CS 598CSC: Combinatorial Optimization	Lecture date: April 20, 2009
Instructor: Chandra Chekuri	Scribe: Jing Gao

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.

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

$$x(E[U]) \leq |U| - 1 \quad U \subseteq V$$

$$x \geq 0$$

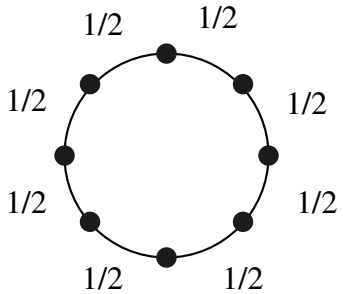
If we drop the constraint $x(E) = n - 1$, then we obtain the convex hull of the characterstic vectors of forests in G, called the forest polytope of G; note that forests are the independent sets in the graphic matroid of G.

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 $x(E) = n - 1$.



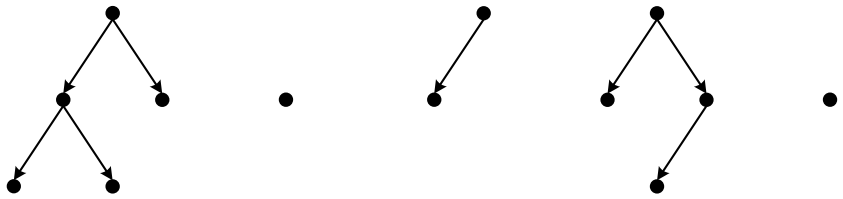
Exercise 1 *Show that even if we add the constraint $x(E) = n - 1$ to the cut-constraints, it still does not determine the spanning tree polytope.*

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.

2 Arborescences and Branchings

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.



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 2 *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.*

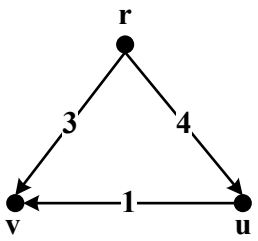
Branchings and Matroid Intersection: We saw earlier that branchings in a directed graph $D = (V, A)$ can be viewed as the common independent sets in the intersection of two matroids on A. Let $M_1 = (A, \mathcal{I}_1)$ where $\mathcal{I}_1 = \{A' \subseteq A \mid |A' \cap \delta^{-1}(v)| \leq 1 \quad \forall v \in V\}$. 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 $M_1 = \{A' \subseteq A \mid |A' \cap \delta^{-1}(v)| \leq 1 \quad \forall v \in V \backslash \{r\}$ 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.



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 3 *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 4 *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 which is comparable to the standard MST algorithms.

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.

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 5 *$\mathcal{P}_{r-arborescence}(D)$ is determined by*

$$x(a) \geq 0 \quad a \in A$$

$$x(\delta^{-1}(v)) = 1 \quad v \in V \backslash \{r\}$$

$$x(\delta^{-1}(U)) \geq 1 \quad U \subseteq V \backslash \{r\}$$

Proof: We give an iterated rounding based proof to show that the following set of inequalities

$$0 \leq x(a) \leq 1 \quad a \in A$$

$$x(\delta^{-1}(U)) \geq 1 \quad U \subseteq V \backslash \{r\}$$

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 $\mathcal{F} = \{U \in V \backslash \{r\} \mid x(\delta^{-1}(U)) = 1\}$ be the collection of tight sets.

Claim 6 *$\mathcal{F}$ is closed under intersections and unions.*

Claim 7 *Let $\mathcal{L}$ be a maximal laminar family in $\mathcal{F}$. Then span($\{\mathcal{X}(U) \mid U \in \mathcal{L}\}$)=span($\{\mathcal{X}(U) \mid U \in \mathcal{F}\}$).*

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. □

In fact, one can show that the system of inequalities is TDI and this implies a min-max result as well. See [1] for more details.

3 Arc-Disjoint Arborescences

A beautiful theorem of Edmonds is the following.

Theorem 8 *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 [2]). 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 $|\delta^+(U) \setminus A_1| \geq k - 1 \ \forall U \subset S, r \in U$. If we reach $S = V$, we are done.

The goal is 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
- $X \cup S \neq V$ and
- $|\delta^+(X) \setminus A_1| = k - 1$.

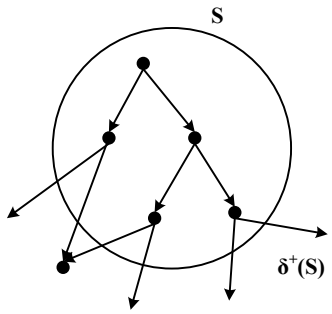


Figure 1: S and $\delta^+(S)$

We observe that if there are no critical sets, any arc $a \in \delta^+(S)$ can be used to augment A_1. 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 9 *If X, Y are critical and $X \cup Y \neq V$, then $X \cap Y$ and $X \cup Y$ are critical.*

Proof: Let $G' = G[A \backslash A_1]$, then $|\delta^+_{G'}(X)| = k-1$ and $|\delta_{G'}(Y)| = k-1$. 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,

$$k-1+k-1 \geq |\delta_{G'}(X \cap Y)| + |\delta_{G'}(X \cup Y)| \Rightarrow \delta_{G'}(X \cap Y) = \delta_{G'}(X \cup Y) = k-1.$$

□

Let X be a inclusion-wise maximal critical set.

Claim 10 *There exists an arc $(u,v) \in A$ s.t. $u \in S \setminus X$ and $v \in V \setminus (S \cup X)$.*

Proof: Note that $A_1 \cap \delta^+(S) = \emptyset$ since S, by defintion, is a set of reachable nodes in A_1. Since $|\delta^+(S \cup X)| \geq k$ (by assumption) and $|\delta^+_{G-A_1}(X)| = k-1$ (since X is critical), we have an arc as desired. □

Now let $A'_1 = A_1 + (u,v)$. The claim is that for all $U \in \mathcal{C}$, $|\delta^+(U) \setminus A'_1| \geq k-1$. Suppose not. Then let Y be such that $|\delta^+(Y) \backslash A'_1| < k-1$ but this implies that $|\delta^+(Y) \backslash A_1| = k-1$, that is Y is

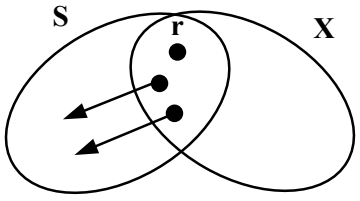


Figure 2: All arcs from $A_1 \cap \delta^+(S)$ go from X to S

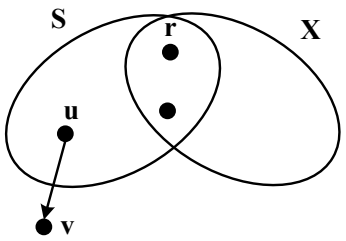


Figure 3: Claim 10

critical and $(u, v) \in \delta^+(Y)$. But consider Y, X both critical and $Y \cup X \neq V$ since $v \notin Y$, $v \notin X$. 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 [1] for details.

Edmonds derived the arc-disjoint r-arborescences theorem from a more general theorem on disjoint branchings. We refer the reader to [1].

References

[1] A. Schrijver. *Combinatorial Optimization.* Springer-Verlag Berlin Heidelberg, 2003.

[2] B. Korte and J. Vygen. *Combinatorial Optimization: Theory and Algorithms.* Springer-Verlag, Fourth Edition, 2008.