Thirty-Five-Point Rectilinear Steiner Minimal Trees in a Day

Jeffrey S. Salowe*

Department of Computer Science, University of Virginia, Thornton Hall, Charlottesville, Virginia 22903

David M. Warme†

Telenex Corporation, 7401 Boston Blvd., Springfield, Virginia 22153

Given a set of terminals in the plane, a rectilinear Steiner minimal tree is a shortest interconnection among these terminals using only horizontal and vertical edges. We present an algorithm that constructs a rectilinear Steiner minimal tree for any input terminal set. On a workstation, problems involving 20 input terminals can be solved in a few seconds, and problems involving 30 input terminals can be solved, on average, in 30 min. Previous algorithms could only solve 16- or 17-point point problems within the 30 min time bound. Problems involving 35 points can be solved, on average, within a day. Our experiments were run on uniformly distributed data on an integer grid. © *1995 John Wiley & Sons, Inc.*

1. INTRODUCTION

Let V be a set of n *terminals* in the plane. A *rectilinear Steiner minimal tree* for V is a tree of shortest length containing V as its vertex set, where the distance between two vertices is measured in the rectilinear metric. Rectilinear Steiner minimal trees are useful in VLSI physical design, particularly global routing and wire-length estimation. Garey and Johnson [11] showed that the decision problem corresponding to the rectilinear Steiner minimal tree problem is NP-complete. NP-completeness strongly suggests that there is no polynomial-time algorithm to construct rectilinear Steiner minimal trees, and, in fact, the known exact algorithms are dreadfully inefficient (not applicable to size 20 terminal sets on a workstation). Relevant papers supporting this statement are mentioned in Section 3. We describe and study an algorithm that can solve 20-point problems in a few seconds on a workstation (a 48 Meg Sun SPARCstation IPC) and can solve 30-point problems, on average, in approximately 30 min. Our experiments were run on uniformly distributed data on an integer grid.

We begin by considering several methods to construct rectilinear Steiner minimal trees. We show that one natural class of algorithms, the "monotonic iterative algorithms," are unable to compute Steiner minimal trees for all point sets. This material appears in Section 4.

We then consider a methodology inspired and influenced by recent results of Winter [28] and Cockayne and Hewgill [5] on exact algorithms for Euclidean Steiner minimal trees. Specifically, this method focuses on *full topologies,* which are Steiner minimal trees having the property that all terminals are leaves, and *full sets,* which are terminal sets, all of whose rectilinear Steiner minimal trees are full topologies. It is well known there is a rectilinear Steiner minimal tree for V that can be decomposed

* Supported by NSF Grants MIP-9107717 and CCR-9224789. Current address: QuesTech, Inc., 7600A Leesburg Pike, Falls Church, VA 22043. E-mail address: jsalowe@nvl.army.mil.

† Some of the work was done as a graduate student at the University of Virginia. E-mail address: warme@telenex.com.

into a collection of full topologies on full sets. The resulting algorithm is described and partially analyzed.

The algorithm is divided into two parts, generating candidate full sets and backtrack searching. The first part is based on a series of properties that limit the number of *candidate full sets,* which are full sets that might appear in a Steiner minimal tree. These properties are described and justified in Sections 5.1.1 and 5.1.2. Empirical findings are given in Section 5.1.3, and some theoretical results corroborating the empirical findings appear in Section 5.1.4.

The second part of the algorithm consists of a backtracking component that combines these candidate full sets into a rectilinear Steiner minimal tree. This algorithm is described in Section 5.2. We implemented a computer program to compute rectilinear Steiner minimal trees and collected statistics. Some of the more interesting statistics are presented and discussed in Sections 5.2.2 and 5.2.4. Various improvements are discussed in Sections 5.2.3 and 5.2.5. A rectilinear Steiner minimal tree for a randomly generated 50-point problem is depicted in Figure 1.

It turns out that our results differ significantly from the Euclidean Steiner minimal tree results obtained by Cockayne and Hewgill [5]. We discuss this and other related issues in Sections 6 and 7.

2. TERMINOLOGY

This section is a repository for terminology in the paper. Let a and b be points in the plane. Let $\|a - b\|$ be the distance from a to b as measured in the rectilinear metric, i.e., if $a = (x_1, y_1)$ and $b = (x_2, y_2)$, then $\|a - b\| = |x_1 - x_2| + |y_1 - y_2|$.

Let V be a set of points in the plane called *terminals.* A *Steiner tree* for V is a tree containing V as its vertex set. The *length of an edge* $e = (u, v)$, *length*(e) [sometimes abbreviated as *length*(u, v)], is given by $\|u - v\|$, and the length of a Steiner tree T, *length*(T), is the sum of its edge lengths. A *rectilinear Steiner minimal tree* for V is a shortest Steiner tree for V.

A Steiner minimal tree T for V is said to have a *full topology* if every vertex in V is a leaf in T. A terminal set V is said to be a *full set* if every Steiner minimal tree for V is a full topology. A terminal set that is a full set and also has size k is said to be a *full set of size* k. With respect to a point set V, a set $S \subseteq V$ is said to be a *full set with respect to* V if S is a full set, and there is some Steiner minimal tree for V that contains a full topology of S as a subgraph.

Let $MST(V) = (V, E_{MST})$ be a minimum spanning tree for V, let $SMT(V) = (\hat{V}, E_{SMT})$ be a Steiner minimal tree for V, where $V \subseteq \hat{V}$, and let $SMT_k(V)$ be a shortest Steiner tree for V containing exactly k Steiner points. Also,

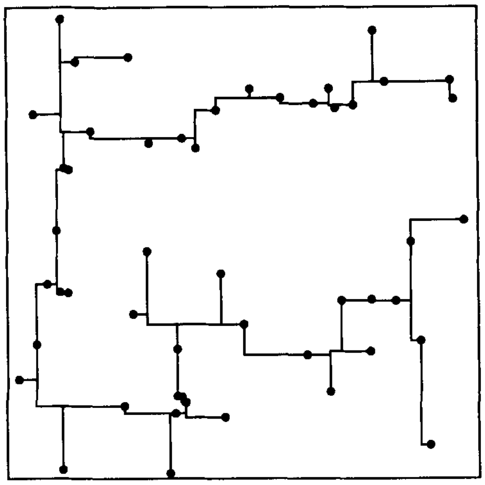

Fig. 1. A rectilinear Steiner minimal tree on 50 points.

let $P_T(a, b)$ be the path between vertex a and vertex b in tree T, and let $\delta_T(a, b) = \max_{e \in P_T(a,b)} length(e)$.

A rectilinear Steiner tree can be embedded in the plane so that all edges consist solely of vertical and horizontal line segments, generically called *segments.* The endpoints of a segment are called *nodes.* Segments intersect only at their endpoints, i.e., no node is in the relative interior of a segment. (Relative interior is used in its usual geometric sense.) Without loss of generality, we require that a degree two node having incident collinear segments be a terminal.

We define a *line* to be a sequence of one or more adjacent, collinear segments with no terminals in its relative interior. A *complete line* is a line of maximal length; it does not have terminals in its relative interior, but it may have terminals as endpoints. A *corner point* is a degree two node that is not a terminal. A corner point is incident to exactly one horizontal complete line and exactly one vertical complete line. These complete lines are the *legs* of the *corner.* A *T-node* is a degree three node that is not a terminal. A T-node is incident to two complete lines, the *head* of the T, which extends on both sides of the T, and the *body* of the T. Finally, a *cross-node* is a degree four node that is not a terminal.

A segment e is *properly incident* to a line l if e intersects l's relative interior. Two segments e and f properly incident to l are said to be *neighboring segments* if $e \cap l \neq f \cap l$ and the portion of l connecting $e \cap l$ and $f \cap l$ is a segment. Neighboring segments are said to *alternate* along l if no two neighboring segments are on the same side of l and all Steiner points in the relative interior of l are T-nodes; note that if neighboring segments alternate along l, no cross-node is in the relative interior of l.

3. LITERATURE SURVEY

At the time the survey paper of Hwang and Richards [16] was written, little effort had gone into devising exact algorithms for rectilinear Steiner minimal trees. They stated that only one algorithm, due to Yang and Wing [30], was proposed specifically for the rectilinear problem and that the Yang and Wing algorithm is limited to terminal sets of size less than 10. Since that time, there have been several papers. One paper, written by Sidorenko [25], appeared in Russian in 1989. A translation of the abstract states that the algorithm is applicable to terminal sets of size 11 or less. Lewis et al. [20] gave an algorithm with approximately the same applicability on an IBM 3090. Thomborson et al. [27] and Ganley and Cohoon [9] presented dynamic programming algorithms. Thomborson et al. focused on implementation details of the Dreyfus and Wagner algorithm, described below, and they could solve 16-point problems within 30 min on a Sun SPARCstation IPC. Ganley and Cohoon used Hwang's theorem on full sets to describe a dynamic programming algorithm, and they could solve 18-point problems on a Sun SPARCstation IPC in 30 min. Thomborson et al. [26], Deneen et al. [7], and Shute [24] described, but did not implement and empirically study, other rectilinear algorithms. Shute's algorithm focuses on full topologies. We note that the empirical running times mentioned above more than double with the inclusion of an additional point, so improved technology has only a limited effect on the results.

Other algorithms for the rectilinear problem are actually algorithms for the *graphical Steiner minimal tree problem.* In the graphical Steiner minimal tree problem, the input is a weighted graph $G = (\hat{V}, E)$ and a terminal set $V \subseteq \hat{V}$. A *graphical Steiner tree for G* is a subtree of G that contains V in its vertex set. A *graphical Steiner minimal tree for G* is a shortest graphical Steiner tree for G.

An important theorem of Hanan [13] states that there is a rectilinear Steiner minimal tree for a terminal set V that is a subtree of the following graph, called the *grid graph.* Draw a horizontal and vertical line through each terminal in V. The vertex set of the grid graph is the set of intersection points of these lines, and the edge set consists of the line segments connecting pairs of vertices. Hanan's theorem shows that the rectilinear Steiner problem is a special case of the Steiner problem in graphs, so any graphical Steiner minimal tree algorithm can also be used to find a rectilinear Steiner minimal tree.

Detailed descriptions of algorithms for the graphical Steiner tree problem appear in Winter's survey [28]. We briefly summarize two of the algorithms that are amenable to asymptotic analysis: Hakimi's algorithm and the Dreyfus and Wagner algorithm. Our discussion is confined to the rectilinear problem.

Hakimi's algorithm finds $SMT_k(V)$ for each k from 0 to $n - 2$, selecting the overall minimum as the Steiner minimal tree. It is well known that at most $n - 2$ Steiner points appear in a Steiner minimal tree on a set of n terminals. The time complexity of Hakimi's algorithm is $O(n^2 2^{n^2-n})$.

The Dreyfus and Wagner algorithm is a dynamic programming algorithm. The key notion is that any Steiner minimal tree can be split at a degree 2, 3, or 4 node into two trees, each of which is a Steiner minimal tree of an easily described point set. Its time complexity is $O(n^2 3^n)$.

