THEORY OF COMPUTING, Volume 2 (2006), pp. 53–64
http://theoryofcomputing.org

An Improved Approximation Ratio for the Covering Steiner Problem*

Anupam Gupta[†] Aravind Srinivasan[‡]

Received: August 4, 2005; published: February 17, 2006.

Abstract: In the *Covering Steiner* problem, we are given 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. The Covering Steiner problem is a common generalization of the k-MST and the Group Steiner problems; indeed, when all the vertices of the graph lie in one group with a requirement of k, we get the k-MST problem, and when there are multiple groups with unit requirements, we obtain 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

*A preliminary version of this work appears in the *Proc. Foundations of Software Technology & Theoretical Computer Science*, 2003. Part of this work was done while the authors were at Lucent Bell Laboratories, 600-700 Mountain Avenue, Murray Hill, NJ 07974-0636, USA.

[†]The research of this author was supported in part by a National Science Foundation CAREER award CCF-0448095, and by an Alfred P. Sloan Fellowship.

[‡]The research of this author was supported in part by the National Science Foundation under Grant No. 0208005 and ITR Award CNS-0426683.

ACM Classification: C.2.0, F.2.2, G.1.6, G.3

AMS Classification: 68W20, 68W25, 90C59

Key words and phrases: Algorithms, Approximation Algorithms, Steiner Trees, Randomized Algorithms, Network Design

 DOI: 10.4086/toc.2006.v002a003

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 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 approximation ratio of the general Covering Steiner problem is at least as large as for the case of all unit requirements, which is just the Group Steiner problem. In this work, we improve on the known approximation algorithms for the Covering Steiner problem given by Even et al. [3] and Konjevod et al. [11]. Our results match the approximation guarantee of the known randomized algorithm for the Group Steiner problem due to Garg et al. [6] (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.

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 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. [6], 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.

Given an instance of the Covering Steiner tree problem, one can get an $O(k)$ approximation algorithm using well-known ideas without resorting to the reduction to tree instances outlined above. Indeed, one can use the 2-approximation algorithm (given by Garg [5]) for the rooted r_i-MST on each group g_i separately, and return the union of the k trees obtained. However, in case the number of groups k is large, one may prefer an algorithm whose dependence on the number of groups k is better than linear, perhaps at the expense of additional logarithmic terms. And indeed, such results are possible: 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, 11]; 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 [11], 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 the above approximation ratios *get worse* as the requirements increase, i.e., as K increases. This is unusual for covering problems, where the approximability usually *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., [12]). 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 1.1. *There is a randomized polynomial-time approximation algorithm for the covering Steiner problem which, with high probability, produces an feasible solution with an approximation ratio 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 $\Theta((\log n)/\log\log n)$ 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 approximable to within $O(k)$ as noted earlier, and so if k was 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 [8] shows that, for any constant $\varepsilon > 0$, an $O((\log(n+k))^{2-\varepsilon})$-approximation algorithm for the Covering Steiner problem implies that $\mathrm{NP} \subset \mathrm{ZTIME}[\exp\{(\log n)^{O(1)}\}]$. (Technically, the hardness is shown for the rooted version of the problem, but this is without loss of generality, since the rooted version can be reduced to the unrooted version by introducing a new group g_0 containing only the root vertex r and having unit requirement.) Furthermore, since we adopt a natural linear programming (LP) relaxation considered by Garg et al. [6] and Konjevod et al. [11], we would like to note that the integrality gap of this relaxation for tree-instances of Group Steiner is $\Omega(\log k \cdot$

$\log N/\log\log N)$ [7]. Since this integrality gap naturally extends to the Covering Steiner problem as well, this suggests that our techniques cannot improve the approximation guarantee substantially.

1.1 Our Techniques

Our approach to solving the Covering Steiner problem is to iteratively round an LP relaxation of the problem suggested by Konjevod et al. [11]. 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 or a direct deterministic approach.

