A greedy approximation algorithm for the group Steiner problem

Chandra Chekuri * Guy Even † Guy Kortsarz ‡

August 31, 2005

Abstract

In the group Steiner problem we are given an edge-weighted graph $G = (V, E, w)$ and m subsets of vertices $\{g_i\}_{i=1}^m$. Each subset g_i is called a *group* and the vertices in $\bigcup_i g_i$ are called *terminals*. It is required to find a minimum weight tree that contains at least one terminal from every group.

We present a poly-logarithmic ratio approximation for this problem when the input graph is a tree. Our algorithm is a recursive greedy algorithm adapted from the greedy algorithm for the directed Steiner tree problem [20, 8]. This is in contrast to earlier algorithms that are based on rounding a linear programming based relaxation for the problem [14, 24]. We answer in positive a question posed in [14] on whether there exist good approximation algorithms for the group Steiner problem that are not based on rounding linear programs. For every fixed constant $\varepsilon > 0$, our algorithm gives an $O((\log \sum_i |g_i|)^{1+\varepsilon} \cdot \log m)$ approximation in polynomial time. As pointed out in [14], approximation algorithms on trees can be extended to arbitrary undirected graphs by probabilistically approximating the graph by a tree [1, 2, 11]. This results in an additional multiplicative factor of $O(\log |V|)$ in the approximation ratio, where $|V|$ is the number of vertices in the graph. The approximation ratio of our algorithm on trees is slightly worse than the ratio of $O(\log(\max_i |g_i|) \cdot \log m)$ provided by the LP based approaches [14, 24].

*Bell Labs, 600 Mountain Ave., Murray Hill, New Jersey 07974, USA. E-mail:chekuri@research.bell-labs.com

†Department of Electrical-Engineering, Tel-Aviv University, Israel. E-mail:guy@eng.tau.ac.il

‡Computer Sciences Department, Rutgers University - Camden. E-mail:guyk@camden.rutgers.edu

1 Introduction

The Steiner tree problem is among the fundamental problems in network design. The input to the Steiner tree problem is an undirected edge-weighted graph $G = (V, E, w)$ and a set of terminals $\mathcal{T} \subseteq V$. The objective is to find a minimum weight tree T that spans the terminals in $\mathcal{T}$. The Steiner tree problem is known to be NP-hard [13] and also APX-hard [5]. In this paper we consider the group Steiner problem which is a generalization of the Steiner tree problem. The input to this problem also consists of an edge-weighted graph $G = (V, E, w)$, however instead of a single set of terminals we are given a collection of possibly intersecting subsets of vertices $\{g_i\}_i$. Each subset g_i is called a group. The objective is to find a minimum weight tree that contains at least one vertex from each group. Throughout, we denote the number of groups by m, the number of terminals $|\cup_{i=1}^m g_i|$ by n, and the size of the largest group $\max_i |g_i|$ by N. We note that the sum of the group sizes $\sum_{i=1}^m |g_i|$ is at most mn.

The group Steiner problem was introduced by Reich and Widmayer [22] motivated by applications to wire routing with multi-port terminals in physical VLSI design. See [14] for additional references to this problem. The problem is of interest not only because of its applications but also because of its relation to the Steiner tree problem in both undirected and directed graphs. The search for good approximation algorithms for this problem has inspired new technical ideas [14, 18, 19, 24].

The group Steiner problem is a strict generalization of the Steiner tree problem, and in [14] it is shown that very special cases of the group Steiner problem are harder to approximate than the Steiner tree problem: in particular it is shown that the set cover problem can be reduced in an approximation preserving way to the group Steiner problem on star graphs. From the hardness of approximating set cover [12, 21], it follows that the group Steiner problem on stars, and hence trees, is NP-hard to approximate to within a factor better than $c \ln m$ for some constant c, or to a factor better than $(1 - o(1)) \ln m$ unless $NP \subseteq DTIME(n^{\log \log n})$. In recent work, Halperin and Krauthgamer [16] improved the hardness of approximation. They showed that for every $\epsilon > 0$, the group Steiner problem on trees is hard to approximate to within a factor better than $\Omega(\log^{2-\epsilon} m)$, unless NP problems can be solved by quasi-polynomial time Las-Vegas algorithms.

In terms of upper bounds, the first sub-linear approximation ratio for this problem was an $O(\sqrt{m})$ ratio given by Bateman *et al.* [4]. Garg *et al.* [14] improved this substantially and obtained the first poly-logarithmic approximation ratio for this problem. They gave an $O(\log N \log m)$ approximation algorithm for the problem on *trees* based on an elegant randomized rounding of the natural linear programming relaxation for the problem. It should be mentioned that their algorithm achieves a ratio of $O(\min\{h, \log N\} \log m)$ on trees of height h. They extended their algorithm for trees to general undirected graphs by using Bartal's [1, 2] probabilistic approximation of a finite metric by tree metrics. Using the best possible probabilistic approximation obtained in [11], the approximation ratio for the group Steiner problem on general graphs is $O(\log N \log m \log |V|)$ where $|V|$ is the number of vertices in G. Zosin and Khuller [24] achieved similar results in a later paper; they also use the natural LP relaxation, however their algorithm is dual based and is substantially different from the primal based algorithm in [14]. In [6] the algorithm of [14] is derandomized to give a deterministic algorithm achieving the same approximation ratio. Note that the approximation ratio for trees is almost matched by the hardness factor in [16]. It is also known that the integrality gap of the natural linear programming relaxation is $\Omega(\log^2 n)$ even on trees [15].

Approximation algorithms for the group Steiner problem have been obtained through a different direction by reducing it to the *directed* Steiner tree problem. The directed Steiner tree problem is a generalization of the Steiner tree problem to directed graphs and is defined as follows. We are given an arc-weighted directed graph $G = (V, A)$, a set of terminals $\mathcal{T} \subseteq V$ and a special vertex $r \in V$ called the root. The objective is to find a minimum weight out-tree T rooted at r in which r has a directed path to every terminal. The directed group Steiner problem is also defined similarly as a generalization of the group Steiner problem: we are given a root in addition to the groups and the goal is to find a min-weight tree such that there is a directed path from the root to at least one vertex of each group. It is an easy observation that the directed Steiner tree problem and the directed group Steiner problem are equivalent. For the directed Steiner tree problem, the current best known approximation algorithm is by Charikar *et al.* [8]. They gave an algorithm that given an integer parameter $i \geq 1$, achieves an $O(i^3 m^{1/i})$-approximation[1] and has a running time of $O(n^i)$. Hence, for any fixed ε, an $O(m^\varepsilon)$ approximation can be obtained in polynomial time. More interestingly, an $O(\log^3 m)$ approximation can be obtained in quasi-polynomial time (i.e., $O\left(n^{\log m}\right)$ time) giving strong evidence for the conjecture that the problem has a poly-logarithmic approximation ratio. These results carry over to the group Steiner problem. We note that the algorithm in [8] is based on a greedy framework [23, 20, 8].

Our Results: The two known poly-logarithmic approximation algorithms for the group Steiner problem are both based on rounding a solution to the linear programming relaxation [14, 24]. In [14] the following question is asked: is there a "combinatorial" poly-logarithmic approximation algorithm for the group Steiner problem? By combinatorial they imply an algorithm that is not based on solving an LP relaxation. In this paper we answer their question and give such an algorithm for trees. For any fixed $\varepsilon > 0$, the approximation ratio obtained by our algorithm is $O(\frac{1}{\varepsilon} \cdot (\log n)^{1+\varepsilon} \cdot \log m)$ which is only slightly worse than the ratio of $O(\log N \log m)$ given in [14, 24]. Following [14], an approximation algorithm on trees allows us to obtain an approximation algorithm for general graphs: the input graph is approximated probabilistically by tree metrics [2, 7, 11]. Given an input graph, the algorithm in [11] produces a tree such that the expected distance between any pair of vertices in the tree is at most $O(\log |V|)$ times the distance between the pair in the graph. We simply run our algorithm on this tree. The algorithm in [11] is randomized. It can be derandomized by using ideas from [7]. We refer the reader to [7, 11] for more details.

