Provably Good Routing Tree Construction with Multi-Port Terminals*

C. Douglass Bateman, C. S. Helvig, Gabriel Robins, and Alexander Zelikovsky

Department of Computer Science, University of Virginia, Charlottesville, VA 22903-2442

Abstract

Previous literature on VLSI routing and wiring estimation typically assumes a one-to-one correspondence between terminals and ports. In practice, however (say, in a gridded routing regime), each "terminal" consists of a large collection of electrically equivalent ports, a fact that is not accounted for in layout steps such as wiring estimation. The presence of multiple ports for a given terminal gives rise to the *group Steiner minimal tree problem*. In this paper, we address the general problem of minimum-cost routing tree construction in the presence of multi-port terminals. Our main result is the first known heuristic with a *sub-linear* performance bound. In particular, for a net with k multi-port terminals, previous heuristics have a performance bound of $(k-1) \cdot OPT$, while our construction offers an improved performance bound of $(1 + \ln \frac{k}{2}) \cdot \sqrt{k} \cdot OPT$. Our Java implementation is available on the World Wide Web.

1 Introduction

Previous works on routing often assume a one-to-one correspondence between terminals and ports. In other words, they either implicitly or explicitly require each terminal to consist of a *single* port. However, in actual layout, a "terminal" to which a wire is to be routed can consist of a large collection of separate ports. Even though a wire may connect to any one of these ports, this degree of freedom is often not fully exploited in routing or in wiring estimation.

In this paper, we address the general problem of minimum-cost Steiner tree construction in the presence of multi-port terminals. Clearly, the problem of interconnecting a net with multi-port terminals is a direct generalization of the NP-hard Steiner problem, and is therefore itself NP-hard (in the classical Steiner problem each terminal contains exactly one port). The Steiner Minimal Tree problem is defined as follows.

The Steiner Minimal Tree (SMT) problem: given an undirected weighted graph $G = (V, E)$ and $M \subseteq V$, find a minimum-cost tree which spans all of M.

The *Steiner* nodes $V - M$ may be optionally used in order to reduce the overall cost of the tree spanning the set of terminals M. In this paper, we address the generalization of the Steiner Minimal Tree problem to nets having multi-port terminals, formalized as follows.

The Group Steiner Minimal Tree (GSMT) problem [4, 10]: given an undirected weighted graph $G = (V, E)$ and a family $N = \{N_1, ..., N_k\}$ of disjoint groups of nodes $N_i \subseteq V$, find a minimum-cost tree which contains at least one node (i.e., port) from each group N_i.

As in the classical Steiner problem, we are allowed to include optional (i.e., Steiner) nodes, in order to reduce the cost of the spanning tree interconnecting the groups of N.

We note that there is a second version of the GSMT problem, namely, the *strong connectivity* version, which allows different connections to the same group to attach to *different* nodes in that group (i.e., all the nodes of a group are implicitly connected to each other, which allows the solution to the GSMT problem to be a forest - see Figure 1(b)). On the other hand, the GSMT problem studied in this paper (i.e., the one defined above) involves *weak connectivity* 1(a), where the solution to the GSMT problem must be strictly a tree (and intra-group edges must be explicitly part of the solution).

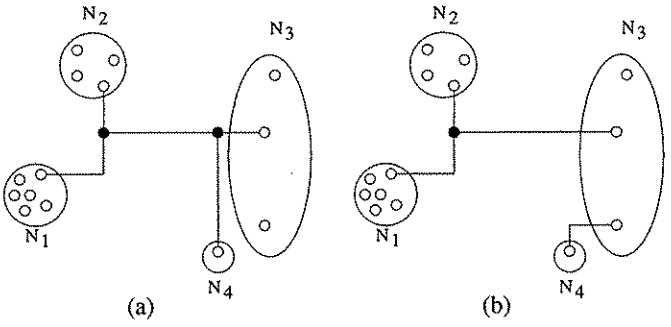

Figure 1: A feasible solution for (a) the weak-connectivity version of GSMT, and for (b) the strong-connectivity version of GSMT. Ovals represent terminals (i.e., groups), hollow dots represent ports within a terminal, and solid dots represent Steiner nodes.

*This work was supported in part by a Packard Foundation Fellowship, by NSF Young Investigator Award MIP-9457412, and by Volkswagen-Stiftung. Corresponding author is Professor Gabriel Robins, Department of Computer Science, Thornton Hall, University of Virginia, Charlottesville, VA 22903-2442, USA, Email: robins@cs.virginia.edu, phone: (804) 982-2207, http://www.cs.virginia.edu/~robins/

from Proceedings of ACM/SIGDA International Symposium on Physical Design, Napa Valley, CA, April, 1997.

