VLSI DESIGN
1998, Vol. 7, No. 1, pp. 15–30
Reprints available directly from the publisher
Photocopying permitted by license only

On Rectilinear Distance-Preserving Trees*

GUSTAVO E. TÉLLEZ and MAJID SARRAFZADEH†

Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL 60208

Given a set of terminals on the plane $N=\{s, v_1,\ldots,v_n\}$, with a source terminal s, a Rectilinear Distance-Preserving Tree (RDPT) $T(V, E)$ is defined as a tree rooted at s, connecting all terminals in N. An RDPT has the property that the length of every source to sink path is equal to the rectilinear distance between that source and sink. A Min-Cost Rectilinear Distance-Preserving Tree (MRDPT) minimizes the total wire length while maintaining minimal source to sink linear delay, making it suitable for high performance interconnect applications.

This paper studies problems in the construction of RDPTs, including the following contributions. A new exact algorithm for a restricted version of the problem in one quadrant with $O(n^2)$ time complexity is proposed. A novel heuristic algorithm, which uses optimally solvable sub-problems, is proposed for the problem in a single quadrant. The average and worst-case time complexity for the proposed heuristic algorithm are $O(n^{3/2})$ and $O(n^3)$, respectively. A 2-approximation of the quadrant merging problem is proposed. The proposed algorithm has time complexity $O(\alpha^2 T(n)+\alpha^3)$ for any constant $\alpha > 1$, where $T(n)$ is the time complexity of the solution of the RDPT problem on one quadrant. This result improves over the best previous quadrant merging solution which has $O(n^2 T(n)+n^3)$ time complexity.

We test our algorithms on randomly uniform point sets and compare our heuristic RDPT construction against a Minimum Cost Rectilinear Steiner (MRST) tree approximation algorithm. Our results show that RDPTs are competitive with Steiner trees in total wire-length when the number of terminals is less than 32. This result makes RDPTs suitable for VLSI routing applications. We also compare our algorithm to the Rao-Shor RDPT approximation algorithm obtaining improvements of up to 10% in total wirelength. These comparisons show that the algorithms proposed herein produce promising results.

Keywords: Timing-driven routing, rectilinear steiner arborescence, distance-preserving trees, routing, layout

1. INTRODUCTION

With recent progress in VLSI technology, interconnection delay has become increasingly significant in determining the circuit speed. Recent studies show that interconnection delay contributes up to 70% of the clock cycle in the design of high performance circuits [7, 15]. Thus, with

*A preliminary version of this paper appeared in the proceedings of ISCAS-95. "On Rectilinear Distance Preserving Trees", *Proc. of the IEEE International Symposium on Circuits and Systems*, Vol. 1, 163–166, April 30, 1995. © 1995 IEEE. Reprinted with permission.

†Corresponding author.

submicron device dimensions and up to a million transistors integrated on a single microprocessor, on-chip and chip-to-chip interconnections play a major role in determining the performance of digital systems.

Therefore, performance-driven routing has received increasing attention in the past several years. For a given signal net the goal of performance-driven routing is to route a net while minimizing average or maximum source to sink delay. Early work on routing focused on minimizing total net length using minimum Steiner tree approximations [9, 11]. More recently, a routing heuristic in [4] simultaneously considered both the cost C (total edge length) and the radius R (longest source to sink path length) of a routing tree. A more general formulation is given in [5] where a parameter ε guides the trade-off between cost and radius minimization. In this approach the Bounded-Radius, Bounded Cost (BBRC) algorithm produces a solution with radius $(1+\varepsilon)R$ and cost $(1+2/\varepsilon)C$. Other methods of obtaining cost-radius trade-offs are given in [1]. The cost-radius trade-off used by the BBRC method can also be viewed as one between minimum-cost Steiner trees (MST) and shortest-path tree (SPT) constructions. Using this approach, [2] proposed the AHHK algorithm which achieves a direct MST-SPT trade-off. For examples of these tree constructions see Figure 1.

In path-oriented timing-driven routing, we encounter the problem of routing nets with critical sinks. For these problems, the primary objective is to minimize the delay to the critical sinks of the net. For this type of routing one would like to use SPTs to construct the connections to the critical sinks and MSTs to construct the remaining connections, thus the AHHK approaches are preferable in this setting. In these algorithms the computation of the SPTs has mostly been overlooked since in general there are few critical sinks. However, in special nets, such as clocks, the number of critical sinks is large, hence the SPT problem becomes important. Furthermore, SPTs are also important because they minimize the objectives of path length and total tree cost simultaneously.

In this paper we focus on the problem of constructing *Min-Cost Rectilinear Distance-Preserving trees* (*MRDPT*), which are also called *Min-Cost Shortest-path Steiner trees* (*MSPST*) and *Min-Cost Rectilinear Steiner Arborescences* (*MRSA*). It is known that the Min-Cost Distance Preserving tree problem in general graphs and in several special case graphs is NP-Hard [3]. The

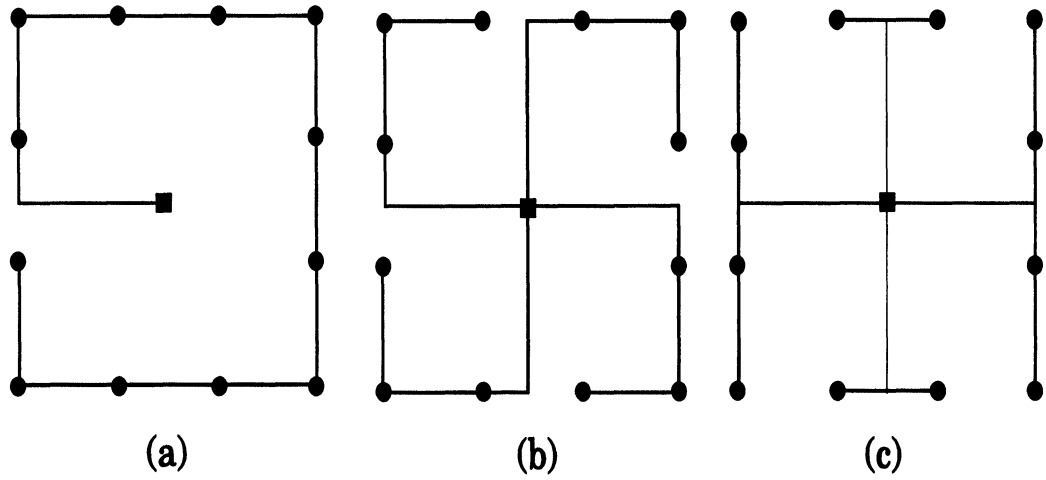

FIGURE 1 Three examples of Steiner trees on the plane. The trees shown are a) a Minimum Steiner Tree, b) Bounded Radius, Bounded Cost Tree and c) a Min-Cost Rectilinear Distance Preserving Tree.

complexity of the MRDPT problem is not known but it is conjectured to be NP-Hard [10]. Exponential time algorithms for MRDPT and several special case polynomial-time algorithms have been proposed in [10]. In [14] Rao *et al.*, propose the Rectilinear Steiner Arborescence algorithm (RSA), which is to date the best known approximation algorithm. The cost performance of the RSA algorithm is shown to be $C(RDPT)/C(MRDPT) \leq 2$. In [6] the A-tree algorithm is proposed, which is an enhancement to the RSA algorithm. Finally, an Iterated 1-Arborescence (I1ARB) algorithm is proposed in [12]. All of these algorithms are reported to have similar worst case and average cost performances.

In this paper we propose a heuristic algorithm (so-called *pRDPT*) that computes an RDPT by partitioning the problem into sub-problems which we can then solve optimally in polynomial time. In Section 2 we present some definitions and the formulation of the problem. In Section 3 we summarize our heuristic algorithm, and describe our partitioning of the problem. We then proceed to characterize the MRDPT problem in Section 4. Once characterized, we present two MRDPT problems that are solvable in polynomial time: namely the single layer, single source MRDPT problem in Section 5, and the single layer, multiple source problem in Section 6. In Section 7 we propose our heuristic algorithm which uses the two previous algorithms to produce a good RDPT solution in one quadrant. We then use another heuristic algorithm in Section 8 to produce the full RDPT tree. Finally in Sections 9 and 10 we present our experimental results and conclusions.