Even and Kortsarz [9] claimed an approximation algorithm for the group Steiner problem on trees with an approximation ratio of $O(\log^2 n / \log \log n)$. The analysis presented in [9] contains an error. We rely on some of the methods used in [9]: the greedy framework [23, 20, 8], geometric search, and avoiding low coverage trees. The greedy algorithms in [20, 8] run in quasi-polynomial time to obtain a poly-logarithmic ratio. In this paper we use several technical ideas to reduce the running time to be polynomial when the input graph is a tree. Some of our ideas are relevant for the directed Steiner tree problem. It is our belief that further ideas along these lines may yield a polynomial time algorithm with a poly-logarithmic ratio for the directed Steiner tree problem.

Techniques: Our algorithm follows the greedy methodology of [20, 8]. The *density* of a partial solution F is the ratio of the weight of F divided by the number of groups covered by F. If an algorithm guarantees a partial solution with density at most α times the density of the optimal tree, then this algorithm can be used iteratively to find a tree that covers all groups and the resulting approximation ratio will be $O(\alpha \log m)$.

Given a tree of height h, the algorithm in [8] yields a partial solution with density $O(h)$ times the density of the optimal tree. However, the running time of the algorithm is exponential in h. Obtaining polynomial running time requires modifications that reduce both the exponent (i.e.

[1]In [8], a ratio of $O(i^2 m^{1/i})$ is claimed but this relied on an erroneous lemma in [23]. The lemma in [23], when fixed (see [17]), results in a worsening of the approximation ratio claimed in [8].

the height) and the base (i.e. number of iterations, number of demand values per iteration, and number of children per node). We accomplish this using several ideas. The first of these involve preprocessing the input tree to satisfy certain height and degree requirements.

- Height reducing transformation: for any α we give a transformation that reduces the height of the tree to $O(\log_\alpha n)$ while incurring a multiplicative factor of $O(\alpha)$ in the weight of the optimal solution.
- Degree reducing transformation: given a parameter $\beta \geq 3$ we reduce the maximum degree of the tree to $\beta + 1$ while (additively) increasing the height of the tree by $O(\log_{\beta/2} n)$ and not increasing the weight of the optimal solution.

By choosing $\alpha = \log^\varepsilon n$ and $\beta = \log n$ we obtain a tree with height $O(\frac{1}{\varepsilon} \log n / \log\log n)$ and maximum degree $O(\log n)$. Further, we are guaranteed that there is a solution in this tree of weight at most $O(\log^\varepsilon n)$ times the weight of an optimal solution in the input tree.

Finally, the greedy algorithm is modified so as to reduce the number of recursive calls by using geometric search and avoiding sub-trees that cover few groups. These modifications combined with the preprocessing mentioned above result in a polynomial time algorithm. Although our algorithm runs in polynomial time, it is not efficient. Our goal is to investigate the greedy approach to this problem and we have not made much effort to choose the best possible parameters to optimize the running time.

Our height reduction procedure is non-trivial and we use it to obtain an algorithm for trees. However, for arbitrary graphs our first step is to reduce the graph to a tree via probabilistic approximation. The trees returned by the algorithms in [1, 2, 7, 11] are HSTs (hierarchically well separated) and height reduction for these structured trees that achieves the same bounds as our procedure is straightforward. For an example of this latter kind of height reduction, see Bartal, Charikar, and Raz [3].

Organization Section 2 contains the formal problem definition and useful notation for the rest of the paper. In Section 3 we present our greedy algorithm with geometric search and analyze its performance. We show that the running time of the algorithm is polynomial for trees with height and degree appropriately bounded. The height and degree reducing transformations are presented in Sections 4 and 5. We conclude in Section 6 with some remarks.

2 Preliminaries

Problem Formulation The group Steiner problem restricted to trees is defined as follows. The input consists of (i) a rooted tree $T = (V, E)$ with a root r, (ii) non-negative edge weights $w(e)$, and (iii) a collection of vertex subsets $\{g_i\}_i$. The subsets g_i are referred to as *groups* and vertices in $\cup_i g_i$ are referred to as *terminals*. A group g_i is *covered* by a subtree $T' = (E', V')$ if $g_i \cap V' \neq \phi$. A *cover* of the groups is a tree T' that covers every group and contains the root r. The goal is to find a minimum weight cover.

Note that we consider a rooted version of the group Steiner problem. This is not a restriction since the unrooted version and the rooted version are polynomially equivalent (i.e., if no root is specified, simply run the algorithm $|V|$ times each time with a different node assigned as the root).

A tree $T' \subseteq T$ is *z-cover* if it covers at least z groups. We consider also the problem of finding a minimum weight z-cover.

Notation and Definitions The input to our problem is a rooted tree. A child-parent and an ancestor-descendant relation is naturally induced over the vertices. The parent of a non-root node v is denoted by $p(v)$. The subtree rooted at a node v is denoted by T_v. Let $e = (u, v)$ be an edge where u is the parent of v. The subtree induced by the edge (u, v) is the tree $T_v \cup \{(u, v)\}$, namely, the tree T_v in addition to the edge (u, v) and the node u. We denote the subtree induced by the edge (u, v) by $T_{(u,v)}$. We denote the set of leaves of a tree T by $\mathcal{L}(T)$.

Let m denote the number of groups, $n = | \cup_{i=1}^{m} g_i|$ denote the number of terminals, and $s = \sum_{i=1}^{m} |g_i|$ denote the sum of the group sizes. Note that s might be significantly greater than n, and therefore, s is used to measure the input length.

Let $n(T')$ denote the number of terminals in T'. For a rooted subtree T_u, we denote $n(T_u)$ simply by n_u. The number of groups covered by T' is denoted by $m(T')$. The sum of the edge weights in a subtree T' is denoted by $w(T')$. The *density* of a subtree tree T' is defined as

$$\gamma(T') \triangleq \frac{w(T')}{m(T')}.$$

The height of a tree is the maximum number of edges along a simple path from the root to a leaf. The height of a tree T is denoted by $h(T)$.

We interpret every subtree T' that contains the root r as a *partial cover*, that is, T' covers a subset of the groups. A partial cover transforms a problem instance into a *residual problem*. The residual problem is obtained by declaring all the terminals in the groups covered by T' to be nonterminals. (Weights of edges in T' could be zeroed in the residual problems, but the analysis we present does not benefit by this, so we keep the edge weights unchanged.) Observe that if T^* covers the groups in T, then T^* covers the groups in every residual problem. Hence the weight of an optimal solution to the residual problem is no greater than the weight of an optimal solution to the initial problem.

Preprocessing. We preprocess the tree as follows so that the total number of nodes is $O(n)$ and every node is a terminal:

1. Eliminate every nonterminal leaf. Obviously, this does not affect the set of feasible solutions. Hence, we may assume that every leaf is a terminal.

2. Eliminate every nonterminal interior node v of degree two. This is done by replacing the length two path $p = x\text{-}v\text{-}y$ that traverses v by a new edge (x, y). The weight of the new edge is $w(x, v) + w(v, y)$. Hence, we may assume that every interior node has at least two children, and therefore, the number of nodes is $O(n)$.

3. Add a new dummy group that contains all the nodes. Obviously, this does not affect the set of feasible solutions.

We remark that if one is not interested in the distinction between n and s, then preprocessing can make the groups disjoint. Simply hang new terminals from every old terminal v so that there is one new terminal per group that v belongs to. This reduction insures that the groups are disjoint at the cost of increasing n so that it equals s.

Finally, by scaling edge weights, we may assume that, for every edge e, $w(e) > 0$ implies that $w(e) \geq 1$.