Many other graphical algorithms have been proposed [16, 29], but it does not appear that their applicability to the rectilinear Steiner minimal tree problem has been seriously studied. In stark contrast to the rectilinear problem, the Euclidean problem has received substantial and focused attention. Melzak [21] gave the first finite algorithm; as in all known exact algorithms for the Euclidean Steiner minimal tree problem, full topologies and full sets are central, since there is a Steiner minimal tree that is the union of full topologies on full sets. Winter [28] provided an important contribution with his program GEOSTEINER, which was able to solve problems of up to 15 points by limiting the number of full sets. (Specifically, problems of size 15 can be solved in several minutes. We record the author's claim of applicability since there is no standard measuring time period, and it is difficult to extrapolate running times to larger point sets. The reader is referred to the papers for specifics on the running times.) Cockayne and Hewgill [4, 5] devised a better way to combine full sets to increase the solvable range first to 30 and then to 100 points. We use this methodology in our rectilinear Steiner minimal tree algorithm.

4. INADEQUACY OF ITERATIVE METHODS

A natural idea to construct a short Steiner tree is the following: Start with the original terminal set V, and successively add Steiner points, one Steiner point at a time; the Steiner tree corresponding to a point set (consisting of terminals and Steiner points) is a minimum spanning tree for that point set. Note that once added a Steiner point cannot be removed. We call such a method an *iterative method.* An algorithm that implements an iterative method is called an *iterative algorithm.*

During the course of an iterative algorithm, a sequence of point sets $V = V_0 \subseteq V_1 \subseteq V_2 \cdots$ is constructed. Call an iterative algorithm *monotonic* if the length of $MST(V_i)$ does not exceed the length of $MST(V_{i-1})$, $i \geq 1$, i.e., if one considers the sequence of the lengths of the Steiner trees $MST(V_0)$, $MST(V_1)$, . . . , the sequence is monotonically nonincreasing. We show that there are terminal sets for which no monotonic iterative algorithm can construct a Steiner minimal tree.

Theorem 1. *There is a terminal set V with the property that length$(SMT(V))$ < length$(MST(V))$, but length-*

$(SMT_1(V)) = length(MST(V))$. Further, there is a terminal set V for which any monotonic iterative algorithm fails to construct a Steiner minimal tree for V.

Proof. Consider the terminal set V given in Figure 2, where a solid line indicates a collection of terminals, very close together, a small circle indicates a single terminal, and a dashed line with an associated number indicates a distance. It is easily checked that terminal set V satisfies the first part of the theorem. To prove the second part of the theorem, note that any Steiner point added to V without increasing the length of the resulting minimum spanning tree must lie on the embedding of an edge in the (unique) minimum spanning tree, and the resulting point set has the same property. ■

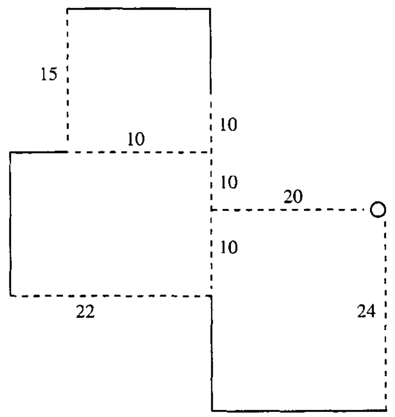

Fig. 2. The terminal set V.

The iterated 1-Steiner heuristic of Kahng and Robins [17] is an example of a monotonic iterative algorithm. It is a greedy algorithm. Given point set V, the next point set consists of V and the Steiner point in $SMT_1(V)$. The 1-Steiner heuristic terminates after n iterations or when $length(SMT_1(V_i)) = length(MST(V_i))$, whichever comes first. The algorithm does not always produce $SMT(V)$, though it appears to produce shorter trees on average than any other known heuristic. (See Kahng and Robins [17], as well as our results below.) Theorem 1 also implies that Kahng and Robins' algorithm may terminate with a point set V_i for which $length(MST(V_i)) > length(SMT(V_i))$.

We now show that any point set satisfying Theorem 1 must be "degenerate" in the sense described below. Let T be a Steiner tree. Then, T is said to be a *monotonic chain* if the following two conditions are satisfied: First, T must be a chain. Second, if the chain is traversed from one leaf to the other, the x-coordinates of the vertices either monotonically increase or monotonically decrease, as do the y-coordinates. A terminal set is said to be in *general position* if all the x-coordinates in that terminal set are distinct, as are all the y-coordinates.

Theorem 2. *Let $V \subset \Re^2$ be in a general position. Then, $length(SMT(V)) < length(MST(V))$ if and only if $length(SMT_1(V)) < length(MST(V))$.*

Proof. If $length(SMT_1(V)) < length(MST(V))$, then $length(SMT(V)) < length(MST(V))$.

Now suppose that $length(SMT_1(V)) = length(MST(V))$. Then, in each possible embedding of the edges of $MST(V)$ in the plane, no two segments can coincide; otherwise, a new Steiner point could be added along a coincident segment. The resulting tree would be shorter than $MST(V)$ and contains a Steiner point, contradicting the claim that the length of $SMT_1(V)$ is the same as the length of $MST(V)$.

Since V is in a general position, all edges incident to a particular terminal have embeddings consisting of at least two segments. As a consequence, each terminal has degree at most 2 in $MST(V)$, so $MST(V)$ must be a chain. If the vertices on this chain are not x- and y-monotone, it is easy to see that there would be two edges whose embeddings partially coincide. This implies that $MST(V)$ is a monotonic chain.

Yang and Wing [30] showed that there is a Steiner minimal tree for V completely contained in the rectilinear convex hull of V. (A rectilinear convex hull of V is a smallest rectilinearly convex body containing V; see Richards and Salowe [23].) The rectilinear convex hull of points on a monotonic chain is a monotonic chain, and it is also a minimum spanning tree for V. Therefore, $length(SMT(V)) = length(MST(V))$. ■

Corollary 1. *If V is a terminal set in general position, then $length(SMT(V)) = length(MST(V))$ if and only if the rectilinear convex hull of V is a monotonic chain.*

5. THE ALGORITHM

We now describe an algorithm that will construct a rectilinear Steiner minimal tree. This algorithm consists of two basic steps: In the first step, point sets are identified that may appear in a Steiner minimal tree as full sets; these are called "candidate full sets" below. The second step combines these candidate full sets into a Steiner minimal tree. This process is depicted in Figure 3. The set of all "candidate full sets" for a particular eight-terminal problem is depicted, along with a Steiner minimal tree. A full topology for each candidate full set is given in black, while the Steiner minimal tree appears in gray in the background. The Steiner minimal tree is a combination of some of these candidate full sets: it is the shortest such

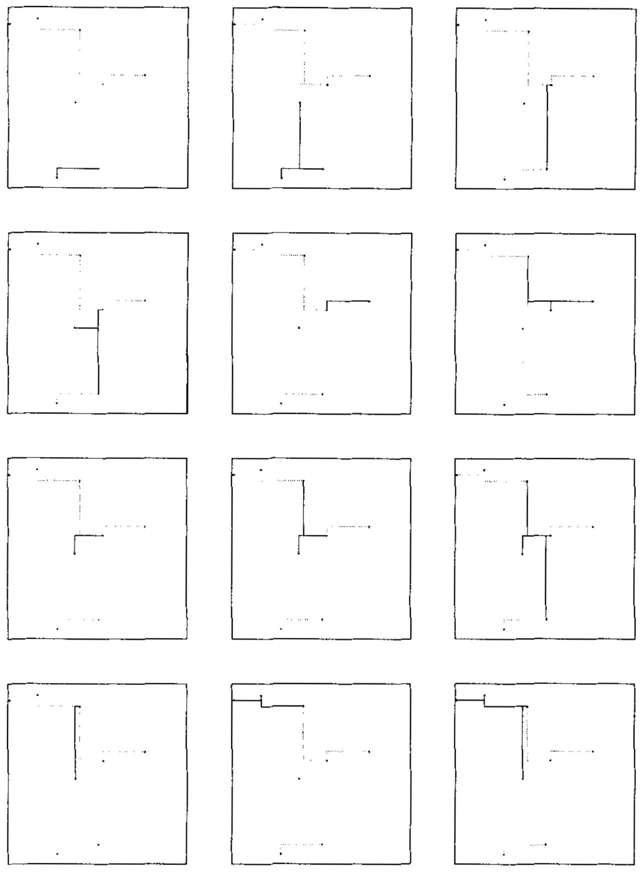
Fig. 3. A Steiner minimal tree and its candidate full sets.

combination that interconnects the terminals into a single component.

5.1. Step 1: Full Topology Decompositions

This section concentrates on Step 1, the identification of full sets. Let a *candidate full set with respect to V* be a set of points that may be a full set with respect to V. Our goal is to limit the number of candidate full sets with a collection of "easily applied tests." By easily applied, we limit the time complexity of our tests to $O(n^3)$ so that 100-point problems can be considered. Our tests fall into two basic categories: First, we have tests that reject terminal sets of small cardinality as candidate full sets, i.e., either the tests are specifically designed to assess whether small-sized terminal sets are candidate full sets or small-sized terminal sets seem more likely to fail the tests than are large-sized terminal sets. These tests are described in Section 5.1.1. Second, we have tests that tend to reject terminal sets of large cardinality as candidate full sets. These tests are described in Section 5.1.2. In Section 5.1.3, we supply implementation details and study the behavior of the tests in practice. Our conclusion is that the current implementation of Step 1 is sufficiently powerful to permit the solution of 100 terminal problems. Section 5.1.4 contains some theoretical results pertaining to Step 1.

5.1.1. Small Cardinality Restrictions

Theorem 3. *Let MST and SMT be, respectively, a minimum spanning tree and a Steiner minimal tree on V. Then,* $\delta_{MST}(a, b) \geq \delta_{SMT}(a, b)$ *for any* $a \in V$ *and* $b \in V$.

Proof. Suppose to the contrary that $\delta_{MST}(a, b) > \delta_{MST}(a, b)$ for some a and b. Consider the path $P_{SMT}(a, b)$. Let $e = (v, w)$ denote an edge on this path of length $\delta_{SMT}(a, b)$. Remove e from SMT. This causes SMT to break into two subtrees, T_a and T_b. Follow the path $P_{MST}(a, b)$ from a to b. There must be an edge $f \in P_{MST}(a, b)$ with exactly one endpoint in T_a; its length is less than the length of e. If e is replaced by f, the resulting structure interconnects V and has shorter length than SMT, a contradiction. ∎

An immediate corollary to Theorem 3 is that a candidate full set with respect to V of size 2 must be an edge in some minimum spanning tree. The following corollary to Theorem 3 generalizes this statement.