The GSMT problem captures several practical scenarios in VLSI layout design:

- In grid-based maze routing regimes, "dot-models" or similar techniques are commonly used to create multi-node routing abstracts for each terminal within the multi-layer grid. A complicated terminal geometry can easily have 30 or more ports located on multiple fabrication layers. These ports form a group in the GSMT formulation. The ports are electrically equivalent, and a routing tree may connect to more than one port of a given terminal: here, the *strong connectivity* model applies.
- Alternatively the possibility of rotating and mirroring a module also induces multiple port locations for a given terminal (for a general module, there are up to 8 possible orientations) [10]. These locations correspond to a group in the GSMT formulation. Use of these ports is mutually exclusive, and a version of the *weak connectivity* model applies.
- Finally, there are cases in practice where a number of ports (say, around the boundary of a block) are electrically equivalent (by virtue of being connected inside the module, e.g., by feedthroughs); again, the GSMT problem applies with these sets of ports forming groups.

Even though such scenarios are very real, the freedom to connect to any of several port locations has not been explored in geometric or graph-based routing tree constructions, nor has it been accounted for in wiring estimation.[1] In deep-submicron technologies, the error incurred by simply approximating multiple ports with, say, their center of gravity can be substantial – especially when the error propagates to delay or slew-time estimations through multiple logic stages. Furthermore, instances of this problem become common when hierarchical design methodologies are applied (e.g., when global nets can be partially pre-routed).

The strong connectivity version, though NP-hard as well, is somewhat more tractable than the weak version. In particular, an instance of the strong connectivity version can be approximated by converting the GSMT instance into an instance of the graph Steiner problem, and then setting to zero the weight of every intra-group edge. This enables efficiently solving the strong-GSMT problem to within a factor of 2 or less of optimal using known graph-based Steiner tree algorithms [8] [12] [13].

The only existing approximation algorithms for the weak GSMT problem produce solutions $k-1$ times worse than optimal [5]. Here, we propose a new heuristic with an improved performance ratio[2] of $(1+\ln\frac{k}{2})\cdot\sqrt{k}$, where k is the number of groups. On the negative side, it is known that this problem cannot be efficiently approximated with a performance bound of less than $\ln k \cdot$ OPT [2] [6].

[1] Detailed maze routers implicitly exploit this degree of freedom, but their solutions may have unbounded error.

[2] The *performance ratio* is an upper bound on the ratio of heuristic solution cost divided by optimal solution cost, over all problem instances (i.e., the worst-case of $\frac{cost(APPROX)}{cost(OPT)}$).

2 Depth-bounded Steiner trees

In this section, we will introduce the concept of depth-bounded trees and prove that an optimal depth-2 -bounded Steiner tree approximates the optimal group Steiner tree to within a factor of $\sqrt{k}$.

In general, the given graph G may violate the triangle inequality, i.e., there may be edges (u,v) in G whose cost is greater than the the cost of the minimum u-to-v path in G. Clearly, an optimal group Steiner tree (GSMT) 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 will replace G with its *metric closure*; i.e., 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.

In order to simplify our analysis, we modify G as follows. For any port v we will create a new node v' and a new zero-cost edge (v,v'); now v' supplants the role of v, i.e., v' becomes a port and v becomes a non-port (see Figure 2). Clearly, an optimal tree in the modified graph will have the same cost as an optimal tree in the original, unmodified graph. This transformation insures that in the optimal Steiner tree every port is a leaf.

Figure 2: All ports become leaves.

Let T be an arbitrary tree with a fixed node r called the *root*. We define the *depth* of a rooted tree T as the maximum number of edges in root-to-leaf paths. Our heuristics for the group Steiner problem aim to construct rooted trees of either depth one or depth two. We define *d-stars* to be rooted trees of depth of at most d (see Figure 3(a)-(b)).

Our motivation for defining d-stars is two-fold: (i) they can be used to approximate optimal group Steiner trees well, and (ii) they can be constructed efficiently, as discussed in the next section. The goal of the rest of this section is to show that for any arbitrary (but henceforth fixed) tree T there exists a low-cost 1-star and a low-cost 2-star spanning the leaves of T. In particular, this implies that an optimal group Steiner tree can be approximated by a group *Steiner* 2-star (that is a 2-star which spans all the groups).

