Improved Approximation Bounds for the Group Steiner Problem*

C. S. Helvig Gabriel Robins Alexander Zelikovsky†

Department of Computer Science, University of Virginia, Charlottesville, Virginia 22903-2442
†Department of Computer Science, UCLA, Los Angeles, California 90095-1596

Abstract

Given a weighted graph and a family of k disjoint groups of nodes, the Group Steiner Problem asks for a minimum-cost routing tree that contains at least one node from each group. We give polynomial-time $O(k^{\epsilon})$-approximation algorithms for arbitrarily small values of $\epsilon > 0$, improving on the previously known $O(k^{\frac{1}{2}})$-approximation. Our techniques also solve the graph Steiner arborescence problem with an $O(k^{\epsilon})$ approximation bound. These results are directly applicable to a practical problem in VLSI layout, namely the routing of nets with multi-port terminals. Our Java implementation is available on the Web.

1 Introduction

The classical Steiner problem can be formulated as follows: given an undirected weighted graph $G = (V, E)$ and $M \subseteq V$, find a minimum-cost tree that spans all of M. Nodes in $V - M$ (referred to as *Steiner* nodes) may be optionally included in order to reduce the total tree cost [11]. In this paper, we address a generalization of this problem, namely the Group Steiner Problem, which was first formulated in [15] (a comprehensive review was given in [7]). The problem is formalized as follows:

> **The Group Steiner Problem** [7, 15]: given an undirected weighted graph $G = (V, E)$ and a family $N = \{N_1, ..., N_k\}$ of k disjoint groups of nodes $N_i \subseteq V$, find a minimum-cost tree which contains at least one node from each group N_i.

As in the classical Steiner problem, we allow optional Steiner nodes in order to reduce the cost of the spanning tree interconnecting the groups (Figure 1(c)).

The Group Steiner Problem captures practical scenarios in VLSI layout design [2, 15]. For example, a circuit module may be rotated and flipped when positioned on a VLSI chip. This induces multiple potential connection points for the given circuit module, one for each of the eight possible orientations (Figure 1(a)). These locations correspond to a group of (up to eight) nodes in the Group Steiner Problem (Figure 1(b)). This formulation also captures the *pin assignment* problem in VLSI physical design [14] where the locations of interconnection points on module boundaries are determined. Finally, the Group Steiner Problem is applicable whenever multi-port terminals exist.

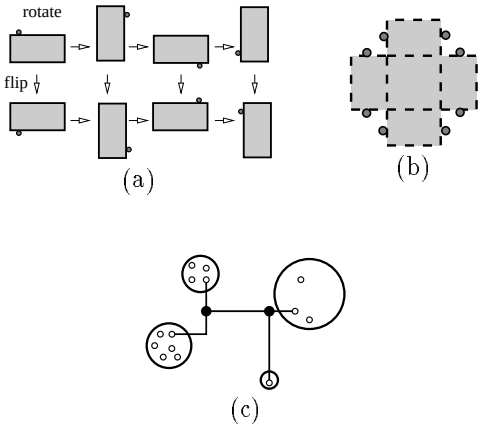

Figure 1: (a) A module is rotated and flipped to induce a group of up to eight virtual positions (b). (c) A group Steiner tree (solid dots represent Steiner nodes).

The first published approximation algorithm for the Group Steiner Problem produced solutions $O(k)$ times worse than optimal [7, 8, 15]. Recently, we improved this result by giving a heuristic with an approximation bound of $O(k^{\frac{1}{2}})$ times optimal [2]. The main result of this paper is a new series of heuristics with an improved performance ratio[1] of $O(k^{\epsilon})$ for arbitrarily small values of $\epsilon > 0$, where k is the number of groups. On the negative side, this problem cannot be efficiently approximated with a performance ratio of less than $\ln k$ times the optimal [5] [9].

*This work was supported by a Packard Foundation Fellowship and by National Science Foundation Young Investigator Award MIP-9457412. The corresponding author is Professor Gabriel Robins, Department of Computer Science, University of Virginia, Charlottesville, Virginia 22903-2442, robins@cs.virginia.edu, http://www.cs.virginia.edu/~robins/

Appeared in Proc. Conf. on Design Automation and Test in Europe (DATE'98), Paris, February, 1998.

[1] The *performance ratio* is an upper bound on the ratio of a heuristic solution cost divided by the optimal solution cost.

The rest of the paper is organized as follows. Section 2 introduces *depth-d -bounded* Steiner trees and proves that they approximate the optimal group Steiner tree to within a factor of $2d \cdot \sqrt[d]{k}$. Section 3 presents our main heuristic for approximating optimal depth-d -bounded Steiner trees to within a factor of $(2 + \ln(2k))^{d-1}$. The overall performance ratio of our heuristic is the product of these two bounds, which yields our main result. Section 4 analyzes the time complexity and suggests practical enhancements. Section 5 describes how to further improve the performance ratio when groups of size one are present, and Section 6 extends our construction to minimize the radius as well as the tree cost. In Section 7, we generalize our results to *directed Steiner trees*. We discuss our experimental results in Section 8.

