



The Polymatroid Steiner Problems*

G. CALINESCU

Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616

calinesc@iit.edu.

A. ZELIKOVSKY

Department of Computer Science, Georgia State University, Atlanta, GA 30303

alexz@cs.gsu.edu.

Received March 15, 2005; Accepted March 15, 2005

Abstract. The Steiner tree problem asks for a minimum cost tree spanning a given set of terminals $S \subseteq V$ in a weighted graph $G = (V, E, c)$, $c : E \rightarrow R^+$. In this paper we consider a generalization of the Steiner tree problem, so called Polymatroid Steiner Problem, in which a polymatroid $P = P(V)$ is defined on V and the Steiner tree is required to span at least one base of P (in particular, there may be a single base $S \subseteq V$). This formulation is motivated by the following application in sensor networks – given a set of sensors $S = \{s_1, \dots, s_k\}$, each sensor s_i can choose to monitor only a single target from a subset of targets X_i , find minimum cost tree spanning a set of sensors capable of monitoring the set of all targets $X = X_1 \cup \dots \cup X_k$. The Polymatroid Steiner Problem generalizes many known Steiner tree problem formulations including the group and covering Steiner tree problems. We show that this problem can be solved with the polylogarithmic approximation ratio by a generalization of the combinatorial algorithm of Chekuri et al. (2002).

We also define the Polymatroid directed Steiner problem which asks for a minimum cost arborescence connecting a given root to a base of a polymatroid P defined on the terminal set S . We show that this problem can be approximately solved by algorithms generalizing methods of Charikar et al. (1999).

Keywords: wireless sensor networks, Steiner trees, polymatroid, approximation algorithms

1. Introduction

This paper is motivated by the following lifetime problem in energy-constrained sensor networks. Let S be a set of (stationary) sensors which can be employed for monitoring a set X of (possibly moving) targets. Each sensor $s_i \in S$ can monitor at most one target chosen from $X_i \subseteq X$, a subset of targets visible to s_i . All targets are supposed to be simultaneously monitored by activated sensors which should continuously transmit collected data to the base possibly using multi-hop connections through other sensors, i.e., the activated sensors and the base should be connected with a Steiner tree. A *schedule* is a set of pairs (T, t) , where T a Steiner tree connecting sensors capable of monitoring all targets and t is time during which T is used. A simple energy model assumes that all sensors transmit with a single unit power and the Steiner tree is derived from the unit-disk graph. Then the energy consumption of each sensor is proportional to the time t during which it is used.

*A preliminary version of this paper appeared in ISAAC 2004

Target-monitoring sensor network lifetime problem. Find a schedule of the maximum total time span such that each sensor $s_i \in S$ does not exceed given initial energy supply b_i .

Previously, several versions of the communication ad-hoc network lifetime problems as well as sensor network lifetime problem have been explored in Calinescu et al. (2003) and Berman et al. (2004), respectively. A provably good approach to the lifetime problems consists of the following steps:

- (i) formulating the lifetime problem as a packing linear program (with exponentially many variables - the feasible trees),
- (ii) approximately solving the feasibility problem of the dual covering linear program linear program,
- (iii) applying the primal-dual algorithm (Garg and Konemann, 1998) for solving the primal packing linear program with almost the same approximation factor as for the feasibility problem of the dual covering linear program.

For details, we refer to Calinescu et al. (2003). Therefore, our focus here is on the following problem which solves the feasibility problem of the dual to the target-monitoring sensor network lifetime problem by putting appropriate costs on the edges of the graph (as in (Berman et al., 2004)).

Target-monitoring sensor covering problem. Find minimum cost Steiner tree spanning the base and a set of sensors capable of simultaneous monitoring of all targets.

Consider a bipartite graph with vertex set $B = S \cup X$ and edges connecting sensors with visible targets. Then any set of sensors capable of simultaneous monitoring of all targets is a set of S -endpoints of a matching completely covering X . Therefore, all minimal feasible sets of sensors form a set of bases of a matroid or, in general, a polymatroid. The following problem generalizes the Steiner tree problem in a very natural way.

Polymatroid Steiner problem (PSP). Given a graph $G = (V, E, c)$ with costs on edges and a polymatroid $P = P(V)$ on vertices of G , find minimum cost tree T within G spanning a base of P .

Equivalently, let $r : 2^V \rightarrow \{0, 1, \dots\}$ be a function on the set of vertices of G (called the *rank* function of the polymatroid $P(V)$) satisfying

- $r(A \cap B) + r(A \cup B) \leq r(A) + r(B)$, for all $A, B \subseteq V$ (submodularity)
- $r(\emptyset) = 0$
- if $A \subseteq B$ then $r(A) \leq r(B)$ (non-decreasing).

Then PSP asks for a minimum cost tree T spanning a maximum rank subset of V .

PSP generalizes various Steiner tree problem formulations. For example, setting the rank function $r(A) = |A \cap S|$, $A \subseteq V$ where $S \subseteq V$ is a given set of terminals, we

obtain the classical *Steiner tree problem* which asks for a minimum cost tree spanning terminals S . The *group Steiner tree problem* searches a tree spanning at least one vertex from each of given groups (subsets of vertices) $V_1, \dots, V_k \subseteq V$ – it is also an instance of PSP with the rank function $r(A) = |\{V_i | A \cap V_i \neq \emptyset\}|$. The *covering Steiner tree problem* (see (Konjevod and Ravi, 2000; Konjevod et al., 2002; Even et al., 2002)) generalizes the group Steiner tree problem by requiring at least k_i vertices from a group V_i to be spanned – the corresponding rank function is $r(A) = \sum_{i=1}^k \min\{k_i, |A \cap V_i|\}$. Finally, the target-monitoring sensor covering problem is reduced to PSP by adding a single auxiliary target matching the base and setting $r(A)$ equal to the maximum number of targets that A can match.