2. FORMULATION

A net is defined by a set of terminals $N=\{s, v_1,\ldots,v_n\}$, with a *source* node s, where $N \subset \Re^2$, $\Re$ is the set of real numbers. Let the coordinates of each terminal be denoted by $v_i=(x_i, y_i)$ then the rectilinear distance between two terminals v_i, v_j is $\|v_i, v_j\| = |x_i - x_j| + |y_i - y_j|$.

Given a tree $T(N)=G_T(V, E)$, on a net N, we denote a path between two vertices u, $v \in V$ as $u \rightsquigarrow v$, the length of a path between these two vertices as $\ell(u \rightsquigarrow v)$ and the shortest path between these two vertices as $u \overset{*}{\rightsquigarrow} v$. A *Rectilinear Distance Preserving Tree* (RDPT) $T(N)$ is a tree rooted on s, with $N \subseteq V$, and $\ell(s \overset{*}{\rightsquigarrow} v_i) = \|s, v_i\|$. An edge $e_{ij} \in E$ connects a parent node v_i and a child node v_j. The set of vertices $u \in V-N$ are called *Steiner vertices*.

The total length of a tree T, denoted $C(T)$, is defined as $C(T) = \Sigma_{e_{ij} \in E} \|v_i, v_j\|$. An RDPT T^* is said to be a *Min-Cost Rectilinear Distance Preserving Tree* (MRDPT) if for any RDPT T', $C(T') \geq C(T^*)$. We can now formulate the MRDPT problem.

Min-Cost Rectilinear Distance Preserving Tree Problem (MRDPT)

Given a signal net $N=\{s, v_1, \ldots, v_n\} \subset \Re^2$, with a source s, construct a routing tree $T(N)=G_T(V, E)$ such that $N \subseteq V, \ell(s \overset{*}{\rightsquigarrow} v_i) = \|s, v_i\|$ and $C(T)$ is minimized.

3. SUMMARY OF APPROACH

In this section we outline the heuristic algorithm pRDPT we propose to solve the MRDPT problem. Before we outline the algorithm some definitions are necessary.

First we define a partition of the plane based on the source terminal s, located at (x_s, y_s). Using a vertical axis at $x=x_s$ and a horizontal axis at $y=y_s$, partition the plane into four quadrants: $Q_0=\{(x, y) | (x \geq x_s) \text{ and } (y \geq y_s)\}$, $Q_1=\{(x, y) | (x \leq x_s) \text{ and } (y \geq y_s)\}$, $Q_2=\{(x, y) | (x \leq x_s) \text{ and } (y \leq y_s)\}$, and $Q_3=\{(x, y) | (x \geq x_s) \text{ and } (y \geq y_s)\}$. Next we define a partition of the terminals in a quadrant. Without loss of generality, we assume that we are dealing with terminals in quadrant Q_0 and that $s=(0, 0)$. Given two terminals u and v, u is said to *dominate* v, denoted $u \succ v$, if $x_u > x_v$ and $y_u > y_v$. Similarly, u and v are said to be *independent* if u does not dominate v and v does not dominate u, denoted $u \not\succ v$. A *layer* L of terminals is a set of

independent terminals, i.e., $L=\{v_i \in N | \forall v_i, v_j \in L, v_i \not\prec v_j\}$. An example of a quadrant and layer partition is shown in Figure 2.

The pRDPT algorithm consists of the following steps:

1. Partition the plane into quadrants $Q_i, i=0,\ldots,3$. The partitioning of the plane divides the terminals into one-quadrant MRDPT problems (1Q-MRDPT). Points on an axis will appear in the 1Q-MRDPT problems of the adjacent quadrants. Thus given the quadrant partitions we define corresponding nets N_i for each quadrant.
2. Solve the 1Q-MRDPT problem for each quadrant, obtaining a tree $T_i(N_i)$ for quadrant i as follows:
   (a) Partition the points into dominance layers $L_i, i=0,\ldots,k$.
   (b) Traverse the layers in depth-first order, solving the 1-layer, k-source MRDPT problem ($(1,k)$-MRDPT) which consists of computing an MRDPT that connects one layer of points to the next.
3. Merge the solutions for each quadrant, thus obtaining the RDPT $T=\bigcup_{i=0}^{3} T_i(N_i)$.

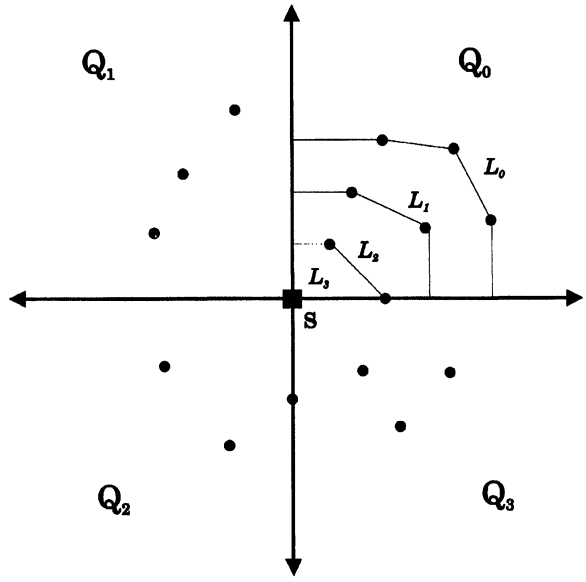

FIGURE 2 Sample partitioning of an MRDPT problem. Points are divided into four quadrants $Q_0,\ldots,Q_3$. The points in quadradrant Q_0 are partitioned into dominance layers L_0, $\ldots,L_3$.

The formulations for the 1Q-MRDPT, and $(1,k)$-MRDPT problems will be given in the following sections. Also, the algorithms that compute the $(1,k)$-MRDPT and the RDPT merge are presented.

4. CHARACTERIZATION OF MRDPT

In this section we simplify the MRDPT problem. We will show that the MRDPT problem can be partitioned into four problems in each quadrant as described in the previous section. Furthermore we will show that the minimum cost solution can be obtained in polynomial time by merging the solutions in each quadrant. We use some of the results from [10] in this section.

LEMMA 4.1 *A Steiner vertex $u \in V$ in an MRDPT $T(V, E)$ satisfies* (1) v_v, $v_h \succ u$ *and* (2) $u=(x_v, y_h)$, *where* $v_v, v_h \in N$.

Proof (1) must be satisfied to guarantee that $v_v \overset{*}{\rightsquigarrow} s$ and $v_h \overset{*}{\rightsquigarrow} s$ are both shortest paths. Furthermore $v_v \overset{*}{\rightsquigarrow} u$ and $v_h \overset{*}{\rightsquigarrow} u$ must also be shortest paths. The optimality of T and the two previous facts lead to condition (2). □

Note that Lemma 4.1 implies that there exists a grid where possible Steiner vertices must be located. This grid is in fact a subset of the Hanan[1] grid for the general Minimum Rectilinear Steiner tree problem. Furthermore, using the same arguments as in the proof, we can show that if a vertex $v \in N$ also satisfies $v \in Q_i$, then any vertex $u \in v \overset{*}{\rightsquigarrow} s$ must also satisfy $u \in Q_i$. With these two results we have shown that the source to sink paths of an MRDPT in one quadrant do not enter any other quadrant, and at most intersect the paths of

[1]The set of intersection points $H(N)$ that are obtained when horizontal and vertical gridlines are drawn through every point of N is called the Hanan grid. Hanan [8] showed that there exists an RSMT whose Steiner vertices are all chosen from $H(N)$.

adjacent quadrants at the axes. The 1Q-MRDPT problem is then simply the MRDPT problem with the restriction that $s=(0,0)$ and $N \subset Q_0$. A simple transformation on the points on the other quadrants yields this problem.

We can also conclude from this discussion that there must be at most one Steiner vertex per axis, so called *corner vertex* (see Fig. 3). Using Lemma 4.1, the number of possible corner vertices per axis is at most $n\binom{n}{4} = O(n^4)$ possible combinations of corner vertices. Let $\{c_0, c_1, c_2, c_3\}$ denote a set of four corner vertices, one for each axis. We can then compute each 1Q-MRDPT with the two corner vertices on the axes of the quadrant, $T_i(N_i \cup \{c_i, c_{i+1 \bmod 4}\})$ and the resulting RDPT $T = \bigcup_{i=0}^{3} T_i(N_i, \{c_i, c_{i+1 \bmod 4}\})$, will also have minimum cost given the choice of corner vertices. Hence the merge step amounts to selecting the best set of corner vertices. In Section 8 we will improve on the complexity of the merging algorithm.