Corollary 2. *Let* $a \in V$ *and* $b \in V$. *If a and b are in a full set with respect to V of size k, then*

$$k \geq \frac{\|a - b\|}{\delta_{MST}(a, b)} + 1.$$

Proof. By Theorem 3, for any $a \in V$ and any $b \neq a \in V$,

$$\delta_{MST}(a, b) \geq \delta_{SMT}(a, b).$$

The total length of $P_{SMT}(a, b)$ is at least $\|a - b\|$, and the number of edges in $P_{SMT}(a, b)$ is at most $k - 1$, so there is an edge in $P_{SMT}(a, b)$ with length at least $\|a - b\|/(k - 1)$. Therefore,

$$\delta_{MST}(a, b) \geq \delta_{SMT}(a, b) \geq \frac{\|a - b\|}{k - 1},$$

proving the corollary. ∎

Note that Theorem 3 and Corollary 2 are also applicable to Euclidean Steiner minimal trees.

We now consider candidate full sets with respect to V of size 3. Define the *circumrectangle of a point set* to be its smallest enclosing rectangle. We say that a rectangle is *empty* if there are no terminals in V in its relative interior.

Theorem 4. *Let $a, b, c \in V$. If $\{a, b, c\}$ is a full set with respect to V, then the circumrectangle of $\{a, b, c\}$ must be empty.*

Proof. It is easy to see that if one of a, b, or c is in the circumrectangle of $\{a, b, c\}$, then $\{a, b, c\}$ is not a full set.

Now suppose that a, b, and c lie on the boundary of their circumrectangle. Without loss of generality, assume that a, b, and c are in the positions depicted in Figure 4. Given these positions, the Steiner minimal tree for $\{a, b, c\}$ is easily determined; let s be the Steiner point in the figure, and let the segments be labeled α, β, γ, and δ as depicted. By assumption, there is a Steiner minimal tree T for V that contains this Steiner minimal tree for $\{a, b, c\}$.

Let v be a point in the relative interior of the circumrectangle of $\{a, b, c\}$. Either v is in area I (possibly on the horizontal line through b), area II, area III, or area IV. Also, v is connected to s be a path containing either a, b, or c, in which case we say v is associated with a, b, or c, respectively.

Suppose that v is in area I and that $\{a, b, c\}$ is a full set with respect to V of size 3. The edge from s to a can be embedded so that it contains v, inducing a cycle in T, thereby contradicting the claim that T is a Steiner minimal tree.

Now suppose that v is in area II. If v is associated with a, then the length of a vertical edge from v to γ is less than the length of β, implying that T is not a Steiner minimal tree. If v is associated with b, then the length of a horizontal edge from v to β is less than the length of γ. If v is associated with c, then a contradiction arises if the Steiner minimal tree for $\{a, b, c\}$ is replaced by the edges from b to v and from v to a.

If v is in area III, the argument is similar to the one for area II.

Finally, suppose that v is in area IV. If v is associated with a, replace the edge from a to s with an edge from v to δ. If v is associated with b, the argument is similar to the one for area II when v is associated with c. If v is associated with c, then a contradiction arises if the Steiner minimal tree for $\{a, b, c\}$ is replaced with the Steiner minimal tree for $\{a, b, v\}$, since the latter tree is shorter than the former. ■

5.1.2. Large Cardinality Restrictions

In 1976, Hwang [15] showed that full topologies have a restrictive structure, given below.

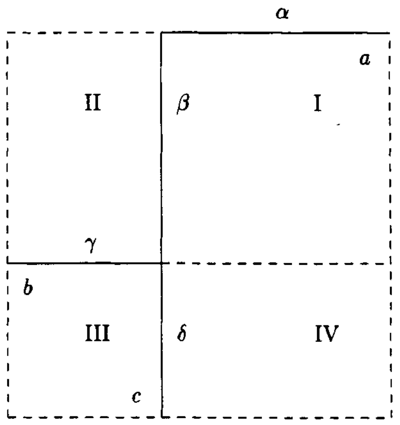

Fig. 4. The Steiner minimal tree of terminal set $\{a, b, c\}$.

Theorem 5 (*Hwang*). *Let $|V| = n > 4$, and suppose that V is a full set. Then, there is a rectilinear Steiner minimal tree for V consisting of either (1) a complete line with $n - 2$ alternating incident segments, (2) a complete corner with $n - 2$ alternating segments incident to a single leg, or (3) a complete corner with $n - 3$ alternating segments incident to one leg and a single segment incident to the other leg.*

Theorem 5 provides an efficient means to construct a Steiner minimal tree if the terminal set is full, a fact used by Agarwal and Shing [1], Cohoon et al. [6], and Richards and Salowe [23], among others. In our algorithm, we use a slightly stronger corollary:

Corollary 3. *Suppose that no two points in V share the same x- or y-coordinates and that V is a full set. Then, there is a rectilinear Steiner minimal tree for V consisting of either a complete corner with $n - 2$ alternating segments incident to a single leg or a complete corner with $n - 3$ alternating segments incident to one leg and a single segment incident to the other leg.*

We discuss a perturbation scheme in the Appendix that ensures that Corollary 3 applies to all candidate full sets.

Theorem 5 also provides a necessary, but not a sufficient, condition to decide if a set of terminals is a full set. We now describe additional, easily checked restrictions on candidate full sets.

The first restriction focuses on T's. A corollary to Theorem 5 is the observation that one of the endpoints of the body of a T is a terminal (the other endpoint is a T-node). Let l_b be the complete line forming the body of a T, let t be the terminal endpoint of l_b, and let s be the T-node. Further, let l_h be the complete line forming the head of a T.

If δ is a real number greater than 0 and a is a point in the plane, define

$$C(a, \delta) = \{b: \|a - b\| < \delta\}.$$

If A is a set of points in the plane, let

$$C(A, \delta) = \{b: \|a - b\| < \delta, \text{some } a \in A\}.$$

Theorem 6.

$$C(t, \|s - t\|) \cap C(l_h, \|s - t\|) \cap V = \varnothing.$$

Proof. Suppose to the contrary that $\alpha \in C(t, \|s - t\|) \cap C(l_h, \|s - t\|) \cap V$. Removing l_b from the Steiner minimal tree T breaks T into two components, T_t and T_s. Component T_t is the subtree containing t, and component T_s is the subtree containing s. If α is in T_t, then connecting α to its closest point in l_h gives a Steiner tree for V with length less than a Steiner minimal tree for V, a contradiction. On the other hand, if α is in T_s, then adding the edge between α and t also produces a Steiner minimal tree for V with length less than a Steiner minimal tree. ■

Now let s_1 and s_2 be two adjacent Steiner points.

Theorem 7.

$$C(s_1, \|s_1 - s_2\|) \cap C(s_2, \|s_1 - s_2\|) \cap V = \varnothing.$$

Proof. Similar to the proof of Theorem 6. ■

There is also an additional, straightforward test for candidate full sets with respect to V: A terminal set cannot be a full set if there is some Steiner tree with length less than the length of a shortest tree having the form given in Theorem 5.

5.1.3. Empirical Results

We have empirically studied the effectiveness of these tests on uniformly distributed input terminals in the unit square. Seven tests were devised in Sections 5.1.1 and 5.1.2:

1. The test in Theorem 3, relating the length of a longest edge on the path between two terminals in a Steiner minimal tree to the length of a longest edge on the path between the same two terminals in a minimum spanning tree.
2. The test in Corollary 2. This can be applied to pairs of points to give a lower bound on the size of a prospective backbone; see below.
3. The test in Theorem 4, applicable to triples of points. If $\{a, b, c\}$ is a full set with respect to V, then the circumrectangle of $\{a, b, c\}$ must be empty.
4. The test in Theorem 5, specifying permissible forms of full topologies.
5. The test in Theorem 6, stating that no terminal can be simultaneously "close" to the terminal endpoint of the body of a T and the head of a T.
6. The test in Theorem 7, stating that no terminal can be closer to each of two adjacent Steiner points than they are to each other.
7. The test stating that no full set can have a shorter Steiner tree than the shortest corresponding full topology.

Some care must be used in applying these tests and in determining the order in which these tests should be applied. There are 2^n subsets of the n terminals, many of which may be full sets. Certain tests may be able to rule out a number of sets from candidacy as full sets. It is essential that some tests discard several sets from full set candidacy; otherwise, the time complexity of Step 1 would be prohibitively high. We therefore divide Step 1 into two substeps: the *coarse tests* and the *set-specific tests.*

We first describe the coarse tests. Let the complete corner described in Theorem 5 be called the *backbone.* The endpoints of each backbone are terminals, say a and b; in this case, we speak of the backbone $\widehat{ab}$. We try to identify the candidate full sets associated with a particular backbone $\widehat{ab}$. To this end, we apply tests to determine lower and upper bounds on the number of points that can appear in a full topology with backbone $\widehat{ab}$. We use Tests 2, 3, 5, and 6 to provide lower bounds on the number of Steiner points in backbone $\widehat{ab}$, and we use Tests 4, 5, and 6 to provide upper bounds on the number of Steiner points.

Specifically, Test 2 is applied first. For each backbone $\widehat{ab}$, the quantity

$$\frac{\|a - b\|}{\delta_{MST}(a, b)} - 1$$

is computed; this is a lower bound on the number of Steiner points that must be associated with $\widehat{ab}$. The next test applied is Test 3 on triples of points. Any backbone $\widehat{ab}$ for which the lower bound on the number of Steiner points is 1 is checked to see if there is a third point that can be in an empty circumrectangle. If there are empty circumrectangles, at least one of the corresponding full topologies must pass Tests 5 and 6. If there is no empty circumrectangle, or no empty circumrectangle passes Tests 5 and 6, the lower bound is raised to 2.

The upper bounds are computed in the following way. For each backbone $\widehat{ab}$, Hwang's theorem and Theorem

6 are used to identify points that can possibly appear in a full topology with backbone $\widehat{ab}$. Hwang's theorem implies that incident segments must alternate off of the backbone, so this limits the number of possible Steiner points. Test 5 is also used to limit the number of Steiner points, namely, the two Steiner points closest to the corner must obey Theorem 7.

Once these tests are applied to determine upper and lower bounds on the number of Steiner points associated with $\widehat{ab}$, the characterization in Theorem 5 (Test 4) is used to enumerate potential full sets. Note that backbones in which the lower bound exceeds the upper bound need not be considered. To eliminate the possibility that some full topologies are enumerated several times, only two backbone orientations are considered. In both of these orientations, the horizontal complete line lies to the right of the vertical complete line.

We give empirical results on the number of full sets enumerated by the coarse tests as a function of n. For each n, 100 terminal sets were considered. Each terminal was chosen from a 10,000 × 10,000 grid according to the uniform distribution. The results are displayed in Figure 5. Each point represents the average over 100 runs.

We now describe the set-specific tests. Given a reasonably small number of candidate full sets, Tests 1, 5, 6, and 7 can be run on each set, drastically reducing the number of candidate full sets. Tests 1, 5, and 6 are all straightforward to apply to a given set and backbone. To make these tests more effective, we also apply the tests to the full topology resulting from a "corner flip," a transformation described in Richards and Salowe [23]. After the corner is flipped, it is fruitful to test whether Hwang's theorem is still satisfied. If not, the set can be eliminated from full set candidacy.