The complexity of PSP can be derived from the recent papers (Halperin et al., 2003; Halperin and Krauthgamer, 2003). Halperin and Krauthgamer (Halperin and Krauthgamer, 2003) showed that for every fixed $\epsilon > 0$, Group Steiner Tree problem admits no $\log^{2-\epsilon} n$ -approximation, unless NP has quasi-polynomial Las Vegas algorithms.

When applying primal-dual algorithm of Garg and Konemann (1998) it is necessary to solve weighted target-monitoring sensor covering problem, i.e., the version in which each sensor has a certain weight and the cost of the solution is the sum of weights of chosen sensors rather than just number of chosen sensors. PSP does not seem to generalize the node-weighted version, but it can be reduced to the following:

Polymatroid directed Steiner problem (PDSP). Given a directed graph $G = (V, E, c)$ with costs on edges and a polymatroid $P = P(V)$ on vertices of G . Find minimum cost arborescence T within G connecting a given root $s \in V$ to all vertices of at least one base of P .

The PDSP generalizes the Directed Steiner Tree Problem which can be obtained from PDSP by setting rank of a subset to its size. The best known approximation algorithm, due to Charikar et al. (1999), has running time $O(n^i k^{2i})$ and approximation ratio $i^2(i-1)k^{1/i}$ for any fixed integer $i > 1$. Thus, in polynomial time, their approximation ratio is $O(k^\epsilon)$, while in quasipolynomial time ($O(n^{c \lg n})$, for constant c) they achieve a polylogarithmic approximation ratio of $O(\log 3k)$.

The simple energy model for the target-monitoring sensor network lifetime problem can be enhanced by allowing sensors to choose the power of transmission. Then the energy consumption of each sensor s_i is proportional to the cost of the hop connecting s_i to the next sensor on the path to the base as well as time t during which T is used. This model straightforwardly reduces the target-monitoring sensor network lifetime problem to PDSP. It turns out (as in Calinescu et al. (2003)) that the version of PDSP used for solving the feasibility problem of the dual of the lifetime program is equivalent to PDSP – the cost of each edge e should be multiplied by the weight of the beginning of e .

In the next section we show how to adapt the algorithm of Charikar et al. (1999) to solve PDSP with almost the same approximation factor. The Section 3 is devoted to generalization of the algorithm of Chekuri et al. (2002) to solve PSP with polylogarithmic approximation ratio.

2. The polymatroid directed Steiner Tree problem

In this section we establish performance bounds of the generalization of the algorithm of Charikar et al. (1999) to the Polymatroid Directed Steiner Tree problem.

First we introduce a version of Polymatroid Directed Steiner Trees which allows the presentation of the algorithm. Without loss of generality, we assume that the directed graph is complete and $c(u, v)$ equals the minimum cost path from u to v . All the edges and trees in this section are directed. Given a set of nodes $X \subseteq V$, we denote by r_X the rank function with X contracted; that is $r_X(Z) = r(X \cup Z) - r(X)$. The rank function of a polymatroid is submodular implying that if $X \subseteq Y \subseteq V$, then for any $Z \subseteq V$, we have $r_X(Z) \geq r_Y(Z)$. Let $PDST(k, v, X)$ denote the problem of finding the minimum-cost tree T rooted at v with $r_X(V(T)) \geq k$, where $V(T)$ is the vertex set of the tree T . Note that the new version is in fact equivalent with the standard version as r_X is another submodular rank function. One can think of r_X as the residual rank function.

We ensure that all the nodes v with $r(v) > 0$ do not have outgoing edges by “duplicating” v as follows: if v has outgoing edges and positive rank, we introduce another node v' with $r(v') = 0$, replace every edge incident to v by a corresponding edge incident to v' , and introduce the edge (v', v) of cost 0. This allows us to write $r(T) := r(V(T)) = r(L(T))$ for any directed tree T with vertex set $V(T)$ and leafs $L(T)$ assuming T 's root has rank 0 (as will be the case for all our trees: even the original root is duplicated if it has positive rank).

Let $c(T)$ be the cost of the directed tree T (the sum of the costs of the edges of T). Then we define the *density* of the tree T with respect to vertex set X as $d_X(T) = c(T)/r_X(T)$.

An l -level tree is a tree where no leaf is more than l edges away from the root. Helvig et al. (2001) give:

Lemma 1. *For all $l \geq 1$ and any tree $T \subseteq G$, there exists an l -level tree $T' \subseteq G$ with $L(T') = L(T)$ and $c(T') \leq l \cdot |L(T)|^{\frac{1}{l}} c(T)$.*

An earlier claim from Zelikovsky (1997) that $c(T') \leq |L(T)|^{\frac{1}{l}}$, used in Charikar et al. (1999), has a gap in the proof.

2.1. The algorithm

We describe the Charikar et. al. algorithm Charikar et al. (1999), adapted for our more general problem. The recursive algorithm $A_i(k, v, X)$ appears in Figure 1. The parameters passed down are the desired rank k , the desired root v , the maximum height i , and a pointer to a vector X describing the vertices already in the tree. The algorithm returns a pointer to an i -level tree $T = T_i(k, v, X)$ rooted at v satisfying $r_X(T) \geq k$, or \emptyset if no such tree exists. The base case is $i = 0$ (as opposed to $i = 1$ in Charikar et al. (1999)), as a trivial reduction from Set Cover shows that it is NP-Hard to compute a minimum weight star (a directed tree with one level) of given rank in the polymatroid setting. When $i = 1$, the recursive algorithm copies the vector X and uses the copy during its execution, while when $i = 1$ the vector X is not modified.