Faithful trees. Consider a tree T rooted at r. Every subset of nodes $S \subseteq V(T)$ induces a subtree $T[S]$ consisting of the union of all the paths in T from the root r to the nodes in S.

The setting for the definition of faithful trees is as follows. Let A and B be two rooted trees. Let $\pi : V(A) \to V(B)$ be a function (non necessarily one-to-one) that maps the nodes of A to those of B. In this mapping, A will be the original tree and B will be the height reduced tree. For $S \subseteq V(A)$, we denote the image of S by $\pi(S)$. The preimage of $S' \subseteq \pi(V(A))$ is denoted by $\pi^{-1}(S')$. We map the group Steiner instance on A to that in B in a straightforward way as follows. For every group $g_i \subseteq V(A)$, we define a group $g'_i \subseteq V(B)$ by $g'_i = \pi(g_i)$.

Fact 2.1 *If $S \subseteq V(A)$ covers g_i in A then $\pi(S)$ covers g'_i in B. Similarly, if $S' \subseteq \pi(V(A))$ covers g'_i in B then $\pi^{-1}(S')$ covers g_i in A.*

We denote the edge weight function of a tree T by w_T.

Definition 2.2 *The tree B is an α-*faithful *representation of the tree A if there is a mapping $\pi : V(A) \to V(B)$ such that the following two conditions hold:*

1. *For every $S \subseteq V(A)$, $w_B(B[\pi(S)]) \leq \alpha \cdot w_A(A[S])$.*
2. *For every $S' \subseteq \pi(V(A))$, $w_A(A[\pi^{-1}(S')]) \leq w_B(B[S'])$.*

The following claim summarizes the approximation preserving properties of α-faithful trees.

Claim 2.3 *Let B denote an α-faithful representation of A. A β-approximate z-cover in B induces an $(\alpha \cdot \beta)$-approximate z-cover in A.*

Proof: Let $\pi : V(A) \to V(B)$ denote the mapping to prove that B is an α-faithful representation of A. Let S_1 be a minimum weight z-cover in A. Let $S'_1 = \pi(S_1)$. From Fact 2.1, it follows that S'_1 is a z-cover in B. From α-faithfulness of π, we have that $w_B(B[S'_1]) \leq \alpha \cdot w_A(A[S_1])$. Let S'_2 be a β-approximate z-cover in B. Since S'_1 is a z-cover in B, $w_B(B[S'_2]) \leq \beta \cdot w_B(B[S'_1])$. Let $S_2 = \pi^{-1}(S'_2)$. Again, from Fact 2.1, S_2 is a z-cover in A. From faithfulness, we have that $w_A(A[S_2]) \leq w_B(B[S'_2])$. Putting together, we have that S_2 is a z-cover in A and $w_A(A[S_2]) \leq \alpha \cdot \beta \cdot w_A(A[S_1])$. Since S_1 is a minimum weight z-cover in A, the claim follows. □

Transformations. As discussed in Section 1, we preprocess the input tree to reduce its height and degree before applying our greedy algorithm. We summarize the properties of these transformations in the following claims that are proved in Sections 4 and 5.

Claim 2.4 *Let $\alpha > 1$. There exists a linear time algorithm that, given a rooted tree T with n nodes, computes an $O(\alpha)$-faithful representation T' of T such that $h(T') = O(\log_\alpha n)$.*

Claim 2.5 *Let $\beta \geq 3$. There exists a linear time algorithm that, given a rooted tree T with n nodes, computes a 1-faithful representation T' of T such that: $h(T') \leq h(T) + \lfloor \log_{\beta/2} n \rfloor$ and every node in T' has at most β children.*

Throughout the paper we use $\log x$ to denote the logarithm of x to some fixed constant. We do not specify the constant since we are mainly interested in asymptotic ratios. We explicitly specify the base when it is needed.

3 A Recursive Greedy Algorithm with Geometric Search

In this section we present our recursive greedy algorithm which is similar to the greedy algorithms in [20, 8]. The main difference is that we use geometric search to reduce the number of recursive calls. Together with the height and degree reducing transformations, this yields a polynomial running time. To motivate and explain our modification, we first describe the algorithm in [8] for the directed Steiner tree problem, specializing it to the case of the group Steiner problem on trees. The notation and description are only superficially different from those in [8].

3.1 Greedy Algorithm from [8]

Algorithm GS gets as input, a subtree $T_{r'}$ rooted at r', edge weights $w(e)$, groups of terminals $\{g_i\}_i$, and a covering demand z'. To simplify notation, we refer to the input as a pair $(T_{r'}, z')$. The algorithm computes a z'-cover of $T_{r'}$.

A listing of Algorithm GS appears as Algorithm 1. The stopping condition of the algorithm is when the input subtree consists of a single leaf, in which case the subtree is returned as the cover. Here we use the assumption that every leaf is a terminal (see preprocessing in Section 2). When the input subtree is not a single leaf, the algorithm finds a z'-cover by adding augmenting trees, one by one, until a z'-cover is found. The variable z^{res} equals the residual demand, namely, the number of groups that still need to be covered. The residual tree T^{res} is the tree obtained from $T_{r'}$ as follows. A terminal is active if it belongs to a group that has not yet been covered. The tree T^{res} is the minimal subtree of $T_{r'}$ that contains all the active terminals. The while-loop in lines 3-7 iterates while the union of the augmenting trees found so far is not a z'-cover. Each iteration of the while loop consists of three stages: recursion, selection, and update. In line 4, the algorithm is called recursively for all the subtrees hanging from children of r' and for all demand values z'' in the range $[1, z^{res}]$. The tree computed by $\mathrm{GS}(T_u, z'')$ is denoted by $C_{u,z''}$. In line 5, an augmenting tree, T_{aug}, is selected as follows. For every tree $C_{u,z''}$, the weight of the edge (r', u) is added to the weight of $C_{u,z''}$, and T_{aug} is picked to be a tree of lowest density among these trees. In line 6 updating takes place. The selected augmenting tree T_{aug} is added to the cover found so far[2], and the residual demand z^{res} and residual tree T^{res} are updated. When the residual demand is zero, the union of the augmenting trees is a z'-cover, and the algorithm returns this cover.

The following two lemmas adapted from [8] summarize the analysis of the above algorithm which yields an $O(h(T_{r'}) \log m)$ approximation in time $O(n^{O(h(T_{r'}))})$.

Lemma 3.1 *The running time of Algorithm* GS *is bounded by* $O((\Delta \cdot m^2)^{O(h)})$, *where* $h = h(T_{r'})$ *and* Δ *is the maximum degree of* $T_{r'}$.

Let $\mathrm{OPT}(T^{res}_{r'}, z^{res})$ denote a min-weight z^{res}-cover in $T^{res}_{r'}$. The following lemma shows that the density of T_{aug} is at most $h(T_{r'})$ times the density of $\mathrm{OPT}(T^{res}_{r'}, z^{res})$. Note that augmenting trees are computed only if r' is not a leaf. If r' is a leaf, then the density of $T_{r'}$ is zero, and is obviously optimal.

Lemma 3.2

$$\gamma(T_{aug}) \leq h(T_{r'}) \cdot \gamma(\mathrm{OPT}(T^{res}_{r'}, z^{res})).$$

[2]The algorithm could reduce the cost of the edges in T_{aug} to zero after adding T_{aug} to the cover. There does not seem to be a way to use this to improve the analysis.

Algorithm 1 GS($T_{r'}, z'$) - A recursive greedy algorithm for the Group Steiner Problem.

