Performance Driven Routing with Multiple Sources *

Jason Cong
Patrick H. Madden
UCLA Computer Science Dept
(310)206-2775, (310)UCLA-CSD Fax
cong@cs.ucla.edu pickle@cs.ucla.edu

March 31, 1995

ABSTRACT

Existing routing problems for delay minimization consider the connection of a *single* source node to a number of sink nodes, with the objective of minimizing the delay from the source to all sinks, or a set of critical sinks. In this paper, we study the problem of routing nets with *multiple* sources, such as those found in signal busses. This new model assumes that each node in a net may be a source, a sink, or both. The objective is to optimize the routing topology to minimize the total weighted delay between *all* node pairs (or a subset of critical node pairs). We present a heuristic algorithm for the multiple-source performance-driven routing problem based on efficient construction of minimum-diameter minimum-cost Steiner trees. Experimental results for submicron CMOS IC and MCM technologies show an average of 11.4% and 20% reduction in the maximum interconnect delay, when compared with conventional minimum Steiner tree based topologies.

1 Introduction

The competitive nature of the VLSI industry has created a strong demand for techniques to improve the performance of integrated circuits. Methods to increase speed, and to reduce area or power consumption, are of great interest.

Scaling of device dimensions has resulted in changes to many fundamental design goals: where previously the bulk of system delay had been generated by the switching times of devices, it is now common that the interconnecting wires between devices accounts for the dominating portion of the delay. These changes have created new areas in need of optimization, and new measures by which we gauge solution quality.

With smaller minimum feature size comes a reduction in transistor channel width, and thus smaller resistance; the reduction in wire width on the other hand results in higher unit wire resistance. As a result, the *resistance ratio*, defined in [8] to be the driver resistance divided by the unit length wire resistance, is reduced significantly. This shift produces a situation where the length of the path between a driver and sink can have comparable resistance to that of the transistor channel. Thus, changes to the interconnect length and topology can have a significant impact on delay. The result in [8] showed convincingly that interconnect topology optimization has a considerable effect on interconnect delay reduction when the resistance ratio is small.

A number of optimized interconnect topologies have been proposed, including bounded-radius bounded-cost trees[6], AHHK trees[1], LAST trees[14], maximum performance trees[5], A-trees[8], low-delay trees[3], and IDW/CFD trees[12]. These methods consider both the traditional concern of low total wire length, and the *path length* between the source node and the timing-critical sink nodes.

Although many of these methods effectively reduce the interconnect delay, all of them assume that there is a single source node driving one or more sink nodes and minimize the delay from the unique source to all sinks, or a set of critical sinks.

In practice, many timing-critical nets may have multiple sources, each of them controlled by a tri-state gate and driving the net at a different time. Signal busses are instances of such nets. In these cases, the existing

*This work is partially supported by ARPA/CSTO under Contract J-FBI-93-112, NSF Young Investigator Award MIP9357582, and a grant from Intel Corporation.

	Routing Tree 1				Routing Tree 2			
Driver	p_1	p_2	p_3	p_4	p_1	p_2	p_3	p_4
p_1	-	1.99	2.93	3.18	-	3.19	4.11	4.11
p_2	4.34	-	5.21	5.45	3.19	-	4.11	4.11

Table 1: Topology effects on delay. For a net with multiple sources, the delay to a given sink depends on which node drives the net.

performance-driven routing algorithms for single source nets may perform poorly, as a topology optimized for one source may result in long interconnect delay when some other source becomes active.

Figure 1 presents a pair of routing trees with equal wire length over four nodes. The first routing tree, optimized for node p_1 and has relatively high delay when node p_2 drives the net. The second routing tree provides a lower overall maximum delay when all four nodes might be sources or sinks. Delay times with respect to the driving nodes are shown in Table 1. While it has been shown that both the optimal Steiner tree with minimum wirelength and the optimal *single source* routing tree under the Elmore delay model can be constructed on the Hanan grid[10, 2], the second routing tree, with improved worst-case performance, does not lie completely on the Hanan grid.

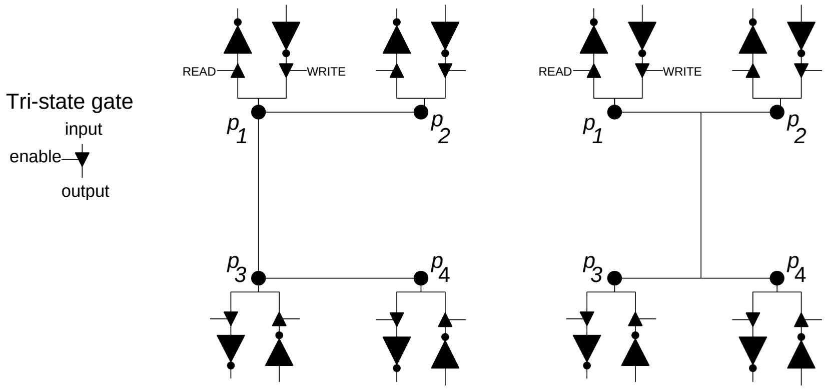

Figure 1: An example of the impact of path length on delay. The first routing tree is optimized for node p_1. When p_2 drives the net, however, performance suffers. The second routing tree provides a lower maximum delay when both p_1 and p_2 can drive the net.

In this paper, we study the problem of routing nets with multiple sources. This new model assumes that each node in a net may be a source, a sink, or both. The objective is to optimize the routing topology to minimize the total weighted delay between *all* node pairs, where the weight between a node pair indicates the priority of delay minimization between this pair of nodes. We present an algorithm for the performance-driven multiple source routing problem based on construction of *minimum diameter A-trees*.

2 Problem Formulation

Given a set of points $P = \{p_1, p_2, ..., p_n\}$ on the Manhattan plane, and a non-negative weight $W(p_i, p_j)$ as the *weight* between each pair of source p_i and sink p_j to indicate the timing criticality between this pair of points, the *performance-driven multiple source routing (PD-MSR) problem* is defined as finding a Steiner tree T which connects all points in P and minimizes the following two objectives:

- Total weighted delay $WD(T)$ between pairs of nodes p_i and p_j. i.e. $WD(T) = \sum_{p_i,p_j} W(p_i,p_j) \times delay(p_i,p_j)$.
- Total tree length $L(T)$, defined as the sum of the lengths of each tree edge.