Informally, in order to do better than the approach of Konjevod et al. [11], 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 [11] 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: either (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 3 is used to bound the number of iterations where case (i) holds; Janson's inequality [9] and some deterministic arguments are used for case (ii). In the interest of the cleanest exposition, we have not attempted to optimize our constants.

2 Preliminaries

The analysis of our algorithm will use an inequality due to Janson [9], which we now state. Let Ω be a set of elements, and let us pick a random set R by picking each $e\in\Omega$ *independently* with some probability p_e. Let $\{A_j\subseteq\Omega \mid j\in J\}$ be some family of subsets of Ω, and we say that $j\sim j'$ if $j\neq j'$ and $A_j\cap A_{j'}\neq\emptyset$. Define $\mathcal{B}_j$ to be the event that $A_j\subseteq R$, and let $\Delta=\sum_{j<j':j\sim j'}\mathbf{Pr}[\mathcal{B}_j\cap\mathcal{B}_{j'}]$. If Y_j is the indicator variable for event $\mathcal{B}_j$, let $\mu_j=\mathbf{E}[Y_j]=\mathbf{Pr}[\mathcal{B}_j]$, and $\mu=\sum_{j\in J}\mu_i$. Then Janson's inequality says that

Theorem 2.1. *For any* $\delta\in[0,1]$,

$$\mathbf{Pr}\Big[\sum_j Y_j\le(1-\delta)\mu\Big]\le\exp\left\{-\frac{\delta^2\mu}{2+(\Delta/\mu)}\right\}. \tag{2.1}$$

3 A Chip Game

Consider the following *chip game*. We initially have chips arranged in k groups, with each group g_i having some number $n_i^{(init)}\le K$ 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: in each round, we choose roughly half of the currently active groups and halve their sizes, until there are no more active groups left. Formally, the game is as follows:

1. At the start of some round, let n_i denote the number of chips in group g_i. Let A denote the number of groups currently active; i.e., $A = |\{i \mid n_i \neq 0\}|$.
2. We choose any set of at least $\lceil A/2 \rceil$ *active* groups.
3. For each of these chosen groups g_i, we remove *at least* $\lceil n_i/2 \rceil$ chips, causing the new number of chips in group g_i to become at most $\lfloor n_i/2 \rfloor$. Note that this may cause some of the groups to become inactive.
4. If there are no more chips left, we stop, else we go back to Step 1.

In the analysis below, it will not matter that two constants in Steps 2 and 3 above were $1/2$ each; any two constants $a, b \in (0,1)$ lead to the same asymptotic result in the following lemma.

Lemma 3.1. *The maximum number of rounds possible in the above chip game is* $O(\log k \cdot \log K)$.

Proof. To bound the number of rounds, we proceed as follows. We now modify the game into an equivalent format. We initially start with $1 + \lfloor \log n_i^{(init)} \rfloor$ chips in each group g_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 at least *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 $k(1 + \log K)$, the number of such rounds N_1 is at most $4(1 + \log K)$. Proceeding in this manner, we see that the total number of rounds is $O(\log k \cdot \log K)$, as claimed. □

The proof above indicates how to prove that the bound proved above in Lemma 3.1 is tight, and there are runs of the chip game which require $\Theta(\log k \cdot \log K)$ rounds. 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.

4 The Algorithm and Analysis

In this section, we present our algorithm for the Covering Steiner problem, and analyse it. As mentioned in the introduction, we will assume that the input consists of a *rooted* tree T, where the root r must be included in the output solution. Furthermore, we assume (without loss of generality) that every vertex belonging to a group is a leaf of T, and that every leaf belongs to some group.