1: **stopping condition: if** r' is a leaf **then** return ($T_{r'}$).
2: **Initialize:** $cover \leftarrow \emptyset$, $z^{res} \leftarrow z'$, and $T^{res} \leftarrow T_{r'}$.
3: **while** $z^{res} > 0$ **do**
4: **recurse:** for every $u \in \text{children}(r')$ and every $z'' \in [1, z^{res}]$

$$C_{u,z''} \leftarrow \text{GS}(T_u^{res}, z'').$$

5: **select:** (pick the lowest density tree)

$$T_{aug} \leftarrow \text{MIN-DENSITY} \left\{ C_{u,z''} \cup \{(r', u)\} \mid u \in \text{children}(r') \ \& \ z'' \in [1, z^{res}] \right\}.$$

6: **update:**

(a) $cover \leftarrow cover \cup T_{aug}$.

(b) $z^{res} \leftarrow z' - m(cover)$.

(c) remove all groups covered by T_{aug} from T^{res}.

7: **end while**
8: **return** ($cover$).

3.2 Geometric search

We now present the Modified-GS Algorithm. The modifications reduce the number of recursive calls per child of r' in each iteration as well as the number of iterations. The increase in the approximation ratio caused by these modifications can be upper bounded by a constant factor.

A listing of the Modified-GS-algorithm is given as Algorithm 2. The new or modified lines are underlined. The main change is in Line 4 where the recursive calls are with demand values that are powers of $(1+\lambda)$ in the range $[\frac{1}{\deg(r')\cdot(1+\frac{1}{\lambda})\cdot(1+\lambda)} \cdot z^{res}, z^{res}]$. This change is referred to as geometric search since the demands are only powers of $(1+\lambda)$. Small subtrees are avoided in the sense that the demand value is at least $\frac{1}{\deg(r')\cdot(1+\frac{1}{\lambda})\cdot(1+\lambda)} \cdot z^{res}$. The second change is that the algorithms stores as $cover_h$ the first partial cover that covers at least $z'/h(T_{r'})$ groups. This modification is used in the proof of Lemma 3.4 instead of the simulation argument in the proof of Lemma 3.2. The final cover that is returned is either $cover$ or $cover_h$, depending on which has a smaller density. Note that if $cover_h$ is returned in the topmost call of Modified-GS, then one needs to invoke Modified-GS again on the residual tree until a full cover is computed.

The proofs of the following two lemmas appear at the end of this section.

Lemma 3.3 *Let Δ be the maximum degree of the tree $T_{r'}$ and let $\beta = \Delta(1+1/\lambda)(1+\lambda)$. The running time of* Modified-GS($T_{r'}, z'$) *is $O(n\alpha^{h(T_{r'})})$ where $\alpha = \beta \cdot h(T_{r'}) \cdot \log z' \cdot \Delta \cdot \log_{1+\lambda} \beta$. If $h(T_{r'}) = O(\log n / \log\log n)$, $\Delta = O(\log n)$ and $1 \leq 1/\lambda = O(\log n)$, then the running time is polynomial in n and in m.*

The following lemma proves that if $1/\lambda \geq h(T_{r'})$, the modifications affect the density of the augmentation tree only by a constant factor.

Algorithm 2 Modified-GS($T_{r'}, z'$) - Modified GS Algorithm (uses geometric search).

1: **stopping condition:** **if** r' is a leaf **then** return $(T_{r'})$.
2: **Initialize:** $cover \leftarrow \emptyset$, $z^{res} \leftarrow z'$, and $T^{res} \leftarrow T_{r'}$.
3: **while** $z^{res} > 0$ **do**
4: **recurse:** for every $u \in \text{children}(r')$ and for every z'' power of $(1+\lambda)$ in $\underline{[\frac{1}{\deg(r')\cdot(1+\frac{1}{\lambda})\cdot(1+\lambda)} \cdot z^{res}, z^{res}]}$

$$C_{u,z''} \leftarrow \text{Modified-GS}(T_u, z'').$$

5: **select:** (pick the lowest density tree)

$$T_{aug} \leftarrow \text{MIN-DENSITY}\left\{C_{u,z''} \cup \{(r',u)\}\right\}.$$

6: **update:**

(a) $cover \leftarrow cover \cup T_{aug}$.

(b) $z^{res} \leftarrow z' - m(cover)$.

(c) remove all groups covered by T_{aug} from T^{res}.

(d) **if** first time $m(cover) \geq z'/h(T_{r'})$ **then** $cover_h \leftarrow cover$.

7: **end while**
8: **return** (lowest density tree $\in \{cover, cover_h\}$).

Lemma 3.4

$$\gamma(T_{aug}) \leq (1+\lambda)^{2h(T_{r'})} \cdot h(T_{r'}) \cdot \gamma(\text{OPT}(T_{r'}^{res}, z^{res})).$$

We obtain the following theorem from the above two lemmas.

Theorem 3.5 *Let I be an instance of the group Steiner problem on a tree T of height $O(\log n/\log\log n)$ and maximum degree $O(\log n)$. Then* Modified-GS *runs in time polynomial n and m and gives an $O(h(T)\log m)$-approximation.*

Proof: Choose $\lambda = 1/h(T)$ in Modified-GS. For this choice of λ and the bounds on the height and degree of T it follows from Lemma 3.3 that Modified-GS runs in time polynomial in n and m.

From Lemma 3.4, we obtain that $\gamma(T_{aug}) \leq (1+1/h(T))^{2h(T)} \cdot h(T) \cdot \gamma(\text{OPT}(T^{res}, z^{res}))$. Therefore $\gamma(T_{aug}) \leq e^2 h(T)\gamma(\text{OPT}(T^{res}, z^{res}))$. It follows that we obtain an $O(h(T)\log m)$ approximation. □

Corollary 3.6 *For any fixed $\varepsilon > 0$, there is a polynomial time recursive greedy algorithm for the group Steiner problem on trees with an approximation ratio of $O(\frac{1}{\varepsilon} \cdot (\log n)^{1+\varepsilon} \cdot \log m)$.*

Proof: Use Claim 2.4 with $\alpha = \log^{\varepsilon} n$ to reduce the height of the input tree to $O(\log n/\log\log n)$ and use Claim 2.5 with $\beta = \log n$ to reduce the maximum degree of the tree to $O(\log n)$ while still keeping the height $O(\log n/\log\log n)$. These transformations worsen the approximation ratio by a multiplicative factor of $O(\log^{\varepsilon} n)$. Applying the algorithm Modified-GS to the transformed tree gives the desired result. □

Now we prove Lemmas 3.3 and 3.4.

Proof of Lemma 3.3: Let $t(h, z)$ denote the running time of Modified-GS on a tree of height h and with z terminals. The recurrence relation for $t(h, z)$ is obtained by bounding the number of recursive calls in Modified-GS. In line 4 each child of r' is called with at most $\log_{1+\lambda} \beta$ values of z'' since we do a geometric search with powers of $(1+\lambda)$ in the range $[z^{res}/\beta, z^{res}]$. Hence the total number of calls in line 4 is $\Delta \cdot \log_{1+\lambda} \beta$. A call to Modified-GS with a covering requirement of z' returns a tree with either z' groups or $z'/h(T_{r'})$ groups. Hence in every iteration of the while loop in line 3, z^{res} is reduced by at least a factor of $(1 - 1/(\beta h(T_{r'})))$. Hence the number of iterations of the while loop is at most $\beta h(T_{r'}) \log z'$. Let α be the total number of recursive calls. From the above we can bound α by $\beta h(T_{r'}) \log z' \cdot \Delta \log_{1+\lambda} \beta$.

For each recursive call, the amount of processing required to compute the density of the returned tree and remove the covered groups is linear in $s = \sum_i |g_i|$. Hence we can write a recurrence relation for $t(h, z')$ as $t(h, z') \leq \alpha t(h-1, z') + cs$ for some constant c. From this we obtain $t(h, z') \leq cs\alpha^h$.