We assume that the first objective has higher priority than the second one. For simplicity, one may assume that $W(p_i, p_j) \in [0, 1]$, i.e. non-critical pairs of points have weight zero and critical pairs have weight one. The delay between a pair of points, $delay(p_i, p_j)$, may be estimated using an appropriate model, such as the linear delay model (where delay is proportional to path length), the distributed Elmore delay model [9], or calculated using SPICE.

Given a point $p_i \in P$, we use $p_i(X)$ and $p_i(Y)$ to denote the x and y coordinates of point p_i respectively. For any two points p_i and p_j, we define the distance $d(p_i, p_j)$ between them as their Manhattan distance, $|p_i(X) - p_j(X)| + |p_i(Y) - p_j(Y)|$. Given a tree T, we define the distance between nodes p_i and p_j in T as $d_T(p_i, p_j)$, the sum of edge lengths along the unique path between the points. The *diameter* D of tree T over a set of points P, $D_T(P)$, is defined as the maximum $d_T(p_i, p_j)$ over all pairs p_i, p_j. Given a point set P, we define the diameter $D(P)$ of the set to be the maximum distance between any pair of points in the set. Clearly, we always have $D_T(P) \geq D(P)$.

Note that if the weights of all pairs are zero, the PD-MSR problem as we have formulated it becomes the classical minimum Steiner tree problem, which is NP-hard. Therefore, the PD-MSR problem is also NP-hard. However, if we don't constrain the total wire length, and only wish to minimize the total weighted delay, the complexity of the problem is not known.

When the delay bound of each pair of timing-critical nodes is given, one can also formulate the *constrained multiple source routing problem* as finding a Steiner routing tree which satisfies the delay constraint between every timing-critical pair and minimizes the total tree length.

In the following, we treat the sources and sinks of the routing problem as nodes in a graph, or points on a plane.

3 Minimum Diameter Tree Construction

For our algorithm, we minimize the maximum path length between any pair of critical source and sink, in order to minimize the maximum linear delay between any pair of critical source and sink.

In the case of a single driver, such minimization can be obtained by *radius minimization*, with direct paths between the driver and all sink nodes. Shortest path trees rooted at the source achieve this goal.

When there are multiple sources and sinks, path length minimization can be achieved by minimizing the maximum distance between any pair of nodes, which leads to *diameter minimization*. Our goal is to construct a minimum diameter routing tree with minimum total tree cost, as measured by a combination of maximum path length, average path length, and total tree length.

A number of results for minimum diameter trees on the Euclidean plane were presented in [11]. In particular, it was shown that the diameter of the smallest enclosing circle for a set of points also gives the minimum diameter for a tree connecting those points. After determination of this minimum diameter circle, a star topology connecting the center of the circle to each point in the set was shown to have the minimum diameter possible of any Steiner tree over the points.

However, the Manhattan minimum diameter Steiner tree problem has not been studied in the literature. Our work studies the construction of minimum diameter Steiner trees in the Manhattan plane, and wire length minimization of such trees.

In the next two subsections, we discuss general constraints related to Manhattan minimum diameter trees, present a pair of lemmas which give the lower bound for tree diameter, and then present a simple method, the *Minimum Diameter A-Tree (MD A-Tree)* algorithm, to construct trees that obtain this lower bound. A third subsection presents the *Minimum Cost Minimum Diameter A-Tree (MC MD A-Tree)* algorithm, which provides a method to optimize the tree construction.

3.1 Manhattan Tree Diameter Minimization

We define a *tilted rectangle (TR)* as a region defined by a rectangle with sides at 45 degree angles with respect to the X and Y axis. Such a region may be defined by a set of four equations, $-X + Y \geq B_{SE}$, $X - Y \geq B_{NW}$, $-X - Y \geq B_{NE}$, and $X + Y \geq B_{SW}$. Using typical compass directions, the constants B_{SE}, B_{NW}, B_{NE}, and B_{SW} represent the Southeastern, Northwestern, Northeastern, and Southwestern boundaries, respectively.

In the Manhattan plane, the analog of the Euclidean circle is the *tilted square (TS)*, the set of points p such that $d(p, c) \leq D/2$ for some center point c and diameter D. A *TS* is a special case of a *TR*, where the distances between opposite sides are equal. Obviously, the maximum distance between any pair of points contained in a

TS of diameter D is less than or equal to D. The distance between points on opposite sides of an *STS* will be D.

Given a point set P, *STR(P)* and *STS(P)* are the *smallest tilted rectangle* and a *smallest tilted square* enclosing P, respectively. Clearly, *STS(P)* contains *STR(P)*. Figure 2 shows an *STR* and two *STS*s for a point set. Note that *STR(P)* is unique, but there may be a class of *STS*s for a given point set.

Lemma 1 *For a point set P, the diameter of the point set $D(P)$ is equal to the diameter D of an STS containing the points.*

Proof: Let p_i and p_j be points on opposite sides of the *STR* which is also part of opposite sides of an *STS* (as in Figure 2). Clearly, the diameter of the *STS* is equal to the distance between p_i and p_j, i.e. $D = d(p_i, p_j)$. Let the point c denote the center of the *STS*.