2 Depth-Bounded Steiner Trees

This section introduces Steiner depth-bounded[2] trees, with a two-fold motivation: (1) depth-d -bounded trees can be used to approximate optimal group Steiner trees to within a factor of $2d \cdot \sqrt[d]{k}$, and (2) optimal depth-bounded Steiner trees in turn can be approximated efficiently, as discussed in the next section. Our overall method is thus a composition of these two approximations, and the overall performance bound is therefore the product of the two corresponding bounds.

In general, the given weighted graph G may violate the triangle inequality, i.e., there may be edges (u, v) in G whose cost is greater than the cost of the minimum u-to-v path in G. Clearly, an optimal group Steiner tree will contain no such edges, since replacing such edges with the corresponding shortest paths will decrease the total tree cost. Therefore, without loss of generality, we replace G with its *metric closure*[3]. In order to further simplify our analysis, we also modify G as follows. For any node $v \in N_i$, we create a new node v' and a new zero-cost edge (v, v'); thus, we let v' take on the role of v. This transformation preserves the cost of Steiner trees, while allowing us to consider only Steiner trees in which every group node is a leaf.

We define *d-stars* to be rooted trees of depth at most d. The rest of this section will show that for any arbitrary (but henceforth fixed) tree T with *root* r, there exists a low-cost d-star spanning the leaves of T. This will imply that an optimal group Steiner tree can be approximated by a low-cost *group Steiner d-star* (defined as a Steiner d-star that spans all of the groups). It is known that even a 1-star (Figure 2(b)) provides an $O(k)$-approximation to optimal group Steiner trees [8]. Our overall strategy is to specify a low-cost d-star and derive upper bounds on its cost.

[2]We define the *depth* of a rooted tree T as the maximum number of edges in any root-to-leaf path.

[3]The metric closure is defined as the complete graph where the cost of each edge (u, v) is equal to the cost of the minimum u-to-v path in G.

We now construct a low-cost d-star S_d from the tree T with the same root and the same set of leaves L. Thus, to completely specify the d-star S_d, we select an appropriate set of intermediate nodes that lie on paths between the root r and the leaves. Note that because our approximate tree is a d-star, it may have at most $d-1$ levels of intermediate nodes. The leaves form the set of level-d nodes in S_d. Similarly, we refer to the lowest level of intermediate nodes as level-$(d-1)$ nodes and so on. Finally, the root is connected to the level-1 intermediate nodes.

We determine an appropriate set of intermediate nodes for a d-star S_d in T as follows. First, we sort the nodes of T in a depth-first manner (so that we may later extract ordered subsequences of nodes from this sorted list). Then, we partition the sorted subsequence of leaves into contiguous blocks of fixed size b (later we will determine an appropriate value for b). We select the level-$(d-1)$ nodes of T to be the set of least common ancestors[4] of the leaves in each of the blocks. Next, we partition these level-$(d-1)$ intermediate nodes into contiguous blocks of fixed size b. Thus, we obtain the level-$(d-2)$ intermediate nodes by selecting the least common ancestors of each of these blocks (Figure 3(a)).

We repeatedly apply this procedure to define each level of intermediate nodes until we reach the root. Thus, in our final tree S_d, each level-i node is connected to b level-$(i+1)$ nodes below and to one level-$(i-1)$ node above.

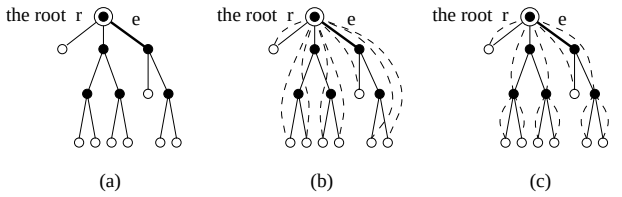

Figure 2: (a) A tree T rooted at r may have arbitrary depth. (b) A 1-star and (c) a 2-star are represented by dashed lines that connect the root r to all leaves. The edge e is (re)used three times by edges of the 1-star in (b) and twice by edges of the 2-star in (c).

In deriving upper-bounds on the cost of d-stars, we sum the costs of tree paths between nodes that are adjacent in d-stars. Since such paths are not necessarily disjoint, the same tree edge may be counted multiple times in this sum, a situation referred to as *edge reuse* (Figure 2). For the tree T, edge reuse provides an upper bound on the ratio $cost(\text{d-star})/cost(T)$: if no edge is used more than j times when replacing edges of a d-star by the corresponding paths in T, then the d-star has cost no more than j times the cost of the tree T. Our strategy for deriving upper bounds on the cost of S_d is to bound its edge reuse.

[4]We define node u to be an *ancestor* of v (i.e., v *descends* from u) if the path from the root to v passes through u.

Let $reuse_T(S_d)$ denote the maximum number of times that any tree edge is used in tree paths connecting nodes adjacent in S_d. We distinguish two types of paths that contribute to the edge reuse of the resulting d-star: (I) paths from the root to level-1 intermediate nodes (top part of Figure 3(a)), and (II) paths from level-i intermediate nodes to level-$(i+1)$ nodes, where $1 \leq i \leq d-1$ (Figure 3(b)). The number of paths of type (I) is bounded by the number of level-1 intermediate nodes. Note that each level contains a factor of b fewer intermediate nodes than the level below it. Thus, there are no more than $|L|/b^{d-1}$ level-1 nodes, where b is the block size (Figure 3(a)).