When $1/\lambda$, Δ, and $h(T_{r'})$ are $O(\log n)$ it is easy to verify that α is poly-logarithmic in s (or, equivalently, in n and m). If $h(T_{r'})$ is $O(\log n/ \log\log n)$ it follows that $\alpha^{h(T_{r'})}$ is polynomial in n and m. $\square$

Proof of Lemma 3.4: The proof is by induction on the height of $T_{r'}$. The induction basis for $h(T_{r'}) = 1$ follows from the fact that the density of T_{aug} is optimal. This follows simply by the fact that T_{aug} is an edge to a closest leaf.

The induction step is proved as follows. Let $\{u_1, u_2, \ldots, u_k\}$ denote the set of children of r'. Let $Q^* = \textsc{opt}(T_{r'}^{res}, z^{res})$. Decompose Q^* into the trees $Q^*_{(r',u_1)} \cup Q^*_{(r',u_2)} \cup \cdots Q^*_{(r',u_k)}$. Recall that $Q^*_{(r',u_i)}$ is the tree $Q^*_{u_i}$ together with the edge (r', u_i).

We distinguish between subtrees that cover a large number of groups and those that cover few. A subtree $Q^*_{(r',u_i)}$ is *bad* if $m(Q^*_{(r',u_i)}) < \frac{z^{res}}{\deg(r')\cdot(1+1/\lambda)}$; otherwise it is *good*. Observe that the union of all bad subtrees covers at most $z^{res}/(1+1/\lambda)$ groups. Hence the union of all good subtrees, denoted by Q^*_{big}, covers at least $z^{res}/(1+\lambda)$ groups. It follows that $\gamma(Q^*_{big}) \leq \gamma(Q^*) \cdot (1+\lambda)$. By a simple averaging argument it also follows that the density of at least one of the good subtrees is at most $\gamma(Q^*_{big})$. Without loss of generality, assume that $Q^*(r', u_1)$ is good and that $\gamma(Q^*(r', u_1)) \leq \gamma(Q^*_{big})$. It follows that

$$\gamma(Q^*_{(r',u_1)}) \leq \gamma(Q^*_{big}) \leq (1+\lambda) \cdot \gamma(Q^*). \tag{1}$$

Let $z_i^* = m(Q^*_{(r',u_i)})$, namely, z_i^* is the number of groups covered by $Q^*_{(r',u_i)}$. Let z_1 denote the integral power of $(1+\lambda)$ such that $z_1 \leq z_1^* < (1+\lambda) \cdot z_1$. Note that z_1 is in the range of powers of $(1+\lambda)$ considered in Line 4 (in fact, this is why the threshold between bad and good subtrees is divided by an extra factor of $(1+\lambda)$). Consider the execution of $C_{u_1,z_1} \leftarrow$ Modified-GS$(T_{u_1}^{res}, z_1)$ in Line 4. The tree C_{u_1,z_1} is incrementally constructed from a sequence of augmenting trees, denoted by $\{R_1, R_2, \ldots\}$. Let i denote the smallest integer such that

$$m(\cup_{j \leq i} R_j) \geq \frac{z_1}{h(T_{r'})}. \tag{2}$$

By the definition of i, it follows that $cover_h = \cup_{j \leq i} R_j$ in the execution of Modified-GS$(T_{u_1}^{res}, z_1)$. Observe that, during all the iterations of the while loop in which C_{u_1,z_1} is computed, the subtree $Q^*_{u_1}$ is a cover that covers the residual demand. This implies that the weight of a min-weight cover of the residual demand is not greater than $w(Q^*_{u_1})$. Moreover, the residual demand when R_j is computed, for $j \leq i$, is at least $z_1 - z_1/h(T_{r'})$. Therefore, the induction hypothesis when applied to R_j, for $j \leq i$, implies

$$\begin{aligned}
\gamma(R_j) &\leq (1+\lambda)^{2h(T_{u_1})} \cdot h(T_{u_1}) \cdot \frac{w(Q^*_{u_1})}{z_1 - z_1/h(T_{r'})} \\
&= (1+\lambda)^{2h(T_{r'})-2} \cdot (h(T_{r'})-1) \cdot \frac{w(Q^*_{u_1})}{z_1 - z_1/h(T_{r'})} \qquad \text{(using } h(T_{u_1}) = h(T_{r'})-1\text{)} \qquad (3) \\
&= (1+\lambda)^{2h(T_{r'})-2} \cdot h(T_{r'}) \cdot \frac{w(Q^*_{u_1})}{z_1}.
\end{aligned}$$

Since $\gamma(\cup_{j\leq i} R_j) \leq \max_{j\leq i} \gamma(R_j)$, it follows that

$$\gamma(\cup_{j\leq i} R_j) \leq (1+\lambda)^{2h(T_{r'})-2} \cdot h(T_{r'}) \cdot \frac{w(Q^*_{u_1})}{z_1}. \qquad (4)$$

Since T_{aug} is selected to be a tree of min-density among *cover* and $cover_h$ in the execution of Modified-GS($T^{res}_{u_1}, z_1$), it follows that

$$\begin{aligned}
\gamma(T_{aug}) &\leq \frac{w(\cup_{j\leq i} R_j) + w(r', u_1)}{m(\cup_{j\leq i} R_j)} \\
\text{(by Eqs. 4 \& 2)} &\leq (1+\lambda)^{2h(T_{r'})-2} \cdot h(T_{r'}) \cdot \frac{w(Q^*_{u_1})}{z_1} + \frac{w(r', u_1)}{z_1/h(T_{r'})} \\
\text{(since } z_1^* \leq z_1(1+\lambda)\text{)} &\leq (1+\lambda)^{2h(T_{r'})-2} \cdot h(T_{r'}) \cdot \frac{w(Q^*_{u_1})}{z_1^*/(1+\lambda)} + h(T_{r'}) \cdot \frac{w(r', u_1)}{z_1^*/(1+\lambda)} \\
&\leq (1+\lambda)^{2h(T_{r'})-1} \cdot h(T_{r'}) \cdot \frac{w(Q^*_{u_1}) + w(r', u_1)}{z_1^*} \\
\text{(by definition)} &= (1+\lambda)^{2h(T_{r'})-1} \cdot h(T_{r'}) \cdot \gamma(Q^*_{(r', u_1)}) \\
\text{(by Eq. 1)} &\leq (1+\lambda)^{2h(T_{r'})} \cdot h(T_{r'}) \cdot \gamma(Q^*).
\end{aligned}$$

This proves the lemma. □

4 Height reducing transformation

In this section we present a height reducing transformation that proves Claim 2.4. The transformation is defined by the following recursive procedure. Given a rooted tree T, a special subtree $Q \subseteq T$, called an *α-decomposition*, is found. The subtree Q is a prefix of T (i.e., the root of T is also the root of Q and the leaves of Q may be internal vertices of the original tree T). Loosely speaking, the subtree Q induces a partition of T into subtrees that have $1/\alpha$ as many nodes as the whole tree. The subtree Q is substituted by an $O(\alpha)$-faithful representation Q' of Q. This $O(\alpha)$-representation is a height-3 tree with the same root and leaf-set. This procedure is then applied recursively to the subtrees rooted at the leaves of Q. Namely, as a leaf u in Q roots a subtree T_u in T, we recursively repeat this modification in T_u. The reduction by a factor of α in the number of terminals per three levels reduces the height to $3 \cdot \log_\alpha n$. Since every α-decomposition is substituted by an $O(\alpha)$-faithful representation, the penalty incurred by this transformation is $O(\alpha)$. We choose $\alpha = \log^\varepsilon n$ to reduce the height to $O(\log n / \log\log n)$ so that the recursive greedy algorithm runs in polynomial time. Interestingly, setting α to a constant reduces the height to $O(\log n)$ while incurring only a constant (multiplicative) penalty. Our transformation runs in linear time.

4.1 α-decompositions

An α-decomposition of a tree T_r is a partition of T_r into α sub-trees, each subtree containing n_r/α terminals. However, such a partition may not be possible; consider, for example, the case when T_r is a star. We therefore need to deal with the situation that there are many "light" descendants.

