On the Covering Steiner Problem*

Anupam Gupta[1] and Aravind Srinivasan[2]

[1] Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15232, USA. anupamg@cs.cmu.edu

[2] Department of Computer Science and University of Maryland Institute for Advanced Computer Studies, University of Maryland at College Park, College Park, MD 20742, USA. srin@cs.umd.edu

Abstract. The *Covering Steiner* problem is a common generalization of the k-MST and Group Steiner problems. An instance of the Covering Steiner problem consists of an undirected graph with edge-costs, and some subsets of vertices called *groups*, with each group being equipped with a non-negative integer value (called its *requirement*); the problem is to find a minimum-cost tree which spans at least the required number of vertices from every group. When all requirements are equal to 1, this is the Group Steiner problem.

While many covering problems (e.g., the covering integer programs such as set cover) become easier to approximate as the requirements increase, the Covering Steiner problem remains at least as hard to approximate as the Group Steiner problem; in fact, the best guarantees previously known for the Covering Steiner problem were *worse* than those for Group Steiner as the requirements became large. In this work, we present an improved approximation algorithm whose guarantee equals the best known guarantee for the Group Steiner problem.

1 Introduction

We present an improved approximation algorithm for the Covering Steiner problem. This is a covering problem has the following property that goes against the norm for covering problems: its approximability cannot get better as the covering requirements increase. Thus the approximability of the general Covering Steiner problem is at least as high as for the case of all unit requirements, which is just the Group Steiner problem. In this work, we improve on the current-best approximation algorithms for the Covering Steiner problem given by Even et al. [3] and Konjevod et al. [9]. Our results match the approximation guarantee of the current-best randomized algorithm for the Group Steiner problem due to Garg et al. [5] (see the paper of Charikar et al. [2] for a deterministic algorithm). A suitable melding of a randomized rounding approach with a deterministic "threshold rounding" method leads to our result.

* This material is based upon work supported in part by the National Science Foundation under Grant No. 0208005 to the second author. Part of this work was done while the authors were at Lucent Bell Laboratories, 600-700 Mountain Avenue, Murray Hill, NJ 07974-0636, USA.

Let $G = (V, E)$ be an undirected graph with a non-negative cost function c defined on its edges. Let a family $\mathcal{G} = \{g_1, g_2, \ldots, g_k\}$ of k subsets of V be given; we refer to these sets $g_1, g_2, \ldots, g_k$ as *groups*. For each group g_i, a non-negative integer $r_i \leq |g_i|$ is also given, called the *requirement* of the group. The Covering Steiner problem on G is to find a minimum-cost tree in G that contains at least r_i vertices from each group g_i; the special case of unit requirements (i.e., $r_i = 1$ for all i) corresponds to the Group Steiner tree problem. We denote the number of vertices in G by n, the size of the largest group by N, and the largest requirement of a group by K. Logarithms in this paper will be to the base two unless specified otherwise.

As in the paper of Garg et al. [5], we focus on the case where the given graph $G = (V, E)$ is a tree, since the notion of probabilistic tree embeddings [1] can be used to reduce an arbitrary instance of the problem to a instance on a tree. Specifically, via the result of Fakcharoenphol et al. [4], a ρ–approximation algorithm on tree-instances implies an $O(\rho \log n)$–approximation algorithm for arbitrary instances. In fact, we can assume that the instance is a *rooted* tree instance where the root vertex must be included in the tree that we output; this assumption can be discharged by running the algorithm over all choices of the root and picking the best tree.

For the special case of the Group Steiner tree problem where $K = 1$, the current-best approximation bound for tree instances is $O((\log k) \cdot (\log N))$. For the Covering Steiner problem, the current-best approximation algorithm for tree instances is $O((\log k + \log K) \cdot (\log N))$ [3,9]; also, an approximation bound of

$$O\left(\frac{(\log N) \cdot \log^2 k}{\log(2(\log N) \cdot (\log k)/\log K)}\right)$$

is also presented in [9], which is better if $K \geq 2^{a(\log k)^2}$ where $a > 0$ is a constant. (As mentioned above, we need to multiply each of these three approximation bounds by $O(\log n)$ to obtain the corresponding results for general graphs.)