In deriving upper-bounds on the cost of 2-stars, we will sum the costs of tree paths between nodes which are adjacent in 2-stars. However, since such paths are not necessarily disjoint, the same tree edge may be counted multiple times in this sum/bound. We refer to this situation as *edge reuse* (see Figure 3(c)). For the tree T, edge reuse provides a loose upper bound on the ratio cost(2-star)/cost(T). Indeed, if no edge is used more than j times when replacing edges of a 2-star by the corresponding paths in T, then the 2-star has cost no more than j times the cost of the tree T. Our overall strategy below is to derive upper bounds on the edge reuse for 2-stars.

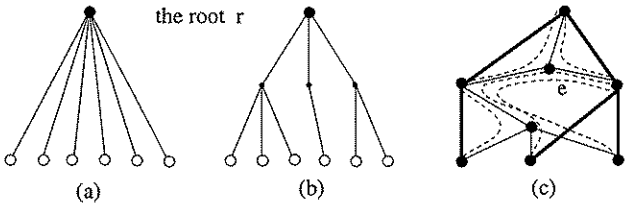

Figure 3: (a) A 1-star, and (b) a 2-star with root r. (c) Graph edges are thin, while 2-star edges are thick; the edge e is (re)used here three times in dashed paths between nodes adjacent in the 2-star.

More formally, given a tree T, a 1-star S_1, and a 2-star S_2 (collectively denoted S_d), 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. In order to establish an upper bound on the cost of a 1-star, we first select an appropriate common root r for S_1 and S_2.

The following is a known fact from graph theory (which we prove here for the sake of completeness).

Lemma 1 *Any tree T has at least one node r, called the* center, *such that each connected component of $T - \{r\}$ contains at most half the leaves of T (See Figure 4).*

Proof: We direct each edge $e = (u, v)$ in T from u to v if the number of leaves in the connected component of $T - \{e\}$ containing v is strictly more than the number of leaves in the other component (See Figure 4(a)). Since T is a tree, we can start from an arbitrary node and walk along directed edges, until we reach a node r without any incident outgoing edges. The node r is a center since no connected component of $T - \{r\}$ contains more leaves than the sum of the leaves in all of the other (disjoint) components. □

Placing the root r of S_1 at a center of the tree T minimizes the edge reuse of S_1, as follows.

Lemma 2 *Let r be a center of the tree T with the set of leaves L. The 1-star S_1 with the root r and leaves L costs no more than $\frac{|L|}{2} \cdot cost(T)$.*

Proof: Since r is a center of T, $reuse_T(S_1)$ is not larger than $|L|/2$ (See Figure 4(b)). Indeed, the reuse of an edge e of T is the number of root-to-leaf paths in T that contain e. But since r is a center, any edge of T lies on at most half of all such paths. □

Now, we will construct a 2-star S_2 from the tree T with cost no more than $2 \cdot \sqrt{|L|} \cdot cost(T)$, where L is the set of leaves of T. Note that a center r of the tree T will serve as a root of both S_1 and S_2. Note also that S_2 and T (and S_1) have the same set of leaves, namely L. Thus, to completely specify the 2-star S_2, we need to select an appropriate set of intermediate nodes that lie on paths between the root r and the leaves.

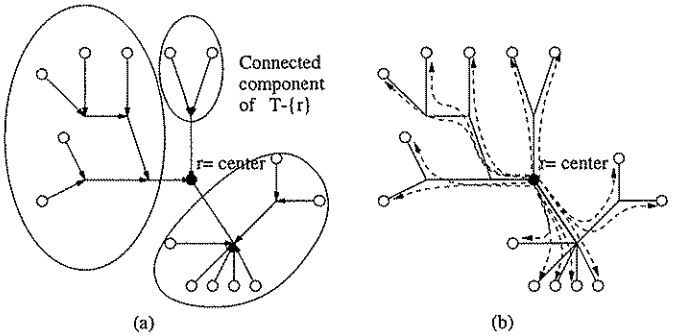

Figure 4: (a) When a tree center is removed, no subtree has more than $|L|/2$ leaves. (b) When we place the root at a center of the tree, no edge is reused more than $|L|/2$ times (in dashed paths). In this example, the edge reuse of the 1-star is 7.

Let C_i be connected components of $T - \{r\}$. In order to connect nodes in different C_i, we must pass through the root r. Define T_i as a component C_i plus the root r and the (unique) edge between the root and C_i (Figure 4(a) shows three such components C_i). Finally, we denote by $L_i = C_i \cap L$ the set of leaves of T that are contained in T_i. Because the root r is a center of T, the size of L_i is at most $|L|/2$ in each rooted subtree T_i. Now, we will construct a 2-star S_2 for the entire tree T by taking the union of the 2-stars for each of the subtrees T_i. Note that the edge reuse of S_2 in T is the maximum of edge reuses over all T_i. We define a node u as an *ancestor* of v if the path from the root to v passes through u.