Input: k, v, X

Output: An i -level tree $T = T_i(k, v, X)$ rooted at v with $r_X(T) \geq k$

0. Let $L(v)$ be the vertices reachable from v . If $r_X(L(v)) < k$, return \emptyset .
 1. If $i = 0$, return the tree with no edges and vertex set $\{v\}$ if $r_X(v) \geq k$, or \emptyset if $r_X(v) < k$.
 2. $j \leftarrow 1$; $k_j \leftarrow k$; $X_j \leftarrow X$
 3. while $k_j > 0$
 - 3.1 $T_{BEST}(j) \leftarrow \emptyset$
 - 3.2 for each vertex $u \in V$ and each $k', 1 \leq k' \leq k_j$
 - 3.2.1 $T' \leftarrow A_{i-1}(k', u, X_j) \cup \{(v, u)\}$
 - 3.2.2 if $d_{X_j}(T_{BEST}(j)) > d_{X_j}(T')$ then $T_{BEST}(j) = T'$
 - 3.3 $k_{j+1} \leftarrow k_j - r_{X_j}(T_{BEST}(j))$; $X_{j+1} \leftarrow X_j \cup L(T_{BEST}(j))$; $j \leftarrow j + 1$
 4. Return $\cup_{q=1}^{j-1} T_{BEST}(q)$
-

Figure 1. Algorithm $A_i(k, v, X)$.

Note the following invariant of the algorithm: $r_X(X_j) = k - k_j$. Indeed, $r_X(X_1) = r(X) - r(X) = 0 = k - k_1$, and $r_X(X_{j+1}) = r(X_{j+1}) - r(X) = r(X_{j+1}) - r(X_j) + r(X_j) - r(X) = r_{X_j}(T_{BEST}(j)) + r_X(X_j) = (k_j - k_{j+1}) + (k - k_j) = k - k_{j+1}$. In particular, we have $r_X(T_i(k, v, X)) \geq k$, so the returned solution is valid.

Let $T_{OPT}^{(i)}(k, v, X)$ be an optimum i -level tree solving PDST(k, v, X). The lemma below is the counterpart of Lemma 3 of Charikar et al. (1999), and is proved by the same method. It is interesting to see where submodularity plays a crucial role in the proof.

Lemma 2. For all $i \geq 1$, each tree $T_{BEST}(j)$ chosen by algorithm $A_i(k, v, X)$ satisfies

$$d_{X_j}(T_{BEST}(j)) \leq i \cdot d_{X_j}(T_{OPT}^{(i)}(k_j, v, X_j)).$$

Proof: The proof is by induction on i , with the base case $i = 1$ being immediate since $A_0(k', v, X_j)$ is called for all $k' \leq k_j$ and $v \in V$ and A_0 always returns the optimum.

Assume now the statement of the lemma holds for all k, v, X , and $i - 1$. $T_{OPT}^{(i)}(k_j, v, X_j)$ consists of several edges (v, u_p) and subtrees T_p rooted at u_p , such that $r_{X_j}(\cup_p T_p) \geq k_j$. Submodularity implies that $\sum_p r_{X_j}(T_p) \geq r_{X_j}(\cup_p T_p)$, and therefore by an averaging argument and renumbering, we have

$$\frac{c(v, u_1) + c(T_1)}{r_{X_j}(T_1)} \leq d_{X_j}(T_{OPT}^{(i)}(k_j, v, X_j)) \quad (1)$$

Consider the execution of the algorithm $A_{i-1}(k_j, u_1, X_j)$. Trees R_1, R_2, \dots are selected in this order, and let $Q_p = \cup_{q=1}^p R_q$, with $Q_0 = \emptyset$ for convenience. Now, when $A_{i-1}(k', u_1, X_j)$ is called for $k' = r_{X_j}(Q_p)$, then $Q_p \cup \{(v, u_1)\}$ is returned and is a candidate for $T_{BEST}(j)$.

Our goal is to wisely choose such p and show that $Q_p \cup \{(v, u_1)\}$ has an appropriate density:

$$\frac{c(v, u_1) + c(Q_p)}{r_{X_j}(Q_p)} \leq i \frac{c(v, u_1) + c(T_1)}{r_{X_j}(T_1)} \quad (2)$$

which together with Eq. (1) would imply the theorem. We pick p to be the smallest integer such that $r_{X_j}(Q_p) \geq r_{X_j}(T_1)/i$, which implies

$$\frac{c(v, u_1)}{r_{X_j}(Q_p)} \leq i \frac{c(v, u_1)}{r_{X_j}(T_1)} \quad (3)$$

It remains to prove that

$$\frac{c(Q_p)}{r_{X_j}(Q_p)} \leq i \frac{c(T_1)}{r_{X_j}(T_1)} \quad (4)$$

as this equation together with Eq. (3) implies Eq. (2).

By the induction hypothesis, we have for all $q \leq p$

$$\frac{c(R_q)}{r_{Q_{q-1} \cup X_j}(R_q)} \leq (i-1) \frac{c(T_1)}{r_{Q_{q-1} \cup X_j}(T_1)} \quad (5)$$

since R_q is $T_{\text{BEST}}(q)$ when executing $A_{i-1}(r_{X_j}(T_1), u_1, X_j)$. We picked p such that $r_{X_j}(Q_{p-1}) < \frac{1}{i} r_{X_j}(T_1)$, and therefore $r_{X_j \cup Q_{p-1}}(T_1) = r(X_j \cup Q_{p-1} \cup T_1) - r(X_j \cup Q_{p-1}) \geq r(X_j \cup T_1) - r(X_j \cup Q_{p-1}) = (r(X_j \cup T_1) - r(X_j)) - (r(X_j \cup Q_{p-1}) - r(X_j)) = r_{X_j}(T_1) - r_{X_j}(Q_{p-1}) \geq \frac{i-1}{i} r_{X_j}(T_1)$.