5. 1-LAYER, 1-SOURCE MRDPT

In this section we consider the problem of computing an MRDPT in one quadrant with one dominating layer. For simplicity we restrict the problem to Q_0. We assume that the terminals are ordered as follows: for any two terminals v_i, v_j with $i<j$, $x_i \leq x_j$ and $y_i \geq y_j$, which is consistent with the definition of a layer (see Fig. 4). We denote the MRDPT $T(\{v_i, \ldots, v_j\})$ with $T_{i,j}$. From [14] the (1,1)-MRDPT problem can be solved using a dynamic programming algorithm based on the following recurrence formula:

$$C(T_{a,b}) = \min_{a<k<b} \{C(T_{a,k}) + C(T_{k+1,b}) + (x_{k+1} - x_a) + (y_k - y_b)\} \quad (1)$$

The resulting algorithm requires $O(n^3)$ time to compute the solution to the (1,1)-MRDPT problem. The complexity of the problem can be reduced as aresult of the following theorem. For each MRDPT $T_{a,b}$, the index k which satisfies Equation (1) is called the branching of $T_{a,b}$.

THEOREM 5.1 *[10] Let l be the branching for $T_{a,b-1}$ and r be the branching for $T_{a+1,b}$ then there exists a branching k for $T_{a,b}$ between l and r, or $l \leq k \leq r$.* □

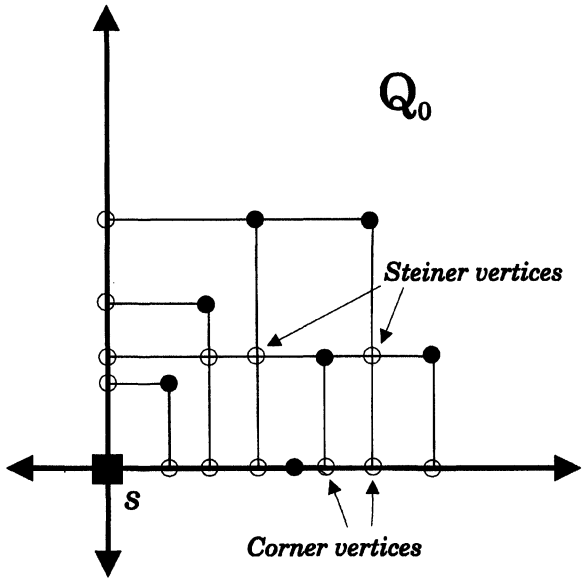

FIGURE 3 One quadrant MRDPT problem with necessary Steiner and corner vertices. Sinks are shown as solid circles.

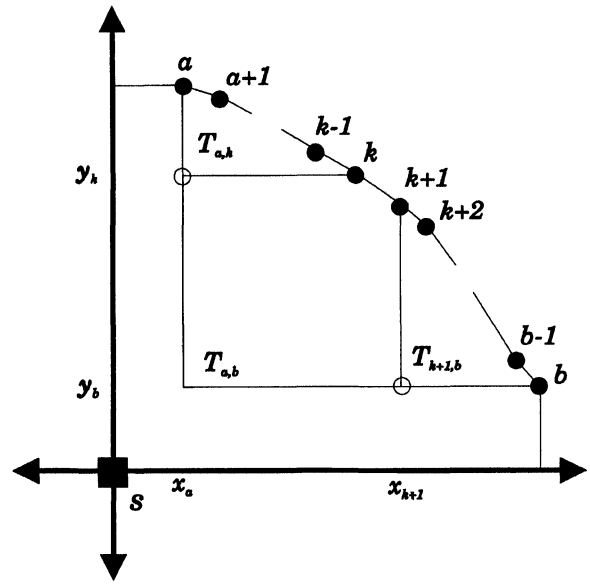

FIGURE 4 Illustration of the (1,1)-MRDPT formulation.

Table I shows the pseudo-code for algorithm (1,1)-MRDPT based on Theorem 5.1. From the theorem and the pseudo-code the complexity of the dynamic programming algorithm is reduced to $O(n^2)$.

6. 1-LAYER, K-SOURCE MRDPT

The problem studied in this section arises from the partitioning of the general MRDPT problem proposed in Section 3. The idea is that we have one layer of sinks, and several sources in a second layer. We would like to connect every sink to the layer of sources with shortest paths and minimum cost. In this problem the layer of sources forms a decreasing staircase boundary (or *source boundary*), and any point on this boundary is considered to be a valid source terminal. For example, see Figure 5. In general the problem can be formulated as follows.

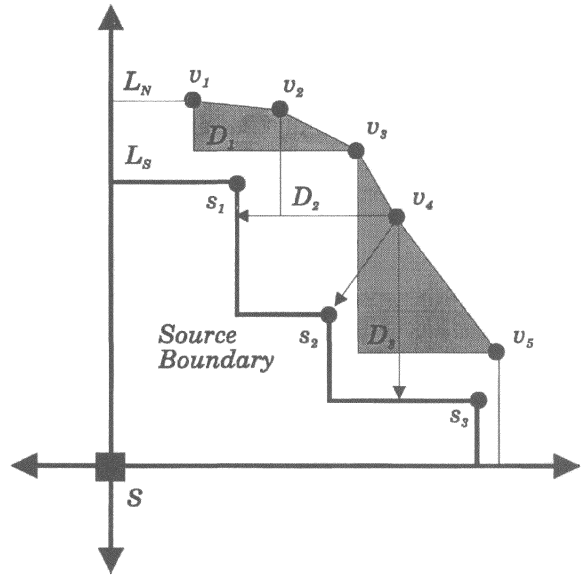

FIGURE 5 Illustration of the (1, k)-MRDPT problem. The sink layer $L_N = \{v_1,\ldots,v_5\}$ and the source layer $L_S = \{s_1,\ldots,s_3\}$. The set of dominance triangles for this problem is $\mathcal{D} = \{D_1,\ldots,D_3\}$. Note that for vertex v_4 three connections are possible: a horizontal connection to the source boundary between s_1 and s_2, a vertical connection to the source boundary between s_2 and s_3, and a connection to the source s_2.

TABLE I Pseudo-code for Algorithm (1,1)-MRDPT. The array $B[i,j]$, $C[i,j]$ and $v_{i,j}$ contain the branching cost and root vertex for the sub-tree $T_{i,j}$, respectively

Algorithm (1,1)-**MRDPT**

Input: N

Output: $T(N)$

```
for i = 1 to n do
  B[i, i]←i;
  C[i, i]←0;
  B[i, i+1]←i;
  C[i, i+1]←‖v_i, v_{i+1}‖;
  v_{i,i+1}←(x_i, y_{i+1});
end for
for ℓ=2 to n−ℓ do
  for a=1 to n−ℓ do
    b←a+ℓ;
    C[a, b]←∞;
    or k=B[a, b] to B[a+1, b+1] do
      C← C[a, k]+C[k+ 1, b]+‖ v_{a, k}, v_{k+1, b}‖;
      if C<C[a, b] then
          C[a, b]←C;
          B[a, b] ← k;
      endif
    endfor
    v_{a,b}← (x_a, y_b);
  endfor
```

1-layer, k-source Min-Cost Rectilinear Distance Preserving Tree Problem((1, k)-MRDPT)

Given a set of sources $S = \{s_1,\ldots,s_k\} \subset \Re^2$ and a set of sinks $N = \{v_1,\ldots,v_n\} \subset \Re^2$ such that $\forall\ s_i$, there exists a v_j, such that $s_i \prec v_j$, and such that N and S form a dominance layer L_S and L_N, respectively, construct a routing forest $F(S,N) = G_F(V,E)$ such that $F, S \subseteq V$, $\ell(u \overset{*}{\rightsquigarrow} v_i) = \|u, v_i\|$, is a point on the decreasing staircase formed by S, and $C(F)$ is minimized.

We extend the (1,1) -MRDPT algorithm to solve this problem. The challenge is to extend the algorithm while maintaining an $O(n^2)$ time complexity. As in the previous section the tree $T_{a,b}$ denotes the MRDPT $T(\{v_a,\ldots,v_b\})$ and $\{v_a,\ldots,v_b\} \subseteq N$. In addition, we denote the shortest connection from the root of a tree $T_{a,b}$ to a point on the boundary S as $R_{a,b}$. The algorithm's extension

consists of the following steps:

1. Compute the costs of every tree $T_{a,b}$ using the (1,1)-MRDPT algorithm.
2. Compute the closest source in S to the root of each tree $T_{a,b}$ (compute $R_{a,b}$).
3. Compute the minimum cost combination of trees to make the $(1,k)$-MRDPT forest using a dynamic programing algorithm.

