A polylogarithmic approximation algorithm for the group Steiner tree problem

Naveen Garg* Goran Konjevod† R. Ravi‡

Abstract

Given a weighted graph with some subsets of vertices called groups, the group Steiner tree problem is to find a minimum-weight subgraph which contains at least one vertex from each group. We give a randomized algorithm with a polylogarithmic approximation guarantee for the group Steiner tree problem. The previous best approximation guarantee was $O(i^2k^{1/i})$ in time $O(n^ik^{2i})$ (Charikar, Chekuri, Goel and Guha).

Our algorithm also improves existing approximation results for network design problems with location-based constraints and for the symmetric generalized traveling salesman problem.

Key Words: Steiner tree, approximation algorithms, set cover, randomized rounding, network design, tree decompositions

1 Introduction.

1.1 Motivation.

The group Steiner problem was introduced by Reich and Widmayer [27]. The problem arises in wire routing with multi-port terminals in physical VLSI design. The traditional model assuming single ports for each of the terminals to be connected in a net of minimum length is a case of the classical Steiner tree problem. When the terminal is a collection of different possible ports, so that the net can be connected to any one of them, we have a group Steiner tree problem: each terminal is a collection of ports and we seek a minimum length net containing at least one port from each terminal group.

The multiple port locations for a single terminal may also model different choices of placing a single port by rotating and/or mirroring the module containing the port in the placement. The choice allows for more interaction between the placement and routing phases of physical VLSI design, potentially allowing for better optimization of the design.

The group Steiner tree problem can be stated formally as follows: we are given a graph $G = (V, E)$ with the cost function $c : E \to \mathbb{R}^+$, and subsets of vertices $g_1, g_2, \ldots g_k \subset V$. We call $g_1, \ldots, g_k$ *groups*. The objective is to find the minimum cost subtree T of G that contains at

*Computer Science and Engineering, Indian Institute of Technology, New Delhi 110016. Work done while the author was at the Max-Planck Institut für Informatik, Im Stadtwald, 66123 Saarbrücken, Germany. Email: `naveen@cse.iitd.ernet.in`

†Dept. of Math. Sciences, Carnegie Mellon University, Pittsburgh, PA 15213-3890. Supported in part by an NSF CAREER grant CCR-9625297. Email: `konjevod@andrew.cmu.edu`

‡GSIA, Carnegie Mellon University, Pittsburgh, PA 15213-3890. Supported in part by an NSF CAREER grant CCR-9625297. Email: `ravi@cmu.edu`

least one vertex from each of the sets g_i. Formally, find a connected subgraph $T = (V', E')$ that minimizes $\sum_{e \in E'} c_e$, such that $V' \cap g_i \neq \emptyset$ for all $i \in \{1, \cdots, k\}$. We use n to denote $|V|$ and N to denote the size of the largest group, $N = \max_i |g_i| \leq n$. The following transformation allows us to assume that the groups are pairwise disjoint: if a vertex v occurs in p groups, $p \geq 1$, attach p new vertices to v with zero-cost edges. Each leaf of this star is assigned to one of the groups while v does not belong to any group.

The group Steiner tree problem is a generalization of the classical Steiner tree problem [30], and therefore NP-hard. In fact, it is also a direct generalization of the even harder set cover problem [15, 20, 29]. In the set cover problem, we are given a collection of weighted subsets of a given ground set and seek a minimum-weight sub-collection whose union is the entire ground set. To reduce this problem to a group Steiner problem, build a star with a leaf for each set. Every element in the set cover problem defines a group of leaves in the star in a natural way, namely, the leaves corresponding to the sets that contain this element. The equivalence is completed by giving the edges the weights of the corresponding sets. (Even if we require the groups to be disjoint, this construction can be realized by the transformation described above.) Therefore, the group Steiner tree problem cannot be approximated to a factor $o(\ln k)$ unless $P = NP$ [3, 9, 26].

1.2 Previous Work.

The papers of Ihler [14, 15, 16], and Ihler, Reich and Widmayer [17, 18] contain some early work on the group Steiner tree problem. (In some of these papers the group Steiner problem is called the class Steiner problem.) In particular, in [14] it is proved that the heuristic introduced by Reich and Widmayer [27] has an approximation ratio of $k - 1$ (k is the number of groups). The related problem of minimum diameter group tree is shown to be polynomially solvable in [17]. Ihler [16] gives a polynomial algorithm for a special case of the group Steiner problem where the groups of points are intervals on two parallel lines. Reich and Widmayer [18] show that the group Steiner tree problem is NP-hard even if the graph is a subgraph of a square grid in the plane, and each group has at most 3 vertices.

A special case of the group Steiner problem is the connected dominating set problem where given an unweighted undirected graph, the problem is to find a connected subgraph with the smallest number of vertices whose neighborhood covers all the vertices in the graph. This is a case of the group Steiner problem where every vertex defines a group which is its neighborhood in the graph and all edges have unit costs. The connected subgraph found as a solution is a tree without loss of generality and in the unweighted setting, the number of edges in the tree is one less than the number of vertices in the tree. An approximation algorithm for this special case with ratio $O(\log \Delta)$ where Δ is the maximum degree of the graph is presented by Guha and Khuller [12].

Slavík [29] considered the group Steiner problem on rooted trees and gave an algorithm with an approximation ratio of $B \cdot H(N) = B \cdot O(\ln N)$, where B is the maximum number of vertices of a group in a subtree of the root, and $H(N)$ is the N-th harmonic number.