Submodularity of the rank function implies that for all $q \leq p$,

$$r_{X_j \cup Q_{q-1}}(T_1) \geq r_{X_j \cup Q_{p-1}}(T_1) \geq \frac{i-1}{i} r_{X_j}(T_1) \quad (6)$$

and, therefore,

$$\begin{aligned} \sum_{q=1}^p c(R_q) &\leq \sum_{q=1}^p r_{Q_{q-1} \cup X_j}(R_q) (i-1) \frac{c(T_1)}{r_{Q_{q-1} \cup X_j}(T_1)} \\ &\leq i \frac{c(T_1)}{r_{X_j}(T_1)} \sum_{q=1}^p r_{Q_{q-1} \cup X_j}(R_q) \end{aligned} \quad (7)$$

But

$$\begin{aligned} r_{X_j}(Q_p) &= r(X_j \cup Q_p) - r(X_j) \\ &= r(X_j \cup Q_p) - r(X_j \cup Q_{p-1}) + r(X_j \cup Q_{p-1}) - \dots - r(X_j \cup Q_0) \\ &= \sum_{q=1}^p r_{X_j \cup Q_{q-1}}(R_q) \end{aligned}$$

Thus Eq. (7) implies Eq. (4), finishing the proof of Lemma 2. \square

Theorem 1. For $k = r_X(V)$ and for every $i > 1$, $v \in V$ and $X \subseteq V$, the algorithm $A_i(k, v, X)$ provides an $i^3 k^{1/i}$ approximation to $\text{PDST}(k, v, X)$ in time $O(n^{i+1} k^{2i+2} q(n))$, where $q(n)$ is the time an oracle returns $r_X(v)$ for an arbitrary $X \subseteq V$.

Proof: The proof follows closely Charikar et al. (1999). Note that for all j $T_{\text{OPT}}(k, v, X)$ is a valid solution for $\text{PDST}(k_j, v, X_j)$ since

$$\begin{aligned} r_{X_j}(T_{\text{OPT}}(k, v, X)) &= r(X_j \cup T_{\text{OPT}}(k, v, X)) - r(X_j) \\ &\geq r(X \cup T_{\text{OPT}}(k, v, X)) - r(X) - (r(X_j) - r(X)) \\ &\geq r_X(T_{\text{OPT}}) - r_X(X_j) = k - (k - k_j) = k_j, \end{aligned}$$

where we used the invariant of the procedure $A_i(k, v, X)$ and $X \subseteq X_j$. Therefore $c(T_{\text{OPT}}(k_j, v, X_j)) \leq c(T_{\text{OPT}}(k, v, X))$ and we also have by Lemma 2 $c(T_{\text{OPT}}^{(i)}(k_j, v, X_j)) \leq i k_j^{1/i} c(T_{\text{OPT}}(k_j, v, X_j))$. By Lemma 1, we have

$$\begin{aligned} d_{X_j}(T_{\text{BEST}}(j)) &\leq i \cdot d_{X_j}(T_{\text{OPT}}^{(i)}(k_j, v, X_j)) \\ &\leq i \cdot i \cdot k_j^{1/i} \cdot \frac{c(T_{\text{OPT}}(k, v, X))}{k_j}. \end{aligned}$$

So we know that

$$\frac{c(T_{\text{BEST}}(j))}{r_{X_j}(T_{\text{BEST}}(j))} \leq i^2 k_j^{1/i} \frac{c(T_{\text{OPT}}(k, v, X))}{k_j}.$$

Let f be the value of j at the end of $A_i(k, v, X)$. Using the invariant and $k = r_X(V)$ we must have $k_f = k - r_X(X_f) \geq 0$ since $k_f < 1$ (which signals the exit from the **while** loop at line 3), we conclude $k_f = 0$. The algorithm gives that for all $j = 1, 2, \dots, f - 1$ we have $r_{X_j}(T_{\text{BEST}}(j)) = k_j - k_{j+1}$, and we obtain

$$c(T_{\text{BEST}}(j)) \leq i^2 c(T_{\text{OPT}}(k, v, X)) (k_j - k_{j+1}) \frac{k_j^{1/i}}{k_j}$$

Summing over j (cf. Lemma 1 of Charikar et al. (1999)) and using $k_1 = k$ and $k_f = 0$ and the fact that the function $f(x) = x^{1/i-1}$ is non-increasing, we obtain

$$\begin{aligned} c(T_i(k, v, X)) &\leq i^2 c(T_{\text{OPT}}(k, v, X)) \sum_j (k_{j+1} - k_j) \frac{k_j^{1/i}}{k_j} \\ &\leq i^2 c(T_{\text{OPT}}(k, v, X)) \int_0^k x^{1/i-1} dx \\ &= i^3 k^{1/i} c(T_{\text{OPT}}(k, v, X)) \end{aligned}$$

The procedure A_i invokes A_{i-1} at most nk^2 times, and the bound on the running time follows. \square

If the input is a star (directed tree with one level), the algorithm becomes the Greedy algorithm for the Submodular Set Covering problem of (Wolsey, 1982). Simplifying and following the proof of Theorem 1 one gets.

Corollary 1. *The approximation ratio of the Greedy Algorithm applied to the Submodular Set Covering problem is at most $1 + \ln k$.*

Wolsey (1982) investigated this problem and has shown that the Greedy algorithm has in fact an approximation ratio of H_q , where $q = \max_v r(v)$. Wolsey's ratio is no worse than the one from the corollary, and is sometimes better.