Let u denote a descendant of r. Let $\alpha > 1$. A node u is *α-light* with respect to T_r if $n_u \leq n_r/\alpha$. A node u is *α-heavy* with respect to T_r if $n_u > n_r/\alpha$. A node u is *minimally α-heavy* if u is α-heavy and v is α-light, for every child v of u. A node u is *maximally α-light* if u is α-light and $p(u)$ is α-heavy. We fix α upfront and hence, for ease of notation, we refer to α-heavy nodes as *heavy* and to α-light nodes as *light*.

Definition 4.1 *A subtree $Q \subseteq T_r$ is an* α-decomposition *of T_r if $r \in Q$ and every leaf of Q is maximally α-light.*

Definition 4.2 *The* skeleton *of an α-decomposition Q is the subtree* $\mathrm{sk}(Q) \subset Q$ *induced by all the α-heavy nodes in Q.*

Returning to the example in which T_r is a star, note that in this case $Q = T_r$ is an α-decomposition of T_r, and the skeleton is simply $sk(Q) = \{r\}$. An α-decomposition of T_r is easy to compute: explore the subtree T_r via depth first search stopping the exploration of a node's children if it is a maximally light node.

Note that every leaf in a skeleton is minimally α-heavy, and therefore, the number of leaves in the skeleton at most α. We refer to the edges in Q that are incident to light leaves as the *fluff* of Q. Every edge of an α-decomposition Q is either an edge in the skeleton $sk(Q)$ or an edge in the fluff of Q, but not both.

4.2 Promotion of α-decompositions

In this section we describe how the height reducing transformation substitutes an α-decomposition Q of T_r by a tree Q' of height 3. We also describe a mapping π from the nodes of Q to those in Q' which will be used to establish the $O(\alpha)$-faithfulness of the transformation.

Branches. A *branch* is a maximal subpath in $sk(Q)$ between two branching points (i.e., nodes with at least two children). There are at most $(2\alpha - 1)$ branches since there are at most α leaves in $sk(Q)$. To avoid inclusion of branching points in multiple branches, we assume that (except for the root) a branching point belongs to the branch above it. The root belongs to one of the branches that emanate from the root.

Bunches. Fix a branch B of $sk(Q)$. Denote the endpoint of B closer to the root of Q by v. Form bunches $B_0, B_1, B_2 \ldots$ of vertices along B as follows. The first bunch B_0 is defined as follows:

$$B_0 \triangleq \{u \in B \mid w(path(v,u)) = 0\}.$$

For every positive integer i, the i'th bunch, denoted by B_i, is defined as follows:

$$B_i \triangleq \{u \in B \mid w(path(v,u)) \in [2^{i-1}, 2^i)\}.$$

Recall that nonzero edge weights are at least 1, so there are no vertices between B_0 and B_1. Since the start-vertex v of a branch B belongs to the branch above it, v does not belong to the bunch B_0.

Promotion. We now create a height-3 tree Q' which has the same number of leaves as Q by *promoting* bunches in branches as follows. Figure 1 depicts the promotion of bunches along a single branch. Intuitively, a path from r to a light leaf ℓ is divided into 3 parts. The first part is the path from r to v, the start-vertex of the bunch of ℓ. The second part is the path from v to $p(\ell) \in B$, and the third part is the edge $(p(\ell), \ell)$. This path is replaced with a path of length 3; the weight of first and last edges in this path equals the weight of the corresponding part in $path(r, \ell)$. The weight of the middle edge is a power of two and approximates the weight of $path(v, p(\ell))$.

For every branch B, the following subtree is constructed. Let r' denote the root of Q'. Add a node $v(B)$ in Q', that corresponds to v, and an edge $(r', v(B))$. The edge $(r', v(B))$ is given weight equal to the weight of the path from r to v. The bunches B_i are promoted as follows. For every non-empty bunch B_i, add a new node b_i and an edge $(v(B), b_i)$. For every leaf $\ell \in \mathcal{L}(Q)$ hanging from a node in B_i, we create a leaf $\ell' \in \mathcal{L}(Q')$ that hangs from b_i. Weights are assigned as follows: (a) $w(v(B), b_0) = 0$, (b) $w(v(B), b_i) \leftarrow 2^i$, if $i > 0$, and (c) $w(b_i, \ell') \leftarrow w(p(\ell), \ell)$, for a leaf ℓ hanging from a vertex in B_i.

The mapping π maps the nodes $V(Q)$ to $V(Q')$ as follows. The root of Q is mapped to the root of Q'. For a branch B, all the nodes in B_i are mapped to the node b_i. Every leaf $\ell \in \mathcal{L}(Q)$ is mapped to its counterpart $\ell' \in \mathcal{L}(Q)$.

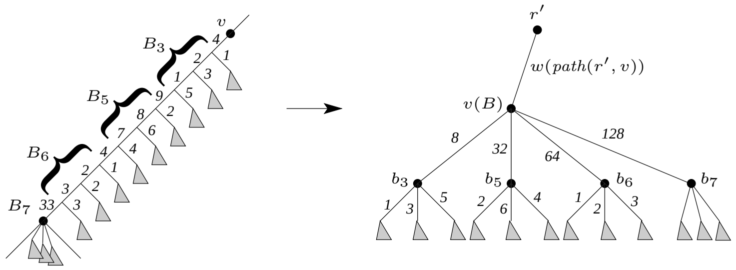

Figure 1: Promotion of bunches along a single branch. Depth of light leaves after promotion is three.

4.3 The transformation

The height reducing transformation proceeds as follows. If r is a leaf, then it returns a copy of T_r. Otherwise, an α-decomposition Q is computed. A height 3 subtree Q' is created from Q. The transformation is then applied recursively to the subtrees rooted at the leaves of Q and they are attached appropriately to Q'. The mapping π is defined in every step of the recursion as described above.

4.4 Analysis of the height reducing transformation

Let T' denote the tree resulting when the height reducing transformation is applied to the tree T. In this section we prove that $h(T')$ is $O(\log_\alpha n)$ and that T' is an $O(\alpha)$-faithful representation of T.

Consider a single promotion step applied to an α-decomposition $Q \subseteq T$. Promotion substitutes Q by a tree Q' of depth 3. It follows that the height $h(n)$ of a reduced tree with n terminals satisfies the recurrence $h(n) \leq 3 + h(n/\alpha)$. This yields the following claim.

Claim 4.3 $h(T') \leq 3 \cdot \log_\alpha n$.

The following proves the faithfulness of the height reducing transformation.

Claim 4.4 *T' is an $O(\alpha)$-faithful representation of T.*

Proof: Let $Q_1, Q_2, \ldots, Q_k \subseteq T$ denote the sequence of α-decompositions computed during the height reduction transformation. By construction, the edge sets of the subtrees in this sequence partition the edge set of T. Let Q'_i denote the height-3 subtree of T' that is used to promote Q_i. By definition, the edge sets of $\{Q'_i\}_i$ also partition the edge set of T' into disjoint parts. The transformation is local in the sense that $\pi(V(Q_i)) = V(Q'_i)$.

We say that a node $v \in V(T)$ is a *border point* if it belongs to more than one subtree Q_i. Note that v is a border point iff it is a light leaf in one Q_i and a root of another Q_j.

Consider a set of vertices $S \subseteq V(T)$. We may assume that S contains all the border points in $T[S]$. Namely, we add all border points in $T[S]$ to S, and this does not affect $T[S]$. Let $S_i = S \cap V(Q_i)$, $S' = \pi(S)$, and $S'_i = \pi(S_i)$. It follows that

$$w_T(T[S]) = \sum_{i=1}^{k} w_{Q_i}(Q_i[S_i])$$
$$w_{T'}(T'[S']) = \sum_{i=1}^{k} w_{Q'_i}(Q'_i[S'_i]).$$

The above equalities hold if $S' \subseteq \pi(V(T))$ and $S = \pi^{-1}(S')$. Hence, it suffices to prove that Q'_i is an $O(\alpha)$-faithful representation of Q_i, for every $1 \leq i \leq k$.