Now we determine an appropriate set of intermediate nodes for a 2-star S_2 in each T_i. First, we sort the leaves of T_i in an arbitrary depth-first manner, labeling them $l_1, l_2, ..., l_{|L_i|}$. Next, we partition this sequence into fixed-size blocks of contiguously-numbered leaves. We define the intermediate nodes of our 2-star for T_i as the set of least common ancestors[3] of the leaves in each of the blocks. In our 2-star, the root is connected to these intermediate nodes, and each intermediate node is connected to the leaves of the corresponding block.

Note that the edge reuse is determined by two kinds of paths: (i) paths from the root to intermediate nodes (Figure 5(a)), and (ii) paths from intermediate nodes to leaves (Figure 5(b)). The number of paths of type (i) is clearly bounded by the number of blocks, i.e., $|L_i|/b$, where b is the block size.

Now we estimate the contribution to edge reuse of paths of type (ii). Since we have sorted the nodes in depth-first order, any node v will be a common ancestor for a sequence of contiguously-numbered leaves, say $l_x, l_{x+1}, ..., l_y$. Clearly, v is an ancestor of the least common ancestors of all blocks that are contained completely in this sequence. Thus, the edge between v and its parent u does not lie on the paths from the interme-

[3] a common ancestor of a set of nodes is the *least* common ancestor if it is not an ancestor of any other common ancestor of these nodes.

diate nodes to leaves of such blocks. Therefore, we are concerned only with paths to the leaves outside of such blocks. Such leaves may occupy only the first or the last $b-1$ positions in the sequence $l_x, ..., l_y$. This induces a bound of $2 \cdot (b-1)$ on the contribution of type-(ii) paths to the reuse of edge (u, v).

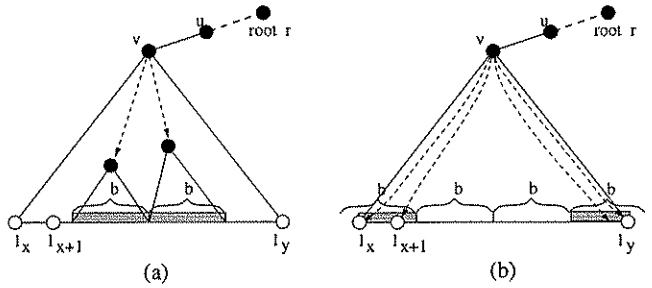

Figure 5: Triangles represent depth-first -ordered subtrees. (a) Type-(i) paths which reuse the edge (u, v) terminate at intermediate nodes below v (there are no more than $|L_i|/b$ of these). (b) Type-(ii) paths which reuse the edge (u, v) terminate at leaves that lie in the leftmost and the rightmost blocks that together contain at most $2 \cdot (b-1)$ leaves.

Thus, the total edge reuse is at most $\frac{|L_i|}{b} + 2 \cdot b$. Choosing $b = \sqrt{|L_i|/2}$ yields an upper bound on edge reuse of $2\sqrt{2 \cdot |L_i|}$. Since the root is the center of T, we have $|L_i| \leq \frac{1}{2}|L|$, and the edge reuse is at most $2\sqrt{|L|}$. This proves that for any tree T with the set of leaves L and a center r, there is a 2-star rooted at r with the same set of leaves L, and with cost at most $2\sqrt{|L|} \cdot cost(T)$. A more sophisticated analysis can improve this bound by a factor of 2. This establishes the following result.

Theorem 3 *Let Opt be an optimal group Steiner tree, let k be the number of groups, and r be a center of Opt. Then:*

1. *the cost of an optimal Steiner 1-star rooted at r is at most $\frac{k}{2} \cdot cost(Opt)$; and*
2. *the cost of an optimal Steiner 2-star rooted at r is at most $\sqrt{k} \cdot cost(Opt)$.*

3 A Provably-Good Heuristic

From the previous section, we know that an optimal Steiner 2-star is a reasonable approximation of an optimal group Steiner tree, denoted *Opt*. Unfortunately, the problem of finding an optimal Steiner 2-star is NP-hard. In this section, we will therefore indirectly approximate *Opt* by approximating an optimal Steiner 2-star. We can show that the problem of approximating a minimum set cover can be embedded in the problem of approximating a minimum Steiner 2-star. Therefore, for any $\epsilon > 0$, it is unlikely that there exists a polynomial-time approximation algorithm with performance ratio $(1-\epsilon) \cdot \ln k$, where k is the number of groups [2]. We present a (sub)optimal algorithm with performance ratio $1 + \ln \frac{k}{2} \approx 0.307 + \ln k$.