6.1. Computing Source-Root Connections

We note that the set of possible trees $T_{a,b}$ in the solution of the $(1,k)$-MRDPT problem is a subset of the trees of the (1,1)-MRDPT problem on the layer defined by the sinks N. Thus we need only one invocation of the (1,1)-MRDPT algorithm to compute these trees. In the remainder of this section we consider the problem of finding the set of $R_{a,b}$ connections and the problem of computing the final forest.

We begin by defining some terms. A *dominance triangle* $D=\{r_D, L_D\}$ consists of a layer of terminals $L_D=\{v_i,\ldots,v_j\}$ and a corresponding *origin* r_D. The origin of a dominance triangle, henceforth also called *triangle*, is a Steiner point which dominates all terminals in the triangle's layer. Let the horizontal and vertical axes of the dominance triangle be defined as $y_h=y_i$ and $x_v=x_j$, then the origin is defined as $r_D=(x_v, y_h)$. A dominance triangle is defined to have an empty interior, thus there may not exist a terminal $u\in N$, such that $r_D \prec u \prec v_i$ and $v_i \in L_D$. A *degenerate dominance triangle* is a dominance triangle with a single terminal in its layer. A *maximal dominance triangle* is one in which the addition of any terminal to its layer would cause some terminal to lie in its interior. It can be shown that an MRDPT can be decomposed into a set of dominance triangles. Furthermore, the MRDPT of one dominance triangle can be found with the (1,1)-MRDPT algorithm. A triangle D is said to be *on* layer N, denoted $D\subseteq N$, if $L_D\subseteq N$. Also, let $\mathcal{D}=\{D_i | D_i \subseteq N,\ i=1,\ldots,d\}$ be the set of maximal triangles on layer N.

THEOREM 6.1 *Any necessary Steiner vertex $v_i\in V$ of the $(1, k)$-MRDPT forest $F(S, N)=G_F(V, E)$ must lie inside a triangle $D\in\mathcal{D}$.*

Proof By contradiction. Suppose there exists a necessary Steiner vertex $p\in V$ such that there exist two paths in F, $u \overset{*}{\rightsquigarrow} p$ and $v \overset{*}{\rightsquigarrow} p$, in $F(S,N)$, u, $v\in N$, and such that p does not lie in any $D\in\mathcal{D}$. Since this is a necessary Steiner vertex, by Lemma 4.1 there must exist two terminals $v_i, v_j\in N$ such that $p=(x_i, y_j)$. Define the triangle $D_p=\{p, L_p\}$, $L_p=\{v_i,\ldots,v_j\}$. Since $D_p\notin\mathcal{D}$ there must also exist a source s_p inside D_p. But if $s_p\neq p$ then the paths to p must cross the decreasing staircase of sources contradicting the assumptions of the problem. □

We will next show how to use the set of maximal dominance triangles on layer N to compute the set of connections $R_{a,b}$ for the $(1,k)$-MRDPT problem. A point may connect to the source boundary in one of three ways: via a path to the closest dominated source on the boundary, via a vertical path to a horizontal section on the boundary or via a horizontal path to a vertical section on the boundary. Consider a maximal dominance triangle D on the sinks $\{v_i,\ldots,v_j\}$, and suppose we know the root-source connections for the sides of the dominance triangle: i.e., we know all $R_{i\{i,\ldots,j\}}$ and $R_{\{i,\ldots,j\},j}$. We now show how to compute the remaining root-source connections.

THEOREM 6.2 *Let $R_{a,b+1}$ and $R_{a-1,b}$ be the closest root-source connections for trees $T_{a,b+1}$ and $T_{a-1,b}$ respectively. Let $T_{a,b+1}$ and $T_{a-1,b}$ be in the interior of the same dominance triangle. Then the closest root-source connection for tree $T_{a,b}$ is obtained from* $\min\{C(R_{a,b+1})+(y_{b+1}-y_b),\ C(R_{a-1,b})+(x_a-x_{a-1})\}$.

Proof Let the root of tree $T_{a,b}$ be located at point $p_{a,b}=(x_a, y_b)$. Let $R_{a,b}=s$ be the closest source dominated by $p_{a,b}$. Since s cannot be in the interior of a dominance triangle, then s can be located in one of three places: Case I: $p_{a-1,b}\succ s$ but $p_{a,b+1}\not\succ s$. Case II: $p_{a,b+1}\succ s$ but $p_{a-1,b}\not\succ s$.

Case III: $p_{a,b+1} \succ s$ and $p_{a-1,b} \succ s$. We know that $\|p_{a,b}, s\| = (x_a - x_s) + (y_a - y_s)$. But for Case I. $x_a - x_s = (x_a - x_{a-1}) + (x_{a-1} - x_s)$ which implies that $\| p_{a,b}, s \| = C(R_{a-1,b}) + (x_a - x_{a-1})$ and for Case II. $y_b - y_s = (y_b - y_{b+1}) + (y_{b+1} - y_s)$, which implies that $\|p_{a,b}, s \| = C(R_{a,b+1}) + (y_b - y_{b+1})$. Finally by similar arguments $\|p_{a,b}, s\| = C(R_{a-1,b}) + (x_a - x_{a-1}) = C(R_{a,b+1}) + (y_b - y_{b+1})$. □

Using Theorem 6.2, the root-source connections for all trees in the interior of the maximal dominance triangles can be computed, given that the root-sink connections are known for the sides of the triangle. This computation is done by propagating the root-source connections from trees $T_{i,i+k}$ to trees $T_{i,i+k-1}$ in $O(n^2)$ time.

Now we show how to compute the root-source connections for the sides of each dominance triangle. We use Theorem 6.2 to simplify the problem of computing closest root-source connection as follows:

1. For a root $p_{i,k}$, $k = i, \ldots, j-1$ of a tree on the left side of the dominance triangle, $T_{i,\{i,\ldots,j-1\}}$, we only need to compute the smallest between horizontal distance to the source boundary, and the sources located between y_k and y_{k+1}.
2. For a root $p_{k,j}$, $k = i+1, \ldots, j$ of a tree on the bottom side of the dominance triangle, $T_{\{i+1,\ldots,j\}j}$ we only need to compute the smallest between the vertical distance to the source boundary and the sources located between x_{k-1} and x_k.
3. For the root $p_{i,j}$ of the tree $T_{i,j}$ the smallest between the vertical and horizontal distance to the source boundary, and the sources dominated by $p_{i,j}$.

To see that the above computations are sufficient, consider a root $p_{i,k}$ on the left side of a maximal dominance triangle. Suppose that the closest source s has $y_s < y_{k+1}$. By Theorem 6.2 this source will be the closest source to some root below $p_{i,k}$. Furthermore, by the order of the propagation of root-source connections, s will end up as the closest root-source connection. Similarly, the same occurs with the bottom side of the triangle. Finally, for the corner of the triangle, we compute all possible source-root connections.

As we will see in Section 7 the sinks and maximal dominance triangles in the dominance layers are ordered by x-coordinate. We use this property in an efficient algorithm that computes the root-source connections on the sides of the maximal dominance triangles. We traverse the maximal dominance triangles on a layer in order, visiting the tree roots on the left side, the corner and then the bottom side of the triangle. Simultaneously we advance on the layer of sources as indicated by the conditions outlined previously. As we advance on the roots and sources, we record the root-source connections. As a result a single scan of the triangles and the layer of sources suffices to compute the source-root connections required by the algorithm.

6.2. Computing the Minimum Cost Forest

In previous sections we have shown how to compute the set of trees that can make up the $(1,k)$-MRDPT forest. The problem that remains is to select the trees that belong to the minimum cost forest. We formulate the problem as follows:

$(1,k)$-MRDPT Forest Construction Problem

Given a set of sinks $N = \{v_1, \ldots, v_n\} \subset \Re^2$ such that N forms a dominance layer, and the set of possible MRDPT trees $T_{i,j}$ on N, with connections to a source $R_{i,j}$, where $1 \le i, j \le n$, construct a routing forest $F = \bigcup_{i=0}^{m} T_{k_i, k_{i+1}-1}$, where $k_0 = 1$, $k_m = n$, and $k_i < k_{i+1}$, $i = 0, \ldots, m$, such that $C(F) = \Sigma_{i=0}^{m} C(T_{k_i,k_{i+1}-1})$ is minimized.