Test 7 is applied in the following way: Among all known heuristics, Kahng and Robins' 1-Steiner [17] seems to have the best performance, and it is very simple to implement. Therefore, we compared the length of the full topology to the length obtained by the 1-Steiner procedure. If the length obtained by 1-Steiner is shorter than the length of the prospective full topology, the set is eliminated from full set candidacy. In Section 5.2.2, we provide evidence that 1-Steiner is, on average, about 0.5% longer than the length of a rectilinear Steiner minimal tree for small-sized point sets. This supports our use of 1-Steiner in Test 7.

We obtained statistics based on the set-specific tests, using the same point sets as in the coarse tests. The results are displayed in Figure 6. Again, each point represents the average over 100 runs.

The empirical results indicate that our step 1 is comparable to Winter's stage 1 in performance, based on limited data appearing in Cockayne and Hewgill [5]. The

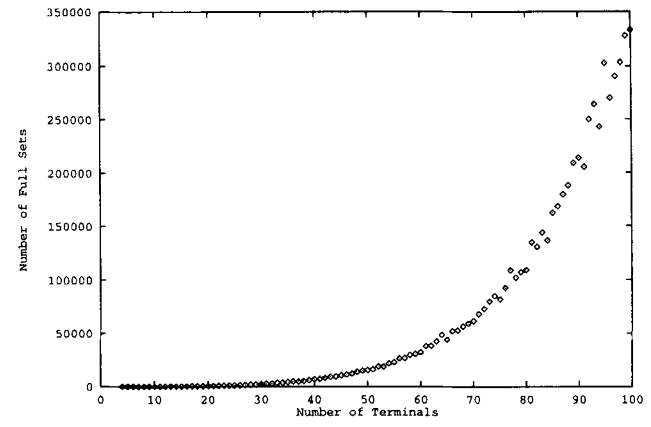

Fig. 5. The number of full sets surviving coarse tests as a function of the number of input terminals. Each point represents the average over 100 runs.

Cockayne and Hewgill data state that the number of Euclidean candidate full sets for 32 and 100 input terminals is 68 and 220, respectively. We also plot the running time of step 1 as a function of the number of input terminals. This plot appears in Figure 7.

5.1.4. A Probabilistic Analysis of the Number of Candidate Full Sets

In this section, we attempt to gain some insight into the performance of the algorithms described above. Recall that a candidate full set with respect to V of size 2 must be an edge in a minimum spanning tree. If the input terminals are uniformly generated in the unit square, the number of edges is $n - 1$.

The data in Figure 6 indicate that, over the range tested, the expected number of full sets surviving the set-specific tests is linear in the number of terminals. In this section, we provide upper bounds on the expected number of full sets. We show that the expected number of full sets of size k, k a constant, is $O(n^2)$. We also show that the expected number of full sets of size 3 is $O(n \log n)$, and we show that if $k = \Omega(n)$ the expected number of full sets is $O(1)$. Ganley and Cohoon [10] showed that the maximum number of full sets of all sizes is $O((1 + \phi)^n)$, where ϕ is the golden ratio.

Theorem 4 has an impact on the number of candidate full sets with respect to V of size 3. There is a close relationship between the number of empty circumrectangles and the number of *direct dominances* in a set of terminals in the plane. A point $z_1 = (x_1, y_1)$ is said to *dominate* $z_2 = (x_2, y_2)$ if $x_1 \geq x_2$ and $y_1 \geq y_2$. Point z_1 is said to directly dominate z_2 if z_1 dominates z_2, and there is no other point z_3 such that z_3 dominates z_2 and z_1 dominates z_3.

Given a set V of n terminals in the plane, let $D(V)$ be the number of direct dominances, where the point set V

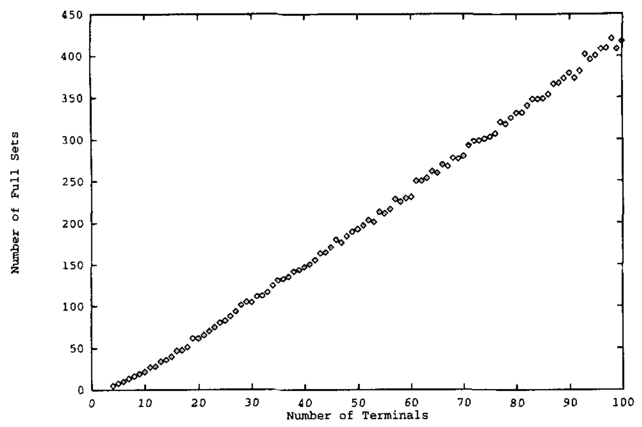

Fig. 6. The number of full sets surviving set-specific tests as a function of the number of input terminals. Each point represents the average over 100 runs.

is rotated either 0, 90, 180, or 270 degrees so that it has the most direct dominances.

Theorem 8. *The number of empty circumrectangles in V is $O(D(V))$.*

Proof. We count the number of points where the lower left corner of the circumrectangle is a terminal. Let the circumrectangle of $\{a, b, c\}$ be empty. Without loss of generality, assume that a is the lower left corner, b is on the top side, and c is on the right side. Then, b and c must directly dominate a. Assign a charge of 1 to the direct dominance pair (a, b), where b is the point on the top side. We claim that no other distinct circumrectangles are charged to (a, b).

Suppose some other rectangle was charged to (a, b). Then, the third point d, on the right side of the circumrectangle, must either be to the left of c, in which case the circumrectangle of $\{a, b, c\}$ is not empty; to the right of c, in which case the circumrectangle of $\{a, b, d\}$ is not empty, or on the same vertical line as c, in which case the circumrectangles are identical. ■

In the worst case, the number of direct dominances is $O(n^2)$, but a consequence of Bentley et al. [2] (see also Güting et al. [12]) is that the expected number of direct dominances for n uniformly distributed points in the unit square is $O(n \log n)$. We can use the algorithm of Güting et al. to find these rectangles in $O(n \log n + m)$ time, where m is the number of direct dominances.

Corollary 4. *Given a set V of n terminals uniformly distributed in the unit square, the expected number of candidate full sets for V of size 3 is $O(n \log n)$.*

We now consider candidate full sets of small size.

Theorem 9. *Let k be a constant. Given a set V of n terminals uniformly distributed in the unit square, the expected number of candidate full sets for V of size k is $O(n^2)$.*

Proof. Let K be a k-set. We bound the probability $P[K]$ that K is a candidate full set for V of size k. The expected number E of candidate full sets is $\binom{n}{k}P[K]$ by linearity of expectation.

To bound $P[K]$, we bound the probability that two terminals in K are rectilinear distance x apart, and then we apply Theorems 6 and 7. Choose an arbitrary terminal $\zeta \in K$. The probability that the furthest terminal from point ζ is between x and $x + dx$ away is at most $(k - 1)2^{2k-1}x^{2k-3}dx$. To obtain this formula, let X_i represent the distance from terminal i to terminal ζ. Then, $P[\max_{1\le i\le k-1} X_i \le x] = P[X_1 \le x]^{(k-1)}$, and $P[X_1 \le x] \le (2x)^2$. Given that the furthest terminal i from terminal ζ is a distance x away, there must be a path involving ζ and i containing at most k links. Theorems 6 and 7 imply that an area of at least $\frac{1}{2}k(x/k)^2$ must be devoid of terminals. The probability that K is a full set with respect to V therefore obeys

$$\begin{aligned}
P[K] &\le \int_0^1 (k-1)2^{2k-1}x^{2k-3}\left(1 - \frac{x^2}{2k}\right)^{n-k} dx \\
&\le (k-1)2^{2k-2}\int_0^1 x^{2k-3}\left(1 - \frac{x^2}{2k}\right)^{n-k} dx \\
&\le (k-1)2^{2k-2}\int_0^1 x^{2k-3}e^{-x^2(n-k)/2k}\,dx \\
&\le (k-1)2^{2k-2}\int_0^\infty x^{2k-3}e^{-x^2(n-k)/2k}\,dx \\
&\le (k-1)2^{2k-2}\frac{(k-2)!}{2\left(\dfrac{n-k}{2k}\right)^{k-2}}.
\end{aligned}$$

This implies that

$$\begin{aligned}
E &= \binom{n}{k}P[K] \\
&\le \frac{n^k}{k!}(k-1)2^{2k-2}\frac{(k-2)!}{2\left(\dfrac{n-k}{2k}\right)^{k-2}} \\
&\le \frac{n^k}{k!}\frac{2^{2k-2}(k-1)!(2k)^{k-2}}{(n-k)^{k-2}} \\
&\le \frac{n^k}{(n-k)^{k-2}}2^{3k-4}k^{k-3} \\
&= O(n^2).
\end{aligned}$$

■

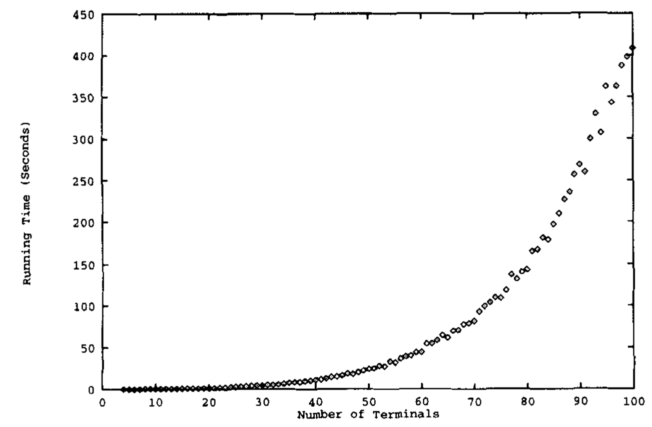

Fig. 7. Running time of Step 1 as a function of the number of input terminals. Each point represents the average over 100 runs.

We now turn our attention to full sets of large size. An interesting corollary to Theorems 5 and 6 is the following result on the maximum length of a Steiner minimal tree of a full set of size k.

Corollary 5. *Let R be the length of the circumrectangle of a full set V of size n. Then, the length of a Steiner minimal tree for V is $O(R \log n)$.*

Proof. Among the cases described in Theorem 5, suppose that all Steiner points lie on a single vertical complete line b, incident to the bottommost terminal. The other cases are similar. Call the n segments incident to b $a_1, a_2, \cdots a_n$, including possibly zero-length segments incident to b's, endpoints. Then, the length $length(T)$ of the Steiner minimal tree T is

$$length(T) = length(b) + \sum_{i=1}^{n} length(a_i).$$

Consider the segments $\lambda_1, \lambda_2, \ldots, \lambda_m$ to the left of b, where the segments are indexed by a decreasing y-coordinate. Let b_i be the portion of b between $\lambda_i \cap b$ and $\lambda_{i+1} \cap b$, $1 \leq i \leq m - 1$. We claim that $length(b_i) \geq \min\{length(\lambda_i), length(\lambda_{i+1})\}$. If this is not the case, then $length(b_i) < length(\lambda_i)$ and $length(b_i) < length(\lambda_{i+1})$. Suppose without loss of generality that $length(\lambda_i) \geq length(\lambda_{i+1})$. Then, the terminal endpoint of λ_i could connect to λ_{i+1} to give a shorter tree.