For this purpose we consider a single α-decomposition Q rooted at r and its height-3 substitute Q'. The main issue in proving that Q' is an $O(\alpha)$-faithful representation of Q is that we have a separate subtree in Q' for every branch in Q. This means that if there are several branches "below" an edge e, then $w(e)$ is counted multiple times. Luckily, the number of branches is $O(\alpha)$, so the increase in weight can be bounded by $O(\alpha)$. However, we also have multiple counting within each branch since bunches are connected separately. Here we utilize the fact that weights of edges $(v(B), b_i)$ increase exponentially with i, and hence their sum is dominated by the heaviest edge. We now provide a rigorous proof.

Consider a single branch B. We use the notation used in the description of the promotion of bunches along a single branch. Let B^+ denote B together with the light leaves hanging from it. We focus now on $S_B = S \cap B^+$ and $S'_B = \pi(S_B)$. Assume that S_B is not empty (and therefore S'_B is also not empty). We claim that

$$w_Q(Q[S_B]) \leq w_{Q'}(Q'[S'_B]) \leq 4 \cdot w_Q(Q[S_B]). \tag{5}$$

Let $u \in B$ denote the "deepest" node in $Q[S_B]$ and let v be start node of B. The subtree $Q[S_B]$ consists of three types of edges: (i) edges along the path from the root r to v, (ii) edges along the path from v to u, and (iii) edges from nodes in B to light leaves in B^+. We therefore rewrite $w_Q(Q[S_B])$ as follows:

$$w_Q(Q[S_B]) = w_Q(path(r, v)) + w_Q(path(v, u)) + \sum_{\ell \in S_B \cap \mathcal{L}(B^+)} w_Q(p(\ell), \ell). \tag{6}$$

Let i denote the index of the bunch that u belongs to (i.e., $u \in B_i$). The subtree $Q'[S'_B]$ consists of three types of edges: (i) the edge $(r', v(B))$ whose weight equals $w_Q(path(r, v))$, (ii) edges $(v(B), b_j)$ whose weight is 2^j, and (iii) edges from nodes in bunch nodes b_j to leaves. We know that $b_i \in Q'[S'_B]$

since $u \in B_i$. Hence, edges of the second type contribute at least 2^i. The bunch nodes b_j in $Q'[S'_B]$ are a subset of $\{b_1, \ldots, b_i\}$. Hence, the edges of the second type contribute at most $\sum_{j=1}^{i} 2^j$. It follows that

$$w_{Q'}(Q'[S'_B]) \leq w_Q(path(r, v(B))) + \sum_{j=1}^{i} 2^j + \sum_{\ell' \in S'_B \cap \pi(\mathcal{L}(B^+))} w_{Q'}(p(\ell'), \ell') \tag{7}$$

$$w_{Q'}(Q'[S'_B]) \geq w_Q(path(r, v(B))) + 2^i + \sum_{\ell' \in S'_B \cap \pi(\mathcal{L}(B^+))} w_{Q'}(p(\ell'), \ell'). \tag{8}$$

Note that the contribution of edges of the third type in $Q[S_B]$ and $Q'[S'_B]$ is identical. It follows that the only difference between $w_Q(Q[S_B])$ and the bounds on $w_{Q'}(Q'[S'_B])$ in the rewriting above is in the middle terms. Since $u \in B_i$, it follows that $2^{i-1} \leq w_Q(path(v,u)) < 2^i$, and Equation 5 follows.

We are now ready to complete the proof of the $O(\alpha)$-faithfulness of Q'. Since each branch B is mapped to a separate subtree in Q', it follows that

$$w_{Q'}(Q'[S']) = \sum_B w_{Q'}(Q'[S'_B]). \tag{9}$$

By Equation 5, the term $\sum_B w_{Q'}(Q'[S'_B])$ is bounded by $4 \cdot \sum_B w_Q(Q[S_B])$. Since there are at most $(2\alpha - 1)$ branches, it follows that $\sum_B w_{Q'}(Q'[S'_B]) = O(\alpha) \cdot \max_B\{w_Q(Q[S_B])\}$. However, $\max_B\{w_Q(Q[S_B])\} \leq w_Q(Q[S])$. It follows that

$$w_{Q'}(Q'[S']) \leq O(\alpha) \cdot w_Q(Q[S]).$$

This completes the first part of the proof.

To prove the second part, we consider a set of terminals $S' \subseteq \pi(V(Q))$ and define $S = \pi^{-1}(S')$. We need to show that

$$w_Q(Q[S]) \leq w(Q'[S']).$$

By Eq. 9 and Eq. 5, it follows that

$$w_{Q'}(Q'[S']) \geq \sum_B w_Q(Q[S_B]).$$

However, $\sum_B w_Q(Q[S_B]) \geq w_Q(Q[S])$, and the claim follows. □

5 Degree reducing transformation

In this section we present a degree reducing transformation that proves Claim 2.5. Given a rooted tree T and an integer $\beta \geq 3$, the transformation produces a 1-faithful representation $\rho(T)$ of T. The rooted tree $\rho(T)$ satisfies: (a) each node in $\rho(T)$ has at most β children, and (b) the height of $\rho(T)$ is at most $h(T) + \lceil \log_{\beta/2} n \rceil$.

Given a tree T rooted at u and a parameter β, the tree $\rho(T)$ is constructed recursively as follows. If u is a leaf, then the algorithm returns u. Otherwise, the subtree induced by the edges between u and its children is locally transformed as follows. Let $v_1, v_2, \ldots, v_k$ denote the children of u.

1. The β-heavy children v_i of u (i.e., such that $n_{v_i} \geq n_u/\beta$) are not changed; the edges (u, v_i) are kept and their weight is not modified.

2. The β-light children of u are grouped arbitrarily into minimal bunches such that each bunch (except perhaps for the last) is β-heavy. Note that the number of leaves in each bunch (except perhaps for the last bunch) is in the interval $[n_u/\beta, 2n_u/\beta)$. For every bunch B, a new node b is created. An edge (u,b) is added as well as edges between b and the children of u in B. The edge weights are set as follows: (a) $w(u,b) \leftarrow 0$, and (b) $w(b,v_i) \leftarrow w(u,v_i)$.

After the local transformation, let $v'_1, v'_2, \ldots, v'_j$ be the new children of u. Some of these children are the original children and some are the new vertices introduced in the bunching. The tree $\rho(T)$ is obtained by recursively processing the subtrees $T_{v'_i}$, for $1 \le i \le j$, in essence replacing $T_{v'_i}$ by $\rho(T_{v'_i})$. Note that after processing, the number of children of u is at most β because the subtrees $\{T_{v'_i}\}_i$ partition the nodes of $V(T_u) - \{u\}$ and each tree except, perhaps one, is β-heavy. The recursion is applied to each subtree $T_{v'_i}$, and hence $\rho(T)$ will satisfies the degree requirement, as claimed. The 1-faithfulness of $\rho(T)$ follows from the fact that the "shared" edges (u,b), that were created for bunching together β-light children of u, have zero weight.

We now bound the height of $\rho(T)$. Given a tree of height h and n nodes let $\gamma(h,n)$ be the height of the tree that results when the above procedure is applied. From the recursive procedure, we have that $h(\rho(T)) = 1 + \max_{i=1}^{j} h(\rho(T_{v'_i}))$. If v'_i corresponds to a β-heavy child of u, then $h(T_{v'_i}) \le h(T) - 1$ and $n(T_{v'_i}) \le n$. If v'_i is formed by bunching together β-light children of u then $n_{v'_i} < 2n_u/\beta$ and $h(T_{v'_i}) \le h(T)$. Therefore $\gamma(h,n)$ satisfies the following recurrence:

$$\gamma(h,n) \le \begin{cases} 0 & \text{if } h = 0 \\ 1 + \max\{\gamma(h-1,n), \gamma(h, \lfloor \frac{2n}{\beta} \rfloor\}) & \text{otherwise.} \end{cases}$$

It follows that the height of $\rho(T)$ is bounded by $h(T) + \lfloor \log_{\beta/2} n \rfloor$, as required.