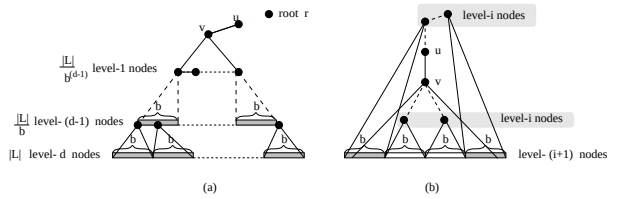

Figure 3: Two types of paths may reuse an edge (u, v): (a) Type-(I) paths terminate at level-1 intermediate nodes below v (there are no more than $|L|/b^{d-1}$ of these); and (b) Type-(II) paths terminate at intermediate nodes (or leaves) contained in only the leftmost and rightmost blocks. These two blocks together contain at most $2b$ level-$(i+1)$ nodes.

We now estimate the contribution of type-(II) paths to the reuse of an arbitrary (but henceforth fixed) edge (u, v) of the tree T with u being the parent of v. Let S be the depth-first -ordered sequence of level-$(i+1)$ intermediate nodes that descend from v. Note that S forms a contiguous subsequence in the sequence of all level-$(i+1)$ intermediate nodes. If a size-b block is completely contained in S, then its least common ancestor (which is a level-i intermediate node) necessarily descends from v (Figure 3). Therefore, the edge (u, v) does not lie on paths to level-$(i+1)$ nodes that belong to such completely-contained blocks, and no contribution to the reuse of the edge (u, v) occurs.

We are thus only concerned with blocks which may be not completely contained inside S, because type-(II) paths ending only in such blocks can contribute to the reuse of the edge (u, v). Since the nodes of S are sorted in depth-first order, the only blocks relevant to this analysis are the leftmost and rightmost blocks. The total contribution to the reuse of edge (u, v) due to the paths ending in the leftmost and rightmost blocks cannot exceed the total size of these two blocks, namely $2b$ (Figure 3).

The edge (u, v) may be used simultaneously in paths starting from different levels. Thus, we bound the total contribution of type-(II) paths for all levels by $2b \cdot (d-1)$, and the total edge reuse is at most $|L|/b^{d-1} + 2b \cdot (d-1)$. Choosing $b = \sqrt[d]{|L|/2}$ yields an upper bound on edge reuse of $2d \cdot \sqrt[d]{|L|/2}$. This proves that for any tree T, with the set of leaves L and root r, there is a d-star rooted at r with the same set of leaves L, having cost at most $2d \cdot \sqrt[d]{|L|/2} \cdot cost(T)$. Therefore, by approximating the optimal group Steiner tree (spanning k groups and thus having $|L| = k$ leaves) with an optimal Steiner d-star, we obtain the following result.

Theorem 1 *Let Opt be an optimal group Steiner tree over k groups, and let r be an arbitrary node of Opt. Then the cost of an optimal Steiner d-star rooted at r and spanning the groups is at most $2d \cdot \sqrt[d]{k/2} \cdot cost(Opt)$.*

3 The Main Heuristic

We have established that an optimal Steiner d-star is a reasonable approximation for an optimal group Steiner tree, (which will henceforth be denoted Opt). It can be shown that even the problem of approximating an optimal Steiner d-star is as difficult as approximating a minimum set cover. Therefore, it is unlikely that there exists a polynomial-time approximation algorithm with performance ratio $(1-\epsilon) \cdot \ln k$, for any $\epsilon > 0$, where k is the number of groups [5]. In this section, we will indirectly approximate Opt by approximating an optimal Steiner d-star. Proofs will be omitted in the rest of the paper due to the lack of space in this extended abstract.

In order to apply Theorem 1, we must first ensure that at least one node in the tree produced by our algorithm belongs to the optimal group Steiner tree Opt. Although we do not know which nodes are in Opt, this is not an obstacle. For each node r of an arbitrary fixed group (say N_1), we construct a low-cost Steiner d-star $Approx_d$ with root r. We then select the least-cost $Approx_d$ over all possible choices of r. Because Opt contains at least one node from each group, this guarantees (within polynomial time) that at least one of the $|N_1|$ trees thus constructed had the proper choice for the root. Therefore, without loss of generality, we may fix the root r, i.e., we consider the rooted version of the Group Steiner Problem.

Let $Opt_d(r)$ be the optimal Steiner d-star rooted at r, for any positive integer d. The main idea in constructing the approximate Steiner d-star $Approx_d$ is to successively refine an initial approximation coinciding with $Opt_1(r)$. There are two advantages to using $Opt_1(r)$ as an initial approximation for $Opt_d(r)$: first, unlike $Opt_d(r)$ for $d \geq 2$, the 1-star $Opt_1(r)$ can be computed efficiently; secondly, the cost of $Opt_1(r)$ is bounded by $k \cdot cost(Opt)$ (from Theorem 1, for $d = 1$). Therefore, to measure the approximation quality of a d-star, we will compare its cost to the cost of an optimal 1-star with the root r and with leaves taken from the same groups spanned by the d-star.