In the above formulation, and in the subsequent discussion we assume that the cost of a tree includes the cost of the connection from its root to a source in the source boundary. The computation of these costs for each tree was outlined in the previous sections.

We note that the above formulation requires that the resulting trees be vertex disjoint, that is,

the trees in the forest are not allowed to share sinks. We also observe that the problem amounts to choosing a set of weighted non-intersecting intervals that cover the sinks and minimize the cost function.

THEOREM 6.3 *Given a set of trees $T_{i,j}$ with costs $C(T_{i,j}), 1 \le i \le j \le n$, the set of trees $F=\{T_{k_i,k_{i+1}-1} | k_0 = 1, k_m = n,$ and $k_i < k_{i+1}, i = 0,\ldots,m\}$ that minimizes $C(F) = \Sigma_{i=0}^{m} C(T_{k_i,k_{i+1}-1})$ is given by:*

$$W(k_i) = \min_{i=1,\ldots,k_i} \{W(i) + C(T_{i+1,k_i})\} \qquad (2)$$

and $W(k_m = n)$.

Proof Proof is by induction on the number of sinks, k. The basis step is trivial. The inductive hypothesis is that we know the solutions for all forests of less that k sinks: i.e., we know all $W(i)$, $i=1,\ldots,k-1$. We must show Equation (2) can compute the minimum cost forest for k sinks. The proof is by contradiction. Suppose there exists a solution for $W'(k) = W'(i) + C(T'_{i+1,k}) < W(k)$, where either **a)** $W'(i) < W(i)$, **b)** $C(T'_{i+1,k}) < C(T_{i+1,k})$ or **c)** both. Consider a) $W'(i) < W(i)$, but this is a contradiction to the inductive hypothesis. Consider b) $C(T'_{i+1,k}) < C(T_{i+1,k})$, this is also a contradiction since $T_{i+1,k}$ is an MRDPT. Finally c) must also be a contradiction, for the same reasons. □

As a result of Theorem 6.3 and the other results discussed previously, we propose algorithm $(1,k)$-MRDPT, shown in Table II. This algorithm has a worst case time complexity of $O(n^2)$ which is dominated by the running of the (1,1)-MRDPT algorithm and the forest computation algorithm.

TABLE II Pseudo-code for Algorithm $(1,k)$-**MRDPT**. The array $C[i, j]$ contain the branching cost (1,1)-MRDPT. The array $W[i]$ and $k[i]$ contain the costs of the i^{th} subs-solution and the best sub-tree of the i^{th}sub-solution, respectively

Algorithm (1, *k*)-**MRDPT**

Input: Layers *N* and *S*

Output: *F*(*N*, *S*)

```
Compute (1,1)-MRDPT on N, producting C;
Compute root-source connections;
for i=1 to n do
    W[i]←C[1, i];
    k[i]←0;
    for j= 1 to i− 1 do
      Cost←W[j]+C[j+ 1, i];
      if Cost < W[i] then
          W[i]←Cost;
          k[i]←j;
      endif
    endfor
endfor
```

7. ONE QUADRANT RDPT HEURISTIC

Up to now we discussed the algorithms used to connect two adjacent layers without justifying their approach. We call the MRDPT tree on one quadrant 1Q-MRDPT. In this section we discuss the problems encountered by the layer partition approach we propose. We then complete the description of the 1Q-RDPT algorithm.The objective of partitioning the problem into layers is to separate the problem into independent parts that we can solve with polynomial time algorithms. The main difficulty with the layer partitioning is to model the connections between adjacent layers. Ideally we seek a solution on one layer that minimizes the combined cost of the two layer solution. Therefore, the cost of connecting the subtrees of one layer to the next and the costs of adding new sources to the next layer should be considered simultaneously. But both of these costs depend on the manner in which the second layer isconstructed, thus the problems are not independent. Therefore we separatethe problem as follows:

1. We partition the plane using the decreasing staircases defined by the layers.
2. We connect the sinks in one layer up to the boundary of the next plane partition. This is the $(1,k)$-MRDPT problem.

Since the trees generated by the approach described above are not necessarily connected, and since the cost of the solution in each layer is minimal, the approach computes a forest whose

cost is a lower bound on the cost of the 1Q-MRDPT. To produce a final 1Q-RDPT we maintain two maximal layers. A maximal layer of sinks L_N contains the roots of the current forest, and any additional sinks from N needed to make L_N a maximal layer. The second layer L_S is made up of the outer layer from the remaining sinks, and makes up the source boundary for the $(1,k)$-MRDPT problem. At each iteration we process the trees in the $(1,k)$-MRDPT problem as follows:

- If the root of a tree is connected by the $(1,k)$-MRDPT algorithm to the source boundary, then the connection is discarded and the root becomes a new sink.
- If the root of a tree is connected to a source in the source layer, then this source becomes a sink, provided it does not dominate other sinks in the sink layer. Otherwise, the root is processed as before.

We will next show that at each step this procedure eliminates at least one terminal from the solution, and thus at most $O(n)$ iterations are needed to compute the 1Q-RDPT.

THEOREM 7.1 *There exists a tree $T_{i,j} \in F(S, N)$ in the $(1,k)$-MRDPT solution such that $R_{i,j} = s_k$ and $s_k \not\prec (x, y)$, where $(x, j) \in F(S, N)$, or else there must exist a tree $T_{i,j} \in F(S, N)$ with $i < j$.*

Proof Let $L_S = \{s_1 \ldots, s_m\}$ and $L_N = \{v_i, \ldots, v_n\}$. Every source s_j must be dominated by at least one sink v_k, by the maximality property of the layers. For each sink three connections are possible: a connection to the dominating source, and a horizontal (vertical) connection to a vertical (horizontal) section on the source boundary. Two important exceptions to this rule occur: first a sink may not dominate a source, in which case only a connection to the source boundary is possible, and a sink may dominate source s_1 or s_m in which case only one (not two) connection to the source boundary is possible. Suppose that $n = m$ and that every sink $v_i \succ s_i$. Then the following connections are possible: $v_i \overset{*}{\rightsquigarrow} s_{i-1}$, $v_i \overset{*}{\rightsquigarrow} s_i$ and $v_i \overset{*}{\rightsquigarrow} s_{i+1}$ for $i = 2, \ldots, n-1$. Since v_i does not dominate s_{i-1}, the $v_i \overset{*}{\rightsquigarrow} s_{i+1}$ connection is to the source boundary. A direct connection to the source would not be a contradiction, so these connections are ignored. Hence a total of $2n-2$ connections for n sinks seem possible. Now examine a pair of sinks v_i and v_{i+1} with connections $v_i \overset{*}{\rightsquigarrow} s_{i+1}$ and $v_{i+1} \overset{*}{\rightsquigarrow} s_i$, respectively. These connections physically cross and are thus not allowed. Hence only $n-1$ non-crossing connections are possible thus the solution must contain at least one $v_i \overset{*}{\rightsquigarrow} s_i$ connection proving the theorem for $n = m$. Now consider the case when there are more sources than sinks $(n < m)$. For any one sink more than three connections are possible, however, the number of connections to the source boundary remains unchanged, hence the proof holds. This argument is also true for $n > m$. □

Theorem 7.1 shows that each step of the 1Q-RDPT algorithm removes at least one point from the set of unconnected terminals in N and ads this terminal to the current sub-solution. Thus the 1Q-RDPT completes after $O(n)$ iterations. We now show that the 1Q-RDPT algorithm produces an RDPT. Firstly each sub-solution consists of a forest of RDPTs. Secondly the last step the problem is reduced to a (1,1)-MRDPT problem and since the roots of the trees in the forest are guaranteed to be in the sink layer L_N, then the final solution is also an RDPT. An important detail of the algorithm remains: the computation of the source and sink layers, and the dominance triangles for the sink layer.

The problem of computing all the dominance layers in a quadrant is a well known computational geometry problem [13]. A scan-line algorithm exists that solves this problem in $O(n \log n)$ time. However, the 1Q-RDPT only needs to compute the two outer-most layers, and it may do so $O(n)$ times. We next describe an $O(n)$ time algorithm that computes the two outermost layers. We note that the roots of the (1,1)-MRDPT forest will always be part of the sink layer, thus we assume that we have an initial sink layer available,

L'_N. Furthermore we note that if a terminal is in the source layer, then it will either remain there or be promoted to the sink layer. Similarly, terminals that have not been assigned a layer may at best end up in the source layer. We can use a single procedure to create both layers. This procedure has as input terminals that currently belong to that layer, denoted L' and a list of candidate terminals denoted N. The resulting layer is denoted L. We traverse the terminals in the solution in reverse lexicographic order by (x, y). We maintain the y-coordinate, y_L of the last inserted sink in the layer. For each point $p=(x_p, y_p)$ we perform the following activities:

- Remove all points in L' which come before p in the reverse lexicographic order and insert them into L
- If there are no points in the layer insert p in L. If $y_p < y_L$ then insert p in L. Otherwise leave p in its original layer.