3. The polymatroid Steiner tree problem

In this section we give the solution of the (undirected) Polymatroid Steiner tree Problem. The first choice for solving this problem is to generalize the linear-program based algorithms of Garg et al. (2000), Konjevod et al. (2002), and Zosin and Khuller (2002). The corresponding linear programs have polynomial-time separation oracle. Unfortunately, it is truly cumbersome to apply rounding to the linear programs of PSP. Instead, our algorithm for PSP relies on the combinatorial approximation algorithm for Group Steiner Tree of Chekuri et al. (2002).

The Chekuri et al. algorithm is obtained by modifying the Charikar et al. (1999) algorithm and is applied after the graph metric has been replaced by a tree metric (Bartal, 1996, 1998; Charikar et al., 1999; Fakcharoenpho et al., 2003), losing a factor of $O(\log n)$ in approximation ratio. Thus we also assume the input is a rooted tree.

Chekuri et al. (2002) preprocess the tree to decrease the depth and degree. This preprocessing approximately preserves the cost of any solution, so we can apply it to Polymatroid Steiner Tree as well. Precisely, every set of leafs $L \subseteq L(T)$ induces a subtree T_l consisting of the union of all paths from the root to the leafs in L . If A and B are two trees with the same root and the same set of leafs, the tree B is a α -faithful representation of the tree A if

$$\forall L \subseteq L(A) : c(A_L) \leq c(B_L) \leq \alpha c(A_L).$$

The preprocessing is stated in the following theorem of Chekuri et al. (2002).

Theorem 2. *Given a tree T with n leafs and integer parameters α and $\beta > 2$, there is a linear time algorithm to transform T into a $O(\alpha)$ -faithful tree T' with height $O(\log_\alpha n + \log_{\beta/2} n)$ and maximum degree $O(\beta)$.*

Thus, by losing a factor of $O(\alpha \log n)$ ($\log n$ comes from embedding the original metric in a tree metric), we can assume that the instance for Polymatroid Steiner Tree is a tree with height $O(\log_\alpha n + \log_{\beta/2} n)$ and maximum degree $O(\beta)$.

For the reader familiar with the Chekuri et al. (2002) paper below you'll find the correspondence between the notions used by this previous paper and our notions. Their theorems and proofs "translate" to PST, some of them directly.

- $w(T) \leftarrow c(T)$
- $z' \leftarrow k$
- $r' \leftarrow v$
- $T_{r'}$ (excluding the leafs already reached) $\leftarrow X$ (the leafs already reached)
- $T_{aug} \leftarrow T_{BEST}(j)$
- $z^{res} \leftarrow k_j$
- $\gamma(T) \leftarrow d_X(T)$
- $cover \leftarrow \cup_{i=1}^j T_{BEST}(j)$
- $m(T) \leftarrow r_X(T)$
- $m(cover) \leftarrow r_X X_j$
- remove groups covered by T_{aug} from $T^{res} \leftarrow X_{j+1} \leftarrow X_j \cup L(T_{BEST}(j))$
- $cover_h \leftarrow j_h$
- Group-Steiner $^*(T_{r'}^{res}, z^{res}) \leftarrow T_{OPT}(k_j, v, X_j)$

The Modified-Group-Steiner algorithm of Chekuri et al. (2002), as adapted for Polymatroid Steiner Tree for trees is described in Figure 2. The recursive procedure $M(k, v, X)$ uses an integer parameter λ as the basis for the geometric search. We use T_v to denote the subtree rooted at v and C_v the set of children of v . $h(T')$ denotes the height of a subtree T' .

In the practical implementations, one can continue the while loop from Step 3 instead of stopping in Step 3.4, returning the lowest density tree from $\cup_{q=1}^{j-1} T_{BEST}(q)$ and $\cup_{q=1}^{j_h-1} T_{BEST}(q)$, where j_h is the value of j at the first moment $r_X(X_j) \geq k/(h(T_v) + 1)$.

The proof of the following lemma follows closely the proof of Lemma 3.3 of Chekuri et al. (2002), except that in the base case (leafs) an oracle call must be made, which we assume takes time $q(n)$, an bookkeeping is slightly more difficult. We include the proof for completeness.

Input: k, v, X

Output: A tree $T = T(k, v, X)$ rooted at v with $r_X(T) \geq k$, or \emptyset if no such tree exists

0. If $r_X(L(T_v)) < k$, return \emptyset .
 1. If v is a leaf, return the tree with no edges and vertex set $\{v\}$
 2. $j \leftarrow 1$; $k_j \leftarrow k$; $X_j \leftarrow X$
 3. while $k_j > 0$
 - 3.1 $T_{BEST}(j) \leftarrow \emptyset$
 - 3.2 for each vertex $u \in C_v$ and each k' power of $(1 + \lambda)$ in $\left[\frac{k_j}{deg(v)(1+\lambda)(1+\lambda)}, k_j \right]$
 - 3.2.1 $T' \leftarrow M(k', u, X_j) \cup \{(v, u)\}$
 - 3.2.2 if $d_{X_j}(T_{BEST}(j)) > d_{X_j}(T')$ then $T_{BEST}(j) = T'$
 - 3.3 $k_{j+1} \leftarrow k_j - r_{X_j}(T_{BEST}(j))$; $X_{j+1} \leftarrow X_j \cup L(T_{BEST}(j))$; $j \leftarrow j + 1$
 - 3.4 If $r_X(X_j) \geq k/(h(T_v) + 1)$, then return $\cup_{q=1}^{j-1} T_{BEST}(q)$
-

Figure 2. Algorithm $M(k, v, X)$.