4.1 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; each iteration produces a subtree rooted at r that provides some additional 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 the group g_i *active* if and only if this residual requirement $r'_i > 0$, 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; with a slight abuse of notation, we 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 these edges 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 relaxation for the residual problem was proposed by Konjevod et al. [11]:

$$Z^* = \min \sum_e c_e x_e$$

$$\text{subject to} \qquad \sum_{j \in g_i} x_{\mathsf{pe}(j)} = r'_i \qquad \forall \text{ active } g_i \tag{4.1}$$

$$\sum_{j \in (T(e) \cap g_i)} x_{\mathsf{pe}(j)} \le r'_i x_e \qquad \forall \text{ active } g_i, \forall e \in E(T) \tag{4.2}$$

$$x_{\mathsf{pe}(e)} \ge x_e \qquad \forall e \text{ not incident on } r \tag{4.3}$$

$$x_e \in [0,1]$$

Note that forcing each variable x_e to have values in $\{0,1\}$ instead of values in the real interval $[0,1]$ gives us a ILP formulation of the Covering Steiner problem; indeed, in this case, the variable x_e being set to 1 corresponds to the edge e being in the solution tree. Since we relax the problem and allow the x_e's to take values in a larger range ($[0,1]$ instead of $\{0,1\}$), it follows that the optimal value Z^* of this LP is a lower bound on OPT, the optimal solution value of the integer linear program, and hence of the original Covering Steiner instance.

In each iteration we start with the residual problem, solve the above LP optimally, and then round the resulting fractional solution as described below in Section 4.2. As noted in [11], it is interesting to note that the integrality gap of the above relaxation 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 t, *re-solve* the LP for the residual problem, and continue.

4.2 Rounding the Fractional Solution

Let $\{x_e\}$ denote an optimal solution to the LP relaxation for some generic iteration t, and let Z^*_t be the optimal LP value. We now show how to round such an optimal solution 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. The constraint (4.1) ensures that total flow into the leaves of a group g_i is exactly r'_i. For $\alpha \ge 1$, 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 proceeds as follows. We define $\mathcal{G}_1$ to be the set of active groups that are $(1/2,4)$-covered; i.e., at least half of the flow into these groups goes to elements that get individual flows of at least a quarter. We will consider two cases based on whether the number of groups in $\mathcal{G}_1$ is at least $k'/2$ or not.

Case I: $|\mathcal{G}_1| \geq k'/2$. In this case, we simply take a 4-scaling of the LP solution and *pick* the edges that are rounded up to 1. By the monotonicity constraint $x_{\mathsf{pe}(e)} \geq x_e$ in the LP, the edges thus picked will form a connected subtree containing the root.

Moreover, we claim that every group in $\mathcal{G}_1$ has at least half of its requirement covered by this process. Indeed, any group $g_i \in \mathcal{G}_1$ is $(1/2,4)$-covered (i.e., it has at least $r'_i/2$ of its member leaves receiving flows of value at least $1/4$); hence the 4-scaling ensures that the picked edges cover at least $r'_i/2$. Since we are in the case when $\mathcal{G}_1$ is large, we know that at least half of the currently active groups have their requirements reduced by at least half. But now Lemma 3.1 for the chip game implies that the total number of iterations where Case I can hold is at most $O(\log k \cdot \log K)$. Furthermore, we pay a cost of at most $4 \cdot Z_t^* \leq 4 \cdot \mathsf{OPT}$ in each such iteration t, implying the following result:

Lemma 4.1. *The total cost incurred in Case-I iterations, where $|\mathcal{G}_1| \geq k'/2$, is*

$$\textit{cost(Case I iterations)} = O(\log k \cdot \log K \cdot \mathsf{OPT}). \tag{4.4}$$

(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: $|\mathcal{G}_1| < k'/2$; i.e., fewer than half the groups are $(1/2,4)$-covered. In this case, we further categorize the groups in $\mathcal{G} \setminus \mathcal{G}_1$ as follows. Let $\lambda = c' \log N$, for a sufficiently large absolute constant c'. We define $\mathcal{G}_2$ to be the set of active groups that are *not* $(3/4,\lambda)$-covered: i.e., at least one fourth of the flow into these groups goes to elements that receive very little flow (at most $1/\lambda$ each). Finally, 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. [6] and Konjevod et al. [11]. Let $\{x'_e\}$ be a λ-scaling of the LP solution; i.e., $x'_e = \min\{\lambda \cdot x_e, 1\}$ for each e. Now, we do the following for each edge independently:

- For every edge e incident on the root, we *pick* e with probability x'_e.
- For every other edge e with its parent denoted f, *pick* e with probability x'_e/x'_f.