Theorem 3 states that there exists a low-cost Steiner 2-star with the root placed in a center of an optimal group Steiner tree *Opt*. Although we do not know which node of the graph G is a center of *Opt*, this is not an obstacle: for each node r of G we construct a low-cost Steiner 2-star $Appr_2(r)$ with root r, and then we select the least-cost $Appr_2(r)$ over all possible choices of r (see Figure 6). Thus, we only need to specify how we will construct a Steiner 2-star $Appr_2(r)$ with a given root r.

Group Steiner Heuristic for arbitrary weighted graphs
Input: a graph $G = (V, E)$, a family N of k disjoint groups $N_1, \ldots, N_k \subseteq V$
Output: a low-cost tree $Appr$ spanning at least one vertex from each group N_i
For each node $r \in V$ **do**
find a low-cost 2-star $Appr_2(r)$ rooted at r intersecting each group $N_i, i = 1, ..., k$
Output the least-cost 2-star $Appr$, i.e. such that $cost(Appr) = \min_{r \in V} cost(Appr_2(r))$

Figure 6: Our main approximation algorithm for the group Steiner problem on arbitrary graphs produces a low-cost Steiner 2-star.

Let $Opt_1(r)$ be the optimal Steiner 1-star rooted at r, and let $Opt_2(r)$ be the optimal Steiner 2-star rooted at r. The main idea in constructing $Appr_2(r)$ is sequential improving of $Opt_1(r)$. There are two advantages to using $Opt_1(r)$ as an initial approximation for $Opt_2(r)$: (i) unlike $Opt_2(r)$, the 1-star $Opt_1(r)$ can be found efficiently, and (ii) the cost of $Opt_1(r)$ is bounded by $\frac{k}{2} \cdot cost(Opt)$ if r is a center of *Opt* (Theorem 3).

Therefore, to measure the approximation quality of a 2-star, we will compare its cost to the cost of an optimal 1-star with the same root and with leaves taken from the same groups spanned by the 2-star. Formally, let S_2' be a 2-star with root r and let $groups(S_2')$ be the set of of groups spanned by S_2'. We denote by S_1' an optimal 1-star with root r connected to $groups(S_2')$ (see Figure 7). We define the *norm* of S_2' as $norm(S_2') = cost(S_2')/cost(S_1')$.

In order to specify our low-cost 2-star $Appr_2(r)$, we need to select the intermediate nodes and also determine the set of groups that should be connected to each intermediate node. It is therefore natural to represent $Appr_2(r)$ as a union of subtrees, each consisting of a single intermediate node that is connected to the root as well as to certain leaf ports. Such rooted subtrees will be called *partial stars* (see Figure 7(a)).

We select the partial stars for $Appr_2(r)$ in the following greedy manner. First, we find a partial star P with the minimum norm (i.e., the minimum ratio of the cost of P over the cost of the corresponding 1-star). Next, we remove the groups that it spans (i.e., $groups(P)$) from the set of all groups. Finally, we determine the next partial star with minimum norm, and iterate until all groups are spanned. Figure 8 gives a formal definition of this procedure.

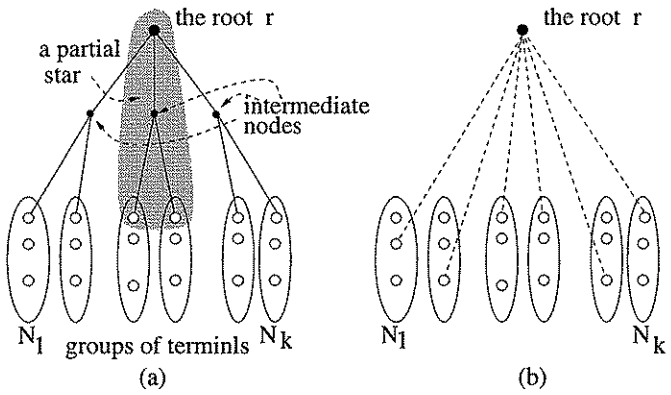

Figure 7: (a) A Steiner 2-star, and (b) the corresponding optimal 1-star. The shaded areas in (a) is a "partial star".

Minimum Partial Star Algorithm

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

Output: The minimum-norm partial star $P(r, M)$ with the root r and leaves from some groups of M

For each $v \in V$ **do**

sort $M = \{N_1, ..., N_k\}$ such that $\frac{cost(v,N_i)}{cost(r,N_i)} \leq \frac{cost(v,N_{i+1})}{cost(r,N_{i+1})}$

find $j \in \{1, ..., k\}$ that minimizes

$$norm(v) = \frac{cost(r,v) + \sum_{i=1}^{j} cost(v, N_i)}{\sum_{i=1}^{j} cost(r, N_i)}$$

$M(v) \leftarrow \{N_1, ..., N_j\}$

Find v with the minimum $norm(v)$

Output the partial star $P(r, M)$ with the intermediate node v adjacent to the root r and groups $M(v)$

Figure 9: Our algorithm for finding a minimum-norm partial star.