Bateman, Helvig, Robins and Zelikovsky [6] gave the first algorithm with a sub-linear performance guarantee. Their algorithm gives an approximation ratio of $(1 + \frac{\ln k}{2})\sqrt{k}$. This ratio comes from approximating the group Steiner tree by a 2-star (tree of depth 2), and then approximating the covering problem on the 2-star within a logarithmic factor.

Charikar, Chekuri, Goel and Guha [7] describe a family of algorithms for the directed Steiner tree problem with running time $O(n^i k^{2i})$ and approximation ratio $O(i^2 k^{1/i})$. The directed Steiner tree problem is defined on a rooted directed weighted graph, and the objective is to find a minimum-weight set of edges that contains a path from the root to each of k given terminal vertices. Since the directed Steiner tree problem is a generalization of the group Steiner problem [20], these algorithms are directly applicable.

1.3 Our results.

For any $\epsilon > 0$, we give a polynomial time algorithm that with probability $1-\epsilon$ finds a group Steiner tree of cost $O(\log^2 n \log\log n \log k)$ times the cost of the optimal group Steiner tree. The main technical result is a randomized algorithm that solves the problem on trees with an $O(\log k \log N)$ approximation ratio. The extension to arbitrary graphs uses the result of Bartal [4, 5], and the approximation ratio for general graphs is $O(\log^2 n \log\log n \log k)$ (the size of the largest group, N, is at most the number of vertices n). The results of [21] used in place of Bartal's improve the performance ratio to $O(\log n \log\log n \log k)$ on graphs that exclude $K_{s,s}$ as a minor for some fixed constant s. An example is planar graphs that exclude $K_{3,3}$. Since planar graph distances approximate distances in the two-dimensional Euclidean plane well [8], the improvement also carries over to group Steiner problems in the plane.

Our approximation algorithm for the case of tree metrics first solves a linear programming relaxation of the group Steiner tree problem. Then an extension of randomized rounding is employed to get the solution subtree. The bound on the cost of the tree follows from the rounding process. On the other hand, to show that the solution tree actually covers all the groups with reasonable probability, we use Janson's inequality [19].

As a corollary to the performance guarantee, we also get an upper bound on the integrality gap of our linear programming relaxation.

Our algorithm works with similar performance bounds when applied to the errand scheduling problem of [29] also known as the generalized TSP [10, 13, 28], to the service-constrained network design problems of [22, 23], and the traveling purchaser problem in [25].

In the remainder of the paper, we first present our linear programming formulation and our rounding procedure for trees, and then prove the performance guarantee. Then, we describe the reduction of the general case to the case of tree metrics, and close with applications to related problems.

2 Linear program.

We consider the group Steiner tree problem on a tree $T'' = (V, E)$ with nonnegative costs c on its edges. We study the rooted version where a pre-specified root vertex r is required to be in the solution subtree. To solve the unrooted version, we can run through the different vertices in a smallest group as the choice for the root r, and pick the best solution among these runs. For any subset of vertices $S \subseteq V$, let $\delta(S)$ denote the set of edges with exactly one endpoint in S. We use the following linear programming relaxation of the (rooted) group Steiner tree problem:

$$\begin{aligned} & \min \sum_{e \in E} c_e x_e \\ (1) \qquad & \sum_{e \in \delta(S)} x_e \geq 1, \quad \text{for all } S \subseteq V \text{ such that } \quad r \in S \text{ and } S \cap g_i = \emptyset \text{ for some } i \\ & 0 \leq x_e \leq 1, \quad \forall e \in E. \end{aligned}$$

This linear program can be solved in polynomial time, despite the exponential number of constraints. This follows, for example from [11] and the fact that a separation oracle can be constructed using a minimum cut procedure. A more direct way to see the polynomial-time solvability of the program is to add new variables and re-interpret the constraints using the max-flow min-cut theorem. The constraints require that any cut separating the root from all the vertices of a given group must have capacity at least one. We can think of adding a new source vertex for this group with edges to all the vertices in it of infinite capacity and interpret the value x_e as the capacity of the edge e. Then the linear constraints and the max-flow min-cut theorem imply that any solution x must support a flow of at least one unit from this source to the root—in other words, the installed capacity x is sufficient to support a total flow of value at least one from the vertices of any group to the root. This can be written as a polynomial-sized set of linear constraints involving one set of flow variables for each group. The resulting formulation is equivalent to the above.

Note that the linear program (1) remains a valid relaxation when T'' is not a tree. In fact, we show in Section 5 that the integrality gap of the program is small even when no restrictions are imposed on the underlying graph.

Let x be the optimal solution of the linear program (1), and T' the support of x (the graph consisting of all edges e such that $x_e > 0$). Since T'' is a tree, T' is a tree as well. We denote by z^* the optimal value of the objective function.

3 Random experiment.

In this section we explain our rounding process and prove the main technical results. Our rounding may be seen as an extension of traditional randomized rounding [24] for the set cover problem to this "tree version" of the problem.

Assume without loss of generality that all group vertices are leaves of T' (internal group vertices can be made leaves by inserting a zero-cost edge).

Consider the following random experiment. For every edge $e \in E(T')$, include e in a forest F with probability x_e/x_f, where f is the edge adjacent to e and closer to r (the parent edge of e). Note that $x_e \leq x_f$ because T' is a tree. If e is incident on r, include it with probability x_e (we think of a fictitious edge above r with unit flow as the parent edge of e denoting that r is always included in F). Then delete all components of F not containing the root r, as well as every edge that is not contained in a path from r to a group vertex. Let T denote the resulting tree.