We first apply this algorithm to the L'_N and L_S layers, thus obtaining a maximal L_N layer and a partial L'_S layer. We then apply the same algorithm to the L'_S layer and the set of remaining unassigned terminals N. The result is a maximal layer L_S. To initialize this procedure the first L_N and L_S layers are extracted from N. Note that terminals will go from being unassigned in N, to sources in L_S, to sinks in L_N. The proposed algorithm then consists of a linear pass that visits each terminal in the problem at most twice, thus has $O(n)$ time complexity. Note that the lists must be in lexicographic and remain in lexicographic order through out the process. Thus an initialsorting of the terminals is required.

Once the L_N and L_S layers are constructed, the maximal dominance triangles for the L_N layer are computed. Since the terminals in both layers are ordered in lexicographic order, we traverse the terminals successively computing the vertical and horizontal axes of the dominance triangles. The proposed algorithm creates at most one maximal dominance triangle per sink, thus at most $O(n)$ maximal dominance triangles are created. Furthermore, the creation of all new maximal dominance triangles costs at most $O(n)$ total time. Also as a result of the traversal order, the dominance triangle list is also ordered as expected by the $(1,k)$-MRDPT algorithms.

We now analyze the time complexity of the 1Q-RDPT algorithm. Pseudo-code for the 1Q-RDPT algorithm is given in Table III. The time complexity of the algorithm is dominated by the $(1,k)$-MRDPT algorithm. In particular, the $(1,k)$-MRDPT algorithm contains two steps that have $O(|L_N|^2)$ time complexity. Therefore if the 1Q-RDPT algorithm requires k iterations to complete, the total time complexity is $O(\sum_{i=1}^{k} |L_i|^2)$. Since $k=O(n)$ and $|L_i|=O(n)$, the worst case time complexity of the1Q-RDPT algorithm is $O(n^3)$.

We now consider the average complexity of the 1Q-RDPT algorithm. We assume that an average 1Q-MRDPT problem is represented by a randomly uniform pointset. Thus, the number of layers $k = O(\sqrt{n})$ and also $|L_i| = O(\sqrt{n})$. As a result we expect the average time complexity of the 1Q-RDPT algorithm on randomly uniform pointsets to be $O(n^{\frac{3}{2}})$.

8. MERGING THE QUADRANTS

Now consider an RDPT tree obtained as follows: take the 1Q-MRDPT tree for each quadrant and

TABLE III Pseudo-code for Algorithm 1Q-RDPT

Algorithm 1Q- RDPT

Input: Sinks N in one quadrant

Output: RDPT $T(N)$

```
Sort sinks in N in reverse lexicographic order on (x, y);
Compute layer L_N from N;
Compute layer L_S from N;
while L_S ≠ {s} do
  L'_N ← (1, k)-ORDPT (L_N, L_S);
  Compute maximal layer L_N from L'_N and L_S;
  L'_N ← L_N − L_S;
  Compute layer L_S from L'_N and N;
  N ← N − L_S;
endwhile
T(N) ← (1,1)-RDPT(L_N);
```

arbitrarily connect to the source s. We can show that this simple RDPT solution has total cost of at most 2 times the cost of the final MRDPT.

THEOREM 8.1 *Any choice of corners yields a* 2-*approximation RDPT.*

Proof Note that all the savings in the cost of the merged MRDPT result from the sharing the common axis by the trees on two adjacent quadrants. Any additional connections to these axes will only result in added costs. Furthermore, notice that the 1Q-MRDPT solutions have two supporting axes. Therefore the merged tree may simply end up merging two parallel axes of adjacent 1Q-MRDPT solutions into a single axis. But since for each axis pair we can obtain at best a single axis, the cost savings of this merge step cannot exceed 2. □

Based on Theorem 8.1 we propose the following algorithm. To recap from Section 4, $\{c_0, c_1, c_2, c_3\}$ denote a set of four corner vertices, $\mathcal{C}_0$, $\mathcal{C}_1$, $\mathcal{C}_2$ and $\mathcal{C}_3$ denote the sets of possible corner vertices for each axis, and $T_i(N_i, \cup\{c_i, c_{i+1 \bmod 4}\})$ denotes a 1Q-MRDPT solution with two corners, then MRDPT T can be computed from the following formula:

$$T = \min_{c_i \in \mathcal{C}_i} \left\{ \bigcup_{i=0}^{3} T_i(N_i \cup \{c_i, c_{(i+1) \bmod 4}\}) \right\} \qquad (3)$$

We now outline a heuristic algorithm that obtains a solution to this problem. Some ideas behind this algorithm can be found in [10]. We choose $O(\alpha)$ corners from each axis, i.e., let $\mathcal{C}_i^\alpha = \{c_j^\alpha | c_j^\alpha \in \mathcal{C}_i$ and $i = 1, \ldots, \alpha\}$. We then compute the RDPT by the following steps:

1. Compute all possible 1Q-RDPT solutions for each quadrant N_i. Since we have selected $O(\alpha)$ corners for each axis, there are $O(\alpha^2)$ corner combinations, each of which corresponds to a $T_i(N_i \cup \{c_i^\alpha, c_{(i+1) \bmod 4}^\alpha)$ tree. Note that for each tree we have to compute a full 1Q-RDPT solution.
2. For each combination of corners on an axis find the least cost corners on the opposite axis. This procedure finds the best $T(N_i \cup N_{i+1} \cup \{c_i^\alpha, c_{i+1}^\alpha, c_{(i+1) \bmod 4}^\alpha\})$ tree. Since the solutions of each quadrant are known, this computation can be done in $O(\alpha)$ time for each corner pair combination. We keep the best combination, since α^2 corner pair combinations on an axis are possible, then the total time complexity of this step is $O(\alpha^3)$. Note that no further RDPT computations are needed for this step, since in the previous step we computed all the necessary costs.

The above procedure can be improved by taking advantage of two observations that lead to corner pruning heuristics:

- If a sink is already located on an axis, we are guaranteed a connection between that sink and the origin. Therefore the corners between that sink and the origin can be discarded.
- Consider quadrants Q_0 and Q_1. Let y_0 and y_1 denote the largest y-coordinates of the terminals in each quadrant, respectively. Since there are terminals on only one of the quadrants between y_0 and y_1, we know that no connections will be made on the y-axis in this span. Thus all corners in the (min (y_0, y_1), max (y_0, y_1)] interval can be discarded. Similarly, this applies to the remaining axes.

We now summarize the time complexity of the heuristic RDPT algorithm. The algorithm is dominated by the computations of the 1Q-RDPT solutions, thus the algorithm at worst takes $O(\alpha^2 n^3 + \alpha^3)$ time and on average $O(\alpha^2 n^{\frac{3}{2}} + \alpha^3)$ time to compute the RDPT tree. The algorithm, as described, requires at most $O(n^2 + \alpha^2)$ storage, which is dominated by the storage requirements of the (1, 1)-MRDPT algorithm and of the cost matrices for the quadrant tree merging algorithm. We note that the Rao-Shor RSA algorithm including quadrant merging has a complexity of $O(n^3 \log n + n^3)$. Our quadrant merging method improves this result, while maintaining the perfor-

mance bound on the solution, to $O(\alpha^2 n \log n + \alpha^3)$ for any constant $\alpha > 1$.

9. EXPERIMENTAL RESULTS

In this section we will present several experimental results obtained with the pRDPT algorithm. The pRDPT algorithm was implemented in $C++$ on a SUN workstation. We use randomly uniform test cases for our comparisons, which we generate by selecting $n-1$ random sink locations on a 1000 × 1000 grid centered on (0,0), and by placing $s = (0,0)$.