Let S be a d-star with a root $v \in V$ and let $groups(S)$ be the set of groups spanned by S. We denote by S' an optimal 1-star with the root r connected to $groups(S)$. We define the *norm* of S as $norm(S) = cost(S)/cost(S')$. Note that S and S' have different roots (Figure 4(a)).

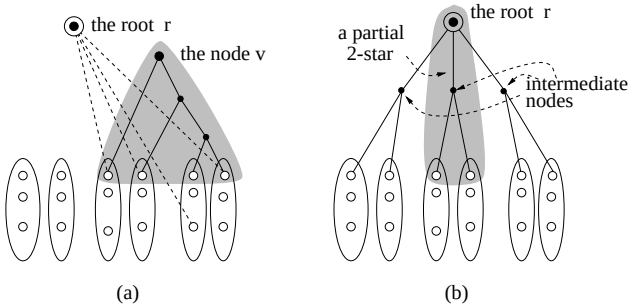

Figure 4: (a) A d-star S rooted at v (shaded area) and the corresponding optimal 1-star S' rooted at r (dashed lines). (b) A Steiner 2-star, with a (shaded) partial 2-star.

We represent our low-cost d-star $Approx_d$ as a union of subtrees, each consisting of the root, exactly one level-1 intermediate node, and all of the descendants of this intermediate node. Such rooted subtrees of depth d will be called *partial d-stars* (Figure 4(b)). We select partial d-stars for $Approx_d$ in the following greedy manner. First, we find a partial d-star P with an approximately minimum norm. Next, we remove the groups that are spanned by P (i.e., $groups(P)$) from further consideration. Finally, we determine the next partial d-star with an approximately minimum norm and iterate this process until all groups are spanned. Figure 5 describes the algorithm, and an execution example (for $d = 2$) is given in Figure 8.

Rooted Steiner d-Star Heuristic

Input: A graph $G = (V, E)$, a family N of k disjoint groups $N_1, \ldots, N_k \subseteq V$, a root $r \in V$, and a node $v \in V$

Output: A low-cost d-star $Approx_d$ rooted at node v and intersecting each group N_i

$Approx_d \leftarrow \{v\}$
$N' \leftarrow N$
While $N' \neq \emptyset$ **do**
  Find a low-norm (with respect to r) partial d-star $P = Partial_d(v, N')$
  $N' \leftarrow N' - groups(P)$
  $Approx_d \leftarrow Approx_d \cup P$
Output $Approx_d$

Figure 5: The greedy d-star heuristic for a given fixed root. At each iteration of the loop, we approximate the lowest-norm partial d-star, add it to the solution, and remove its groups from future consideration. The algorithm terminates when no groups remain to be spanned, at which point all groups are spanned in the solution.

In order to complete the description of our heuristic, we need to describe an efficient procedure that, given a root r and set of groups M, finds a low-norm partial d-star $Partial_d(r, M)$ rooted at r spanning some of the groups of M. We call this procedure the Partial d-Star Heuristic. First, we will describe how to find minimum-norm partial 2-stars, and then we will show how to approximate minimum-norm partial d-stars for $d \geq 3$.

Figure 6 describes a procedure for finding minimum-norm partial 2-stars. Note that in this procedure we use $cost(v, N_i)$ to denote the cost of the shortest edge between v and any node in group N_i.

Partial d-Star Heuristic (for $d = 2$)

Input: A graph $G = (V, E)$, a family $M \subseteq N$ of disjoint groups and root $r \in V$

Output: The minimum-norm partial 2-star $P(r, M)$ rooted at r with leaves from some groups of M

For each $v \in V$ **do**
  Sort $M = \{N_1, ..., N_{|M|}\}$ such that:
  $\frac{cost(v,N_i)}{cost(r,N_i)} \leq \frac{cost(v,N_{i+1})}{cost(r,N_{i+1})}$
  Find $j \in \{1, ..., |M|\}$ that minimizes
  $norm(v) \leftarrow \frac{cost(r,v)+\sum_{i=1}^{j} cost(v,N_i)}{\sum_{i=1}^{j} cost(r,N_i)}$
  $M(v) \leftarrow \{N_1, ..., N_j\}$
Find v having minimum $norm(v)$
Output the partial 2-star $P(r, M)$ with the intermediate node v, root r, and groups $M(v)$

Figure 6: Algorithm for finding a minimum-norm partial 2-star (i.e., d-star for $d = 2$). For each candidate v for the intermediate node, we sort groups N_i according to the potential improvement of inserting node v between the root r and each group. Then, we include consecutive groups from the list as long as their inclusion decreases the norm of the partial 2-star.

Although we can prove that minimum-norm partial 2-stars can be found efficiently, it can be shown that finding minimum-norm partial d-stars becomes NP-hard for $d \geq 3$. Thus, we now describe our heuristic for finding low-norm partial d-stars.

Our Partial d-Star Heuristic finds a low-norm partial d-star $Partial_d = Partial_d(r, M)$ with a given root r spanning elements from some groups of a given subfamily $M \subseteq N$, where N is the family of all groups (Figure 7 gives a formal description). The Partial d-Star Heuristic is recursive: in order to find a low-norm partial d-star, we need to first obtain low-norm partial $(d-1)$-stars.