Note that these current-best approximations *get worse* as the requirements increase, i.e., as K increases. This is unusual for covering problems, where the approximability *gets better* as the coverage requirements increase. This is well-known, for instance, in the case of covering integer programs which include the set cover problem; the "multiple coverage" version of set cover is one where each element of the ground set needs to be covered by at least a given number of sets. In particular, the approximation ratio improves from logarithmic to $O(1)$ (or even $1 + o(1)$) for families of such problems where the minimum covering requirement B grows as $\Omega(\log m)$, when m is the number of constraints (see, e.g., [10]). In light of these results, it is natural to ask whether the approximation guarantee for the Covering Steiner problem can be better than that for the Group Steiner problem.

This question can easily be answered in the negative; indeed, given a rooted tree instance of the Group Steiner problem and an integer K, we can create an instance of the Covering Steiner problem as follows: increase the requirement of every group from 1 to K, connect $K-1$ dummy leaves to the root with edge-cost

zero and add these leaves to all the groups. It is easily seen that any solution to this Covering Steiner instance can be transformed to a solution to the original Group Steiner instance with no larger cost, and that the two instances have the same optimal solution value. Therefore, the Covering Steiner problem is at least as hard to approximate as the Group Steiner problem. (This fact was pointed out to us by Robert Krauthgamer.)

We are thus led to the question: can the Covering Steiner problem be approximated as well as the Group Steiner problem? The following theorem answers this question in the affirmative:

Theorem 1. *There is a randomized polynomial-time approximation algorithm for the covering Steiner problem which, with high probability, produces an approximation of: (i)* $O((\log N) \cdot (\log k))$ *for tree instances, and (ii)* $O((\log n) \cdot (\log N) \cdot (\log k))$ *for general instances.*

This implies an improvement of $\Theta((\log n)/\log\log n)$ can be obtained over previous results in some situations; indeed, this is achieved when, say, $k = \log^{2+\Theta(1)} n$ and $K = n^{\Theta(1)}$. (The reason we take $k \gg \log^2 n$ in this example is that the problem on trees is easily approximable to within k; so, if k were small, we would have a good approximation algorithm anyway.)

The bounds for tree instances are essentially the best possible in the following asymptotic sense: the paper of Halperin and Krauthgamer [7] shows that, for any constant $\epsilon > 0$, an $O((\log(n+k))^{2-\epsilon})$-approximation algorithm for the Covering Steiner problem implies that $NP \subseteq ZTIME[\exp((\log n)^{O(1)})]$. Furthermore, we adopt a natural linear programming (LP) relaxation considered by Garg et al. [5] and Konjevod et al. [9]; it has been shown that the integrality gap of this relaxation for tree-instances of Group Steiner is $\Omega((\log k)\cdot(\log N)/\log\log N)$ [6].

1.1 Our Techniques

Our approach to solve the Covering Steiner problem is to iteratively round an LP relaxation of the problem suggested by Konjevod et al. [9]. Given a partial solution to the problem, we consider the fractional solution of the LP for the current residual problem, and extend the partial solution using either a randomized rounding approach and a direct deterministic approach.