Let H denote the subgraph of G induced by the edges that were picked; the solution we return is T_H, the connected subtree of H that contains the root r. 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 $e \in T_H$ for some edge e is x'_e (see, e.g., [6, Lemma 3.1]), and hence linearity of expectation implies:

$$\mathbf{E}[\text{cost}(T_H)] = \sum_{e \in E} x'_e \leq \lambda \cdot \sum_{e \in E} x_e = \lambda \cdot \mathsf{OPT}. \tag{4.5}$$

We next analyze the expected coverage properties of this randomized rounding procedure. Let us first 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 $(3/4 - 1/2)r'_i = r'_i/4$. Since the flow into any one of these leaves is no more than $1/4$, 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_H, and hence every group in $\mathcal{G}_3$ has all of its requirement satisfied with probability 1.

This leaves us with groups in $\mathcal{G}_2$. For any such group $g_i \in \mathcal{G}_2$, 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. The following fact follows from the very definition of $\mathcal{G}_2$.

Fact 4.2. For any $g_i \in \mathcal{G}_2$, the flow $f_i \geq r'_i/4$.

If we define Y_{ij} for each leaf $j \in g'_i$ to be the indicator variable indicating that j was chosen, then it suffices to give a lower bound on $X_i = \sum_{j \in g'_i} Y_{ij}$, the number of elements of g'_i that get covered by the above randomized rounding. For this, we plan to employ Janson's inequality (Theorem 2.1). Since each of the flows f_i is at most $1/\lambda$, the scaled-up value $x'_{\mathsf{pe}}(j) \leq 1$ for each $j \in g'_i$ even after the λ-scaling, and hence $\mu_i \doteq \mathbf{E}[X_i] = \sum_{j \in g'_i} \mathbf{E}[Y_{ij}] = f_i \lambda$. We need a few further definitions in order to apply Janson's inequality. Let A_j be the path from the leaf j to the root r, and let A'_j be the prefix of A_j, the edges e of which satisfy $x_e < 1/\lambda$. (When we say "prefix" here, we view A_j as starting from the leaf j and going up to the root r.) Note that in our rounding, the edges e of A_j which *do not* lie in A'_j satisfy $x_e \geq 1/\lambda$, and hence will be chosen with probability 1. Thus, Y_{ij} is 1 if and only if A'_j is contained in the chosen subtree T_H. Now for two leaves $j, j' \in g'_i$, we will define two relations:

- $j \sim j'$ if and only if (a) $j \neq j'$ and (b) the paths $A_j, A_{j'}$ intersect in at least one edge; i.e., 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'.
- $j \sim' j'$ if and only if (a) $j \neq j'$ and (b) the paths $A'_j, A'_{j'}$ intersect in at least one edge. This is basically the same as the previous definition, except that we now work with the paths A'_j instead of A_j. Also note that if $j \sim' j'$, then $\mathsf{lca}(j, j')$ is well-defined.

To use Janson's inequality, define

$$\Delta'_i = \sum_{j, j' \in g'_i:\ j \sim' j',\, x_{\mathsf{lca}(j,j')} > 0} \frac{x'_{\mathsf{pe}(j)} x'_{\mathsf{pe}(j')}}{x'_{\mathsf{lca}(j,j')}} \ . \tag{4.6}$$

It is easy to verify that this is indeed $\sum_{j \sim' j'} \mathbf{E}[Y_{ij} Y_{ij'}]$; indeed, if we condition on the event $Y_{ij} = 1$, we know that the $\mathsf{lca}(j, j')$ must be contained in T_H, and hence the conditional probability of the event $Y_{ij'} = 1$ can be shown to be $x'_{\mathsf{pe}(j')}/x'_{\mathsf{lca}(j,j')}$. Since $f_i \geq r'_i/4$ by Fact 4.2, we get $r'_i \leq 4 f_i = 4\mu_i/\lambda$, which is at most $\mu_i/2$ if c' is large enough. Now Janson's inequality shows that

$$\mathbf{Pr}[X_i \leq r'_i] \leq \exp\Big\{-\frac{\mu_i/4}{2 + \Delta'_i/\mu_i}\Big\} \ . \tag{4.7}$$