The following Lemma gives the performance ratio for the Rooted Steiner d-Star Heuristic described in Figure 5.

Lemma 2 *Let $Opt_d(v)$ be an optimal Steiner d-star rooted at v, and let $Opt_1(r)$ be the optimal Steiner 1-star rooted at r. The cost of the tree produced by the Rooted Steiner d-Star Heuristic is at most:*

$$\left[2 + \ln \frac{cost(Opt_1(r))}{cost(Opt_d(v))}\right]^{d-1} \cdot cost(Opt_d(v))$$

Partial d-Star Heuristic (for $d > 2$)

Input: A graph $G = (V, E)$, a family $M \subseteq N$ of disjoint groups and root $r \in V$

Output: A low-norm partial d-star $Partial_d(r, M)$ with root r and leaves from some groups of M

For each $v \in V$ **do**
  $Partial_d \leftarrow (r, v)$
  $norm(Partial_d) \leftarrow \infty$
  $M' \leftarrow M$
  While $M' \neq \emptyset$ **do**
    Use the Partial d-Star Heuristic for depth $d-1$:
      $Partial_{d-1} \leftarrow Partial_{d-1}(v, M')$
    If $norm(Partial_d) \leq norm(Partial_{d-1})$ **then**
      Exit while
    $Partial_d \leftarrow Partial_d \cup Partial_{d-1}$
    $M' \leftarrow M' - groups(Partial_{d-1})$
  $norm(v) \leftarrow norm(Partial_d)$
Find v having minimum $norm(v)$
Output the partial d-star $Partial_d(r, M)$
  with intermediate node v

Figure 7: Our heuristic for finding a low-norm partial d-star for $d \geq 3$.

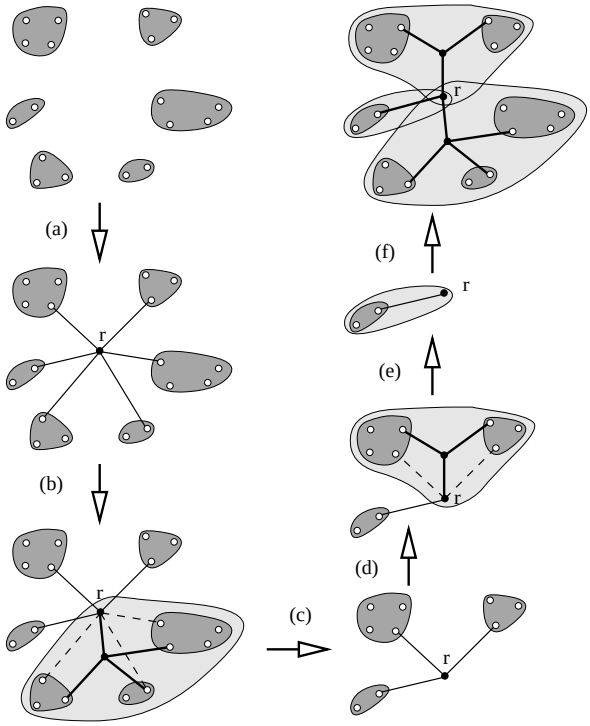

Figure 8: Given an instance of the Group Steiner Problem, for each possible root r, our heuristic (a) finds the optimal 1-star, (b) finds a low-norm partial d-star (shaded region), (c) saves this star and removes its groups from future consideration, (d) finds the next low-norm partial d-star (shaded region), (e) repeats step (c) for all partial d-stars, and finally (f) finds the last low-norm partial d-star and outputs the union of all saved partial d-stars.

Together with Theorem 1, Lemma 2 implies our main result:

Theorem 3 *The Rooted Steiner d-star Heuristic (Figure 5) solves the Group Steiner Problem with performance ratio* $2d \cdot (2 + \ln(2k))^{d-1} \cdot \sqrt[d]{k}$, *where* k *is the number of groups.*

Corollary 4 *The optimal group Steiner tree can be approximated in polynomial time with a performance ratio of* $O(k^{\epsilon})$ *for arbitrarily small* $\epsilon > 0$.

4 Runtime and Enhancements

In this section, we analyze the runtime of our heuristic and discuss several ways of improving its runtime and performance in practice. The time complexity for preprocessing is dominated by the time to compute all-pairs shortest paths in a graph (we denote this time complexity by τ). All node-to-group distances (i.e., distances between each node and the closest node to it in each group) can be computed in time less than τ.

The time complexity for a graph $G = (V, E)$ of the Partial d-Star Heuristic (Figure 6) is $O(|V|^{d-1} \cdot k^d)$, where k is the number of groups, and d is the tree depth bound. Therefore, the Rooted d-Star Heuristic (Figure 5) has runtime $O(|V|^{d-1} \cdot k^{d+1})$. To find a low-cost Steiner d-star, we run our Rooted d-Star Heuristic for all possible roots from the smallest group (i.e., size at most $|V|/k$). Therefore, an overall runtime of $O(\tau + (|V| \cdot k)^d)$ is sufficient to find a group Steiner tree with cost at most $2d \cdot (2 + \ln(2k))^{d-1} \cdot \sqrt[d]{k}$ times optimal.