Lemma 3. *Let Δ be the maximum degree of the tree and $b = \Delta(1 + 1/\lambda)(1 + \lambda)$. The running time of $M(k, v, X)$ is $O((n + q(n))\mu^{h(T_v)})$ where $\mu = b \cdot h(T_v) \cdot \log k \cdot \Delta \cdot \log_{1+\lambda} b$.*

Proof: For a given k_j , the number of recursive calls done in line 3.2.1 is $\Delta \log_{1+\lambda} b$. A successful call (that is, not returning \emptyset) has $r_{X_j}(T') \geq \frac{k_j}{b} \frac{1}{h(T_v)}$ and therefore $k_{j+1} \leq k_j - \frac{k_j}{b} \frac{1}{h(T_v)} = k_j(1 - \frac{1}{bh(T_v)}) \leq k_j e^{-\frac{1}{bh(T_v)}}$. As soon as $k_j < 1$ the procedure stops, and therefore the number of times the **while** loop in line 3 is executed is at most $bh(T_v) \log k$. Thus the total number of recursive calls done by $M(k, v, X)$ is μ .

The time spent by $M(k, v, X)$ is at most $(n + q(n))$ times the number of recursive calls. With $i = h(T_v)$, we obtain the recurrence for the running time of $M(k, v, X)$ $t(i) \leq \mu(t(i-1) + n + q(n))$, with the base case $t(0) = n + q(n)$. Solving the recurrence gives $t(i) \leq (n + q(n))(\mu^i + \sum_{q=1}^i \mu^q)$ and for all $\mu \geq 2$ we get $t(i) = O((n + q(n))\mu^i)$. \square

The main lemma needed for establishing the approximation ratio is the equivalent of Lemma 3.4 of Chekuri et al. (2002).

Lemma 4.

$$d_{X_j}(T_{\text{BEST}}(j)) \leq (1 + \lambda)^{2h(T_v)} h(T_v) d_{X_j}(T_{\text{OPT}}(k_j, v, X_j))$$

Proof: The proof follows closely (Chekuri et al., 2002). The base case for us requires some extra arguments. It is also interesting to see where the submodularity of the rank function is used.

We use $\gamma^* = d_{X_j}(T_{\text{OPT}}(k_j, v, X_j))$. The proof is by induction on $h(T_v)$, the height of the subtree rooted at v . Both the base case $h(T_v) = 1$ and the general case need the following argument.

Let u_1, u_2, \dots, u_d be the children of v in $T_{\text{OPT}}(k_j, v, X_j)$, and $T_i, 1 \leq i \leq d$, be the subtree of $T_{\text{OPT}}(k_j, v, X_j)$ rooted at u_i . Thus $r_{X_j}(\cup_{i=1}^d T_i) \geq k_j$. We divide the set $1, 2, \dots, d$ into the set B giving ‘‘big’’ trees: those i with $r_{X_j}(T_i) \geq \frac{k_j}{\deg(v)(1+1/\lambda)}$, and the set S giving ‘‘small’’ trees: those i with $r_{X_j}(T_i) < \frac{k_j}{\deg(v)(1+1/\lambda)}$. Then $r_{X_j}(\cup_{i \in S} T_i) \leq \sum_{i \in S} r_{X_j}(T_i) < \frac{k_j}{1+1/\lambda}$ and, therefore

$$\begin{aligned} \sum_{i \in B} r_{X_j}(T_i) &\geq r_{X_j}(\cup_{i \in B} T_i) \\ &\geq r(\cup_{i \in B} T_i \cup X_j) - r(X_j) \\ &= r(\cup_{i \in B} T_i \cup (\cup_{i \in S} T_i) \cup x_j) - r(X_j) \\ &\quad - (r(\cup_{i \in B} T_i \cup (\cup_{i \in S} T_i) \cup x_j) - r(\cup_{i \in B} T_i \cup X_j)) \\ &= r_{X_j}(\cup_{i=1}^d T_i) - r_{\cup_{i \in B} T_i \cup X_j}(\cup_{i \in S} T_i) \\ &\geq k_j - r_{X_j}(\cup_{i \in S} T_i) \\ &\geq k_j - \frac{k_j}{1 + 1/\lambda} \\ &= \frac{k_j}{1 + \lambda}, \end{aligned}$$

where we used $r_{X_j}(\cup_{i \in S} T_i) \geq r_{\cup_{i \in B} T_i \cup X_j}(\cup_{i \in S} T_i)$, which follows from submodularity. By an averaging argument, there is a big tree (which we renumber T_1) such that

$$\begin{aligned} \frac{c(v, u_1) + c(T_1)}{r_{X_j}(T_1)} &\leq \frac{\sum_{i \in B} c(v, u_i) + c(T_i)}{\sum_{i \in B} r_{X_j}(T_i)} \\ &\leq (1 + \lambda) \frac{c(T_{\text{OPT}}(k_j, v, X_j))}{k_j} \\ &= (1 + \lambda) \gamma^*. \end{aligned} \quad (8)$$

Let z be the power of $(1 + \lambda)$ such that $z \leq r_{X_j}(T_1) < (1 + \lambda)z$ and note that z is in the range of powers of $(1 + \lambda)$ considered in Line 3.2 of the algorithm. Therefore $M(z, u_1, X_j)$ is called.

In the base case, u_1 is a leaf, and therefore a candidate for $T_{\text{BEST}}(j)$ is the tree T' with only the edge (v, u_1) , having density $d_{X_j}(T') = \frac{c(v, u_1)}{r_{X_j}(T_1)} \leq (1 + \lambda) \gamma^*$. Thus the base case of induction holds.