Lemma 3.1. *The expected cost of the tree T picked by the random experiment is z^*, the cost of the optimal solution to the linear program.*

Proof. We show that the probability of including any edge e in T is x_e, and the lemma follows from the linearity of expectation.

An edge e is included in T if and only if all the edges in the path from r to e, say $e_0, e_1, \ldots, e_p = e$ are picked in their respective independent random trials. This event happens with probability $x_{e_0} \cdot \prod_{i=1}^{p}(x_{e_i}/x_{e_{i-1}}) = x_e$. □

To analyze this experiment, we use Janson's inequality ([19], see also [1], p. 95), which can be stated as follows: let Ω be a universal set, and $R \subseteq \Omega$ determined by the experiment in which each element $r \in \Omega$ is independently included in R with probability p_r. In what follows I will denote a finite index set. Let $\mathcal{A} = \{A_i \mid i \in I\}$ be a family of subsets of Ω, and denote by B_i the event that $A_i \subseteq R$. Write $i \sim j$ if B_i and B_j are not independent. Define $\Delta = \sum_{i \sim j} \Pr[B_i \cap B_j]$ (the sum is over ordered pairs). Let $\mu = \sum_i \Pr[B_i]$, and ϵ be such that $\Pr[B_i] \leq \epsilon$ for all i.

Theorem 3.2. *(Janson's inequality.) With the notation as above, if $\Delta \geq \mu(1-\epsilon)$, then*

$$\Pr\left[\bigcap_i \overline{B_i}\right] \leq e^{\frac{-\mu^2(1-\epsilon)}{2\Delta}}.$$

In our case, $\Omega = E(T')$, and $p_e = x_e/x_f$. The family $\mathcal{A}$ is the family of edge-sets of paths from r to leaves belonging to a fixed group g, and $\bigcap_i \overline{B_i}$ is the event that we don't reach g in the experiment. In the sequel, we provide an upper bound on $\Pr\left[\bigcap_i \overline{B_i}\right]$ by using Janson's inequality, which implies a lower bound on the probability of including a group's vertex.

To prove the main result we need a simple lemma.

Lemma 3.3. *If T and T' are trees that differ only in the capacity of an edge e, and $x_T(e) \geq x_{T'}(e)$, then for any group g, the probability of including a vertex from g is no greater in T' than in T.*

Proof. Let A be the event that we pick a vertex of g from the subtree below e. Then

$$\Pr\left[\overline{A}\right] = 1 - x_e + x_e \Pr\left[\overline{A} \mid e \text{ is picked}\right].$$

Note that in the above expression the coefficient of x_e is always negative. Hence if x_e is decreased, $\Pr\left[\overline{A}\right]$ increases, so that $\Pr[A]$ decreases. ☐

Theorem 3.4. *If we run the random experiment on a feasible solution to LP (1), then for every group g, the probability of including a vertex from g in the chosen tree T is $\Omega(1/\log N)$ where N is the maximum size of a group.*

Proof. Consider the tree spanned by the paths from r to the leaves of a fixed group g. We will transform this tree into one where it will be easier to estimate the success probability. In the process we only decrease the success probability, so that a lower bound carries over to the original tree.

Since this tree comes from a feasible solution x to LP (1), as argued before, the capacity function x supports a flow of at least one between r and the vertices of g. Decrease the capacities on the edges so that x_e equals the value of the flow from r to g supported by the edge e. Because the groups are disjoint, the total flow to g is exactly 1. By Lemma 3.3, this only decreases the success probability.

We now have a tree with flow of 1 between the root and g. Round down all the capacities to next powers of 2. This in the worst case halves the flow from r to g. Let $N_g \leq |g|$ be the number of leaves in this tree, and let $d = \lceil \log N_g \rceil$. Delete all edges of capacity less than $1/2^{d+2}$. This reduces the flow again, but since there were only N_g leaves to begin with, the total flow we lose now is at most $N_g 2^{-(d+2)} \leq 1/4$. If there is a leaf with flow at least $1/4$, we remove all the edges except the ones on the path to this leaf. Then we reduce the capacities along this path to $1/4$. Otherwise, we remove some leaves until the flow to the remaining ones is exactly $1/4$, and then delete the edges used to carry flow only to the deleted leaves. Assume that the flow is now exactly $1/4$. Finally, shrink every edge (except the ones incident on leaves) that is preceded (on the path from the root) by another edge of the same capacity. This doesn't change the success probability, and reduces the depth of the tree to $d+1$. We abuse notation slightly and continue to denote by x the resulting capacity function carrying a flow of value exactly $1/4$ to group g. We denote by T^g the support tree of x.

For clarity, we say that an edge is a *leaf edge*, if it is incident on a leaf. We will show that $\Delta = O(\log N_g)$ in T^g. Consider a leaf edge e, and let

$$\Delta_e = \sum_{f \sim e} \frac{x_e x_f}{x_g},$$

where g is the least common ancestor of e and f. Recall that $f \sim e$ implies that f is another leaf edge to a vertex in this group whose path to r shares at least one edge with the path from e to r. Thus Δ_e is the contribution to Δ of the edge e, and $\Delta = \sum_e \Delta_e$.

Suppose edge e goes from level i to level $i+1$ of T^g, and denote by $e_j = v_j v_{j+1}$ the edges on the path from the root to e ($j = 0, \ldots, i$, $v_0 = r$ and $e_i = e$). Further, let T_j be the subtree of T^g whose root is v_j, and which does not include e_j (Figure 1). Let f_j be the total flow from subtree T_j to the root. Then we have

$$\Delta_e = \sum_{j=0}^{i} \frac{x_e f_j}{x_{e_{j-1}}},$$

where we define $x_{e_{-1}} = 1$.