Informally, in order to do better than the approach of Konjevod et al. [9], the main technical issue we handle is as follows. Let OPT denote the optimal solution value of a given tree instance. In essence, each iteration of [9] constructs adds an expected cost of $O(\mathsf{OPT} \cdot \log N)$ to the solution constructed so far, and reduces the total requirement by a constant fraction (in expectation). Since the initial total requirement is at most $k \cdot K$, we thus expect to run for $O(\log k + \log K)$ iterations, resulting in a total cost of $O(\mathsf{OPT} \cdot (\log k + \log K) \log N)$ with high probability. To eliminate the $\log K$ term from their approximation guarantee, we show, roughly, that every iteration has to satisfy one of two cases: (i) a good fraction of the groups have their requirement cut by a constant factor via a threshold rounding scheme, while paying only a cost of $O(\mathsf{OPT})$, or (ii) a randomized rounding scheme is expected to fully cover a constant fraction of

the groups. A *chip game* presented in Section 2 is used to bound the number of iterations where case (i) holds; Janson's inequality [8] and some deterministic arguments are used for case (ii). We do not attempt to optimize our constants.

2 A Chip Game

Consider the following *chip game*. We initially have chips arranged in k groups, with each group i having some number $n_i^{(init)} \leq K$ of chips. Chips are iteratively removed from the groups in a manner described below; a group is called *active* if the number of chips in it is nonzero. The game proceeds in rounds as follows. Let n_i denote the number of chips in group i at the start of some round. Letting A denote the number of groups currently active, we may choose any set of at least $\lceil A/2 \rceil$ *active* groups; for each of these chosen groups i, we remove *at least* $\lceil n_i/2 \rceil$ chips, causing the new number of chips in group i to at most $\lfloor n_i/2 \rfloor$.) Informally, we choose roughly half of the currently active groups, and halve their sizes. (In the analysis below, it will not matter that these two constants are $1/2$ each; any two constants $a, b \in (0, 1)$ will lead to the same asymptotic result.)

Due to these chips being removed, some groups may become inactive (i.e., empty); once a group has become inactive, it may be removed from consideration. The game proceeds until there are no chips left. The following lemma will be useful later.

Lemma 1. *The* maximum *number of rounds possible in the above chip game is* $O((\log k) \cdot (\log K))$.

Before we prove this lemma, let us note that this bound is tight, and $\Theta((\log k) \cdot (\log K))$ rounds may be required. Suppose all the groups start off with exactly K chips. We first repeatedly keep choosing the *same* set of $\lceil k/2 \rceil$ groups for removing chips, and do this until all these groups become inactive; we need $\Theta(\log K)$ rounds for this. We are now left with about $k/2$ active groups. Once again, we repeatedly keep removing chips from the same set of about $k/4$ active groups, until all of these become inactive. Proceeding in this manner, we can go for a total of $\Theta((\log k) \cdot (\log K))$ rounds.

Proof (Lemma 1). To bound the number of rounds, we proceed as follows. We first round up the initial number of chips $n_i^{(init)}$ in each group to the closest power of 2 greater than it. Next, we may assume that in each round, each chosen active group i has its number of chips n_i reduced by the least amount possible; i.e., to *exactly* $\lfloor n_i/2 \rfloor$. This ensures that each active group always has a number of chips that is a power of two. We can now modify the game into an equivalent format. We initially start with $1 + \log n_i^{(init)}$ chips in each group i; once again, a group is active if and only if its number of chips is nonzero. Now, in any round, we choose at least half of the currently-active groups, and remove *one* chip from each of them. Note that this simple "logarithmic transformation" does not cause the maximum number of rounds to decrease, and hence we can analyze the game on this transformed instance. Let N_1 be the number of rounds in which at least

$k/2$ groups are active. In each of these rounds, at least $k/4$ chips get removed. However, since the total number of chips is at most $k(1 + \log K)$, we must have $N_1 \leq 4(1 + \log K)$. Proceeding in this manner, we see that the total number of rounds is $O((\log k) \cdot (\log K))$, as claimed.

3 Algorithm and Analysis

As mentioned in the introduction, we will assume that the input consists of a *rooted* tree, where the root r must be included in the output solution. Furthermore, we make the following assumptions without loss of generality: every vertex belonging to a group is a leaf, and every leaf belongs to some group.

3.1 The Basic Approach

As in previous papers on the Covering Steiner and Group Steiner problems, the broad idea of the algorithm is to proceed in iterations, with each iteration producing a subtree rooted at r that provides some coverage not provided by the previous iterations. This process is continued until the required coverage is accomplished, and the union of all the trees constructed is returned as output.