For any other points p_k and p_l, we have

$$d(p_k, p_l) \leq d(p_k, c) + d(c, p_l) \leq \frac{D}{2} + \frac{D}{2} = D$$

Thus, the *STS* has diameter D equal to $d(p_i, p_j)$, and no other pair of points has a greater distance. □

Lemma 2 *A shortest path tree T rooted at the center c of an STS for a point set P is a minimum diameter tree.*

Proof: Let D be the diameter of $STS(P)$, then a shortest path tree rooted at the center c of $STS(P)$ with edges to each point p_i will have a radius of no more than $\frac{D}{2}$; the distance between any pair of points will therefore be no more than $2 \times \frac{D}{2}$. So for any pair of points p_i and p_j, we have

$$d_T(p_i, p_j) \leq d_T(p_i, c) + d_T(c, p_j) = d(p_i, c) + d(c, p_j) \leq \frac{D}{2} + \frac{D}{2} = D = D(P)$$

This implies $D_T(P) \leq D(P)$. Since $D_T(P) \geq D(P)$ for any tree T, we have $D_T(P) = D(P)$, i.e. tree T has the minimum diameter. □

For the Manhattan plane, an *STS* can easily be found in linear time. We first find the *STR* enclosing the points by computing the B_{SE}, B_{NW}, B_{NE}, and B_{SW} values of the boundary equations as follows.

$$\begin{aligned}
B_{SE} &= min(-p_i(X) + p_i(Y)) && \forall p_i \in P \\
B_{NW} &= min(p_i(X) - p_i(Y)) && \forall p_i \in P \\
B_{NE} &= min(-p_i(X) - p_i(Y)) && \forall p_i \in P \\
B_{NW} &= min(p_i(X) + p_i(Y)) && \forall p_i \in P
\end{aligned}$$

We can then adjust one of the boundary equations to produce an *STS* covering the points.

Also note that the center of an *STS* can be obtained directly from the B_{SE}, B_{NW}, B_{NE}, and B_{SW} values. The two lines $-X + Y = \frac{B_{SE}+B_{NW}}{2}$ and $X + Y = \frac{B_{NE}+B_{SW}}{2}$ bisect the sides of the *STR*, and will therefore intersect at the center of the *STR*. The center of the *STR* is also the center of an *STS*.

3.2 Minimum Diameter A-Tree Algorithm

Our algorithm presented in this paper for Manhattan minimum diameter tree construction consists of two basic steps (as shown in Figure 3). The first step is to identify the root point r of the tree, which could be the center of an *STS*, or another point obtained by methods we will describe below. The second step is to construct a shortest path tree rooted at r with low tree length. As the *A-Tree*[8] algorithm has proven to be very effective in generating a shortest path Steiner tree on the Manhattan plane with minimal total wirelength, it is used in the second step of our algorithm.

The most basic variation of our algorithm is called the *Minimum Diameter A-Tree (MD A-Tree)* algorithm. It simply computes the *STS* for the point set P, and then constructs an A-tree rooted at the center of the *STS*. The A-Tree is a shortest path tree; as Lemma 2 indicates, the resulting routing tree will have minimum diameter.

The A-tree algorithm is described briefly in the next section.

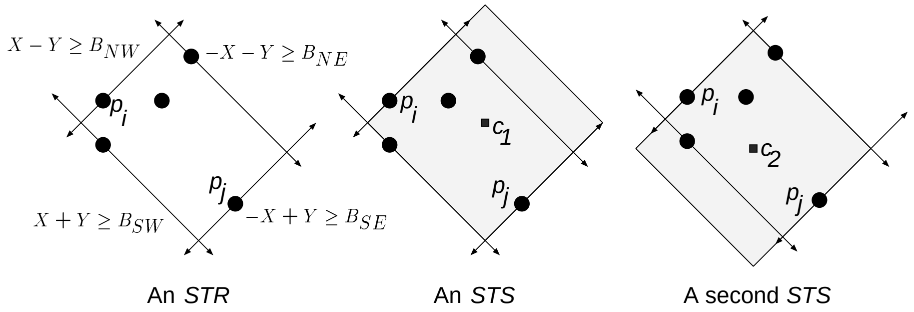

Figure 2: The smallest tilted rectangle *(STR)* containing the points, and a pair of smallest tilted squares *STS*s which contain the rectangle.

1. Identify a root point r for the point set P.
2. Construct a shortest path tree connecting the root point r with each point in P.

Figure 3: Basic steps of the *Minimum Diameter A-Tree (MDAT)* algorithm.

3.2.1 Review of the A-Tree Algorithm

In [8], the authors presented an algorithm for the construction of *A-trees,* which are defined to be Steiner trees rooted at a source, with shortest path connections between the root and all sinks. An optimal A-tree has minimum wire length.

The A-tree algorithm in [8] constructs trees by starting with a forest of points (the source and all sinks), and then iteratively merges subtrees until all components are connected. The merges progress such that new edges form shortest paths to the root. Their algorithm applies three "safe" merge moves which preserve the length optimality of the construction process; if only safe moves are applied, the tree will have optimal length. Two heuristic moves are used in the event that no safe move can be found. On average, it was shown that 94% of merge moves were optimal, and the trees constructed were within 4% of their optimal length.

Figure 4 shows the three types of safe moves.

The S1 move involves the merging of two sub-trees. The path from point p_i to point p_j is shorter than p_i's path to any other point, or to any other location that could be on some point p_k's shortest path to the root. As no other connection of p_i to any possible (shortest path) tree can have lower length, the optimality of the construction process is preserved.

The S2 and S3 moves "grow" (either horizontally or vertically) a sub-tree p to a location p', with growth directed towards the root. p' is a location at which another sub-tree may be able to pick up the connection. The sub-tree then "waits" at this location for connection by another safe move, or a heuristic move if no safe move can be found. Examination of the locations of other sub-trees can establish instances where growth by these rules preserves the optimality of the construction process.

When none of the safe moves are possible, the heuristic moves shown in Figure 5 are applied.

In an A-tree T rooted at point r, for any point p_i in P, $d_T(p_i, r) = d(p_i, r)$, as the routing within the tree is guaranteed to be a shortest path.

As was noted in Figure 2, the *STS* of a set of points in not necessarily unique, and so there may be a number of acceptable "root" points for the center of a minimum diameter tree. In fact, it is not always necessary to place the root of the A-tree at the center of an *STS*. This freedom leads to an optimization approach that is discussed in the next subsection.