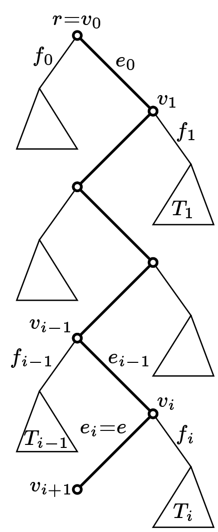

Figure 1.

Since the capacities on these edges are a result of rounding down to powers of 2, it follows that $f_j \leq 2x_{e_{j-1}}$. (Indeed, assume $f_j > 2x_{e_{j-1}}$. In the rounding, $x_{e_{j-1}}$ was at most halved, and so before the rounding it must have been true that $f_j > x_{e_{j-1}}$. But this would contradict the flow-conservation constraints that were satisfied before the rounding). So,

$$\Delta_e \leq x_e \sum_{j=0}^{i} 2 = 2(i+1)x_e.$$

Therefore,

$$\Delta = \sum_e \Delta_e \leq \sum_e 2(d+2)x_e = \frac{1}{2}(d+2) \leq \log_2 N_g.$$

Now we can apply Janson's equality with $\mu = 1/4$, $\Delta = \log N_g$ and $\epsilon = 1/2$. We get

$$\Pr\left[\text{fail to reach } g\right] \leq e^{-\frac{1}{64\Delta}} = e^{-\frac{1}{64 \log_2 N_g}} = 1 - \frac{1}{64 \log_2 N_g} + \frac{1}{4096 \log_2^2 N_g} - \cdots,$$

and we see that we will reach group g with probability of about $1/(64 \log_2 N_g)$. Since $N_g \leq N$, the maximum size of any group, the theorem follows. $\square$

Our analysis in the above theorem is tight up to a constant factor as can be seen by considering T^g to be the complete binary tree where all capacities in a level are equal and where the capacities decrease by a factor of two as we go down the tree. The success probability p_d (when this tree is of depth d) satisfies the recurrence relation $p_d = p_{d-1}(1 - p_{d-1}/4)$, and $p_0 = 1$. It can be shown that $\lim_d dp_d/4 = 1$. Thus the probability of success is $\Theta(1/\log n)$ where n is the number of leaves.

4 Building the Steiner tree.

Now we show how to use the result of the previous section and amplify the probability of success, while keeping the final cost low.

When we pick a single tree randomly, the probability that it covers g is at least $1/(64 \log_2 N)$ for any group g. If we pick $64 \log_2 N$ trees, their union will cover g with a probability of at least $1-1/e$. If we pick $128 \log_2 N \log 2k$ trees, the probability of missing a given group is at most $1/4k$, and by subadditivity, the probability of missing any group is at most $1/4$. So, if we pick $128 \log_2 N \log 2k$ trees, their union will cover all groups with probability at least $3/4$.

The total cost of the union of these trees is at most the sum of their costs. Denote this by $c(\mathcal{T})$. Recall that z^* denotes the cost of the optimal solution to the LP (1). Then by Lemma 3.1 and Markov's inequality,

$$\Pr\left[c(\mathcal{T}) \geq 4 \cdot 128 \cdot \log N \log 2k \; z^*\right] \leq 1/4.$$

Thus the tree $\mathcal{T}$ has low cost with probability at least $3/4$.

Since the two "good" events each occupy at least three quarters of the probability space, they must overlap in at least one half, and so with probability at least $1/2$, we cover all groups with a tree of cost $O(\log N \log k \; z^*)$. Since z^* is a lower bound on the cost of an optimal group Steiner tree, we obtain the following theorem.

Theorem 4.1. *There is a randomized polynomial time algorithm that, with probability at least* $1/2$*, finds a group Steiner tree on an underlying graph which is a tree, of cost no more than* $O(\log N \log k)$ *times the optimum, where* N *is the maximum size of a group and* k *is the number of groups.*

5 General graphs.

We need a few more definitions before explaining how to extend the above to the general case.

Definition 5.1. *A set of metric spaces* $\mathcal{S}$ *over* V *is said to* α*-probabilistically approximate a metric space* M *over* V*, if* **(1)** *for all* $x, y \in V$ *and* $S \in \mathcal{S}$, $d_S(x,y) \geq d_M(x,y)$*, and* **(2)** *there exists a probability distribution* D *over metric spaces in* $\mathcal{S}$ *such that for all* $x, y \in V$, $\mathbf{E}[d_D(x,y)] \leq \alpha d_M(x,y)$.

Bartal [4, 5] proved the following theorem.

Theorem 5.2. *Every weighted connected graph* G *on* n *vertices can be* α*-probabilistically approximated by a set of weighted trees, where* $\alpha = O(\log n \log\log n)$*. Moreover, the probability distribution can be computed in polynomial time.*

Theorem 5.3. *For any* $\epsilon > 0$*, there is a polynomial-time algorithm that with probability* $1-\epsilon$ *finds a group Steiner tree whose cost is* $O(\log N \log n \log\log n \log k)$ *times the cost of the optimal tree.*

Proof. We first state the algorithm.
(1) Randomly choose a tree T from Bartal's distribution.
(2) Solve the linear program (1) on T; let x be the optimal solution found.
(3) Run the rounding procedure described in Section 3 independently $128 \log \frac{1}{\epsilon} \log n \log 2k$ times and let F be the union of trees found. Return F.