Consider a generic iteration; we will use the following notation. Let r'_i denote the residual requirement of group g_i; call g_i *active* if and only if $r'_i > 0$, and let k' be the number of groups currently active. The leaves of g_i already covered in previous iterations are removed from g_i; abusing notation, we will refer to this shrunk version of the group g_i also as g_i. All the edges chosen in previous iterations have their cost reduced to zero: we should not have to "pay" for them if we choose them again. For any non-root node u, let $\mathsf{pe}(u)$ denote the edge connecting u to its parent; for any edge e not incident on the root, let $\mathsf{pe}(e)$ denote the parent edge of e. Finally, given an edge $e = (u, v)$ where u is the parent of v, both $T(v)$ and $T(e)$ denote the subtree of G rooted at v.

The following integer programming formulation for the residual problem was proposed by Konjevod et al. [9], in which there is an indicator variable x_e for each edge e, to say whether e is chosen or not. The constraints are: (i) for any active group g_i, $\sum_{j \in g_i} x_{\mathsf{pe}(j)} = r'_i$; (ii) for any edge e and any active group g_i, $\sum_{j \in (T(e) \cap g_i)} x_{\mathsf{pe}(j)} \leq r'_i x_e$; and (iii) for any edge e not incident on the root r, $x_{\mathsf{pe}(e)} \geq x_e$. Finally, the objective is to minimize $\sum_e c_e x_e$. By allowing each x_e to lie in the real interval $[0, 1]$ as opposed to the set $\{0, 1\}$, we get an LP relaxation. Since chosen edges have their cost reduced to zero for all future iterations, it is easy to see that the optimal value of this LP is a lower bound on OPT, the optimal solution value to the original Covering Steiner instance.

Each iteration will start with the residual problem, and solve the above LP optimally; it will then round this fractional solution as described below in Section 3.2. An interesting point to note is that as pointed out in [9], the integrality gap of the relaxation considered can be quite large: when we write the LP relaxation for the original instance, the integrality gap can be arbitrarily close to K. Hence it is essential that we satisfy the requirements *partially* in each iteration, then *re-solve* the LP for the residual problem, and continue.

3.2 Rounding

We will henceforth use $\{x_e\}$ denote an optimal solution to the LP relaxation; we now show how to round it to partially cover some of the residual requirements. In all of the discussion below, only currently active groups will be considered. For any leaf j, we call $x_{\mathsf{pe}(j)}$ the *flow into* j. Note that the total flow into the leaves of a group g_i is r'_i. For $\alpha \geq 1$, we define a group g_i to be (p, α)-*covered* if a total flow of at least pr'_i goes into the elements (i.e., leaves) of g_i which receive individual flow values of at least $1/\alpha$. An α-*scaling* of the solution $\{x_e\}$ is the scaled-up solution $\{\hat{x_e}\}$ where we set $\hat{x_e} \leftarrow \min\{\alpha\, x_e, 1\}$, for all edges e.

The iteration is as follows. Define $\mathcal{G}_1$ be the set of active groups that are $(1/2, 4)$-covered. We will consider two cases, based on the size of $\mathcal{G}_1$.

Case I: $|\mathcal{G}_1| \geq k'/2$. In this case, we simply perform a 4-scaling, and *pick* the edges that are rounded up to 1. The solution returned in this case is just the connected subtree consisting of the picked edges that contains the root. Note that every group in $\mathcal{G}_1$ has at least half of its requirement covered by this process. Indeed, since any such group g_i is $(1/2, 4)$-covered, it must have at least $r'_i/2$ of its member leaves receiving flows of value at least $1/4$. Thus, by this rounding, at least half of the currently active groups have their requirements reduced by at least half. Now Lemma 1 implies that the total number of iterations where Case I holds is at most $O((\log k) \cdot (\log K))$. We pay a cost of at most $4 \cdot \mathsf{OPT}$ in each such iteration, and hence

$$\text{cost}(\text{Case I iterations}) = O((\log k) \cdot (\log K) \cdot \mathsf{OPT}). \tag{1}$$