3.3 Minimum Cost Minimum Diameter A-Tree Algorithm

By using an A-tree construction, we ensure that the tree distance between any point and the root is equal to the Manhattan distance. Having this, it is easy to see that as long as the root point r satisfies the constraint

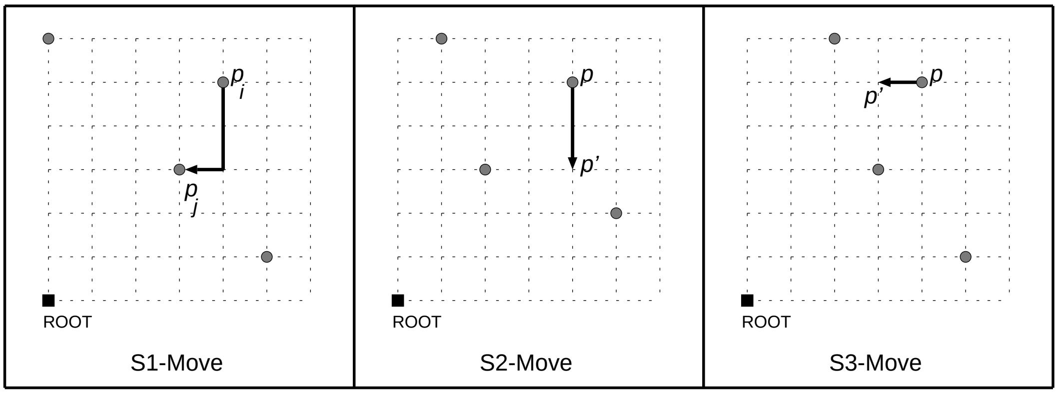

Figure 4: Three types of safe moves used in *A-Tree* construction.

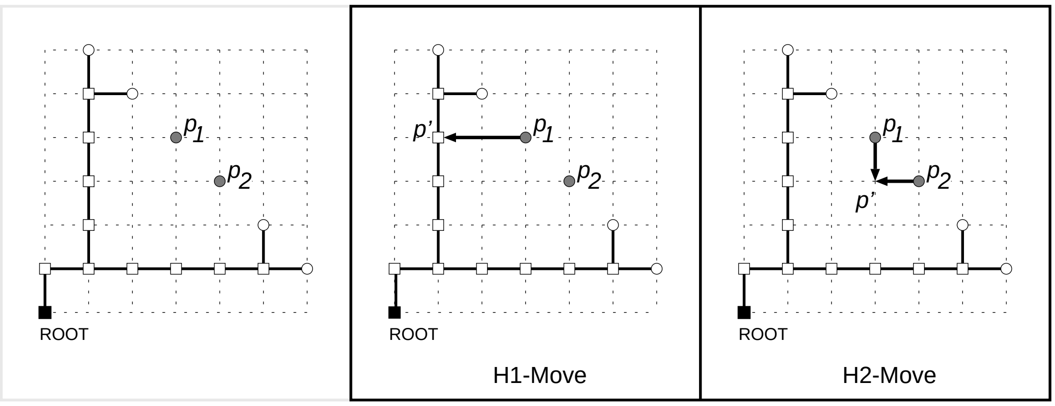

Figure 5: Two types of heuristic moves used in *A-Tree* construction.

$d(p_i, r) + d(r, p_j) \leq D$ for all distinct p_i and p_j, the tree will have minimum diameter. This freedom allows for further optimization, with the possibility of reductions in tree length and weighted path length.

An example of such an instance for the Manhattan plane is given in Figure 6. By shifting the "center" point slightly, a reduction in tree length is obtained without an increase in the maximum diameter of the tree. It should be noted that first topology in this figure is a sub-optimal A-tree, used to illustrate the general principal. More complex examples can be constructed where an optimal A-tree rooted at the center of an STS will have greater wire length than an A-tree rooted at another location.

3.3.1 Feasible Region

As there can be more that one location that can serve as the root of a minimum diameter tree, we wish to define this region formally.

We define the *feasible region* (FR) of a set of points P as

$$FR(P) = \{r \mid d(p_i, r) + d(r, p_j) \leq D\} \forall p_i, p_j \in P$$

If only a subset of point pairs are critical, we can define the *critical feasible region* (FR_c) of a set of points P as

$$FR_c(P) = \{r \mid d(p_i, r) + d(r, p_j) \leq D\} \forall p_i, p_j \in P, W(p_i, p_j) \neq 0$$

For the FR, the diameter D is relative to the feasible diameter over all points. For the FR_c, the diameter D is relative only to those points which are part of some p_i, p_j pair where $W(p_i, p_j) \neq 0$. The diameter for the FR is less than or equal to the diameter for the FR_c. Note that the FR_c does not place constraints on path lengths

Equation	Constraint		
E_N:	$-Y$	$\geq$	B_N
E_S:	Y	$\geq$	B_S
E_E:	$-X$	$\geq$	B_E
E_W:	X	$\geq$	B_W
E_{NE}:	$-X + -Y$	$\geq$	B_{NE}
E_{NW}:	$X + -Y$	$\geq$	B_{NW}
E_{SE}:	$-X + Y$	$\geq$	B_{SE}
E_{SW}:	$X + Y$	$\geq$	B_{SW}

Table 2: Octilinear bound equations used to define either an *OE* or and *OR*.

between non-critical pairs, so the path length between a non-critical point and another point may exceed the minimum diameter. When all node pairs are critical, the FR and FR_c are equivalent, but in general, the two regions are not equivalent and may even be disjoint.

In the Euclidean plane, the regions which satisfy these constraints are simply the intersections of ellipses formed by the pairs p_i, p_j. A similar property holds for the Manhattan plane.

We define an *octilinear segment* to be a segment that is either horizontal, vertical, or have slope ± 1; an *octilinear bound equation* is an equation of the form $\alpha X + \beta Y \geq B$ ($\alpha, \beta \in \{0, 1, -1\}$, with at least one being non-zero), for some constant B. In the Manhattan plane, the set of points satisfying $d(p_i, r) + d(r, p_j) \leq D$ is an *octilinear ellipse (OE)*. An *OE* is bounded by no more than eight octilinear segments, and can also be represented by the intersection of eight *octilinear bound equations* with constants B_N, B_S, B_E, B_W, B_{NE}, B_{NW}, B_{SE}, and B_{SW} denoting the Northern, Southern, Eastern, Western, Northeastern, Northwestern, Southeastern, and Southwestern boundaries of the *OE*, respectively; the octilinear bound equations are shown in Table 2. If $d(p_i, p_j) = D'$, the *OE* contains the bounding box of the points with a "fringe" of $\frac{D-D'}{2}$. An example is shown in Figure 7.