The performance ratios derived in previous sections pertain to *worst-case* analysis. In practice, however, we are also interested in the average-case behavior of our heuristics, in terms of both runtime and solution quality. For example, one practical improvement entails omitting the removal of the set of groups spanned by the minimum-norm d-star (see the inner loop of the algorithm in Figure 5). Instead, every time we accept an intermediate node, we update the best possible current star by calculating the distance to a particular group, not from the root, but rather from the closest already-accepted intermediate node. This will tend to improve the solution cost in practice.

A practical enhancement for reducing time complexity entails computing a *group minimum spanning tree* instead of a group Steiner tree - i.e., a minimum spanning tree for a set of nodes containing at least one node from each group. It can be shown that the optimal group minimum spanning tree is at most twice longer than the optimal group Steiner tree [12, 17]. Thus, when approximating the group Steiner tree by a group minimum spanning tree, we lose at most a factor of 2, which does not asymptotically increase the overall bound, yet yields substantial savings in runtime.

We may further modify our algorithm with a post-processing step that finds the minimum spanning (or approximate Steiner) tree for the set of intermediate nodes chosen by the group Steiner heuristic. We may also make local (one at a time) node substitutions in groups to re-arrange the tree topology in order to reduce the overall cost.

Although provably-good heuristics are frequently outperformed by local optimization methods, the output of the former can serve as a good starting point for local-improvement post-processing schemes. For example, it was shown in [16] that Christofides' heuristic (i.e., the best-known heuristic for traveling salesperson in graphs [13]) also provides excellent initial traveling salesperson tours for further local rearrangements.

5 Cases with Degenerate Groups

We now show how to more effectively handle instances of the Group Steiner Problem with some *degenerate* groups, i.e. groups of size 1. We will see that treating degenerate groups differently yields improvements in solution quality as well as in runtime.

The degenerate groups by themselves induce an instance of the classical Steiner problem, and such an instance can be approximated efficiently by known methods (with a constant performance ratio). Thus, to solve the Steiner problem for degenerate groups, we may choose a provably-good heuristic from among the numerous existing ones [3, 6, 10, 12, 18]. For example, given a graph $G = (V, E)$, we may find in time $O(|V|^3)$ a Steiner tree which is at most $\frac{11}{6}$ times longer than the optimal [19]. The remaining issue now is to effectively combine the Steiner tree over the degenerate groups with a tree spanning the other, non-degenerate groups.

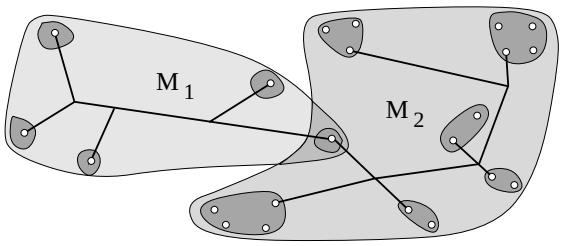

Figure 9: We span the set M_1 of degenerate groups with a Steiner tree $Approx_1$ (left). Then, we span all non-degenerate groups M_2 together with an arbitrary degenerate group using a group Steiner tree $Approx_2$ (right).

We define the *Combined Group Steiner Heuristic* as follows. Let $N = M_1 \cup M_2$ be the partition of all the groups in N into those containing one node (M_1), and those containing at least two nodes (M_2). First, we find the usual Steiner minimal tree $Approx_1$ for the degenerate groups M_1 using the approximation algorithm from [19]. Next, using our group Steiner heuristic, we find the group Steiner tree $Approx_2$ for the family of groups that includes the non-degenerate groups M_2 and an arbitrary degenerate group from M_1. Finally, we output the union $Approx_1 \cup Approx_2$ (Figure 9).

If the number of degenerate groups is large, the Combined Group Steiner Heuristic will enjoy considerable runtime savings, as well as an improved performance ratio that depends only on the non-degenerate groups:

Theorem 5 *The Combined Group Steiner Heuristic solves the Group Steiner Problem for $\bar{k}$ non-degenerate groups with a performance ratio of at most:*

$$\frac{11}{6} + 2d \cdot [2 + \ln(2) + \ln(\bar{k} + 1)]^{d-1} \cdot \sqrt[d]{\bar{k} + 1}$$

for any tree depth bound $d \geq 2$.

Corollary 6 *The optimal group Steiner tree with $\bar{k}$ non-degenerate groups can be approximated in polynomial time with a performance ratio of $O(\bar{k}^{\epsilon})$ for arbitrarily small $\epsilon > 0$.*

6 Bounding the Radius

In deep-submicron VLSI regimes, the objective of delay minimization often induces wiring geometries that are substantially different from those dictated by an optimal-area objective. This has motivated a number of bounded-radius[5] routing constructions [1, 4, 11]. Our basic group Steiner tree approach can be easily extended to a bounded-radius construction, thereby yielding routing trees with source-to-sink pathlengths bounded by a user-specified parameter.