In order to complete the description of our heuristic, we will next present an efficient procedure that, given a root r and set of groups M, finds a minimum-norm partial star $P(r, M)$ rooted at r and spanning some of the groups in M. This procedure is formally described in Figure 9. Note that in this procedure we use $cost(u, N_i)$ to denote the cost of an edge between u and any node in group N_i.

Rooted Steiner 2-star Heuristic

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

Output: A low-cost 2-star $Appr_2(r)$ with the root r intersecting each group N_i

$Appr_2(r) \leftarrow \{r\}$

$M \leftarrow N$

While $M \neq \emptyset$ **do**

find a partial star $P = P(r, M)$ with the minimum norm

$M \leftarrow M \setminus groups(P)$

$Appr_2(r) \leftarrow Appr_2(r) \cup P$

Output $Appr_2(r)$

Figure 8: The greedy heuristic for the rooted graph.

The following lemma establishes a performance ratio for the Rooted Steiner 2-star Heuristic (Figure 8). (The proof is omitted in this extended abstract.)

Lemma 4 *Let $Opt_d(r)$, $d = 1, 2$, be an optimal Steiner d-star rooted at r. The cost of the output of the Rooted Steiner 2-star Heuristic (Figure 8) is at most:*

$$cost(Appr_2(r)) \leq (1 + \ln \frac{cost(Opt_1(r))}{cost(Opt_2(r))}) \cdot cost(Opt_2(r))$$

Together with Theorem 3 this implies our main result:

Theorem 5 *The Group Steiner Heuristic (Figures 6, 8, 9) solves the the Group Steiner Minimal Tree problem with performance ratio $(1 + \ln \frac{k}{2}) \cdot \sqrt{k}$, where k is the number of groups.*

Proof: Our overall Group Steiner Heuristic (Figure 6) runs the Rooted Steiner 2-Star Heuristic (Figure 8) for all possible roots $r \in V$. If the root is a center r_c of the optimal group Steiner tree Opt, then by Theorem 3, we know that $cost(Opt_1(r_c)) \leq \frac{k}{2} \cdot cost(Opt) \leq \frac{k}{2} \cdot cost(Opt_2(r_c))$, where k is the number of groups. Therefore, Lemma 4 implies that the cost of the tree output by our main algorithm is at most $1 + \ln \frac{k}{2}$ times the cost of the optimal Steiner 2-star. Finally, by Theorem 3, we obtain a group Steiner tree that costs no more than $(1 + \ln \frac{k}{2}) \cdot \sqrt{k}$ times the optimum. □

4 Time Complexity and Enhancements

In this section, we first estimate the runtime of our heuristic and then we discuss several ways of improving its runtime and performance ratio. Let n denote the total number of ports in all of the groups, i.e. $n = |\cup_{i=1}^{k} N_i|$. Let α denote the time complexity of computing all-pairs shortest paths in the graph G. As part of our preprocessing, we shall also compute in time $O(n \cdot |V|)$ all vertex-to-group distances (i.e., distances between each vertex and the closest port in each group). The time complexity of the Minimum Partial Star Heuristic (Figure 9) is $O(|V| \cdot k \cdot \log k)$. Therefore, the Rooted 2-star Heuristic (Figure 8) has runtime $O(|V| \cdot k^2 \cdot \log k)$, where k is the number of groups. Thus we obtain the following result:

Theorem 6 *The total runtime of the Group Steiner Heuristic (Figure 6) is $O(\alpha + |V|^2 \cdot k^2 \cdot \log k)$, where k is the number of groups, and α is the time complexity of computing all-pairs shortest paths.*

The performance ratios derived in previous sections pertain to *worst-case* analysis. However, in practice we are also interested in the average-case behavior of our heuristics, in terms of both the solution quality and the runtime. One such practical improvement entails omitting the removal of the set of groups spanned by the minimum-norm 2-star (see the inner loop of the algorithm in Figure 8). 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. We then use this distance to sort the groups in the Minimum Partial Star Algorithm (Figure 9).