An *octilinear region (OR)* is defined to be a region that is bounded by no more than eight octilinear segments, and can also be represented by the intersection of no more than eight octilinear bound equations. An *OE* is an instance of an *OR*.

Without loss of generality, let $E^i \equiv \alpha X + \beta Y \geq B^i$ be an octilinear bound equation of *OR* R_i, and $E^j \equiv \alpha X + \beta Y \geq B^j$ an octilinear bound equation of *OR* R_j. Then $\alpha X + \beta Y \geq max(B^i, B^j)$ defines an octilinear bound equation of the *OR* $R_i \cap R_j$. We will use the term "constrictive" in relation to these equations; if an equation E^i of *OR* R_i defines a region that is a strict subset of E^j from *OR* R_j, then E^i is more constrictive than E^j.

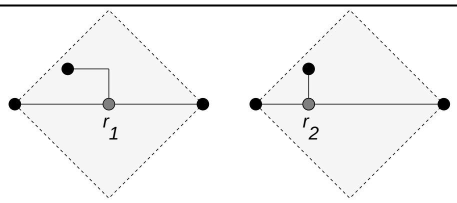

Figure 6: Two trees in the Manhattan plane which satisfy $d(p_i, r) + d(r, p_j) \leq D$. The point r_1 is at the center of the *STS*; point r_2 is within the *feasible region* for the set of points.

In order to determine the set of points which may serve as the center of a minimum diameter tree, we find the intersection of all *OE*s for the point pairs. As the *OE*s are convex, their intersections (and the feasible regions) will also be convex. There are no more than $O(n^2)$ point pairs, resulting in the same number of *OE*s. It can be shown that the intersection of any number of *OE*s can be represented by a single *OR*. The shaded *OR* in Figure 8 shows the feasible region for the root of a shortest path tree that will result in a minimum diameter tree.

Theorem 1 *The $FR(P)$ and $FR_c(P)$ for any set of points P and any set of critical pairs are non-empty.*

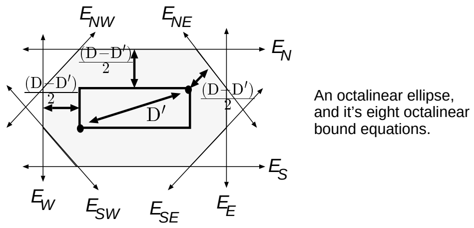

Figure 7: An *octilinear ellipse (OE)* on the Manhattan plane, the set of points which satisfy $d(p_i, r) + d(r, p_j) \leq D$. This region can be defined as either a polygon bounded by no more than eight sides, or as the intersection of eight octilinear bound equations.

Proof: Lemma 1 showed that the minimum diameter of a tree connecting the points was equal to the diameter of an *STS*; let c be the center of such an *STS*. Lemma 2 showed that a shortest path tree rooted at c has minimum diameter. Clearly, c satisfies $d(p_i, c) + d(c, p_j) \leq D$ for any p_i and p_j, and is therefore contained by each *OE*. As this point is contained by each *OE*, it is also contained by the intersection, and therefore the $FR(P)$ and $FR_c(P)$ are non-empty. □

Theorem 2 *Any shortest-path tree rooted at point $r \in FR(P)$ is a minimum diameter tree. Any shortest-path tree rooted at a point $r \in FR_c(P)$ is a minimum diameter tree over the critical points.*

Proof: This arises directly from the definitions of $FR(P)$ and $FR_c(P)$. For any point r in the feasible region, $d(p_i, r) + d(r, p_j) \leq D$ for all pairs p_i and p_j; a shortest path tree T rooted at r ensures that $d_T(p_i, r) = d(p_i, r)$, so $d_T(p_i, r) + d_T(r, p_j) \leq D$. □

As $FR(P)$ and $FR_c(P)$ are non-empty, and points within these sets allow for the construction of minimum diameter trees, we will use them to guide our search for root points of low cost (in terms of path length or tree length) trees.

Clearly, construction of $FR(P)$ and $FR_c(P)$ can be performed in $O(n^2)$ time, by simply intersecting the $\binom{n}{2}$ *OE*s formed by all point pairs. In the next subsection, we will present a method for linear time computation of these regions.

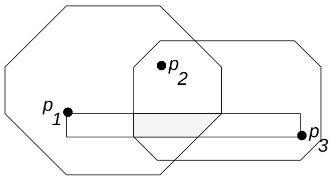

Figure 8: The feasible region $FR(P)$ for the center of a minimum diameter A-tree, the intersection of a number of octilinear ellipses.

3.3.2 Linear Time Computation of Feasible Region

In this section we show how to compute $FR(P)$ in linear time; $FR_c(P)$ can be computed similarly by considering only critical points. We approach the problem by determining which pairs of points generate the most constrictive bounds for each of the eight octilinear bound equations that may define the feasible region. We will consider two cases: one for the uppermost horizontal bound of $FR(P)$, E_N, and one for the upper right diagonal bound, E_{NE}. Other bounds may be obtained by similar methods.

Pseudocode for the algorithms to compute these bounds is given in Figure 9. Proofs that these algorithms are correct are given in the next two Lemmas.

```
find-E_N()
    find points p_i, p_j with smallest and second smallest
        p_i(X) + p_i(Y) and p_j(X) + p_j(Y) values respectively
    find points p_k, p_l with largest and second largest
        p_k(X) − p_k(Y) and p_l(X) − p_l(Y) values respectively
    if   (p_i ≠ p_k)
        Bound is obtained from OE(p_i, p_k)
    else
        Bound is obtained from most constrictive of
        OE(p_i, p_l) and OE(p_k, p_j).

find-E_NE()
    find points p_i, p_j with smallest and second smallest
        p_i(X) + p_i(Y) and p_j(X) + p_j(Y) values respectively
    Bound is obtained from OE(p_i, p_j)
```

Figure 9: Pseudocode for algorithms which find the boundaries for the feasible regions in linear time. Other bounds may be obtained by simply replacing the smallest or largest value criteria with those specified in Tables 3 and 4.