6 Conclusions

We conclude the paper with a few remarks.

An approximation algorithm for the covering Steiner problem on trees: The covering Steiner problem generalizes the group Steiner problem; in addition to the graph and groups we are given an integer demand d_i for every group g_i and the goal is to cover, for each i, at least d_i terminals from g_i. In the $\frac{1}{2}$-group Steiner problem the input is the same as in the group Steiner problem, but the goal is to compute a minimum-weight tree containing a terminal from at least half the groups. Poly-logarithmic approximation algorithms for the covering Steiner problem are given in [18, 19] and these algorithms rely on solving an LP relaxation for the problem. In [10] a simple randomized procedure is applied to show that a ρ ratio approximation for the $\frac{1}{2}$-group Steiner problem can be used to approximate the covering Steiner problem within $\rho \log(\sum_i d_i)$. Our algorithm for the group Steiner problem on trees can be used to derive an $O((\log n)^{1+\varepsilon})$ approximation algorithm for the $\frac{1}{2}$-group Steiner problem on trees and hence, $O(\log^{2+\epsilon} n)$ ratio algorithm for the covering Steiner problem on trees. It can also be used to obtain an $O(\log^{3+\epsilon} n)$ ratio for the covering Steiner problem on graphs, using [11]. Thus we obtain an algorithm that does not rely on solving linear programs.

On improving the $\log^2 n$ ratio: Our algorithm can be modified to give an $O(\log^2 n / \log\log n)$ ratio algorithm for the group Steiner problem on trees which would slightly improve upon the best known ratio [14, 24]. The main idea is to *guess* all the minimally $\log n$-heavy nodes in the optimum solution. We note however that the algorithm would run in quasi-polynomial time. One should contrast this result with the recent $\Omega(\log^{2-\epsilon} n)$ hardness of approximation for this problem [16] where $\varepsilon > 0$ is any fixed constant. The hardness of approximation does not preclude a polynomial time algorithm that achieves an $O(\log^2 n/\log\log n)$ ratio.

Currently the only way to get a poly-logarithmic approximation for the group Steiner problem on graphs is to first reduce it to the tree case. This reduction incurs a logarithmic factor in the approximation ratio. Is it possible to avoid this reduction and work directly with graphs? This would improve the ratio by a logarithmic factor.

Directed Steiner tree problem: Currently there is no poly-logarithmic approximation ratio for the directed Steiner tree problem that runs in polynomial time (the algorithm in [8] runs in quasi-polynomial time). Geometric search and height reduction can be applied to directed acyclic graphs (DAGs). However there is no degree reducing transformation for DAGs that has the same properties as those for trees. We believe that with some more sophisticated ideas, the greedy algorithm can be adapted to give a polynomial time poly-logarithmic approximation for the directed Steiner tree problem.

Height reduction: The height reducing transformation presented in this paper loses only a constant factor to reduce the height of the tree to $O(\log n)$. On the other hand, the reduction procedure of Zelikovsky [23, 17] loses an $\Omega(\log n)$ factor to achieve a similar reduction. The analysis in [8] relies on height reduction and hence it might appear that a logarithmic factor can be saved by using the transformation from this paper. However, the transformation in this paper requires an explicit tree and does not seem to be adequate for the algorithm and analysis in [8].

Acknowledgments Chandra Chekuri thanks Moses Charikar for useful discussions, in particular on the degree reducing transform. We thank the anonymous referees for extensive comments and suggestions that improved the presentation of the paper.

References

[1] Y. Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. *Proc. of FOCS*, 184–93, 1996.

[2] Y. Bartal. On approximating arbitrary metrics by tree metrics. *Proc. of STOC*, 161–168, 1998.

[3] Y. Bartal, M. Charikar, D. Raz. Approximating min-sum k-clustering in metric spaces. *Proc. of STOC*, 11–20, 2001.

[4] C. D. Bateman and C. S. Helvig and G. Robins and A. Zelikovsky. Provably good routing tree construction with multi-port terminals. *Proc. of ACM/SIGDA International Symposium on Physical Design* , 96–102, 1997.

[5] M. M. Bern and P. Plassmann. The Steiner problem with edge lengths 1 and 2. *Inform. Process. Lett*, 32, 171-176, 1989.

[6] M. Charikar, C. Chekuri, A. Goel, and S. Guha. Rounding via trees: deterministic approximation algorithms for group Steiner trees and k-median. *Proc. of STOC*, 114–123, 1998.

[7] M. Charikar, C. Chekuri, A. Goel, S. Guha, and S. Plotkin. Approximating a finite metric by small number of trees. *Proc. of FOCS*, 379–388, 1998.

[8] M. Charikar, C. Chekuri, T. Cheung, Z. Dai, A. Goel, S. Guha and M. Li. Approximation Algorithms for directed Steiner Problems. *Journal of Algorithms*, 33, 73–91, 1999.

[9] G. Even and G. Kortsarz. An approximation algorithm for the Group Steiner Problem. *Proc. of SODA*, 49–58, 2002.

[10] G. Even, G. Kortsarz, and W. Slany. On Network Design Problems: Fixed Cost Flows and the Covering Steiner Problem. *Proc. of SWAT*, 318–327, 2002.

[11] J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *Proc. of STOC*, 448–455, 2003.

[12] U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45:634-652, 1998.

[13] M.R. Garey and D.S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. *W.H. Freeman and Company*, 1979.

[14] N. Garg and G. Konjevod and R. Ravi, A polylogarithmic approximation algorithm for the Group Steiner tree problem. *Journal of Algorithms*, 37, 66-84, 2000. Preliminary version in *Proc. of SODA*, 253–259, 1998.

[15] E. Halperin, G. Kortsarz, R. Krauthgamer, A. Srinivasan, and N. Wang. Integrality ratio for Group Steiner Trees and Directed Steiner Trees. *Proc. of SODA*, 275–284, 2003.

[16] E. Halperin and R. Krauthgamer, Polylogarithmic inapproximability. *Proc. of STOC*, 585-594, 2003.

[17] C. H. Helvig, G. Robins, and A. Zelikovsky. Improved approximation scheme for the group Steiner problem. *Networks*, 37(1):8–20, 2001.

[18] G. Konjevod and R. Ravi, An Approximation Algorithm for the Covering Steiner Problem. *Proc. of SODA*, 338–334, 2000.

[19] G. Konjevod, R. Ravi, and A. Srinivasan. Approximation Algorithms for the Covering Steiner Problem. *Random Structures and Algorithms* 20, 465–482, 2002.

[20] G. Kortsarz and D. Peleg. Approximating the Weight of Shallow Steiner Trees. *Discrete Applied Math*, 93, 265-285, 1999.

[21] R. Raz and S. Safra. A Sub-Constant Error-Probability Low-Degree test and a Sub-Constant Error-Probability PCP Characterization of NP. *Proc. of STOC*, 475–484, 1997.

[22] G. Reich and P. Widmayer. Beyond Steiner's problem: A VLSI oriented generalization. *Proc. of Graph-Theoretic Concepts in Computer Science (WG-89)*, LNCS volume 411, 196–210, 1990.

[23] A. Zelikovsky. A series of approximation algorithms for the acyclic directed Steiner tree problem. *Algorithmica*, 18: 99-110, 1997.

[24] L. Zosin and S. Khuller. On directed Steiner trees. *Proc. of SODA*, 59–63, 2002.