP has exponentially many constraints, we can solve it using Ellipsoid algorithm since the separation oracle for it is just computing a minimum cut which can be solved in polynomial time. Unfortunately, this LP is not integral. For example if one considers the cycle on n nodes, C_n, then $x_e = \frac{1}{2}$ everywhere is a feasible solution to this LP whereas any integral solution needs $n-1$ edges.

So let's find another LP formulation for the MST problem. An important observation is that for any tree T and any set $S \subseteq V$: $E(T) \cap E(S) \leq |S| - 1$. Also, any spanning tree on V must have exactly $|V| - 1$ edges. Thus we obtain the following LP relaxation of MST:

$$\begin{array}{rl} \min & \sum w_e x_e \\ \text{s.t.} & x(E(S)) \leq |S| - 1, \forall S \subset V \\ & x(E(V)) = |V| - 1 \\ & x_e \geq 0. \end{array}$$

We refer to it as LP(2) from now on. Our goal is to prove the following theorem:

Theorem 2.1 *LP(2) is integral, i.e. every bfs of this LP is integral.*

Before that, we need to show that we can solve this LP in polynomial time, and for that it is sufficient to show that the separation oracle (to be used with Ellipsoid algorithm) is polynomial.

2.1 Separation Oracle

Given fractional solution x, the algorithm needs to find a node set $S \subset V$, such that $x(E(S)) > |S| - 1$. For that, itt is sufficient to check if

$$\min_{S \subset V} \{|S| - 1 - x(E(S)) < 0\}$$

We show how to check this using only $2|V| - 2$ calls to a procedure that solves an instance of min-cut. Fix a node $v_0 \in V$. For each $u \in V - v_0$ we build two min-cut instances:

- one checks the inequality for all set S containing v_0 but not u.
- the other checks the inequality for all sets containing u but not v_0.

So for a fixed node r there are a total of $2|V| - 2$ such tests. We describe the procedure for the first type; the second one is similar.

Build a digraph $G' = (V, E')$ in the following way. For each edge $ij \in E$ we add two directed edges (i, j) and (j, i) in E', each having weight $x_{ij}/2$. We also add arcs from each $v \in V - \{v_0, u\}$ to u with weight 1 and from v_0 to each $v \in V - r$ of weight $\sum_{e \in \delta(v)} (x_e/2)$. See Figure 1.

Consider any cut separating v_0 and u, say $(S, V - S)$. The edges of weight 1 contribute exactly $|S| - 1$ to this cut. For each edge $ij \in E$ its corresponding copy in E' contributes exactly $x(\delta(S))/2$ to this cut. The edges from v_0 to the rest of the vertices contribute $\sum_{v \notin S} \frac{x(\delta(v))}{2}$. Thus, the total weight of the edges of the cut is:

$$|S| - 1 + \frac{x(\delta(s))}{2} + \sum_{v \in V - S} \frac{x(\delta(v))}{2} = |S| - 1 + x(E(V)) - x(E(S)),$$

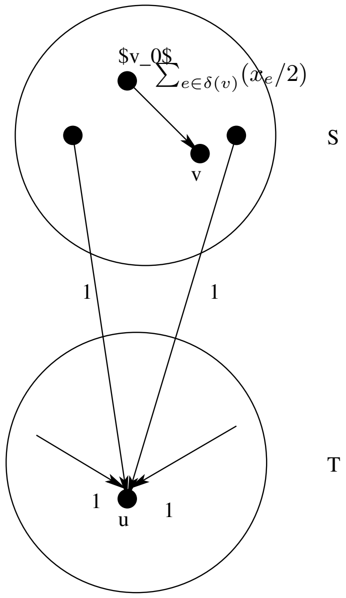

Figure 1: An illustration of edges in G' for each edge $ij \in G$

The total weight of the edges of the whole graph, i.e. $x(E(V))$ is a fixed given value. Therefore, if we find a minimum cut it also minimizes $|S| - 1 - x(E(S))$. If the minimum cut is negative then we have found a violated constraint. Therefore, we can solve this LP in polynomial time.

Next lecture we will complete the proof of Theorem 2.1 by showing that every bfs of this LP is integral.