Lemma 3 *For the two points $p_i, p_k \in P$, $p_i(X) \leq p_k(X)$, which form the most constrictive upper horizontal bound E_N of $FR(P)$, the point p_i will have minimal $p_i(X) + p_i(Y)$ value, and the point p_k will have maximal $p_k(X) - p_k(Y)$ value.*

Proof: Let p_i and p_k be the two points which generate the most constrictive upper horizontal bount E_N, and $p_i(X) \leq p_k(X)$. Consider Figure 10.

We choose a point q along the upper horizontal bound directly above point p_k, and have $d(p_i, q) = d_1$, $d(p_k, q) = d_2$, and $d_1 + d_2 = D$.

Now suppose that there exists a point p_m which satisfies $p_m(X) + p_m(Y) < p_i(X) + p_i(Y)$. These conditions result in the following.

$$\begin{array}{rcl} d(p_m, q) & > & d(p_i, q) \quad = d_1 \\ d(p_m, q) + d_2 & > & D \end{array}$$

This implies that the upper horizontal bound for the OE of p_m and p_k is more constrictive than the one of p_i and p_k, and contradicts the initial assumption. Thus, the p_i element of the pair which generates the most constrictive bound is clearly one with smallest the $p_i(X) + p_i(Y)$ value. Points with equal values will generate equivalent bounds. The p_i component of the pair can be found with a single pass over the points.

By a similar set of arguments, it is clear that the p_k element of the pair must have the largest $p_k(X) - p_k(Y)$ value.

In some instances, the point with the smallest $X + Y$ and the point with the largest $X - Y$ may be the same; in this case, it is clear that the point will either be a "p_i" or a "p_k". We consider the second smallest $p_j(X) + p_j(Y)$ and second largest $p_l(X) - p_l(Y)$ points as well, with the most constrictive bound being formed from the point that the criteria select in common, and one of the secondary points. □

Lemma 4 *The two points $p_i, p_j \in P$ which form the most constrictive upper right diagonal bound E_{NE} of $FR(P)$ will have the two smallest $p_i(X) + p_i(Y)$ and $p_j(X) + p_j(Y)$ values.*

Proof: Assume p_i and p_j are the point pair which most constrictive upper right diagonal bound E_{NE} of $FR(P)$.

Consider Figure 11. Assume that we shift the input point set so that it is contained in the first quadrant (this will not change the shape of the feasible region, or the points which provide the bounds). We label the

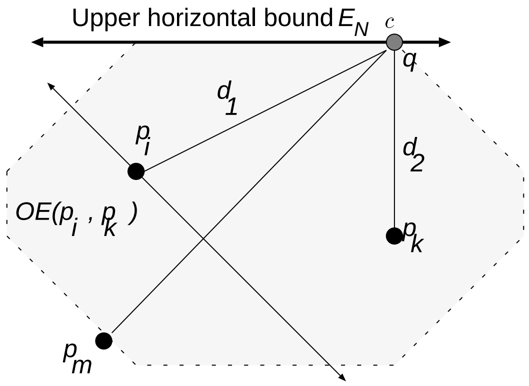

Figure 10: Computation of the upper horizontal bound of the feasible region. If $OE(p_i, p_k)$ generates the most constrictive upper horizontal bound on $FR(P)$, then point p_i must have minimal $p_i(X) - p_i(Y)$ value.

intersection of E_{NE} and the X axis as point q; the bound E_{NE} which generates the minimal q will also generate the most constrictive bound on the FR.

Without loss of generality, we will assume that point p_i is to the left of p_j. For this proof, we will assume that $p_i(Y) \geq p_j(Y)$, the other case being solved similarly. We now have the following set of equations.

$$\begin{aligned} q &= p_i(Y) + \frac{D - d(p_i, p_j)}{2} + p_j(X) \\ &= p_i(Y) + \frac{D}{2} - \frac{(p_j(X) - p_i(X))}{2} - \frac{(p_i(Y) - p_j(Y))}{2} + p_j(X) \\ &= \frac{D}{2} + \frac{p_i(X) + p_i(Y)}{2} + \frac{p_j(X) + p_j(Y)}{2} \end{aligned}$$

Since p_i and p_j generate the most constrictive upper right diagonal bound, q is minimum. Therefore, $p_i(X) + p_i(Y)$ and $p_j(X) + p_j(Y)$ are the smallest and second smallest among all $p(X) + p(Y)$. □

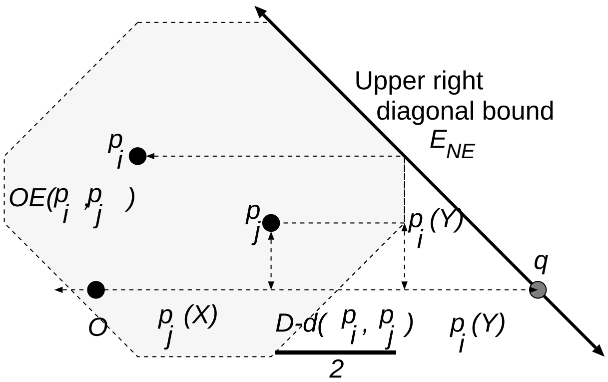

Figure 11: Computation of the upper diagonal bound of the feasible region. The most constrictive bound on $FR(P)$ will be generated by the linear inequality of $OE(p_i, p_j)$ that intersects the X axis at the minimum q.