We will first investigate the effects of choosing α on the results of the algorithm. We tried $\alpha = \{2, 4, 8, 10, 12\}$ on problems of various sizes. A set of 10 random examples were generated for each problem size. To highlight the effect of choosing corners, we forced a gap between the axes and the point sets: *i.e.* no points other that the source were allowed within a certain distance from the axes. Without this gap we observed at most a 3% difference in the tree costs with increased corners. With the gap, we observed at most a 15% difference in the cost of a tree generated with $\alpha = 2$ and $\alpha = 12$. Furthermore for $n \leq 64$ the number of corners per axis was often less than 12. From these results we conclude that a small value for α is sufficient to obtain excellent solutions without sacrificing algorithm performance. Some of these results are shown in Figure 6.

Next we examine the performance of the pRDPT algorithm. We use as a performance complexity measure the value $P = \Sigma_{i=1}^{k} |L_i|^2$ which we have shown to dominate the worst case behavior of the algorithm. A plot of various values for different problem sizes is shown on Figure 7. A regression was made using the performance function $P = c_1 n^{c_2}$ on the worst case points from Figure 7. The value obtained for the coefficient $c_2 = 1.3$, which is considerably better than the expected worst case performance value of $c_2 = 3$, and close to the average performance value of $c_2 = 1.5$.

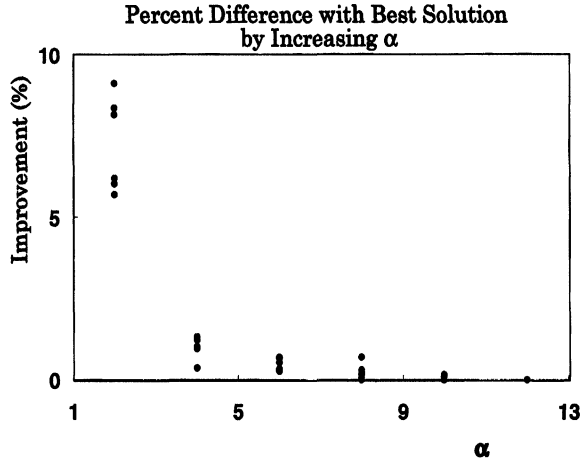

FIGURE 6 Graph plots the percentage difference in total wirelength from a run with a given α to the run using the largest $\alpha = 12$. Runs use $\alpha = \{2, 4, 8, 10, 12\}$. The number of points vary from $n = 16$ to $n = 256$. These results are obtained by forcing a minimum separation between the sinks and the axes.

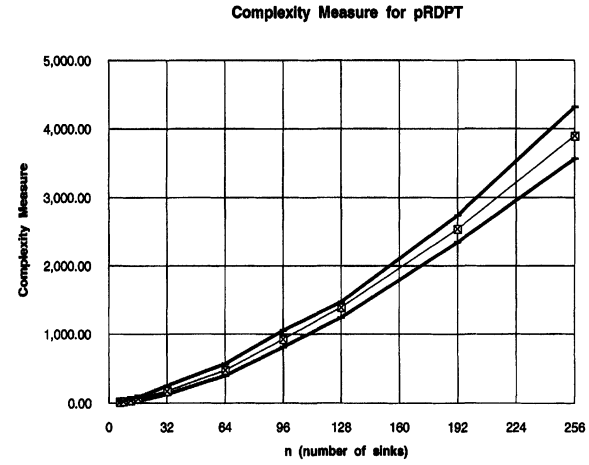

FIGURE 7 Graph plots minimum, average and maximum values of the complexity meansure $\sum_{i=1}^{k} |L_i|^2$ versus the total number of sinks n. The values are obtained for 25 randomly generated samples, with different number of sinks n.

We compare the cost of trees obtained from the pRDPT algorithm to the cost of Rectilinear Steiner trees and to Minimum Rectilinear Spanning trees(MST). The Rectilinear Steiner trees are obtained using the Iterated1-Steiner (I1S) heuristic. Details of the I1S algorithm can be found in [11]. The I1S code used in these experiments was

written by Gabriel Robins from University of Virginia. This algorithm is to date the best Minimum Cost Rectilinear Steiner tree approximation known, and thus it serves well as a basis of comparison for the results of the pRDPT algorithm. The graphs of the C(pRDPT)/C(I1S) and the C(pRDPT)/C(MRST) ratios can be seen in Figures 8 and 9, respectively. The results indicate that the pRDPT algorithm produces trees of almost the same wire-length as the I1S algorithm for small trees (and in a few cases the results are slightly better! An example is given in Figure 10). The quality of the results decreases as the problem size increases, however, up to the largest problem tried ($n \leq 256$) the increase in total wire-length does not exceed 20%.

Finally, we compare the results of the pRDPT heuristic to the Rao-Shor RSA heuristic. Since the RSA heuristic solves the 1Q-RDPT problem, we compare the results of these two 1Q-RDPT and the RSA algorithms on randomly generated one-quadrant problems. According to [12] other RDPT heuristics, namely the A-Tree and the I1Arb heuristics are experimentally equivalent to the Rao-Shor RSA algorithm. Furthermore the RSA algorithm's performance bound is the best known to date. In our results, detailed in Figure 11, we observe that the 1Q-RDPT algorithm shows a slight average improvement over the RSA algorithm, about 1%, with a maximum improvement of about 10%.

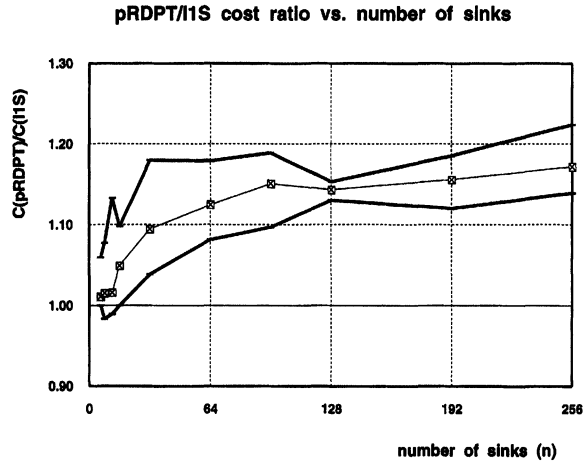

FIGURE 8 Graph plots minimum, average and maximum values of the ratio of the pRDPT tree cost and the I1S approximation tree cost C(pRDTPT)/C(I1S) versus the total number of sinks n. The values are obtained over 25 randomly generated samples, for different number of sinks n.

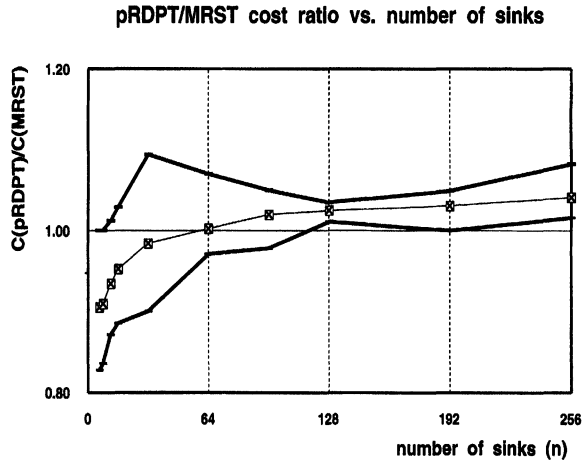

FIGURE 9 Graph plots minimum, average and maximum values of the ratio of the pRDPT tree cost and the Minimum Rectilinear Spanning Tree (MRST) tree cost C(pRDTPT)/C(MRST) versus the total number of sinks n. The values are obtained over 25 randomly generated samples, over different values of n.

10. CONCLUSIONS

The experiments show that the pRDPT algorithm exhibits an average performance of $O(n^{\frac{3}{2}})$ time on uniformly random problems. While the worst case complexity of the Rao-Shor RSA heuristic, namely $O(n \log n)$ is superior to the pRDPT heuristic, in practice both algorithms perform similarly. The results also show that in most cases the pRDPT algorithm obtains results that are a few percent better than the Rao-Shor heuristic, and in some cases improvements of 10% were observed.