(Let us emphasize again that throughout the paper, OPT refers to the cost of the optimal solution for the *original* Covering Steiner instance.) Now suppose Case I does not hold, and thus we consider:

Case II: $|\mathcal{G}_1| < k'/2$. In this case, let $\lambda = c' \log N$, for a sufficiently large absolute constant c'. Let $\mathcal{G}_2$ be the set of active groups that are *not* $(3/4, \lambda)$-covered, and let $\mathcal{G}_3$ be the set of active groups that do not lie in $\mathcal{G}_1 \cup \mathcal{G}_2$.

We now use a modification of the rounding procedure used previously by Garg et al. [5] and Konjevod et al. [9]. For every edge e, define $x'_e = \min\{\lambda \cdot x_e, 1\}$ to be the solution scaled up by λ. For every edge e incident on the root, *pick* e with probability x'_e. For every other edge e with its parent denoted f, *pick* e with probability x'_e/x'_f. This is performed for each edge independently. Let H denote the subgraph of G induced by the edges that were picked, and let T denote the connected subtree of H containing the root r; this is returned as the solution in this case. Crucially, we now set the costs of all chosen edges to 0, so as to not count their costs in future iterations. It is easy to verify that the probability of the event $e \in T$ for some edge e is x'_e; now linearity of expectation implies that

$$\mathbf{E}[\text{cost}(T)] \leq \lambda \cdot \mathsf{OPT}. \tag{2}$$

We next analyze the expected coverage properties of this randomized rounding. For any group g_i, let g'_i be the subset of elements of g_i which have individual

in-flow values of at most $1/\lambda$, and let f_i denote the total flow into the elements of g'_i. We want a lower bound on X_i, the number of elements of g'_i that get covered by the above randomized rounding; for this, we plan to employ Janson's inequality [8].

It is easy to see that $\mu_i \doteq \mathbf{E}[X_i] = f_i\lambda$. Suppose $j, j' \in g'_i$ are two leaves in g'_i. We say that $j \sim j'$ if and only if **(i)** $j \neq j'$ and **(ii)** the least common ancestor of j and j' in G is not the root r. If $j \sim j'$, let $\mathsf{lca}(j, j')$ denote the least common ancestral *edge* of j and j' in T'. To use Janson's inequality, define

$$\Delta_i = \sum_{j,j' \in g'_i:\ j \sim j',\, x_{\mathsf{lca}(j,j')} > 0} \frac{x'_{\mathsf{pe}(j)} x'_{\mathsf{pe}(j')}}{x'_{\mathsf{lca}(j,j')}}.$$

If $f_i \geq r'_i/4$, then Janson's inequality shows that if c' is large enough, then

$$\Pr[X_i \geq r'_i] \geq 1 - \exp\left(-\Omega\left(\frac{\mu_i}{2 + \Delta_i/\mu_i}\right)\right). \tag{3}$$

Furthermore, a calculation from [9] can be used to show that $\Delta_i \leq O((r'_i)^2 \log^2 N)$. Plugging this into (3), along with the facts that $\mu_i = f_i\,\lambda$, and that $\lambda = c' \log N$, it follows that there is an absolute constant $c'' \in (0, 1)$ such that

$$\text{if } f_i \geq r'_i/4, \text{ then } \Pr[X_i \geq r'_i] \geq c''. \tag{4}$$

Recall that $\mathcal{G}_2$ consisted of all the groups g_i which were *not* $(3/4, \lambda)$-covered. Hence, at least $r'_i/4$ flow into g_i goes into leaves with in-flow at most $1/\lambda$, and hence f_i is at $r'_i/4$. This satisfies the condition in (4), and hence the expected number of groups in $\mathcal{G}_2$ that get *all* of their requirement covered in this iteration is at least $c'' \cdot |\mathcal{G}_2|$.