In the general case, let R_1, R_2, \dots, R_p be the trees picked as T_{BEST} during the execution of $M(z, u_1, X_j)$. Let $Q_q = \cup_{i=1}^q R_i$ (with $Q_0 = \emptyset$ for convenience) and let $h_1 = h(T_{u_1})$. Induction gives for $i \in \{1, 2, \dots, p\}$

$$\begin{aligned} \frac{c(R_i)}{r_{X_j \cup Q_{i-1}}(R_i)} &\leq (1 + \lambda)^{2h_1} h_1 d_{X_j \cup Q_{i-1}}(T_{\text{OPT}}(z - r_{X_j}(Q_{i-1}), u_1, X_j \cup Q_{i-1})) \\ &\leq (1 + \lambda)^{2h_1} h_1 \frac{c(T_1)}{r_{X_j \cup Q_{i-1}}(T_1)}, \end{aligned} \quad (9)$$

where the first inequality uses $r_{X_j}(Q_{i-1}) = \sum_{q=1}^{i-1} r_{X_j \cup Q_{q-1}}(R_q)$ and the second inequality follows from the fact that $r_{X_j \cup Q_{i-1}}(T_1) \geq z - r_{X_j}(Q_{i-1})$ (which follows from $r_{X_j}(T_1 \cup Q_{i-1}) \geq r_{X_j}(T_1) \geq z$), and, therefore, T_1 is a candidate for $T_{\text{OPT}}(z, u_1, X_j \cup Q_{i-1})$.

By the return condition of the algorithm, $r_{X_j}(Q_{i-1}) < \frac{1}{h_1+1}z$ for all $i \leq p$, and therefore

$$\begin{aligned} r_{X_j \cup Q_{i-1}}(T_1) &\geq r_{X_j}(T_1) - r_{X_j}(Q_{i-1}) \\ &\geq z \left(1 - \frac{1}{h_1+1}\right) \\ &= \frac{h_1}{h_1+1} z. \end{aligned}$$

Together with Eq. (9), we obtain that, for all $i \leq p$,

$$\begin{aligned} \frac{c(R_i)}{r_{X_j \cup Q_{i-1}}(R_i)} &\leq (1 + \lambda)^{2h_1} (h_1 + 1) \frac{c(T_1)}{z} \\ &\leq (1 + \lambda)^{2h_1+1} (h_1 + 1) \frac{c(T_1)}{r_{X_j}(T_1)}, \end{aligned} \quad (10)$$

since z was chosen such that $z \leq r_{X_j}(T_1) < (1 + \lambda)z$. Thus,

$$\sum_{q=1}^p c(R_q) \leq \sum_{q=1}^p r_{X_j \cup Q_{q-1}}(R_q)(1 + \lambda)^{2h_1+1}(h_1 + 1) \frac{c(T_1)}{r_{X_j}(T_1)}. \quad (11)$$

But

$$\begin{aligned} r_{X_j}(Q_p) &= r(X_j \cup Q_p) - r(X_j) \\ &= r(X_j \cup Q_p) - r(X_j \cup Q_{p-1}) + r(X_j \cup Q_{p-1}) - \dots - r(X_j \cup Q_0) \\ &= \sum_{q=1}^p r_{X_j \cup Q_{q-1}}(R_q), \end{aligned}$$

and therefore,

$$\frac{\sum_{q=1}^p c(R_q)}{r_{X_j}(Q_p)} \leq (1 + \lambda)^{2h_1+1}(h_1 + 1) \frac{c(T_1)}{r_{X_j}(T_1)}. \quad (12)$$

Note that p was picked such that $r_{X_j}(Q_p) \geq \frac{1}{h_1+1}z \geq \frac{1}{(h_1+1)(1+\lambda)}r_{X_j}(T_1)$ and therefore

$$\frac{c(v, u_1)}{r_{X_j}(Q_p)} \leq (h_1 + 1)(1 + \lambda) \frac{c(v, u_1)}{r_{X_j}(T_1)}. \quad (13)$$

Combining the previous equation with Eq. (12) we obtain

$$\frac{c(u, v) + \sum_{q=1}^p c(R_q)}{r_{X_j}(Q_p)} \leq (1 + \lambda)^{2h_1+1}(h_1 + 1) \frac{c(v, u_1) + c(T_1)}{r_{X_j}(T_1)} \quad (14)$$

Using Eq. (8) and the fact that $h(T_v) \geq h(T_{u_1}) + 1 = h_1 + 1$, we obtain

$$\frac{c(u, v) + \sum_{q=1}^p c(R_q)}{r_{X_j}(Q_p)} \leq (1 + \lambda)^{2h(T_v)}h(T_v)\gamma^* \quad (15)$$

Thus, the tree T' returned by $M(z, u_1, X_j)$, which is a candidate for $T_{\text{BEST}}(j)$ in the execution of $M(k, v, X)$, satisfies $d_{X_j}(T') \leq (1 + \lambda)^{2h(T_v)}h(T_v)\gamma^*$. \square

Chekuri et al. (2002) choose (and we do the same) $\alpha = (\log n)^\epsilon$, $\beta = \log n$, $1/\lambda = \log n$. Assuming the oracle computation is polynomial time, plugging this values in Lemma 3 yields a polynomial-time algorithm.

