The Polymatroid Steiner Problems ⋆

G. Calinescu[1] and A. Zelikovsky[2]

[1] Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616.
E-mail: calinesc@iit.edu.
[2] Department of Computer Science Georgia State University, Atlanta, GA 30303,
E-mail: alexz@cs.gsu.edu.

Abstract. The Steiner tree problem asks for a minimum cost tree spanning a given set of terminals $S \subseteq V$ in a weighted graph $G = (V, E, c)$, $c : E \to R^+$. 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 $S = \{s_1, \ldots, s_k\}$, 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 $X = X_1 \cup \ldots \cup X_k$. 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. [7].
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 [6].
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 [4] and [3], 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 [11] 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 [4]. 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 [3]).

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

- $r(A \cap B) + r(A \cup B) \leq r(A) + r(B), \quad \text{for all } A, B \subseteq V$ (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) $V_1, \ldots, V_k \subseteq V$ – 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 [17, 16, 8]) 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 [13, 14]. Halperin and Krauthgamer [14] 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 [11] 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. [6], has running time $O(n^i k^{2i})$ and approximation ratio $i2(i-1)k^{1/i}$ 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 [4]) that 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 [6] to solve PDSP with almost the same approximation factor. The section 3 is devoted to generalization of the algorithm of [7] 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, Chekuri, Cheung, Dai, Goel, Guha, and Li [6] 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. Robins and Zelikovsky [15] 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 [19] that $c(T') \leq |L(T)|^{\frac{1}{l}}$, used in [6], has a gap in the proof.

2.1 The Algorithm

We describe the Charikar et. al. algorithm [6], 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 $i = 1$ (as opposed to $i = 2$ in the original version), as it is NP-Hard to compute a minimum density bunch (a tree with only one vertex with outdegree larger than one) 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.

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.

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 $\cup_{q=1}^{j-1} T_{BEST}(q)$

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

Let $T^{(i)}_{OPT}(k, v, X)$ be an optimum i-level tree solving $PDST(k, v, X)$. The lemma below is the counterpart of Lemma 3 of [6], and is proved by the same method. It is interesting to see where submodularity plays a crucial role in the proof.

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

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

.

Proof. The proof is by induction on i, with the base case $i = 0$ being immediate.

Assume the statement of the lemma holds for all k, v, X, and $i-1$. $T^{(i)}_{OPT}(k_j, v, X_j)$ 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

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

Consider the execution of the algorithm $A_{i-1}(k_j, u_1, X_j)$. Trees $R_1, R_2, \ldots$ are selected in this order, and let $Q_p = \cup_{q=1}^{p} R_q$. 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 is has an appropriate density

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

which together with Equation 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

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

It remains to prove that

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

as this equation together with Equation 3 implies Equation 2.

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

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

since R_q is $T_{BEST}(q)$ 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 $q \leq p$,

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

and, therefore,

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

But

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

Thus Equation 7 implies Equation 4, finishing the proof of Lemma 2. ■

Theorem 1. *For every $i > 1$, $k \geq 0$, $v \in V$ and $X \subseteq V$, the algorithm $A_i(k, v, X)$ provides an $i3k^{1/i}$ approximation to $PDST(k, v, X)$ 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 [6]. Note that $T_{OPT}(k, v, X)$ is a valid solution for $PDST(k_j, v, X_j)$ since

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

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

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

So we know that

$$\frac{c(T_{BEST}(j))}{r_{X_j}(T_{BEST}(j))} \leq i2k_j^{1/i} \frac{c(T_{OPT}(k, v, X)}{k_j}$$

Since $r_{X_j}(T_{BEST}(j)) = k_{j+1} - k_j$, we obtain

$$c(T_{BEST}(j)) \leq i2c(T_{OPT}(k, v, X))\,(k_{j+1} - k_j)\,\frac{k_j^{1/i}}{k_j}$$

Summing over j (cf. Lemma 1 of [6]) and using $k_1 = k$ and $k_{j+1} < 0$, we obtain

$$\begin{aligned} c(T_i(k, v, X)) &\leq i2c(T_{OPT}(k, v, X)) \sum_j (k_{j+1} - k_j) \frac{k_j^{1/i}}{k_j} \\ &\leq i2c(T_{OPT}(k, v, X) \int_0^k x^{1-1/i} dx \\ &= i3k^{1/i} c(T_{OPT}(k, v, X)) \end{aligned}$$

The procedure A_i invokes A_{i-1} at most $nk2$ times, and the bound on the running time follows. ■

If the input is a star (directed tree with one level), the algorithm becomes the Greedy algorithm for the Submodular Set Covering problem [18].

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

Wolsey [18] investigated this problem and has shown that the Greedy algorithm has in fact a slightly better approximation ratio H_q, where $q = \max_{v \mid r(v)>0} r(v)$.

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, Konjevod, and Ravi [10], Konjevod, Ravi, and Srinivasan [16], and Zosin and Khuller [20]. 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, Even, and Korsatz [7].

The Chekuri et. al. algorithm is obtained by modifying the Charikar et. al. algorithm [6], and is applied after the graph metric has been replaced by a tree metric [1, 2, 5, 9], losing a factor of $O(\log n)$ in approximation ratio. Thus we also assume the input is a rooted tree.

Chekuri et. al. [7] 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

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

The preprocessing is stated in the following theorem of Chekuri et. al. [7]:

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. paper [7], 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.