For example, we can utilize the tree produced by our main algorithm as the starting point in the bounded-radius construction of [4]. For an arbitrary instance of the Group Steiner Problem (with k groups), this hybrid approach yields a routing tree with *simultaneous* cost/radius bounds: its radius is $(1 + \epsilon)$ times the optimal radius, and its total cost is $(1 + \frac{2}{\epsilon}) \cdot 2d \cdot (2 + \ln(2k))^{d-1} \cdot \sqrt[d]{k}$ times the optimal cost, for *any* user-specified parameter $\epsilon > 0$. This provides a provably-good smooth tradeoff between tree cost and tree radius for the Group Steiner Problem.

7 Steiner Arborescences

Our techniques can be adapted to address a generalization of the Group Steiner Problem, namely the Steiner problem in directed graphs (also known as the *Steiner Arborescence Problem*) [7]:

The Directed Steiner Problem [7]: given a directed weighted graph $G = (V, E)$, a set of k *terminals* $N \subseteq V$, and a *root* $r \in N$, find a minimum-cost subgraph T that contains directed paths from r to each terminal.

[5] The radius of a graph is defined as the maximum pathlength of any shortest source-to-sink path. Note that the radius of d-stars is implicitly bounded by d times the optimal radius.

Without loss of generality, we may assume that the subgraph T is a *directed tree*[6], otherwise T would contain arcs which can be removed without increasing the cost of T. As in the beginning of Section 2, we replace the directed graph G with its *transitive closure*[7], and we modify the graph so that all terminals become leaves.

We define *directed d-stars* to be directed trees of depth at most d. Again, as in Section 2, it can be shown that the minimum-cost directed d-star costs at most $2d \cdot \sqrt[d]{k/2}$ times the optimal directed Steiner tree. As in Section 3, we represent our low-cost directed d-star $Approx_d$ as a union of *directed partial d-stars*. Each such partial d-star consists of the root r, exactly one arc from the root to a level-1 intermediate node, and the directed subtree rooted at this intermediate node.

Given a directed d-star S, let S' be the corresponding directed 1-star (with the same terminals and root). The norm of S is defined to be $cost(S)/cost(S')$. Now we can use (an analog of) the Partial d-Star Heuristic for directed graphs in order to find a low-norm directed partial d-star. We approximate an optimal directed Steiner d-star by using (an analog of) the Rooted Steiner d-Star Heuristic for directed graphs (Section 3). Starting from an optimal 1-star, we update the current solution with low-norm partial d-stars found by the Partial d-Star Heuristic. The algorithm stops when all terminals are spanned with directed partial d-stars. The output of this algorithm is a d-star with the same cost bound as in Lemma 2. Finally, combining the analog of Lemma 2 with Theorem 1, we obtain the following result.

Theorem 7 *Optimal directed Steiner trees can be approximated in polynomial time with a performance ratio of $O(k^\epsilon)$ for arbitrarily small $\epsilon > 0$.*

8 Experimental Results

We have implemented our main heuristic for the Group Steiner Problem using the Java programming language. Our implementation is available on the Web at:

http://www.cs.virginia.edu/~robins/groupSteiner/

This choice of implementation enables the execution of our code directly off the Web using any Java-enabled browser. We have also implemented a variant of our basic heuristic where the solutions are post-processed with a minimum spanning tree algorithm (i.e., we output the minimum spanning tree over the nodes selected by our main heuristic).

[6]Each node of T is reachable from the root via a unique directed path. This means that replacing all directed edges in T by undirected edges would produce an undirected tree.

[7]The transitive clusure is an analog of metric closure: if a node v is reachable from a node u in G, then in the transitive closure of G there is an arc (u, v) whose cost is equal to the cost of the minimum directed u-to-v path.

We compared our heuristics with the heuristic proposed by Reich & Widmayer [15], denoted by RW. The RW heuristic begins by first finding the minimum spanning tree T for the entire set of nodes of all the groups. If a leaf node is not the last member of its group in the tree T, then it may be removed. The RW heuristic then repeatedly deletes such a leaf node incident to the longest edge among all such nodes.

Figure 10 compares two versions of our heuristic to the RW algorithm [15]. We generated instances of the Group Steiner Problem by uniformly and independently distributing the nodes of each group inside a randomly-placed square area of predetermined size, which we varied among 10%, 50%, and 100% of the overall region. The data represent the average relative improvement over 100 trials, given as a percentage of the RW algorithm tree cost. While our implementation uses the rectilinear metric to determine the distances between nodes, our algorithms are general and apply to arbitrary weighted graphs.

The first version of the group Steiner heuristic that we implemented has the following three modifications over the basic version described in previous sections: (1) Intermediate nodes are selected strictly from among the groups (i.e., all of the constructions benchmarked are *spanning* trees - they do not use Steiner nodes other than group nodes); (2) the root of the d-star is selected from a single randomly-chosen group; and (3) after accepting an intermediate node in the inner loop of Figure 6, it is removed from further consideration in subsequent iterations.

The second version that we implemented adds a minimum spanning tree post-processing step to the first version above (i.e., we output the minimum spanning tree over the nodes selected by our modified heuristic described above). In our implementations, we set $d = 2$. As can be seen from the chart (Figure 10), the second version significantly outperforms the RW algorithm, especially for larger group cardinalities and group areas.