Using exactly the method from Theorem 3.5 and Corollary 3.6 of Chekuri et al. (2002) we prove:

Theorem 3. *There is a combinatorial polynomial-time $O(\frac{1}{\epsilon} \cdot \frac{1}{\log \log n} \cdot (\log n)^{1+\epsilon} \log k)$ -approximation algorithm for Polymatroid Steiner Tree on trees with n nodes, where k is*

the desired rank. For general undirected graphs, there is a combinatorial polynomial-time $O(\frac{1}{\epsilon} \cdot \frac{1}{\log \log n} \cdot (\log n)^{2+\epsilon} \log k)$ -approximation algorithm

Proof: The running time follows from the discussion above. Also, note that the height of the tree is $O(\frac{1}{\epsilon} \frac{\log n}{\log \log n})$ and then computation shows that Lemma 3 gives:

$$d_{X_j}(T_{\text{BEST}}(j)) \leq O\left(\frac{1}{\epsilon} \frac{\log n}{\log \log n}\right) d_{X_j}(T_{\text{OPT}}(k_j, v, X_j)).$$

Standard set-cover analysis (as in Chekuri et al. (2002) or Zosin and Khuller (2002)) gives then a $O(\frac{1}{\epsilon} \frac{\log n}{\log \log n} \log k)$ approximation on the trees of reduced height and degree. Another at most $O((\log n)^\epsilon)$ factor is lost due to preprocessing of the tree, based on Theorem 3, obtaining the $O(\frac{1}{\epsilon} \cdot \frac{1}{\log \log n} \cdot (\log n)^{1+\epsilon} \log k)$ -approximation algorithm for Polymatroid Steiner Tree on trees. To obtain the approximation ratio for general undirected graphs, use the algorithm of Fakcharoenpho et al. (2003) to replace a graph metric by the tree metric. \square

4. Conclusion

Motivated by applications in wireless sensor networks (when sensors can monitor only a single target), we have introduced the Polymatroid (Directed and Undirected) Steiner Tree Problems (PSP). These problems asks for a (directed) Steiner tree spanning a subset of terminals of sufficiently large rank. The undirected version of PSP generalizes all known versions of the Group Steiner Tree Problem and is shown to be solved by a generalization of the algorithm from Chekuri et al. (2002) with polylogarithmic approximation ratio. The directed version of PSP is a generalization of the Directed Steiner Tree Problem as well as Polymatroid Set Cover Problem. We show that this problem can be approximately solved by methods generalizing Charikar et al. (1999).

Acknowledgment

The first author would like to thank Peng-Jun Wan for suggesting some applications to wireless networks.

References

- Y. Bartal, "Probabilistic approximation of metric spaces and its algorithmic applications," in *FOCS*, 1996.
- Y. Bartal, "On approximating arbitrary metrics by tree metrics," in *STOC*, 1998.
- P. Berman, G. Calinescu, C. Shah, and A. Zelikovsky, "Power efficient monitoring management in sensor networks," in *Proc. IEEE Wireless Communications and Networking Conference*, 2004.
- G. Calinescu, S. Kapoor, A. Olshevsky, and A. Zelikovsky, "Network lifetime and power assignment in ad-hoc wireless networks," in *Proc. 11th European Symposium on Algorithms*, 2003.
- M. Charikar, C. Chekuri, A. Goel, and S. Guha, "Approximating a finite metric by a small number of tree metrics," in *STOC*, 1999.
- M. Charikar, C. Chekuri, T. Cheung, Z. Dai, A. Goel, S. Guha, and M. Li, "Approximation algorithms for directed steiner problems," *Journal of Algorithms*, vol. 33, no. 1, pp. 73–91, 1999.

- C. Chekuri, G. Even, and G. Kortsarz, "A combinatorial approximation algorithm for the group Steiner problem," 2002. Submitted for publication. Available on the web.
- G. Even, G. Kortsarz, and W. Slany, "On network design: Fixed charge flows and the covering Steiner problem," in *SWAT 2002*, pp. 318–329, 2002.
- J. Fakcharoenphol, S. Rao, and K. Talwar, "A tight bound on approximating arbitrary metrics by tree metrics," in *STOC*, 2003.
- N. Garg, G. Konjevod, and R. Ravi, "A polylogarithmic approximation algorithm for the Group Steiner Tree problem," *J. Algorithms*, vol. 37, no. 1, pp. 66–84, 2000.
- N. Garg and J. Konemann, "Faster and simpler algorithms for multicommodity flow and other fractional packing problems," in *IEEE Symposium on Foundations of Computer Science*, 1998, pp. 300–309.
- A. Gupta and A. Srinivasan, "On the covering steiner problem," in *em Proc. Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, 2003, pp. 244–251.
- E. Halperin, G. Kortsarz, R. Krauthgamer, A. Srinivasan, and N. Wang, "Integrality ratio for Group Steiner Trees and Directed Steiner Trees," in *14th Annual ACM-SIAM Symposium on Discrete Algorithms*, Jan 2003, pp. 275–284.
- E. Halperin and R. Krauthgamer, "Polylogarithmic inapproximability," in *Proceedings of the 35th ACM Symposium on Theory of Computing*, 2003, pp. 585–594.
- C. H. Helvig, G. Robins, and A. Zelikovsky, "Improved approximation scheme for the group Steiner problem," *Networks*, vol. 37, no. 1, pp. 8–20, 2001.
- G. Konjevod, R. Ravi, and A. Srinivasan. "Approximation algorithms for the covering Steiner problem," *Random Structures and Algorithms*, vol. 20, no. 3, pp. 465–482, 2002. Preliminary version by Konjevod and Ravi in SODA 2000.
- G. Konjevod and R. Ravi, "An approximation algorithm for the covering Steiner problem," in *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 2000, pp. 338–344.
- L.A. Wolsey, "Analysis of the greedy algorithm for the submodular set covering problem," *Combinatorica*, vol. 2, no. 385–392, 1982.
- A. Zelikovsky, "A series of approximation algorithms for the acyclic Directed Steiner Tree problem," *Algorithmica*, vol. 18, 1997.
- L. Zosin and S. Khuller, "On Directed Steiner Trees," in *SODA*, 2002, pp. 59–63.