Theorem 3 *$FR(P)$ and $FR_c(P)$ can be computed in linear time.*

Proof: Lemma 1 gave a linear time method to obtain diameter $D(P)$, while Lemmas 3 and 4 gave linear time methods to identify the pairs of points which generate the most constrictive bounds for E_N and E_{NE}. Other bounds can be obtained in a similar manner, using the criteria shown in Tables 3 and 4. □

Bound	p_i, p_j	p_k, p_l
E_N	smallest $X+Y$	largest $X-Y$
E_S	smallest $X-Y$	largest $X+Y$
E_E	smallest $X+Y$	smallest $X-Y$
E_W	largest $X-Y$	largest $X+Y$

Table 3: Criteria used to determine pairs of points which define the horizontal and vertical bounds of $FR(P)$.

Bound	p_i, p_j
E_{NE}	smallest $X+Y$
E_{NW}	largest $X-Y$
E_{SE}	smallest $X-Y$
E_{SW}	largest $X+Y$

Table 4: Criteria used to determine pairs of points which define the horizontal and vertical bounds of $FR(P)$.

3.3.3 Necessity of Feasible Region

The lemmas given above for the linear time construction of the feasible region are useful in proving another property of the feasible region: any minimum diameter tree *must* intersect $FR(P)$. To prove this, we will need an additional lemma.

Lemma 5 *The upper horizontal bound E_N of $FR(P)$ will intersect $FR(P)$.*

Proof: Consider Figure 12. Let p_i and p_j be the two points with smallest and second smallest $X - Y$ values respectively; Lemma 4 showed that these points generate the most constrictive upper right diagonal bound E_{NE}. Similarly, the points which define the upper right diagonal bound E_{NW} are labeled the points are p_k and p_l. We identify the intersection of these two bounding lines as the point q, resulting in

$$d(p_i, q) + d(p_j, q) = D$$

$$d(p_k, q) + d(p_l, q) = D$$

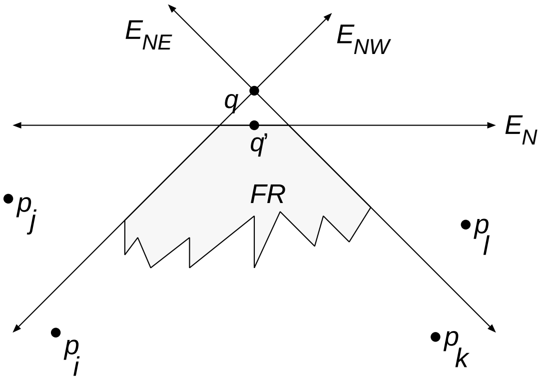

Figure 12: The upper horizontal bound E_N of the feasible region must cross the diagonal bounds E_{NE} and E_{NW} at their intersection, or below their intersection.

Lemma 3 showed that the two points which generate the upper horizontal bound are p_i and p_k. As $d(p_i, q) \geq d(p_j, q)$, and $d(p_k, q) \geq d(p_l, q)$, the distances of p_i and p_k to point q are clearly at least $\frac{D}{2}$. A point q' along the upper horizontal bound E_N cannot have larger Y value that q, as this would result in $d(p_i, q') + d(p_k, q') > D$. $\square$

Similarly the lower horizontal bound E_S, and the two vertical bounds E_E and E_W also intersect the feasible region (if only at a single point). The diagonal bounds cannot restrict the feasible region away from a horizontal or vertical bound (leaving a "gap" between feasible region and one of these bounds).

Theorem 4 *Any minimum diameter tree connecting a set of points P must intersect $FR(P)$.*

Proof: We will prove this by creating a contradiction: if a tree to has minimum diameter over the points and does not intersect $FR(P)$, it contains a cycle.

Assume we have a minimum diameter tree, and that it does not intersect $FR(P)$. Consider Figure 13, with the four most extreme points (smallest $X+Y$, largest $X-Y$, smallest $X-Y$, and largest $X+Y$) labeled p_i, p_j, p_k, and p_l respectively. Lemma 3 showed that these points are the ones which generate the horizontal and vertical bounds of $FR(P)$.

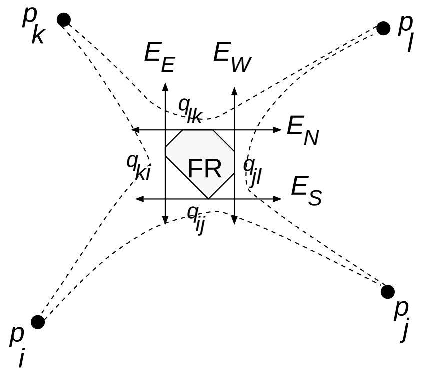

Figure 13: Paths between the extreme points must form a cycle if they do not intersect the feasible region.

In a minimum diameter tree, there must be a path from p_i to p_j of length no greater than D. This path cannot go above the upper horizontal bound E_N, as this bound is generated by these two p_i and p_j. As Lemma 6 showed, E_N must intersect the feasible region, if only at a single point, so if the path is to avoid the feasible region, not only must it go beneath the E_N boundary, but beneath the entire feasible region, passing through some point q_{ij}. Similar constraints are placed on the p_i to p_k path, the p_k to p_l path, and the p_j to p_k path.

Thus, we have a set of paths $p_i \rightarrow q_{ij} \rightarrow p_j$, $p_j \rightarrow q_{jl} \rightarrow p_l$, $p_l \rightarrow q_{lk} \rightarrow p_k$, and $p_k \rightarrow q_{ki} \rightarrow p_i$. Clearly, there is a cycle $q_{ij} \rightarrow q_{jl} \rightarrow q_{lk} \rightarrow q_{ki} \rightarrow q_{ij}$, and the construction cannot be a tree. □

Thus, we show that not only will selection of a root point from the $FR(P)$ allow for the construction of a minimum diameter tree, but also that if a minimum diameter tree is required, the tree must pass through this region.

3.3.4 Summary of *MC MD A-Tree* Algorithm

As was shown in Figure 6, some locations of tree root points will lead to lower cost trees (in terms of path length or tree length) than other points. The basic algorithm given in Figure 3 allows for a number of variations for root point selection in the first step. The options available are briefly described below.