Another practical enhancement entails computing a *group minimum spanning tree* (GMST) instead of a *group Steiner minimal tree* (GSMT), that is, a minimum spanning tree for a set of nodes containing *exactly* one port from each group. It can be shown that the optimal GMST is at most twice longer than the optimal GSMT. Thus, in approximating the GSMT by a GMST, we lose only a factor of 2 which does not asymptotically increase the overall bound of $(1 + \ln \frac{k}{2}) \cdot \sqrt{k}$, yet yields substantial savings in runtime.

We may further modify our algorithm by finding the minimum spanning (or approximate Steiner) tree for the set of intermediate nodes and ports chosen by the Group Steiner Heuristic (Figure 6). We may also make local (one at a time) port substitutions in groups to rearrange the tree topology and 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 [11] that Christifides' heuristic (i.e., the best-known heuristic for traveling salesperson in graphs [9]) also provides excellent initial traveling salesperson tours for further local rearrangements.

In the rest of this section, we will show how to handle more effectively instances of the GSMT problem with some *degenerate* groups, i.e. groups of size 1. We will see that treating degenerate groups differently will yield improvements in solution quality as well as in runtime.

The degenerate groups by themselves induce an instance of the classic Steiner Minimal Tree problem, and such an instance can be approximated efficiently by known methods (with a constant performance ratio). Thus, to solve the SMT problem for degenerate groups, we may choose a provably-good heuristic from among the numerous existing ones [1] [3] [7] [8] [12]. For example, in time $O(|V|^3)$ we may find a Steiner tree which is at most $\frac{11}{6}$ times longer than the optimal [13]. The remaining issue now is to combine the Steiner minimal tree over the degenerate groups with a tree spanning the other, non-degenerate groups.

Let $N = M_1 \cup M_2$ be the partition of N into groups containing exactly one port (M_1), and groups containing at least two ports (M_2). We suggest the following *Combined Group Steiner Heuristic* (CGSH): first, we find the usual SMT for the ports of M_1 using the algorithm from [13]. Next, using our Group Steiner Heuristic (Figure 6), we find the GSMT for the family of groups that includes M_2 and a single arbitrary port from (one of the groups of) M_1. Finally, we output a minimum spanning tree over the union of these two trees.

If the number of degenerate groups is large, then the CGSH heuristic will enjoy considerable runtime savings as compared to the Group Steiner Heuristic (of Figure 6). Moreover, the following theorem shows that this heuristic also enjoys an improved performance bound.

Theorem 7 *The Combined Group Steiner Heuristic solves the GSMT problem with a performance ratio of at most:*

$$\frac{11}{6} + (1 + \ln \frac{k_2}{2}) \cdot \sqrt{k_2}$$

where M_2 is the set of groups of size at least 2, and $k_2 = |M_2| + 1$.

5 Experimental Results

We have implemented our heuristic for the group Steiner problem using the Java programming language. Our implementation is available on the World-Wide-Web at:

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

This choice of implementation will enable researchers to execute our code directly off the World-Wide-Web using any Java-enabled browser (this is the first implementation of a CAD research workbench using Java as far as we know). Preliminary results appear quite promising and are discussed below.

We compared our heuristic with the heuristic proposed by Reich & Widmayer [10]. Their 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 heuristic then repeatedly deletes such a leaf node which is incident to the longest edge among all such nodes (see Figure 10 for a formal description of this algorithm).

The RW heuristic for the Group Steiner Problem [10]

Input: a graph $G = (V, E)$, k disjoint groups $N_1, \ldots, N_k \subseteq V$
Output: a low-cost tree T spanning ≥ 1 nodes from each N_i

$M \leftarrow \cup_{i=1}^{k} N_i$
Find minimum spanning tree T over M, i.e. $T \leftarrow MST(M)$
Repeat forever
  For each leaf l node of T **do**
    find the group $N(l) = N_i$ containing l
    If $N(l) \cap T = l$, **then** mark l
  If all leaves of T are marked **then** exit repeat
  find an unmarked leaf l incident to the longest edge
  remove l from T, i.e. $T \leftarrow T - l$
Output the tree T

Figure 10: The heuristic of Reich & Widmayer [10] for the Group Steiner Problem

Table 1 compares two versions of our heuristic (Figure 6) to the Reich & Widmayer algorithm [10] (Figure 10). We created each group by first defining a randomly-placed square area of predetermined size, and then uniformly and independently distributing nodes inside this square-shaped area. We varied the predetermined group areas among 10%, 50%, and 100% of the overall square routing region. All table numbers represent the average relative improvement over 100 trials, given as a percent improvement over the tree cost of Reich & Widmayer's algorithm [10] (negative numbers represent disimprovements).