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

The Modified-Group-Steiner algorithm of Chekuri et. al. [7], 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 $\cup_{q=1}^{j-1} T_{BEST}(q)$

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 $\left[\frac{k_j}{deg(v)(1+1/\lambda)(1+\lambda)}, k_j\right]$
      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 $\cup_{q=1}^{j-1} T_{BEST}(q)$

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

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 is exactly as the proof of Lemma 3.3 of [7], except that in the base case (leafs) an oracle call must be made, which we assume takes time $q(n)$.

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 $O((n+q(n))\alpha^{h(T_v)})$ where $\alpha = b \cdot h(T_v) \cdot \log k \cdot \Delta \cdot \log_{1+\lambda} b$.*

The main lemma needed for establishing the approximation ratio is a the equivalent of Lemma 3.4 of [7].

Lemma 4.

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

Proof. The proof follows closely [7]. 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 $\gamma^* = d_{X_j}(T_{OPT}(k_j, v, X_j))$. 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 $u_1, u_2, \ldots, u_d$ be the children of v in $T_{OPT}(k_j, v, X_j)$, and T_i, $1 \leq i \leq d$, be the subtree of $T_{OPT}(k_j, v, X_j)$ rooted at u_i. Thus $r_{X_j}\left(\cup_{i=1}^{d} T_i\right) \geq k_j$. We divide the set $1, 2, \ldots d$ 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,

$$\begin{aligned}\sum_{i\in B} r_{X_j}(T_i) &\geq r_{X_j}(\cup_{i\in B}T_i)\\ &\geq k_j - d\frac{k_j}{deg(v)(1+1/\lambda)}\\ &\geq \frac{k_j}{1+\lambda}\end{aligned}$$

By an averaging argument, there is a big tree (which we renumber T_1) such that

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

.

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 $T_{BEST}(j)$ 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 $R_1, R_2, \ldots, R_p$ be the trees picked as T_{BEST} during the execution of $M(z,u_1,X_j)$. Let $Q_q = \cup_{i=0}^{q} R_i$ and let $h_1 = h(T_{u_1})$. Induction gives for $i \in \{1,2,\ldots,p\}$

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

where 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 $r_{X_j}(T_1) \geq z$ and the submodularity of r), and, therefore, T_1 is a candidate for $T_{OPT}(z,u_1,X_j\cup Q_{i-1})$.

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

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

Together with Equation 9, we obtain:

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

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

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

But

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

and, therefore,

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

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

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

Combining the previous equation with Equation 12 we obtain

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

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

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

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

Chekuri et. al. [7] choose (and we do the same) $\alpha = (\log n)^{\epsilon}$, $\beta = \log n$, $1/\lambda = \log n$). Assuming the oracle computation is polynomial time, the proof of Theorem 3.5 and Corollary 3.6 of [7] gives our corresponding statement:

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*

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 [7] 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 [6].

References

1. Y. Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *FOCS*, 1996.
2. Y. Bartal. On approximating arbitrary metrics by tree metrics. In *STOC*, 1998.
3. 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.
4. 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.
5. M. Charikar, C. Chekuri, A. Goel, and S. Guha. Approximating a finite metric by a small number of tree metrics. In *STOC*, 1999.
6. Moses Charikar, Chandra Chekuri, Toyat Cheung, Zuo Dai, Ashish Goel, Sudipto Guha, and Ming Li. Approximation Algorithms for Directed Steiner Problems. *Journal of Algorithms*, 33(1):73–91, 1999.
7. C. Chekuri, G. Even, and G. Kortsarz. A combinatorial approximation algorithm for the group Steiner problem, 2002. Submitted for publication. Available on the web.
8. G. Even, G. Kortsarz, and W. Slany. On network design: fixed charge flows and the covering Steiner problem. In *SWAT 2002*, pages 318–329.
9. Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *STOC*, 2003.
10. N. Garg, G. Konjevod, and R. Ravi. A polylogarithmic approximation algorithm for the group Steiner tree problem. *J. Algorithms*, 37(1):66–84, 2000.
11. Naveen Garg and Jochen Konemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In *IEEE Symposium on Foundations of Computer Science*, pages 300–309, 1998.
12. 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) pages 244–251, 2003.
13. 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*, pages 275–284, January 2003.
14. E. Halperin and R. Krauthgamer. Polylogarithmic inapproximability. In *Proceedings of the 35th ACM Symposium on Theory of Computing*, pages 585–594, 2003.
15. C. H. Helvig, , G. Robins, and A. Zelikovsky. Improved approximation scheme for the group Steiner problem. *Networks*, 37(1):8–20, 2001.

16. G. Konjevod, R. Ravi, and A. Srinivasan. Approximation algorithms for the covering Steiner problem. *Random Structures and Algorithms*, 20(3):465–482, 2002. Preliminary version by Konjevod and Ravi in SODA 2000.
17. Goran Konjevod and R. Ravi. An approximation algorithm for the covering Steiner problem. In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 338–344. Society for Industrial and Applied Mathematics, 2000.
18. L.A. Wolsey. Analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2:385–392, 1982.
19. A. Zelikovsky. A series of approximation algorithms for the acyclic directed Steiner tree problem. *Algorithmica*, 18, 1997.
20. Leonid Zosin and Samir Khuller. On directed Steiner trees. In *SODA*, pages 59–63, 2002.