Let $m \geq 3$. We can consider two groupings of the λ_i into pairs. First, we have the "odd" pairing, where λ_{i+1} is paired with λ_i if $\lceil i + 1/2 \rceil = \lceil i/2 \rceil$. Second, we have the "even" pairing, where λ_{i+1} is paired with λ_i if $\lfloor i + 1/2 \rfloor = \lfloor i/2 \rfloor$. Let B_o be the sum of the lengths of the b_i's located between odd pairs, and let B_e be the sum of the lengths of the b_i's located between the even pairs. By the observation above, the sum of the lengths of the *segment minima* in the "odd" pairing (the shorter of each pair) is less than B_o and the sum of the lengths of the minima in the "even" pairing is less than B_e.

If $m \geq 3$, one of B_o and B_e is at most $[length(b)]/2$, say B_e. Remove the segment minima from the even pairs; the sum of the lengths of these segments is at most $[length(b)]/2$, and at least $\lfloor m - 1/2 \rfloor$ of the λ_i are removed. This process is repeated with the remaining edges. Therefore, the length $\tau(m, \lambda_{\max})$ of the horizontal edges to the left of b is bounded by the following recurrence, where $\lambda_{\max}$ is a longest λ_i to the left of b:

$$\tau(1, length(\lambda_{\max})) = length(\lambda_{\max})$$

$$\tau(2, length(\lambda_{\max})) \leq length(\lambda_{\max}) + length(b)$$

$$\tau(m, length(\lambda_{\max})) < \frac{length(b)}{2} + \tau\left(\left\lceil \frac{m+1}{2} \right\rceil, length(\lambda_{\max})\right), \quad m \geq 3.$$

This recurrence is satisfied by

$$\tau(m, length(\lambda_{\max})) < length(\lambda_{\max}) + \frac{length(b)}{2}(\lfloor \log(m-2) \rfloor + 3), \quad m \geq 3.$$

In the full topology being considered, m is either $\lceil n/2 \rceil$ or $\lfloor n/2 \rfloor$. We therefore have

$$length(T) < length(b) + \tau\left(\left\lceil \frac{n}{2} \right\rceil, length(\lambda_{m_1})\right) + \tau\left(\left\lceil \frac{n}{2} \right\rceil, length(\lambda_{m_2})\right),$$

where λ_{m_1} is a longest edge on the left of b and λ_{m_2} is a longest edge on the right of b.

The bound follows from the observation that

$$length(b) + length(\lambda_{m_1}) + length(\lambda_{m_2}) = \frac{R}{2}. \quad \blacksquare$$

It is interesting to compare Corollary 5 to results about a longest Steiner minimal tree in the unit square, which state that a longest Steiner minimal tree for n terminals in the unit square has length $\sqrt{n} + O(1)$ [3]. In fact, Komlós and Shing [18] showed that the length of a Steiner minimal tree for n uniformly distributed points in the unit square is at least $\sqrt{n}/5$ with probability $1 - o(1)$.

These results make the appearance of large-cardinality candidate full sets improbable, as the following theorem indicates:

Theorem 10. *Let $k = \Omega(n)$. Given a set V of n terminals uniformly distributed in the unit square, the expected number of candidate full sets for V of size k is $O(1)$.*

Proof. We use Corollary 5. A full set of size k in the unit square has length at most $\log k$. We consider the probability $P[K]$ that k randomly generated points K have this length. The expected number E of full sets is $\binom{n}{k}P[K]$.

We consider only vertical backbones here; horizontal backbones are handled in a similar manner. Consider a randomly chosen k-set, generated by a uniform distribution in the unit square. We bound the probability that there is a vertical full topology with length at most $\log k$.

Let A_i be the event that there is a vertical full topology with length at most $\log k$ whose vertical backbone passes through terminal i. We bound the probability $P[K] \le P[A_1 \cup A_2 \cup \cdots \cup A_k] \le \sum_{i=1}^{k} P[A_i] = kP[A_1]$. To calculate $P[A_1]$, consider a vertical strip of width 2ω centered at the x-coordinate of terminal 1. Then, event A_1 only happens if $k - 1 - [(\log k)/\omega]$ or more of the terminals lie inside the strip. If fewer terminals lie inside the strip, then more than $(\log k)/\omega$ terminals lie outside the strip, and the length of the resulting full topology would be greater than $\log k$.

We bound the probability p that $\alpha(k) = k - 1 - [(\log k)/\omega]$ or more terminals lie inside the strip surrounding terminal 1. Let $p[x]$ be the probability that $\alpha(k)$ or more terminals lie inside the strip, given that the x-coordinate of terminal 1 is x. Then, $p = \int_0^1 p[x]dx \le p[1/2]$. Probability $p[1/2] = \sum_{i=\alpha(k)}^{k-1} \binom{k-1}{i}(2\omega)^i(1 - 2\omega)^{k-1-i}$. We bound this probability with the well-known Chernoff bound, which states that $p[1/2] \le ([2e(k - 1)\omega]/[\alpha(k) - 2(k - 1)\omega])^{\alpha(k)-2(k-1)\omega}$.

We see that

$$E \le \binom{n}{k} k \left(\frac{2e(k-1)\omega}{\alpha(k) - 2(k-1)\omega} \right)^{\alpha(k)-2(k-1)\omega}$$

If $k = \Omega(n)$, then there is a choice for ω such that $E = O(1)$. The details are left to the inspired reader. ∎

5.2. Step 2: Backtrack Search

We now describe the second part of the rectilinear Steiner minimal tree algorithm: the backtrack search. The input is the collection of full sets determined in Step 1. The output is a rectilinear Steiner minimal tree. We give a number of backtrack search procedures. The first procedure is a straightforward rectilinear analog of the work of Cockayne and Hewgill [4, 5]. We give a brief overview of the Cockayne and Hewgill procedure, elaborating on methods and tests specific to rectilinear Steiner minimal trees. Empirical results are summarized in Section 5.2.2.

We then consider a second backtrack search strategy called "most-promising-first." In this strategy, we more carefully consider the searching order. Details appear in Section 5.2.3, and empirical results appear in Section 5.2.4.

Finally, in Section 5.2.5, we give a more powerful graph decomposition theorem than that used by Cockayne and Hewgill. Its use in the Steiner minimal tree algorithm will be the subject of future work.

5.2.1. Basic Algorithm Description

To find a Steiner minimal tree, the candidate full topologies must be combined. The resulting object must contain all terminals in V, and it must be of shortest length. To search for a shortest-length tree, a graph $F = (X, A)$ is formed. The set X consists of the candidate full sets remaining after Step 1. The set A consists of pairs of candidate full sets that share a common vertex and may appear simultaneously in a rectilinear Steiner minimal tree.

Let two full sets x and y be *compatible* if they share a common vertex and can appear simultaneously in some Steiner minimal tree. If x and y share a common vertex but cannot appear simultaneously, they are said to be *incompatible*. If x and y do not share a common vertex, they are *disjoint*. The determination of compatibility is an important contribution of Cockayne and Hewgill [5]. They use compatibility to eliminate some full sets and to organize the backtrack search. The number of full sets remaining after compatibility is determined is pictured in Figure 8. The plots in Figures 6 and 8 are superimposed in Figure 9.

Starting with a particular $x \in X$, we use F to find a shortest-length tree containing x. (To find a Steiner minimal tree, we try all choices of x containing a particular vertex.) At any given time, we consider a partial solution containing at least one candidate full set; suppose that $y \in X$ was the last candidate full set added. We try to extend the partial solution with candidate full sets z that are compatible with y. These full sets are the neighbors of y in F. If z is incompatible with any full set in the partial solution, it is rejected. Otherwise, z is either disjoint or compatible with all other full sets in the partial solution. If z is compatible with some other full set w in the partial solution, then w, y, and z must intersect in a single, common terminal. If it is possible to extend the partial solution to also include z, that solution is stored and considered at a later time. More precisely, the solutions are stored in a stack. Some care must be taken so that partial solutions with w, y, and z are considered at most once; also, representation of the stack is important since the number of possible solutions may be large. For brevity, these details are omitted.

Partial solutions are only expanded if they do not exceed the shortest-known Steiner tree in length. We use the 1-Steiner heuristic to initialize this value. If the length of the partial solution is shorter than is the shortest-known Steiner tree, it is tested to determine if it contains all terminals in V. If so, it is retained as the shortest solution. Otherwise, it is expanded in the manner described above.

Central to this procedure is the classification of pairs of full sets as compatible, incompatible, or disjoint. Our initial tests of compatibility are the following, assuming that trees x and y share some common terminals:

1. How many terminals are in both x and y? If the answer is more than one, x and y are incompatible.
2. Do x and y overlap or intersect in either their original or their "corner flip" orientations? If so, they are incompatible.
3. Consider an embedding of x and y. Is it possible to add an edge, thereby creating a cycle, where the added edge is not a longest one on the cycle? If so, x and y are incompatible.

After the initial classification, we use the tests devised by Cockayne and Hewgill [5] to change some of the labels and possibly remove some full sets from candidacy.

Prior to the backtrack search, but after removal of some of the full sets using the classifications above, decomposition theorems are used. These decomposition theorems reduce a Steiner minimal tree problem to several smaller Steiner minimal tree problems. We exploit two decomposition theorems, each based on the composition of the

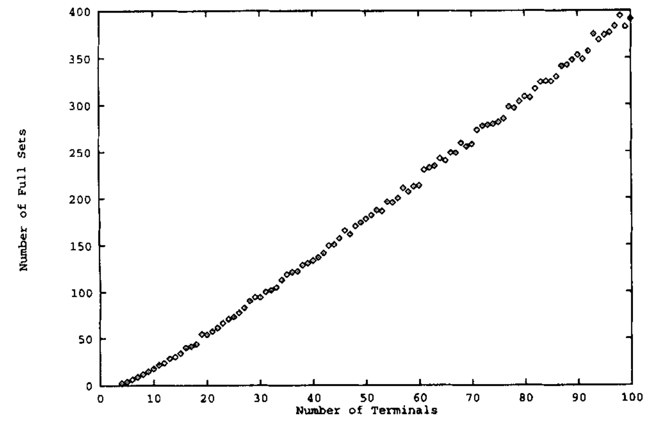

Fig. 8. Number of full sets surviving compatibility tests as a function of the number of input terminals. Each point represents the average over 100 runs.