In the context of VLSI layout, we are primarily concerned with the *rectilinear* (or *Manhattan*) plane, where the cost of routing between two nodes (a_x, a_y) and (b_x, b_y) is defined to be $|a_x - b_x| + |a_y - b_y|$. While our implementation uses the rectilinear metric to determine the distances between ports, our algorithms are general and indeed apply to arbitrary weighted graphs.

The first version of the Group Steiner Heuristic (Figure 6) that we implemented has the following three modifications:

1. Intermediate nodes are selected strictly from among the ports (i.e., all of the constructions benchmarked are *spanning* trees - they do not use Steiner nodes other than ports);

2. The root of the 2-star is selected from a single randomly chosen group;

3. After accepting an intermediate node in the inner loop of Figure 9, it is removed from further consideration in subsequent iterations.

The second version which we implemented is a hybrid approach which is our first version above, except that 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 modified heuristic described above).

The table column labeled "2-star" shows the average percent improvements of our modified heuristic over Reich & Widmayer's algorithm, while data for the hybrid approach is given in the column labeled "2-star + MST". As can be seen from the table, the hybrid approach significantly outperforms the algorithm of Reich & Widmayer as the group sizes and the group areas increase.

Area of each group is 10% of total routing region						
num of groups	group size = 3		group size = 5		group size = 8	
	2-star	2-star +MST	2-star	2-star +MST	2-star	2-star +MST
3	4.0	4.0	5.0	5.0	7.2	7.2
5	-6.2	4.6	-2.0	5.6	-1.2	8.5
10	-23.0	6.5	-14.6	10.1	-10.5	11.0
20	-40.0	8.0	-32.2	12.3	-26.0	16.1
Ave	-16.3	5.8	-11.0	8.3	-7.6	10.7

Area of each group is 50% of total routing region						
num of groups	group size = 3		group size = 5		group size = 8	
	2-star	2-star +MST	2-star	2-star +MST	2-star	2-star +MST
3	10.2	10.2	15.2	15.2	24.7	24.7
5	4.5	10.4	16.4	21.2	23.6	27.6
10	-3.4	15.2	9.0	23.0	20.3	32.4
20	-21.9	14.5	-7.2	23.1	11.2	34.3
Ave	-2.7	12.6	8.4	20.6	20.0	29.8

Area of each group is 100% of total routing region						
num of groups	group size = 3		group size = 5		group size = 8	
	2-star	2-star +MST	2-star	2-star +MST	2-star	2-star +MST
3	13.9	13.9	28.0	28.0	31.4	31.4
5	11.8	17.7	25.9	30.2	28.5	31.3
10	-1.9	14.1	16.8	29.6	26.8	36.2
20	-13.9	18.0	6.6	31.4	15.4	37.2
Ave	-2.5	15.9	19.3	29.8	25.5	34.0

Table 1: Experimental results: all numbers represent the average percent improvement over 100 trials, with respect to the cost of the output tree of Reich & Widmayer's algorithm [10] (negative numbers represent disimprovements).

References

[1] P. Berman and V. Ramaiyer, *Improved Approximations for the Steiner Tree Problem*, in Proc. ACM/SIAM Symp. Discrete Algorithms, San Francisco, CA, January 1992, pp. 325–334.

[2] U. Feige, *A Threshold of ln n for Approximating Set Cover*, in Proc. ACM Symp. the Theory of Computing, May 1996, pp. 314–318.

[3] 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 (1994), pp. 1351–1365.

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

[5] E. Ihler, *Bounds on the quality of approximate solutions to the group Steiner problem*, in Lecture Notes in Computer Science, vol. 484, 1991, pp. 109–118.

[6] E. Ihler, *The complexity of approximating the class Steiner tree problem*, tech. rep., Institut fur Informatik, Universitat Freiburg, 1991.

[7] A. B. Kahng and G. Robins, *A New Class of Iterative Steiner Tree Heuristics With Good Performance*, IEEE Trans. Computer-Aided Design, 11 (1992), pp. 893–902.

[8] L. Kou, G. Markowsky, and L. Berman, *A Fast Algorithm for Steiner Trees*, Acta Informatica, 15 (1981), pp. 141–145.

[9] 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.

[10] G. Reich and P. Widmayer, *Beyond Steiner's problem: A VLSI oriented generalization*, in Lecture Notes in Computer Science, vol. 411, 1989, pp. 196–211.

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

[12] A. Z. Zelikovsky, *An 11/6 Approximation Algorithm for the Network Steiner Problem*, Algorithmica, 9 (1993), pp. 463–470.

[13] A. Z. Zelikovsky, *A Faster Approximation Algorithm for the Steiner Tree Problem in Graphs*, Information Processing Letters, 46 (1993), pp. 79–83.