We show that the expected cost of the optimal group Steiner tree in a tree chosen at random from Bartal's distribution is $O(\log n \log \log n)$ times the cost of the optimal group Steiner tree in the original graph G. Then our claim follows from Theorem 4.1, since by running the tree-rounding algorithm $\log 1/\epsilon$ times we can boost the success probability to $1 - \epsilon$.

Consider a tree T chosen from Bartal's distribution. Replace every edge ij of the optimal group Steiner tree H in G by the (unique) ij-path in T. This produces a group Steiner tree in T whose cost is no more than the sum of the costs of the paths in T. By Theorem 5.2 the expected cost of the ij-path in T is $O(\log n \log \log n)$ times c_{ij}. Thus an optimal group Steiner tree in a tree chosen from Bartal's distribution has expected cost $O(\log n \log \log n)$ times the cost of the optimal group Steiner tree in G. $\square$

6 Integrality gaps and tree-decompositions

The following is directly implied by Theorem 4.1.

Corollary 6.1. *The integrality gap of LP (1) is $O(\log n \log k)$ when the underlying graph is a tree.*

However, a similar result follows for LP (1) in the case of general graphs as well.

Corollary 6.2. *The integrality gap of LP (1) is $O(\log n \log \log n \log N \log k)$ on an n-vertex graph G, when there are k groups each of size at most N.*

Proof. Let z^* be the cost of the optimal solution of LP (1), and T a tree found by Bartal's algorithm. Let z^*_T be the cost of the optimal solution to LP (1) on the tree T. By Bartal's theorem, with constant probability $z^*_T \leq 2z^* \log n \log \log n$. (Let x^* be the optimal solution of LP (1). For every edge $e = ij \in T$ with $x^*_e > 0$, define $w_e = x^*_e$. Define $w_e = 0$ for all other $e \in T$. Then consider in turn every edge e such that $x^*_e > 0$ and $e \notin T$. For every edge f in the path corresponding to e in T, add x^*_e to w_f. By Bartal's theorem, the expected cost of w is $O(\log n \log \log n)z^*$, and by construction w is a fractional group Steiner tree in T.) By Theorem 3.4, the weight of the group Steiner tree output by our algorithm is with constant probability $O(z^*_T \log N \log k)$. $\square$

In the remainder of this section we discuss the idea of decomposing a solution of LP (1) into subtrees. We first define what we mean by a decomposition. Let x be a (fractional) solution to LP (1), and G its support graph. We use the notation $T \sim g$ to denote that a graph T contains a vertex of group g. Similarly we use $T \not\sim g$ to denote that a graph T contains no vertex of group g. The set of subtrees $T_1, T_2, \ldots, T_\ell$ of G, together with weights $\lambda_1, \lambda_2, \ldots, \lambda_\ell$ forms an *α-decomposition* of the solution x, if

$$\sum_{T_i \sim g} \lambda_i \geq 1 \text{ for all groups } g \tag{2}$$

$$\sum_{T_i \ni e} \lambda_i \leq \alpha x_e \text{ for all edges } e \tag{3}$$

$$r \in T_i \text{ for all } i. \tag{4}$$

For instance, suppose $\alpha = 1$ (that is, suppose we can always find a "perfect" packing that covers each group fractionally in the sense of condition (2) above). Then we could think of the subtrees $T_1, \ldots, T_\ell$ as sets in a set cover problem. In this case we could apply the randomized rounding procedure for set cover: independently of all other trees, include edges of T_i with probability λ_i. Taking the union of $O(\log k)$ independent experiments would with probability at least 3/4 give a group Steiner tree. Also with probability at least 3/4, the cost of this tree is bounded by $z \log k$ where z is the cost of the solution x.

More generally, an α-decomposition into trees of any solution of LP (1) gives an algorithm for the group Steiner problem with an approximation ratio of $O(\alpha \log k)$. Our algorithm provides an upper bound on the parameter α.

Theorem 6.3. *Let N denote the size of the largest group. For any solution of LP (1) whose support graph is a tree T, there is an α-decomposition, where $\alpha = O(\log N)$.*

Proof. Let $T_1, \ldots, T_\ell$ be all the possible outcomes of a rounding step from Section 3, and let $p_1, \ldots, p_\ell$ be their respective probabilities. Let Y_e be the indicator variable for picking the edge e in a rounding step. Then for every edge e,

$$\sum_{T_i \ni e} p_i = \mathbf{E} Y_e = x_e.$$

Let N_g denote the size of the group g. By Theorem 3.4,

$$\sum_{T_i \ni g} p_i \geq \frac{c}{\log N_g}.$$

Defining $\lambda_i = p_i \cdot \max_g (\log N_g / c)$, we get a decomposition with $\alpha = (\log N)/c$. ☐

Similarly as above, in the case of general graphs the factor is multiplied by $\log n \log \log n$.

It is not possible to always have $\alpha = 1$, even in the case of trees, as shown by the example in Figure 2, where $\alpha = 6/5$, and no better α is possible for this solution.