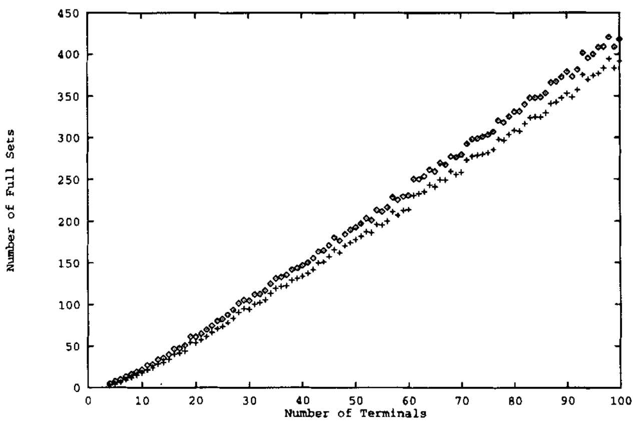

Fig. 9. Number of surviving full sets before and after compatibility tests. Each point represents the average over 100 runs. The number of full sets before the compatibility tests is denoted by a diamond, and the number after the tests is denoted by a cross.

candidate full sets. Consider the graph $H = (V, B)$, where V is the input terminal set, and edge (x, y) is in B if x and y are both in some candidate full set. If H has an articulation point, it can be split into two subproblems at that point. (This is Theorem 2.2 in Cockayne and Hewgill [4].) If H is biconnected, but two vertices can be removed to separate the graph, the set of points can again be split, and several subproblems result. (This is their Theorem 2.3; see Section 5.2.5 for details.)

5.2.2. Empirical Results of Backtrack Search

We tested our rectilinear Steiner minimal tree program, written in C, on point sets of size 5–33, 100 sets each. Each terminal was chosen from a 10,000 × 10,000 grid according to a uniform distribution. In each test, the following information was collected: (1) the number of candidate full topologies surviving the coarse tests, (2) the number of candidate full topologies surviving the set-specific tests, (3) the length of a minimum spanning tree of the input terminals, (4) the length determined by the 1-Steiner heuristic, (5) the length of the tree computed by the program, and (6) the running time, in seconds, on a 48 Meg Sun SPARCstation IPC. We include tables (see Table I) of the following statistics, each a function of the number of input terminals: (1) the average ratio of the length of a minimum spanning tree to the length of a Steiner minimal tree, (2) the average ratio of the length of the 1-Steiner heuristic to the length of a Steiner minimal tree, and (3) the average running time. A graphical plot of the running time, where the time is plotted on a logarithmic scale, is given in Figure 10.

We make three observations: First, the 1-Steiner heuristic is very close in length to the length of a Steiner

TABLE I. Ratios and running times for backtrack search, averaged over 100 runs

	n				
	5	6	7	8	9
MST/SMT	1.106	1.117	1.131	1.117	1.123
1st/SMT	1.000	1.003	1.003	1.004	1.004
Avg. time	0.1	0.1	0.1	0.1	0.2
	10	11	12	13	14
MST/SMT	1.123	1.114	1.116	1.119	1.117
1st/SMT	1.003	1.004	1.004	1.003	1.003
Avg. time	0.3	0.4	0.5	0.6	0.8
	15	16	17	18	19
MST/SMT	1.120	1.127	1.126	1.118	1.125
1st/SMT	1.003	1.006	1.004	1.005	1.006
Avg. time	1.0	1.3	1.6	1.9	2.9
	20	21	22	23	24
MST/SMT	1.129	1.122	1.121	1.124	1.128
1st/SMT	1.004	1.004	1.005	1.004	1.006
Avg. time	4.0	6.3	12.3	20.4	48.2
	25	26	27	28	29
MST/SMT	1.124	1.125	1.120	1.125	1.121
1st/SMT	1.005	1.006	1.004	1.006	1.005
Avg. time	66.2	136	163	380	1640
	30	31	32	33	
MST/SMT	1.125	1.128	1.125	1.126	
1st/SMT	1.005	1.005	1.005	1.006	
Avg. time	2210	3920	6530	13,600	

minimal tree over the range tested. Second, the mean running time is deceptively slow because it is sensitive to outliers. In Table II, we plot the median running times in addition to the mean running times. Finally, we note that our program does not exhibit the spectacular performance of EDSTEINER89 [5]. This result was unexpected because we attempted to incorporate the methodology of EDSTEINER89 into our program.

To be more specific about the last point, Cockayne and Hewgill [5] ran 10 randomly generated problems each of sizes 10, 15, 20, . . . , 100 and found that 181 completed within 20 h on a Sun 3/60; the smallest unfinished point set had size 45. We found, however, that our program was unlikely to solve 50-point problems within a 12 h time bound on a 48 Meg Sun SPARCstation IPC. We attempted 30 problems, and only three finished within the required time bound. In addition, the median running times show the same exponential growth as that of the mean running times. They are both plotted in Figure 11, where the time is placed on a logarithmic scale. A 16 Meg Sun 3/60 is roughly 2.5 to 3 times slower on the same code (though the binaries are different).

5.2.3. The Most-Promising-First Search Strategy

A backtrack search can sometimes be accelerated by scheduling candidate full sets more carefully. The backtrack search scheme described in Section 5.2.1 uses the following methodology: At any point in time, there is a partial solution, which is then extended by adding a compatible full set. Rather than selecting a full set arbitrarily, we select a "promising" full set. We now give a more formal measure of a full set's promise.

We first define the complexity $m(T, X)$ of a set of full sets X with respect to a partial solution T and biconnected component decompositions (a similar measure can be defined with respect to other decompositions; we omit the details). Recall the graph $H = (V, B)$ defined above; edge (x, y) is in B if x and y are both in some candidate full set. We define a new graph $H(T, X)$, where the vertex set consists of those vertices that are not in T and a supervertex corresponding to tree T, and (x, y) is an edge if x and y are in a candidate full set in X. The measure $m(T, X)$ is the number of vertices in a largest biconnected component in the graph $H(T, X)$.

Recall that the original set of full sets is X, and no full sets are in the initial solution. We maintain a triple of full sets, $(\hat{T}, Y, Z)$, where $\hat{T}$ are the full sets in the partial solution, Y are full sets that are compatible with $\hat{T}$, and Z are "forbidden" full sets, $X = \hat{T} \cup Y \cup Z$. Throughout the computation, we assume that there is a Steiner minimal tree T that contains $\hat{T}$ and does not contain any full sets in Z. A set $F \in Y$ is either in T or not in T. If F is in T, then $\hat{T}$ may be extended to include F; the updated

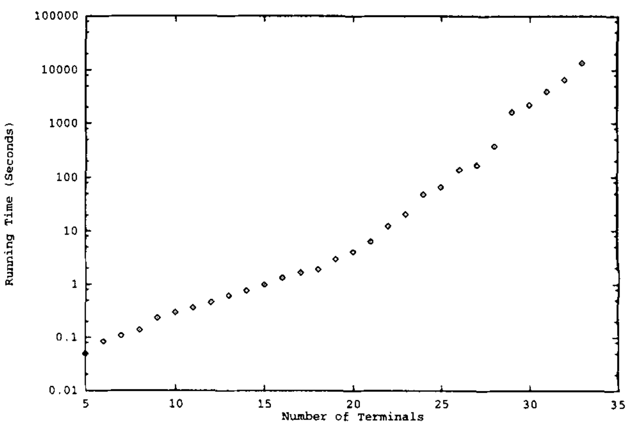

Fig. 10. Running time of backtrack search plotted on a logarithmic scale.

TABLE II. Median running time of backtrack search

	n				
	5	6	7	8	9
Mean	0.1	0.1	0.1	0.1	0.2
Median	0.1	0.1	0.1	0.1	0.2
	10	11	12	13	14
Mean	0.3	0.4	0.5	0.6	0.8
Median	0.3	0.3	0.4	0.6	0.7
	15	16	17	18	19
Mean	1.0	1.3	1.6	1.9	2.9
Median	0.9	1.1	1.4	1.3	1.9
	20	21	22	23	24
Mean	4.0	6.3	12.3	20.4	48.2
Median	2.5	2.6	4.3	10.5	15.9
	25	26	27	28	29
Mean	66.2	136	163	380	1640
Median	20.7	32.2	25.6	60.2	94.1
	30	31	32	33	
Mean	2210	3920	6530	13,600	
Median	171	415	1000	1290	

triple is $(\hat{T} \cup F, Y - F - K, Z \cup K)$, where K is the set of full sets in Y that are incompatible with $\hat{T} \cup F$. If F is not in T, the updated triple is $(\hat{T}, Y - F, Z \cup F)$.

We define a *most promising full set* to be a full set F that minimizes $m(\hat{T} \cup F, Y - F - K)$. The most-promising-first strategy selects the most promising full set as the next one to consider. It spawns two new subproblems, each of which may be recursively subdivided.

We speculate that the most-promising-first strategy can be improved by more carefully scheduling full sets. The definition of promise above is asymmetric—one of the two subproblems is considered in the measure of promise, while the other is not. This represents a greedy approach to a more general scheduling problem, to determine what sequence of full sets will minimize the size of the largest recursively called subproblem. We will study this scheduling problem in future work.

5.2.4. Empirical Study of Most-Promising-First

We repeated the tests in Section 5.2.2. The mean running times are plotted on a logarithmic scale in Figure 12. The mean and median running times are given in Table III, and the plots in Figures 10 and 12 are superimposed in Figure 13. The most-promising-first strategy has little effect on problems of size less than 25, but it seems to facilitate the search when there are between 30 and 35 terminals, roughly halving the search time.

5.2.5. A New Decomposition Theorem

Recall the graph $H = (V, B)$ constructed to state Cockayne and Hewgill's decomposition theorems. We now describe a partitioning scheme that generalizes these theorems. We plan to incorporate these ideas into our program in future work.

Let $H = (V, B)$ be a k-connected graph, i.e., there is a set of k vertices that, when removed, disconnect H, but there is no such set of $k - 1$ vertices. Let X be a set of k vertices whose removal disconnects G into nonempty graphs $\hat{H}_1 = (\hat{V}_1, \hat{B}_1)$ and $\hat{H}_2 = (\hat{V}_2, \hat{B}_2)$; X is called a *vertex separator*. Let $\hat{A}_1$ be the set of edges in B connecting vertices in $\hat{V}_1$ with X. Finally, let $H_1 = (V_1, B_1) = (\hat{V}_1 \cup X, \hat{B}_1 \cup \hat{A}_1)$ and $H_2 = (V_2, B_2) = (\hat{V}_2 \cup X, B - (\hat{B}_1 \cup \hat{A}_1))$.

Let T be a rectilinear Steiner minimal tree for V. The subgraph of T restricted to vertices in V_1 is a forest, as is the subgraph of T restricted to vertices in V_2; call these forests T_1 and T_2, respectively. Note that the candidate full sets can be partitioned into two groups: one group corresponding to a subgraph of H_1 and the other corresponding to a subgraph of H_2.