Next consider any $g_i \in \mathcal{G}_3$; by definition, g_i must be $(3/4, \lambda)$-covered, but *not* $(1/2, 4)$-covered. In other words, if we consider the leaves of g_i whose individual flow values lie in the range $[1/\lambda, 1/4]$, the total flow into them is at least $(3/4 - 1/2)r'_i = r'_i/4$. Hence there are at least r'_i such leaves in the group g_i. Finally, since all these leaves have individual flow values of at least $1/\lambda$, all of them get chosen with probability 1 into our random subtree T, and hence every group in $\mathcal{G}_3$ has all of its requirement satisfied with probability 1.

To summarize, the expected number of groups whose requirement is fully covered, is at least

$$c'' \cdot |\mathcal{G}_2| + |\overline{\mathcal{G}_1 \cup \mathcal{G}_2}| \geq c'' \cdot |\mathcal{G}_2| + c'' \cdot |\overline{\mathcal{G}_1 \cup \mathcal{G}_2}| \geq c'' \cdot |\overline{\mathcal{G}_1}| \geq c''\, k'/2,$$

the last inequality holding since we are in Case II and $\mathcal{G}_1 < k'/2$. Hence each iteration of Case II is expected to fully satisfy at least a constant fraction of the active groups. It is then easy to see (via an expectation calculation and Markov's inequality) that for some constant a, the probability that not all groups are satisfied after $a \log k$ iterations where Case II held, is at most $1/4$.

Also, summing up the expected cost (2) over all iterations (using the linearity of expectation), and then using Markov's inequality, we see that with probability

at least 1/2,

$$\text{cost}(\text{Case II iterations}) \leq O((\log k) \cdot (\log N) \cdot \mathsf{OPT}). \tag{5}$$

Combining (1) and (5) and noting that $K \leq N$, we get the proof of Theorem 1.

4 Open Questions

It would be nice to resolve the approximability of the Group Steiner and Covering Steiner problems on general graph instances. Near-optimal approximation algorithms that are fast and/or combinatorial would also be of interest.

Acknowledgments. We thank Eran Halperin, Goran Konjevod, Guy Kortsarz, Robi Krauthgamer, R. Ravi, and Nan Wang for helpful discussions. We also thank the FST & TCS 2003 referees for their helpful comments.

References

1. Y. Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, pages 184–193, October 1996.
2. M. Charikar, C. Chekuri, A. Goel, S. Guha, and S. Plotkin. Approximating a finite metric by a small number of tree metrics. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, pages 379–388, 1998.
3. G. Even, G. Kortsarz, and W. Slany. On network design problems: fixed cost flows and the Covering Steiner problem. In *Proceedings of the Scandinavian Workshop on Algorithm Theory*, pages 318–327, 2002.
4. J. Fakcheroenhpoel, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 448–455, 2003.
5. N. Garg, G. Konjevod, and R. Ravi. A polylogarithmic approximation algorithm for the group Steiner tree problem. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 253–259, 1998.
6. E. Halperin, G. Kortsarz, R. Krauthgamer, A. Srinivasan, and N. Wang. Integrality Ratio for Group Steiner Trees and Directed Steiner Trees. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 275–284, 2003.
7. E. Halperin and R. Krauthgamer. Polylogarithmic Inapproximability. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 585–594, 2003.
8. S. Janson. Poisson approximations for large deviations. *Random Structures & Algorithms*, 1:221–230, 1990.
9. G. Konjevod, R. Ravi, and A. Srinivasan. Approximation Algorithms for the Covering Steiner Problem. *Random Structures & Algorithms* (Special Issue on *Probabilistic Methods in Combinatorial Optimization*), 20:465–482, 2002.
10. A. Srinivasan. New approaches to covering and packing problems. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 567–576, 2001.