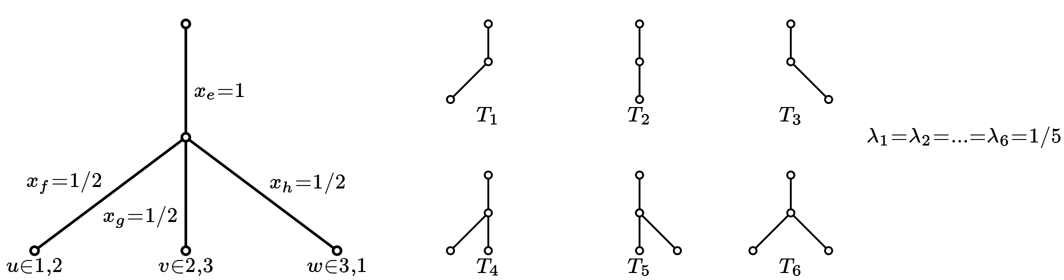

Figure 2.

However, a perfect decomposition always exists when there are only two groups.

Theorem 6.4. *Let x be a solution of LP (1) in an instance of the group Steiner problem on a tree T where there are only two groups denoted by g_1 and g_2. Then there exists a perfect decomposition of x. Moreover, this decomposition can be found in polynomial time.*

Proof. We assume as before that all vertices belonging to any group are leaves of T and that T is the support graph of x. We also assume that the groups g_1 and g_2 are disjoint and that capacities of edges on any path down the tree form a non increasing sequence. First define $y = x$. We will reduce y in each step until finally $y = 0$ to get the decomposition of x. To construct the decomposition, while neither of the two groups are empty, find a pair (v, w) of leaves such that $v \in g_1$ and $w \in g_2$. This pair should be chosen so that the intersection of the $r - v$ path and $r - w$ path is maximal. Denote by u the least common ancestor (lca) of v and w. Let a be the capacity of the leaf edge incident on v and b the capacity of the leaf edge incident on w. Let $c = \min\{a, b\}$. Define T_i to be the union of the three paths $r - u$, $u - v$ and $u - w$, and $\lambda_i = c$. Then add the tree T_i to the decomposition, and reduce y by subtracting the incidence vector of T_i multiplied by λ_i. Update T by deleting edges whose y-values have been reduced to zero. This procedure is repeated until the tree T is empty.

We claim that after reducing y in step i, the flow to each of the two groups reduces by at most λ_i. Indeed, suppose that the flow to g_1 reduces by more than λ_i. Reducing the edge-values along the path from r to v by λ_i only reduces the flow to g_1 by λ_i. Therefore, any further decrease in the flow to g_1 must come from the reduction of an edge outside the $r - v$ path. The only such edges belong to the path from u to w. By the choice of v and w, no vertex of g_1 can share a longer path with w than v and thus no vertex of g_1 receives flow along any portion of the path from u to w.

Condition (2) can be established as follows:

$$\sum_{T_i \sim g} \lambda_i = \sum_{v \in g \cap T_i} \sum_{T_i \ni v} \lambda_i = \sum_{v \in g \cap T} x_v = 1.$$

The first equality follows because in each step we only take out at most one vertex of the group g, and the second because the flow from r to g in the reduced graph decreases by λ_i in step i. The final equality holds because x is a feasible solution to LP (1).

Let I_1^e be the set of all leaves $v \in g_1$ such that e is one of the edges in the unique $r - v$ path in T. Let x_v denote the value of the flow from r to v in x.

Now condition (3) follows as well:

$$\sum_{T_i \ni e} \lambda_i = \sum_{v \in I_1^e} \sum_{T_i \ni v} \lambda_i = \sum_{v \in I_1^e} x_v \leq x_e.$$

The first equality follows since every iteration i reduces the flow to g_1 by exactly λ_i, and the second because every leaf is eventually removed. The final inequality is true because x supports a flow of x_v to v. □

7 Other formulations and applications.

In this section, we first sketch the improvement in the case of graphs that exclude small minors. Then we give some more applications of our results. One is to a bicriteria network design problem that involves location-based constraints, and the other to some generalizations of the traveling salesman problem.

7.1 Improved metric approximations.

The following improvement of Bartal's result to graphs that exclude small minors is presented by Konjevod, Ravi and Salman [21].

Theorem 7.1. *Let G be an n-vertex graph that excludes $K_{s,s}$ as a minor. Then G can be α-probabilistically approximated by a set of weighted trees, where $\alpha = O(s^3 \log n)$. Moreover, the probability distribution can be computed in polynomial time.*

This improved result (for constant s) applies, e.g., to planar graphs, which exclude $K_{3,3}$ as a minor. This theorem, together with the arguments from the previous section, then gives an improved approximation ratio of $O(\log n \log N \log k)$ for such graphs.

Since distances in the Euclidean plane can be approximated to within a factor of 2 by a planar graph [8], the improvements also apply to this case. More formally, if the edge lengths of the resulting planar graph can be assumed to be integers in a polynomial range, then we can probabilistically approximate the original distances by trees with only a logarithmic loss. By identifying some points we can assume the distances to be in $\{1, \ldots, O(n^2)\}$. This can be done so that the optimum value of a group Steiner tree only changes by a factor of $1 + \epsilon$ for any constant ϵ as in [2].

7.2 Service-constrained network design problems.

Marathe, Ravi and Sundaram [22, 23] study the following problem: an instance is given by an undirected graph $G = (V, E)$ with two different (nonnegative) cost functions on the edges, c (modeling the service cost) and d (modeling the construction cost), and a nonnegative function s on the vertices (defining the service-radius constraints). The goal is to find a minimum d-cost tree such that every vertex v in the graph is *serviced* by some vertex in the tree, i.e. every vertex v is within distance s_v (under the c-costs) of some vertex in the tree.