To explain how to compute these forests, let $\equiv$ be an equivalence relation on X. A *rectilinear Steiner minimal forest for V with respect to* $\equiv$ is a shortest rectilinear Steiner forest $\hat{T}$ for V having the following properties: First, distinct terminals $x \in X$ and $y \in X$ satisfying $x \equiv y$ cannot be in the same connected component in $\hat{T}$. Second, there must be a path in $\hat{T}$ from each vertex in $V - X$ to some vertex in X. Finally, if x and y are not in the same equiv-

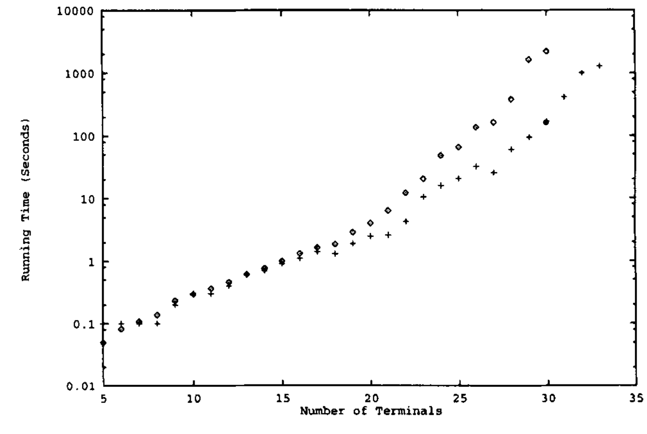

Fig. 11. Mean and median running times of backtrack search, plotted on a logarithmic scale. The mean time is denoted by a diamond, and the median time is denoted by a cross.

TABLE III. Mean and median running times for most promising-first

	n				
	5	6	7	8	9
Mean	0.1	0.1	0.1	0.2	0.2
Median	0.1	0.1	0.1	0.1	0.2
	10	11	12	13	14
Mean	0.3	0.4	0.4	0.6	0.7
Median	0.3	0.3	0.4	0.5	0.5
	15	16	17	18	19
Mean	0.7	1.0	1.5	2.0	2.6
Median	0.6	0.9	1.0	1.3	1.6
	20	21	22	23	24
Mean	3.8	7.0	6.3	16.8	33.7
Median	2.2	3.7	3.7	6.0	6.2
	25	26	27	28	29
Mean	90.2	153	217	177	645
Median	12.5	16.9	42.2	25.5	71.1
	30	31	32	33	34
Mean	1160	1790	3720	5470	
Median	123	230	383	866	

alence class, there must be some terminals $w \in X$ and $z \in X$ such that $w \equiv x$, $z \equiv y$, and there is a path from w to z.

Given T_1, let $x \in X$ and $y \in X$ satisfy $x \equiv_1 y$ if and only if x and y are in the same connected component in T_1; similarly, define $\equiv_2$ for T_2.

We can use our rectilinear Steiner minimal tree algorithm to compute a rectilinear Steiner minimal forest for V_1 with respect to $\equiv$ for some choices of $\equiv$. In the backtracking stage of our algorithm, any set $S \subseteq X$ of points can be interconnected by an imaginary full topology $T_I(S)$ of length zero. In fact, imaginary full topologies for a partition $C = \{S_1, S_2, \cdots\}$ of X can also be constructed. The Steiner minimal tree calculation for V_1 is restricted to candidate full sets corresponding to H_1. If the Steiner minimal tree calculation succeeds in interconnecting all of V_1, the forest $T_1(C)$ formed by removing all $T_I(S_i)$ is a rectilinear Steiner minimal forest for V_1 with respect to the equivalence relation corresponding to C.

Lemma 1. *There is a partition C of X such that length(T_1) = length$(T_1(C))$. Similarly, there is a partition C' of X such that length(T_2) = length$(T_2(C'))$.*

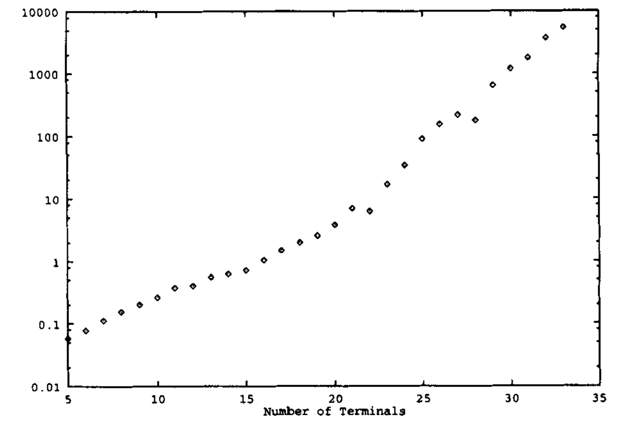

Fig. 12. Running time of most-promising-first search plotted on a logarithmic scale.

Proof. Let C be the partition induced by $\equiv_2$. Then, the union of T_1 and all the $T_I(S)$, $S \in C$, is a tree on V_1 with length identical to the length of T_1. Furthermore, all full sets in T_1 correspond to ones in H_1.

Suppose to the contrary that $T(C)$ is shorter than is T_1. Then, the union of $T(C)$ and T_2 is a rectilinear Steiner tree for V with length less than the length of T, a contradiction. ■

To construct a rectilinear Steiner minimal tree, the following procedure is performed: For each partition C of X, $T_1(C)$ is found using the candidate full sets corresponding to H_1. Given $T_1(C)$, the connected components are found, and the resulting equivalence relation $\equiv_1$ on X is determined. Then, $T_2(C')$ is computed using the candidate full sets corresponding to H_2, where C' is the partition induced by $\equiv_1$. A rectilinear Steiner minimal

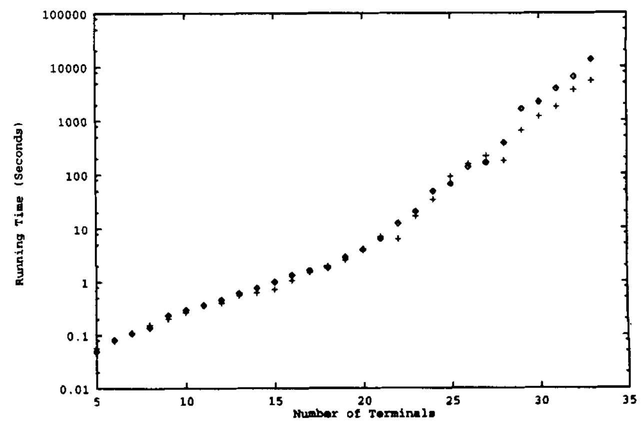

Fig. 13. Running time of the two algorithms, plotted on a logarithmic scale. The diamond represents the original version of backtrack search, and the cross represents the version performing the most-promising-first search.

tree can be found by minimizing the length of the union of $T_1(C)$ and $T_2(C')$ over all choices of C. Correctness follows from Lemma 1 and its proof.

Theorem 11. *Let X be a vertex partition, and let T be a rectilinear Steiner minimal tree. Then, there is a partition C of X such that length$(T_1(C) \cup T_2(C'))$ = length(T), where C' is the partition induced by $\equiv_1$.*

The efficacy of this partition depends on two factors: (1) the time complexity to find rectilinear Steiner minimal forests with respect to $\equiv$ for V_1 and V_2, and (2) the number of such forest calculations. The first criterion dictates that V_1 and V_2 should be about the same size, while the second criterion dictates that the cardinality of X should be small. To be more specific, the number of distinct equivalence classes for a set of k elements is $\sum_{i=1}^{k} S_{k,i}$, where $S_{k,i}$ is a Stirling number of the second kind. If k = 2, 3, 4, 5, 6, 7, and 8, the number of distinct equivalence classes is 2, 5, 15, 52, 203, 877, and 4140, respectively.

Even and Tarjan [8] described an algorithm to determine graph connectivity. If there are n vertices and m edges, the algorithm takes $O(\sqrt{n}m^2)$ time. For this application, however, it would be prudent if the two resulting subproblems were split as evenly as possible. An *edge separator* of a graph is a set of edges that, when deleted, disconnects a graph $H = (V, B)$ into two subgraphs, $H_1 = (V_1, B_1)$ and $H_2 = (V_2, B_2)$ such that $|V_1| \geq |V|/3$ and $|V_2| \geq |V|/3$. A *vertex separator,* a set of vertices whose deletion disconnects the graph, can be constructed from an edge separator by computing a vertex cover of the edges in the edge separator.

Some recent work has focused on the construction of edge separators [19, 22]. We plan to investigate this more extensively.

6. REMARKS ON CORRECTNESS

Correctness of the implementation is a crucial component of this work. It is possible that a purported Steiner minimal tree algorithm occasionally delivers suboptimal trees; such errors are difficult for humans and computers to detect, especially when no competitor programs exist. Such errors destroy the validity of timing and length analyses.

The implementation of Ganley and Cohoon [9] seems to be the fastest existing alternative, so a direct comparison between the two programs is possible for less than 20 terminals. We, however, compared our computed trees to trees constructed by the 1-Steiner procedure. (The Ganley and Cohoon work followed ours.) In several thousand tests, our program never produced a tree with length greater than the length of a tree discovered by the 1-Steiner heuristic. The 1-Steiner heuristic was chosen because it appears to generate shorter trees than does any other known approximation algorithm.

7. CONCLUSIONS

One of the most important questions raised by this work is why results as spectacular as those of Cockayne and Hewgill were not obtained. There are several possible explanations: either one or both programs are in error, some procedure crucial to the Euclidean algorithm was omitted in the rectilinear version, the rectilinear algorithm does not eliminate enough full sets from candidacy, or there is a fundamental difference between the Euclidean and rectilinear problems.

We deal with correctness first, as this seems to be the most likely explanation. There appears to be a problem in the implementation of Theorem 2.3 in [4]; we do not know if some resulting error appears in the code. If the code actually reflects the asserted implementation, then, based on the nature of the problem, proposed solutions for small terminal sets may be optimal, but proposed solutions for larger terminal sets may be suboptimal. This would make the error undetectable if the new program was checked against an old Euclidean Steiner minimal tree program.

Theorem 2.3 is stated as follows: Let $V_0, \ldots, V_{k-1}$ be subsets of V such that (i) every candidate full set is completely contained in some V_i, (ii) $V_i \cap V_j = \emptyset$ if $i \neq j + 1$ and $i \neq j - 1$, and (iii) $|V_i \cap V_{i+1}| = 1$, $0 \leq i < k$. (Index arithmetic is done modulo k.) Then, any Steiner minimal tree for V contains a Steiner minimal tree for $k - 1$ of the sets $V_0, \ldots, V_{k-1}$.

Suppose that $H = (V, B)$ is a graph with no articulation points. Cockayne and Hewgill stated that a decomposition given by Theorem 2.3 can be discovered by finding a vertex $a \in V$ such that the removal of a reduces the connectivity of H from 2 to 1. They stated that the required sets are the biconnected components of the reduced graph $H - \{a\}$, with vertex a added to the "appropriate components." (The graph $H - \{a\}$ is not formally defined in the Cockayne and Hewgill paper, but it is reasonable to assume it is the graph resulting from the deletion of a and all its incident edges from H.)

The suggested implementation is underspecified, as shown in Figure 14. In Figure 14, the graph H is biconnected, and the removal of a induces a graph that contains 2 articulation points. Consequently, there are 3 components in $H - \{a\}$. If a is added back to each of these components, the resulting decomposition does not satisfy Theorem 2.3. The set of "appropriate components" needs to be specified.