Furthermore, the results also indicate that average sink layouts can be routed with little or no penalty in total wire-length using the pRDPT algorithm which uses the 1Q-RDPT algorithm or

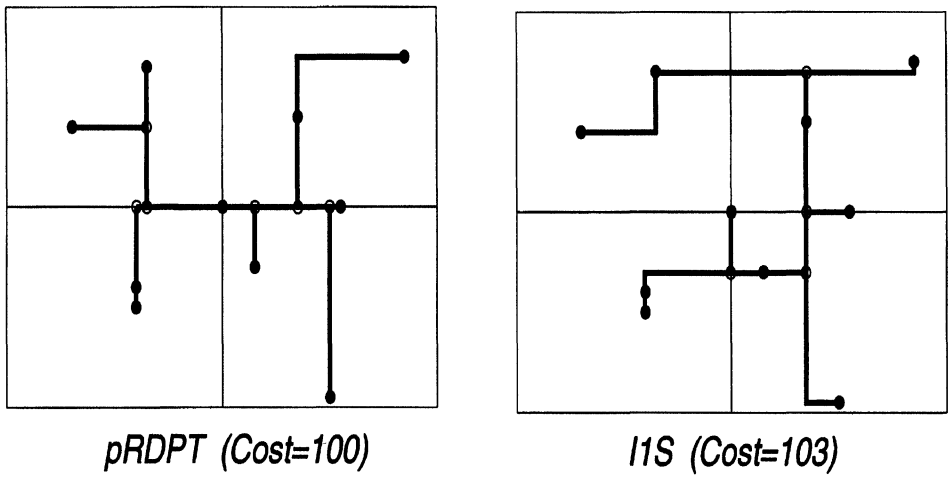

FIGURE 10 Plots of two trees, with $n = 10$. The left tree is generated by the pRDPT algorithm, the tree on the right is generated by the I1S algorithm. This is an example where C(pRDPT) < C(I1S).

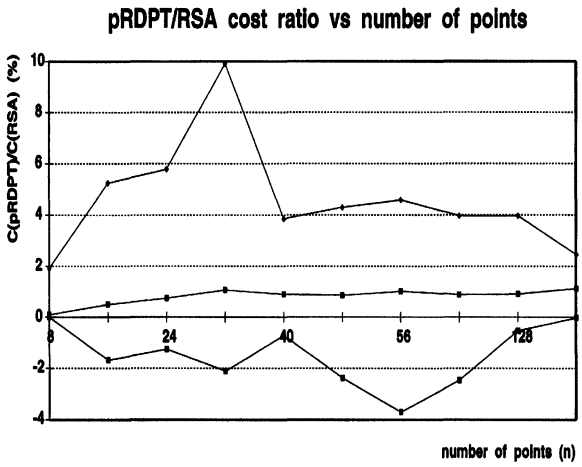

FIGURE 11 Graph shows minimum, average and maximum percent improvement of C(pRDPT)/C(RSA), for values of $n \leq 256$. The results are obtained by running 100 uniformly random experiments for each different problem size.

the RSA heuristic to solve the single quadrant problems. We also conjecture that the pRDPT heuristic has a performance bound ≤ 2. We base this conjecture on the performance bound proofs for the RSA heuristic in [14]. The proof consists in showing that for any 45° cut of an RSA tree, the number of edges that cross the cut can be at most twice the minimum. We can also show this property holds for all pairs of layers used in the 1Q-RDPT algorithm. We have not shown that this is sufficient for the bound to hold.

Several open questions remain. Firstly, it is not known whether the MRDPT problem is NP-complete. Secondly, while the empirical results of RDPT algorithms proposed to date are excellent, their worst case performance ratio remains ≤ 2. However, for the more general Rectilinear Steiner problem algorithms exist with a much better performance ratio $< 11/6$. An open problem is to close this gap, or to show that this is not possible.

We have experimentally shown that the proposed algorithm has good average performance properties. We have also shown experimentally that the trees generated by the pRDPT algorithm exhibit small costs, comparable to those obtained using general Steiner tree and RSA approximation algorithms. Finally, we have proposed algorithmic improvements to the quadrant merging algorithm which make single-quadrant RDPT algorithms attractive for solving VLSI routing problems.

Acknowledgement

This work was supported in by NSF grant MIP-9207267 and by the IBM Ph.D. Resident Study Program.

References

[1] Awerbuch, B., Baratz, A. and Peleg, D. (1990). "Cost-Sensitive Analysis of Communication Protocols". In *Proceedings of ACM Symposium on Principles of Distributed Computing*, 177–187.

[2] Huang, J. H., Alpert, C. J., Hu, T. C. and Kahng, A. B. (1992). "A direct Combination of the Prim and Dijkstra Constructions for Improved Performance-Driven Global Routing". Technical Report CSD-910051, Department of Computer Science, UCLA.

[3] Choi, H. A. and Esfahanian, A. H. (1990). "On Complexity of a Message-Routing Strategy for Multicomputer Systems". In *Proceedings of 16th International Workshop on Graph-Theoretic Concepts in Computer Science, Berlin, Germany, in Lecture Notes in Computer Science*, 170–181. Spring-Verlag.

[4] Cohoon, J. P. and Randall, L. J. (Jan. 1991). "Critical Net Routing". In *Proc. IEEE Intl. Conf. on Computer Design*, 174–177.

[5] Cong, J., Kahng, A., Robins, G., Sarrafzadeh, M. and Wong, C. K. (June 1992). "Provably Good Performance-Driven Global Routing". *IEEE Transactions on Computer Aided Design*, **11**(6), 739–752.

[6] Cong, J., Leung, K.S. and Zhou, D. (Oct. 1992). "Performance-Driven Interconnect Design Based on Distributed RC Delay Model". In *UCLA Computer Science Department Technical Report CSD-920043.*

[7] Donath, W. E., Norman, R. J., Agrawal, B. K. and Bello, S. E. (1990). "Timing Driven Placement Using Complete Path Delays". In *Design Automation Conference*, 84–89. IEEE7/ACM.

[8] Hanan, M. (February 1966). "On Steiners Problem with Rectilinear Distance". *SIAM Journal on Applied Mathematics*, **14**(2), 255–265.

[9] Hwang, F. K. (January 1976). "On Steiner Minimal Trees with Rectilinear Distance". *SIAM Journal on Applied Mathematics*, **30**(1), 104–114.

[10] Ma, T. H., Ho, J. M., Ko, M. T. and Sung, T. Y. (June 1992). "Algorithms for Rectilinear Optimal Multicast Tree Problem". In *Proc. Intl. Symposium on Algorithms and Computation*, 106–115.

[11] Kahang, A and Robins, G (November 1990). "A New Class of Steiner Tree Heuristics with Good Performance: the Iterated 1-Steiner Appraoch". In *International Conference on Computer-Aided Design*, 428–431. IEEE.

[12] Kahng, A. B. and Robins, G. (1995). *On Optimal Interconnections for VLSI*, 91–95. Kluwer Academic Publishers, Norwell, MA.

[13] Preparata, F. P. and Shamos, M. I. (1985). *Computational Geometry: An Introduction*, Springer Verlag.

[14] Hwang, F. K., Rao, S. K., Sadayappan, P. and Shor, P. W. (1992). "The Rectilinear Steiner Arborescence Problem" *Algorithmica*, **7**, 277–288.

[15] Sutanthavibul, S., Shragowitz, E. (1990). "An Adaptive Timing-Driven Layout for High Speed VLST". In *Design Automation Conference*, 90–95. IEEE/ACM.

Authors' Biographies

Gustavo E. Téllez was born in Bogotá Colombia. He received his B.S. and M.S. degrees in Electrical Engineering from Rensselaer Polytechnic Institute, in 1984 and 1985, respectively. From 1986 to 1992 he worked for IBM Corporation in their EDA Development facility in East Fishkill, NY. In 1996 he received his Ph.D. in Computer Science at Northwestern University under the IBM Ph.D. Resident Study Program, and has returned to his position at IBM EDA. His research interests include design and analysis of algorithms, computer architectures, timing and power driven VLSI design, and clock network design.

Majid Sarrafzadeh (M'87, SM'92, F'96) received his B.S., M.S. and Ph.D. in 1982, 1984, and 1987, respectively, all from the University of Illinois at Urbana-Champaign in Electrical and Computer Engineering Department.

He joined Northwestern University as an Assistant Professor in 1987. Since 1991 he has been Associate Professor of Electrical Engineering and Computer Science at Northwestern University. His research interests lie in the area of VLSI CAD, design and analysis of algorithms and VLSI architecture.

Dr. Sarrafzadeh is a Fellow of IEEE. He received an NSF Engineering Initiation award in 1987, two distinguished paper awards in ICCAD-91, and the best paper award for physical design in DAC-93. He has served on the technical program committee of various conferences, for example, ICCAD, EDAC and ISCAS. He is a co-editor of the book "Algorithmic Aspects of VLSI Layout", co-author of a book "An Introduction to VLSI Physical design", on the editorial board of the VLSI Design Journal, co-editor-in-chief of the International Journal of High-Speed Electronics, an Associate Editor of IEEE Transactions on Computer-Aided Design.