An (α, β)-bicriteria approximation for such a problem is an algorithm which finds a solution in which the service constraints are not exceeded by more than a factor of α, and whose cost under d is within a β factor of the optimal one that satisfies the service constraints (under the c-costs). Marathe et al. [22, 23] give a $(1, 2\Delta)$-approximation algorithm, where Δ is the maximum *service degree*, the maximum number of vertices that can service any given vertex.

We observe that if the first approximation factor α is fixed at 1, this problem is equivalent to the group Steiner tree problem.

First we reduce the service-constrained problem to a group Steiner tree problem. We define a set of groups $\{g_v \mid v \in V\}$. Let g_v be the set of vertices w that are within the budget (c-)distance of v,

$$g_v = \{w \in V \mid c(vw) \leq s_v\}.$$

Now any group Steiner tree will satisfy the service constraints, and conversely, any tree that services all vertices within the budget (i. e. such that $\alpha = 1$) will be a group Steiner tree.

Note that our algorithm improves the approximation factor of [23] to $(1, O(\log^2 n \log\log n \log k))$, where $n = |V|$ and k is the maximum service-degree of any vertex (in particular, $k \leq n$).

Next we reduce the group Steiner problem to a version of the service-constrained network design problem. Assume without loss of generality that the groups are disjoint. Let the weights in the given graph represent the d-cost. Define the c-cost as follows: between a pair of vertices in the same group, the c-cost is zero, and between all other pairs, the c-cost is one. The service radius s_v is set to zero for every vertex which is contained in some group and to n for all other vertices. Any solution output by an (α, β)-approximation algorithm for this service-constrained network design problem for any α includes at least one vertex from every group, and is therefore a group Steiner tree of cost at most β times the minimum.

7.3 Generalized traveling salesman problem.

The generalized traveling salesman problem is defined on a graph $G = (V, E)$ with a nonnegative cost function on the edges. Also given are subsets of vertices $S_1, \dots S_m \subseteq V$. These sets are called *clusters*. The goal is to find a minimum-cost cycle containing at least one vertex from each cluster. Unlike the group Steiner problem, it may be assumed that every vertex of the graph belongs to one of the clusters.

The generalized traveling salesman problem was first described by Henry-Labordere [13] and Saksena [28]. More information and further references can be found in the paper by Fischetti, Salazar and Toth [10].

We assume that the cost function satisfies the triangle inequality.

Before our results, the only approximation algorithm for the generalized traveling salesman problem was due to Slavík [29]. The approximation guarantee is $3\rho/2$, where ρ denotes the maximum size of a group.

We note that an ρ-approximation algorithm for the group Steiner problem implies a 1.5ρ-approximation algorithm for the generalized traveling salesman problem. The graphs in both instances are identical, and the groups are defined to be exactly the clusters. Then any group Steiner tree can be shortcut into a generalized traveling salesman tour of at most 1.5 times its cost, using the classical Christofides heuristic. On the other hand, any feasible tour induces a group Steiner tree of no greater cost.

Thus, our algorithm gives an $O(\log^2 n \log\log n \log k)$ approximation for the generalized traveling salesman problem.

7.4 Traveling purchaser problem.

Ravi and Salman [25] study the *traveling purchaser problem* and give approximation algorithms for this problem and a bicriteria version; Their approximation guarantees are similar to ours. Let $G = (V, E)$ be a graph and $c : E \to \mathbb{R}_+$ a weight function. Let a set of *products* P be given together with nonnegative prices d_{pv} for every $p \in P$ and $v \in V$. We think of V as the set of *markets*. A feasible solution of the traveling purchaser problem is a tour on some of the vertices in V. If we visit a vertex $v \in V$ then we can buy any product at the price it is offered at v. Every product must be bought at some market that the tour visits. The objective is to minimize the sum of the length of the tour and all the prices paid for products in P.

This problem generalizes the group Steiner problem since we may think of a group as a product and assign cost 0 for a group g at a vertex v if and only if $v \in g$. If v is not a vertex of g, then the cost is defined to be ∞. Now a tour of total cost z corresponds to a group Steiner tour of equal cost, and vice versa.

However, we can also reduce the traveling purchaser problem to the group Steiner problem. We show that a ρ-approximation algorithm for the group Steiner problem implies a 1.5ρ-approximation algorithm for the metric traveling purchaser problem. Starting with an instance of the traveling purchaser problem, define an extra vertex w_{pv} for each pair $(p, v) \in P \times V$. Add an edge of weight $d_{pv}/2$ from w_{pv} to v. Then define a group corresponding to each product to consist of exactly the new vertices added for this product. A group Steiner tree may be shortcut to a tour visiting each group, increasing the cost by at most a factor of 1.5. The cost of the edges entering and leaving group vertices is then exactly equal to the cost of buying each product at the corresponding market. So our group Steiner algorithm gives an algorithm with similar performance guarantee for the traveling purchaser problem.

7.5 Group prize-collecting Steiner problem.

The group prize-collecting Steiner problem was defined by David Johnson. Like in the group Steiner problem, a weighted graph is given together with a family of subsets, called groups. With each group, we associate a penalty. The goal is to find a connected subgraph that minimizes the sum of the costs of the edges used by the subgraph and the sum of penalties incurred for groups not visited in the subgraph. The reduction to group Steiner is similar to the one from the traveling purchaser problem. For every group, we add a new vertex and connect it to all other vertices of the graph with edges of cost equal to this group's penalty. The group Steiner problem on this new graph is equivalent to the original prize-collecting problem.