To implement Theorem 2.3, one needs to find a pair of vertices such that their removal disconnects H. In prin-

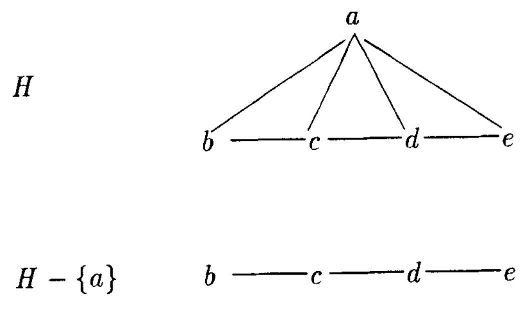

Fig. 14. Graphs H and $H - \{a\}$.

ciple, this could be done using the triconnected component algorithm of Hopcroft and Tarjan [14]. In practice, one can delete a vertex and find the biconnected components of the resulting graph, since the running time of this procedure is small compared to the backtrack time.

Once these vertices are found, the graph is split into two components, H_1 and H_2. The biconnected components of H_1 and H_2 are the sets required in Theorem 2.3. A definitive implementation would find a pair of vertices whose deletion disconnects H and for which the size of a largest biconnected component in H_1 and H_2 is minimized.

The second possible explanation for the discrepancy between the Euclidean and rectilinear results is the omission of one or more important details from the rectilinear version. We tried to incorporate all of the ideas given by Cockayne and Hewgill [4, 5] into our program.

The third explanation is the difference in the number of candidate Euclidean full sets and candidate rectilinear full sets. Winter's stage 1 appears to generate fewer candidate full sets than does ours, based on limited data appearing in Cockayne and Hewgill [5]. Nevertheless, the empirical expected number of candidate full sets for 50-point rectilinear problems is fewer than the expected number of candidate full sets for 100-point Euclidean problems. The resulting graph H would likely be denser in the rectilinear problem than in the Euclidean problem, possibly affecting decomposition theorems. On the other hand, the pruning techniques appearing in Cockayne and Hewgill [5] should be more effective on denser graphs than on sparser graphs, as there are more possible pairs to prune.

The final explanation, a fundamental difference between the rectilinear and Euclidean problems, merits further study. Perhaps the Euclidean problem is more amenable to decomposition than is the rectilinear problem, or perhaps the Euclidean Step 1 is more effective than the rectilinear Step 1 in constructing decomposable problems. At this point, we do not feel that this explanation is as likely as is programming error. Our algorithm was programmed two separate times, once in Pascal and once in C, and it was programmed by two different people. The Pascal version was programmed in a series of stages. We found that the version of our program that was analogous to Winter's performed in a manner similar to Winter's. We found that the inclusion of the ideas incorporated into EDSTEINER86 made modest improvements. We did not, however, obtain drastic improvements with the methods of EDSTEINER89. It is fair to note, however, that the Pascal version contained a number of bugs. The C version increased the range of the Pascal version by approximately 8 points. Its performance does not seem to match EDSTEINER89, either.

There are many additional problems that should be considered: First, more stringent conditions on full set candidacy are needed. Second, a strong theoretical result on the expected number of candidate full sets seems imminent and worthy of attention. Third, graph decomposition theorems should be studied and included in future Steiner tree programs. Fourth, the scheduling problem in the most-promising-first search needs to be formalized and studied. Finally, we believe that the results of Cockayne and Hewgill and our own need to be independently evaluated to resolve their discrepancy.

We thank J. Ganley for enlightening discussions and for reading various drafts of the manuscript. We also thank a referee who clarified the statement of Theorem 3 and a referee who supplied a simple proof of that result.

APPENDIX: A PERTURBATION SCHEME FOR NONDISTINCT INPUTS

In this appendix, we explain how to ensure that no two input terminals share the same x- or y-coordinates. We outline a perturbation scheme that makes all coordinates distinct and allows straightforward construction of a Steiner minimal tree for the original terminal set, given a Steiner minimal tree for the perturbed terminal set. Details on placing the perturbed set onto the integer grid are left to the reader.

Recall that the original terminals are on the integer grid, and consider a small perturbation ϵ (this is simulated by a magnification). The value of ϵ is the maximum perturbation for a given coordinate—all points are perturbed a slightly different amount to ensure that there are no degeneracies. Let T_1 be an Steiner minimal tree for the original (unperturbed) problem, and let T_2 be a Steiner minimal tree for the perturbed problem. We choose ϵ to be sufficiently small to satisfy $5n\epsilon < \frac{1}{2}$, where n is the number of input terminals.

Consider the relationship between T_1 and T_2. Then,

$$length(T_2) \leq length(T_1) + 2n\epsilon,$$

because it is possible to form a Steiner tree on the perturbed terminal set by modifying T_1 to include an edge between the old and new locations of the terminals.

On the other hand, consider T_2, and remember that ϵ is small. Since the Steiner minimal tree must lie on the grid graph, any edge approaching a terminal must be long compared to ϵ—it must have length at least $1 - 2\epsilon$. At most, one edge can "approach" a terminal v in a given direction, so there are at most 4 edges incident to the radius ϵ box surrounding terminal v. Some part of the Steiner minimal tree lies inside this box. Now, if v is moved back to its original location, and each of these incident edges is connected to v, the length of L_2 increases by at most 5ϵ. The tree T_3 constructed by this procedure satisfies

$$length(T_3) \leq length(T_2) + 5n\epsilon.$$

Now, T_3 is a Steiner tree for the original set of points and T_1 is a Steiner minimal tree for the original set of points, so

$$length(T_1) \leq length(T_3),$$

and we get

$$\begin{aligned} length(T_2) - 2n\epsilon \leq length(T_1) &\leq length(T_3) \\ &\leq length(T_2) + 5n\epsilon. \end{aligned}$$

Recall that $5n\epsilon < \frac{1}{2}$, so we have

$$\begin{aligned} length(T_2) - \tfrac{1}{2} < length(T_1) &\leq length(T_3) \\ &< length(T_2) + \tfrac{1}{2}. \end{aligned}$$

Both $length(T_1)$ and $length(T_3)$ are integers, since the original set of terminals lies on the integer grid. This implies that $length(T_1)$ and $length(T_3)$ represent the same integer, so both T_1 and T_3 are Steiner minimal trees.

REFERENCES

[1] P. K. Agarwal and M. T. Shing, Algorithms for the special cases of rectilinear Steiner trees: I. Points on the boundary of a rectilinear rectangle, *Networks* **20** (1990) 453–485.

[2] J. L. Bentley, H. T. Kung, M. Schkolnick, and C. D. Thompson, On the average number of maxima in a set of vectors and applications, *J. ACM* **25** (1978) 536–543.

[3] F. R. K. Chung and R. L. Graham, On Steiner trees for bounded sets. *Geomet. Dedicata* **11** (1981) 353–361.

[4] E. J. Cockayne and D. E. Hewgill, Exact computation of Steiner minimal trees in the plane. *Inf. Proc. Lett.* **22** (1986) 151–156.

[5] E. J. Cockayne and D. E. Hewgill, Improved computation of plane Steiner minimal trees. *Algorithmica* 7 (1992) 219–229.

[6] J. P. Cohoon, D. S. Richards, and J. S. Salowe, An optimal Steiner tree routing algorithm for a net whose terminals lie on the perimeter of a rectangle. *IEEE Trans. Comp.-Aided Design* **9** (1990) 179–189.

[7] L. L. Deneen, G. M. Shute, and C. D. Thomborson, A probably fast, provably optimal algorithm for rectilinear Steiner trees. *Random Struct. Algor.* **5** (1994) 535–557.

[8] S. Even and R. E. Tarjan, Network flow and testing graph connectivity. *SIAM J. Comput.* **4** (1975) 507–518.

[9] J. L. Ganley and J. P. Cohoon, A faster dynamic programming algorithm for exact rectilinear Steiner minimal trees. *Fourth Great Lakes Symposium on VLSI* (1994) 238–241.

[10] J. L. Ganley and J. P. Cohoon, Optimal rectilinear Steiner minimal trees in $O(n^2 2.62^n)$ time. *Proc. 6th Can. Conf. Comput. Geom.* (1994) 308–313.

[11] M. Garey and D. S. Johnson, The rectilinear Steiner problem is NP-complete. *SIAM J. Appl. Math.* **32** (1977) 826–834.

[12] R.-H. Güting, O. Nurmi, and T. Ottmann, Fast algorithms for direct enclosures and direct dominances. *J. Algor.* **10** (1989) 170–186.

[13] M. Hanan, On Steiner's problem with rectilinear distance. *SIAM J. Appl. Math.* **14** (1966) 255–265.

[14] J. E. Hopcroft and R. E. Tarjan, Dividing a graph into triconnected components. *SIAM J. Comput.* **2** (1973) 135–158.

[15] F. K. Hwang, On Steiner minimal trees with rectilinear distance. *SIAM J. Appl. Math.* **30** (1976) 104–114.

[16] F. K. Hwang and D. S. Richards, Steiner tree problems. *Networks* **22** (1992) 55–89.

[17] A. B. Kahng and G. Robins, A new class of iterative Steiner tree heuristics with good performance. *IEEE Trans. Comp.-Aided Design* **11** (1992) 893–902.

[18] J. Komlós and M. T. Shing, Probabilistic partitioning algorithms for the rectilinear Steiner problem. *Networks* **15** (1985) 413–423.

[19] T. Leighton and S. Rao, An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms. *Proc. 29th IEEE Symp. Found. Comput. Sci.* (1988) 422–431.

[20] F. D. Lewis, W. C. Pong, and N. Van Cleave, Optimum Steiner tree generation. *Second Great Lakes Symposium on VLSI* (1992) 207–212.

[21] Z. A. Melzak, On the problem of Steiner. *Can. Math. Bull.* **4** (1961) 143–148.

[22] D. A. Plaisted, A heuristic algorithm for small separators in arbitrary graphs. *SIAM J. Comput.* **19** (1990) 267–280.

[23] D. S. Richards and J. S. Salowe, A linear-time algorithm to construct a rectilinear Steiner minimal tree for k-extremal point sets. *Algorithmica* 7 (1992) 247–276.

[24] G. M. Shute, A reduced grid for rectilinear Steiner minimal trees. *Proc. 2nd Can. Conf. Comput. Geom.* (1990) 309–314.

[25] A. F. Sidorenko, Minimal rectilinear Steiner trees. *Diskret. Mat.* **1** (1989) 28–37.

[26] C. D. Thomborson, L. L. Deneen, and G. M. Shute, Computing a rectilinear Steiner minimal tree in $O(n^{O(\sqrt{n})})$ time. *Parallel Algor. Architect.* (1987) 176–183.

[27] C. D. Thomborson, B. Alpern, and L. Carter, Rectilinear Steiner tree minimization on a workstation, in Computational Support for Discrete Mathematics, DIMACS Series in *Discrete Mathematics and Theoretical Computer Science,* **15,** N. Dean and G. E. Shannon (eds), American Mathematical Society (1994) 119–136.

[28] P. Winter, An algorithm for the Steiner problem in the Euclidean plane. *Networks* **15** (1985) 323–345.

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

[30] Y. Y. Yang and O. Wing, Optimal and suboptimal solution algorithms for the wiring problem. *Proc. IEEE Int. Symp. Circuit Theory* (1972) 154–158.

Received January 1993
Accepted November 1994