9 Conclusion and Future Work

We have addressed a generalization of the classical Steiner problem, with the goal of finding a minimum-cost tree spanning k groups of nodes. Our main result is a series of polynomial-time approximation algorithms with performance ratio $O(k^\epsilon)$ for arbitrarily small $\epsilon > 0$. This result improves the previously known $O(k^{\frac{1}{2}})$-approximation and also applies to the Steiner problem in directed graphs. A Java implementation and benchmark results indicate that our approach is also effective in practice. Future work includes:

- Reducing the performance ratio even further (e.g., to a polylogarithmic function of the number of groups); and
- Improving the time complexity of the heuristics.

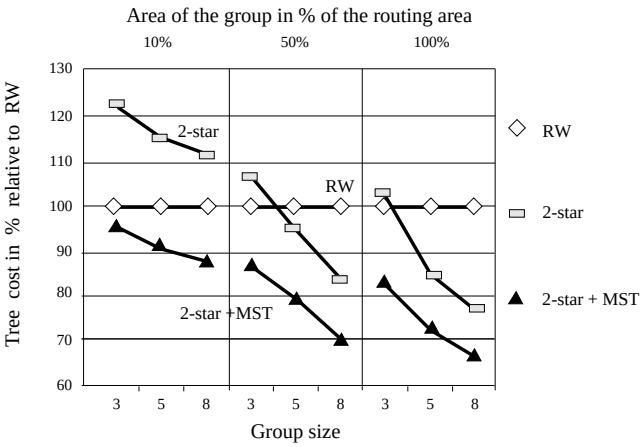

Figure 10: Average tree cost of the output of our main heuristic (rectangles) and our MST post-processing heuristic (triangles), normalized with respect to the RW heuristic [15] (diamonds), for various group sizes.

10 Acknowledgments

The authors are grateful to Andrew Kahng, Doug Bateman, Sarah Friend, John Viega, and Jeff Roach for their comments. Professor Robins was supported by a Packard Foundation Fellowship and by National Science Foundation Young Investigator Award MIP-9457412. Related papers may be found at http://www.cs.virginia.edu/~robins/

References

[1] C. J. Alpert, T. C. Hu, J. H. Huang, and A. B. Kahng. A direct combination of the prim and dijkstra constructions for improved performance-driven global routing. In *Proc. IEEE Intl. Symp. Circuits and Systems*, pages 1869–1872, Chicago, IL, May 1993.

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

[3] P. Berman and V. Ramaiyer. Improved approximations for the steiner tree problem. In *Proc. ACM/SIAM Symp. Discrete Algorithms*, pages 325–334, San Francisco, CA, January 1992.

[4] J. Cong, A. B. Kahng, G. Robins, M. Sarrafzadeh, and C. K. Wong. Provably good performance-driven global routing. *IEEE Trans. Computer-Aided Design*, 11(6):739–752, 1992.

[5] U. Feige. A threshold of ln n for approximating set cover. In *Proc. ACM Symp. the Theory of Computing*, pages 314–318, May 1996.

[6] J. Griffith, G. Robins, J. S. Salowe, and T. Zhang. Closing the gap: Near-optimal steiner trees in polynomial time. *IEEE Trans. Computer-Aided Design*, 13(11):1351–1365, November 1994.

[7] F. K. Hwang, D. S. Richards, and P. Winter. *The Steiner Tree Problem*. North-Holland, 1992.

[8] E. Ihler. Bounds on the quality of approximate solutions to the group steiner problem. In *Lecture Notes in Computer Science*, volume 484, pages 109–118, 1991.

[9] E. Ihler. The complexity of approximating the class steiner tree problem. Technical report, Institut für Informatik, Universität Freiburg, 1991.

[10] A. B. Kahng and G. Robins. A new class of iterative steiner tree heuristics with good performance. *IEEE Trans. Computer-Aided Design*, 11(7):893–902, July 1992.

[11] A. B. Kahng and G. Robins. *On Optimal Interconnections for VLSI*. Kluwer Academic Publishers, Boston, MA, 1995.

[12] L. Kou, G. Markowsky, and L. Berman. A fast algorithm for steiner trees. *Acta Informatica*, 15:141–145, 1981.

[13] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy, and D. B. Shmoys. *The Traveling Salesman Problem: a Guided Tour of Combinatorial Optimization*. John Wiley and Sons, Chichester, New York, 1985.

[14] B. T. Preas and M. J. Lorenzetti. *Physical Design Automation of VLSI Systems*. Benjamin/Cummings, Menlo Park, CA, 1988.

[15] G. Reich and P. Widmayer. Beyond steiner's problem: A vlsi oriented generalization. In *Lecture Notes in Computer Science*, volume 411, pages 196–211, 1989.

[16] G. Reinelt. *The Traveling Salesman: Computational Solutions for TSP Applications*. Springer Verlag, Berlin, Germany, 1994.

[17] H. Takahashi and A. Matsuyama. An approximate solution for the steiner problem in graphs. *Mathematica Japonica*, 24(6):573–577, 1980.

[18] A. Z. Zelikovsky. An 11/6 approximation algorithm for the network steiner problem. *Algorithmica*, 9:463–470, 1993.

[19] A. Z. Zelikovsky. A faster approximation algorithm for the steiner tree problem in graphs. *Information Processing Letters*, 46(2):79–83, May 1993.