8 Conclusion.

We have presented the first algorithm with a polylogarithmic approximation ratio for the group Steiner problem. After reading a preliminary version of our paper, Charikar, Chekuri, Goel and Guha [7] derandomized our algorithm. They gave a general way to use Bartal's theorems in a deterministic setting and, independently of this result, a deterministic version of our tree-rounding procedure.

We leave open several problems. The only known lower bounds for the group Steiner problem are the ones that arise from the hardness of the set cover problem. It is a natural open problem then, to reduce the approximation ratio to $O(\log n)$. This would require avoiding Bartal's construction and solving the problem on the given general graph. A related question is to find a combinatorial algorithm for the group Steiner problem with a (poly)logarithmic approximation guarantee (even on a tree).

Another interesting question is whether better (say, constant ratio) approximations are possible for the Euclidean case. Finally, showing constructively the existence of a better decomposition would yield a simpler algorithm and establish a more direct connection to the set cover problem.

Acknowledgment. We would like to thank Santosh Vempala for a stimulating discussion on decompositions and splitting-off during our early attempts to find an approximation algorithm for the group Steiner problem.

References

[1] N. Alon and J. Spencer. *The Probabilistic Method.* Wiley Interscience, New York, 1992.

[2] S. Arora. Polynomial time approximation schemes for Euclidean TSP and other geometric problems. In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, pages 2–11, 1996.

[3] S. Arora and M. Sudan. Improved low degree testing and its applications. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 485–495, May 1997.

[4] Y. Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, pages 184–193, October 1996.

[5] Y. Bartal. On approximating arbitrary metrics by tree metrics. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 161–168, 1998.

[6] C. D. Bateman, C. S. Helvig, G. Robins, and A. Zelikovsky. Provably good routing tree construction with multi-port terminals. In *Proceedings of ACM/SIGDA International Symposium on Physical Design*, Apr. 1997.

[7] M. Charikar, C. Chekuri, T. Cheung, Z. Dai, A. Goel, S. Guha, and M. Li. Approximation algorithms for directed Steiner problems. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 192–200, 1998.

[8] L. P. Chew. There are planar graphs almost as good as the complete graph. *Journal of Computer and System Sciences*, 39(2):205–219, Oct. 1989.

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

[10] M. Fischetti, J. J. Salazar González, and P. Toth. A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research*, 45:378–394, 1997.

[11] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization.* Springer, 1988.

[12] S. Guha and S. Khuller. Approximation algorithms for connected dominating sets. *Algorithmica*, 20:374–387, 1999.

[13] A. L. Henry-Labordere. The record balancing problem: a dynamic programming solution of a generalized traveling salesman problem. *RIRO*, B-2:43–49, 1969.

[14] E. Ihler. Bounds on the quality of approximate solutions to the group Steiner problem. In *Graph-Theoretic Concepts in Computer Science WG90*, volume 484 of *Lecture Notes in Computer Science*, pages 109–118. Springer, 1991.

[15] E. Ihler. The complexity of approximating the class Steiner tree problem. In *Graph-Theoretic Concepts in Computer Science WG91*, volume 570 of *Lecture Notes in Computer Science*, pages 85–96. Springer, 1992.

[16] E. Ihler. The rectilinear class Steiner tree problem for intervals on two parallel lines. *Math. Programming, Ser. B*, 63:281–296, 1994.

[17] E. Ihler, G. Reich, and P. Widmayer. On shortest networks for classes of points in the plane. In *Computational Geometry—Methods, Algorithms and Applications CG'91*, volume 553 of *Lecture Notes in Computer Science*, pages 103–111. Springer, 1992.

[18] E. Ihler, G. Reich, and P. Widmayer. Class Steiner trees and VLSI design. *Discrete Appl. Math.*, 90:173–194, 1999.

[19] S. Janson. Poisson approximations for large deviations. *Randoms Structures and Applications*, 1:221–230, 1990.

[20] P. N. Klein and R. Ravi. A nearly best-possible approximation algorithm for node-weighted Steiner tree. *J. Algorithms*, 19:104–115, 1995.

[21] G. Konjevod, R. Ravi, and F. S. Salman. On approximating planar metrics by tree metrics. Manuscript, Jul. 1997.

[22] M. V. Marathe, R. Ravi, and R. Sundaram. Service-constrained network design problems. *Nordic J. Comput.*, 3:367–387, 1996.

[23] M. V. Marathe, R. Ravi, and R. Sundaram. Improved results on service-constrained network design problems. In D. Z. Du and P. M. Pardalos, editors, *Proceedings of a DIMACS workshop on Network Design: Connectivity and Facilities Location*, volume 40 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 269–276, 1998.

[24] P. Raghavan and C. Thompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7:365–374, 1987.

[25] R. Ravi and F. S. Salman. On the traveling purchaser problem. In *Proceedings of ESA '99*, Lecture Notes in Computer Science Vol. 1643, pages 29–40, 1999.

[26] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability pcp characterization of np. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 475–484, May 1997.

[27] G. Reich and P. Widmayer. Beyond Steiner's problem: A VLSI oriented generalization. In *Graph-Theoretic Concepts in Computer Science WG89*, volume 411 of *Lecture Notes in Computer Science*, pages 196–210. Springer, 1990.

[28] J. P. Saksena. Mathematical model of scheduling clients through welfare agencies. *CORS J.*, 8:185–200, 1970.

[29] P. Slavík. The errand scheduling problem. Technical Report 97-02, Department of Computer Science, SUNY, Buffalo NY, 1997.

[30] P. Winter. Steiner problem in networks: A survey. *Networks*, 17:129–167, 1987.