Let us now bound Δ'_i, by considering the following closely-related quantity:

$$\Delta_i = \sum_{j, j' \in g'_i:\ j \sim j',\, x_{\mathsf{lca}(j,j')} > 0} \frac{x_{\mathsf{pe}(j)} x_{\mathsf{pe}(j')}}{x_{\mathsf{lca}(j,j')}} \ . \tag{4.8}$$

The ways in which Δ_i differs from Δ'_i are that: (i) the sum is over pairs (j, j') which satisfy $j \sim j'$, instead of $j \sim' j'$; and (ii) the summand involves terms such as $x_{\mathsf{pe}(j)}$ instead of $x'_{\mathsf{pe}(j)}$.

Now, [11, Theorem 3.2] shows that $\Delta_i \le r'_i(r'_i - 1 + r'_i \ln N)$. The reader can verify that any pair (j, j') that appears in the sum for Δ'_i, also appears in Δ_i; furthermore, for any such pair,

$$\frac{x_{\mathsf{pe}(j)} x_{\mathsf{pe}(j')}}{x_{\mathsf{lca}(j,j')}} = \lambda \cdot \frac{x'_{\mathsf{pe}(j)} x'_{\mathsf{pe}(j')}}{x'_{\mathsf{lca}(j,j')}} \ .$$

Thus, $\Delta'_i \le \lambda \cdot \Delta_i$, and hence $\Delta'_i = O((r'_i)^2 \log^2 N)$, by [11, Theorem 3.2]. Now plugging this into (4.7), along with the facts $\mu_i = f_i \lambda = f_i c' \log N$ and $f_i \ge r'_i/4$, we get that there is an absolute constant $c'' \in (0,1)$ such that

$$\Pr[X_i \ge r'_i] \ge c'' \ . \tag{4.9}$$

Applying the linearity of expectation to (4.9) shows us that the expected number of groups in $\mathcal{G}_2$ that get *all* of their requirement covered in this iteration is at least $c'' \cdot |\mathcal{G}_2|$.

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

$$c'' \cdot |\mathcal{G}_2| + |\mathcal{G}_3| \ge c'' \cdot |\mathcal{G}_2 \cup \mathcal{G}_3| \ge c'' k'/2 \ ,$$

the last inequality holding since we are in Case II and $\mathcal{G}_1 < k'/2$. Applying Markov's inequality gives us that we cover a constant fraction of the groups with constant probability, ensuring that the probability that not all groups are satisfied after $a \log k$ iterations in which Case II held is at most $1/4$ (for some large enough a).

Also, summing up the expected cost (4.5) over all iterations (using the linearity of expectation), and then using Markov's inequality, we see that with probability at least $1/2$,

$$\text{cost(Case II iterations)} = O(\log k \cdot \log N \cdot \mathsf{OPT}) \ . \tag{4.10}$$

Combining (4.4) and (4.10) and noting that $K \le N$, we get the proof of Theorem 1.1.

5 Open Questions

It would be nice to resolve the approximability of the Group Steiner and Covering Steiner problems on general graph instances; we currently lose a logarithmic factor due to the step of approximating general graphs by distributions over tree metrics. As a practical matter, it would be interesting to employ frameworks such as those of [10] to approximately solve the linear program by combinatorial means, or to develop a new combinatorial approximation algorithm matching our bounds.

Acknowledgments

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

References

[1] * YAIR BARTAL: Probabilistic approximations of metric spaces and its algorithmic applications. In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, pp. 184–193, 1996. [FOCS:10.1109/SFCS.1996.548477]. 1

[2] * MOSES CHARIKAR, CHANDRA CHEKURI, ASHISH GOEL, AND SUDIPTO GUHA: Rounding via trees: deterministic approximation algorithms for group Steiner trees and k median. In *Proceedings of the 30th Annual Symposium on the Theory of Computing*, pp. 114–123, 1998. [STOC:10.1145/276698.276719]. 1

[3] * GUY EVEN, GUY KORTSARZ, AND WOLFGANG SLANY: On network design problems: fixed cost flows and the covering Steiner problem. *ACM Trans. Algorithms*, 1(1):74–101, 2005. [TALG:10.1145/1077464.1077470]. 1

[4] * JITTAT FAKCHAROENPHOL, SATISH RAO, AND KUNAL TALWAR: A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. System Sci.*, 69(3):485–497, 2004. [JCSS:10.1016/j.jcss.2004.04.011]. 1

[5] * NAVEEN GARG: Saving an epsilon: a 2-approximation for the k-mst problem in graphs. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pp. 396–402, 2005. [STOC:10.1145/1060590.1060650]. 1

[6] * NAVEEN GARG, GORAN KONJEVOD, AND R. RAVI: A polylogarithmic approximation algorithm for the group Steiner tree problem. *Journal of Algorithms*, 37(1):66–84, 2000. [JAlg:10.1006/jagm.2000.1096]. 1, 1, 4.2

[7] * ERAN HALPERIN, GUY KORTSARZ, ROBERT KRAUTHGAMER, ARAVIND SRINIVASAN, AND NAN WANG: Integrality ratio for group Steiner trees and directed Steiner trees. In *14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 275–284, January 2003. [SODA:644108.644155]. 1

[8] * ERAN HALPERIN AND ROBERT KRAUTHGAMER: Polylogarithmic inapproximability. In *Proceedings of the thirty-fifth ACM symposium on Theory of computing*, pp. 585–594. ACM Press, 2003. [STOC:10.1145/780542.780628]. 1

[9] * SVANTE JANSON: Poisson approximation for large deviations. *Random Structures and Algorithms*, 1(2):221–229, 1990. 1.1, 2

[10] * ROHIT KHANDEKAR: *Lagrangian Relaxation based Algorithms for Convex Programming Problems*. PhD thesis, Indian Institute of Technology, New Delhi, March 2004. 5

[11] * GORAN KONJEVOD, R. RAVI, AND ARAVIND SRINIVASAN: Approximation algorithms for the covering Steiner problem. *Random Structures and Algorithms*, 20(3):465–482, 2002. Special Issue on *Probabilistic methods in combinatorial optimization*. [RSA:10.1002/rsa.10038]. 1, 1, 1.1, 4.1, 4.1, 4.2, 4.2

[12] * ARAVIND SRINIVASAN: New approaches to covering and packing problems. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms (Washington, DC, 2001)*, pp. 567–576, Philadelphia, PA, 2001. SIAM. [SODA:365411.365535]. 1

AUTHORS

Anupam Gupta
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15232, USA
anupamg@cs.cmu.edu
http://www.cs.cmu.edu/~anupamg

Aravind Srinivasan
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
http://www.cs.umd.edu/~srin

ABOUT THE AUTHORS

ANUPAM GUPTA is a faculty member at Carnegie Mellon University, Pittsburgh PA. He grew up in Calcutta, India, went to the Indian Institute of Technology at Kanpur for his B. Tech. degree, and received his Ph. D. from the University of California at Berkeley under the guidance of Alistair Sinclair. He also spent two years searching for good and unknown restaurants in New York while he was ostensibly a postdoctoral researcher at Lucent Bell Labs, Murray Hill, NJ. Anupam's interests include approximation algorithms, metric embeddings, networking, eating, and playing squash badly.

ARAVIND SRINIVASAN is a faculty member at the University of Maryland, College Park. He received his Ph. D. degree from Cornell University under the supervision of David Shmoys, and his undergraduate degree from the Indian Institute of Technology, Madras. His research interests are in randomized algorithms, networking, social networks, combinatorial optimization, and related areas. Aravind is a native of Chennai, India, and his "mother tongue" (native language) is Tamil.