Root Restriction. The root point may be restricted to either the center of an *STS*, or a location within the $FR(P)$ or $FR_c(P)$. For the *MD A-Tree* variation we use the center of the *STS*, while the *MC MD A-Tree* variations are allowed to select a root point from within the feasible region.

Tree Root Selection. When there are a number of root points available, one must be selected. We have performed experiments using both an estimate of tree length ($\sum d(p_i, r)$), and an actual A-tree construction to evaluate candidate root points.

We consider the corner points of the feasible region, the Hanan grid points contained within the feasible region, and the intersections of the Hanan grid lines with the feasible region boundaries as candidate root points.

Cost Function. The cost function used to select a root point can either prefer minimum tree length, minimum weighted path length, or a combination of these.

Experimentally, we have found the best performance to be that of the minimum diameter tree rooted within $FR_c(P)$, with tree cost obtained from the actual A-Tree construction, and the preference for minimum tree length. Detailed results are given in the next section.

It should be noted that the when there is an underlying routing grid, the feasible region does not always contain points on this grid. Thus, the restriction of a root point to a grid location may result in a tree with slightly larger diameter than the minimum diameter possible on a gridless surface.

4 Experimental Results

To evaluate the performance of the routing topologies, we used HSPICE to simulate a sized inverter driving a minimum-width wired network. The transistor model used for both 0.5μm CMOS IC and MCM technologies was the 0.5μm CMOS IC technology "nominal" model supplied by MCNC. Interconnect resistance and capacitance parameters are shown in Table 5; these values are the same as those used in [8] and [2]. For each technology, we generated 100 test sets for each set size of 4, 8, and 16 randomly placed nodes.

Technology:	0.5μm CMOS IC	Multi-Chip Modules (MCMs)
Driver (NMOS, PMOS):	10.0μm $\times$0.5μm, 9.5μm $\times$0.5μm	200.0μm $\times$0.5μm, 170.0μm $\times$0.5μm
Unit Wire Resistance:	0.112 $\Omega/\mu m$	0.008 $\Omega/\mu m$
Unit Wire Capacitance:	0.039 $fF/\mu m$	0.060 $fF/\mu m$
Loading Capacitance:	1 fF	1000 fF
Total Area:	$1cm \times 1cm$	$10cm \times 10cm$

Table 5: Technology parameters based on advanced MCM designs.

For the 0.5μm IC technology experiments, the surface area spanned was 1cm square, with a grid size of 10 microns (resulting in a 1000 by 1000 grid). Net segments were modeled as "π" circuits, with the capacitance of each wire segment divided between its endpoints. Segments longer than 1000 microns were broken into smaller segments. Transistor sizes for the inverters were 10.0μm $\times$0.5μm and 9.5μm $\times$0.5μm for NMOS and PMOS, respectively; these sizes were selected to provide roughly equal rise and fall times.

For the MCM experiments, the surface area spanned was 10cm square, with a grid size of 100 microns (resulting in a 1000 by 1000 grid). Segments longer than 10000 microns were broken into smaller segments. Transistor sizes for the inverters were 200.0μm $\times$0.5μm and 170.0μm $\times$0.5μm for NMOS and PMOS, respectively; these sizes were selected to provide roughly equal rise and fall times.

Delay for all cases was measured as the time between the beginning of the input transition and the output signal reaching 90% of the final value (either 0.5V or 4.5V). Rise and fall times for the inputs were 0.6ns for all tests.

For each tree, we compute three types of delay. The first, *maximum delay* (MD) is the maximum delay of any source node to any sink node in a tree. The second, *average maximum delay* (AMD), is the average of the maximum source-sink delays for each source. The third, *average delay* (AD), is the average of all source-sink delays. Delay results are in nanoseconds, while tree and path lengths are in centimeters.

The topologies compared are as follows.

Minimum Diameter A-Trees. The *STS* for the weighted points is obtained, and an A-tree spanning all points is constructed.

Minimum Cost Minimum Diameter A-Trees. The $FR(P)$ or $FR_c(P)$ is constructed, and then possible root points from this region are evaluated by the construction of a tree at each point. The minimum length tree is selected as the final topology.

1-Steiner trees. To obtain a tree with low total length, the 1-Steiner algorithm of Kahng and Robins[13] is used. These trees place no bounds on path length, but have the best known performance for tree length minimization.

n	Topology	MD (ns)	AMD (ns)	AD (ns)	Length	Diameter
4	1-Steiner	1.38	1.26	1.19	1.216	1.052
4	MD A-Tree	1.37	1.27	1.21	1.235	.998
4	MC MD A-Tree	1.36	1.25	1.19	1.217	.998
8	1-Steiner	2.47	2.01	1.81	1.994	1.549
8	MD A-Tree	2.34	2.03	1.88	2.099	1.283
8	MC MD A-Tree	2.32	1.98	1.82	2.049	1.284
16	1-Steiner	4.13	3.05	2.66	2.969	1.999
16	MD A-Tree	3.70	2.98	2.73	3.205	1.487
16	MC MD A-Tree	3.66	2.91	2.66	3.129	1.487

Table 6: Comparison of 1-Steiner, MD A-Tree, and MC MD A-Tree for maximum delay (MD), average maximum delay (AMD), and average delay (AD) through HSPICE simulation with 0.5μm CMOS IC technology parameters. Average tree lengths and diameters are in centimeters.

Simulation results for 0.5μm CMOS IC parameters are given in table 6. While the 1-Steiner algorithm produces wire lengths that are from 1.5% to 8% lower, the average diameter of trees produced by the Minimum Diameter A-Tree and Minimum Cost Minimum Diameter A-Tree algorithms was from 5% to 34% lower. MCM technology examples, shown in Table 7, produced identical length results (the MCM test sets are scaled versions of the CMOS IC test sets). When given the freedom to select a root point, the Minimum Cost Minimum Diameter A-Tree algorithm obtained tree lengths that were from 1% to 2% lower than those of the Minimum Diameter A-Tree algorithm.

MD A-Tree and *MC MD A-Tree* produced similar delay results. Using 0.5μm CMOS IC parameters, the two algorithms produced as much as an 11.4% maximum delay reduction, and a 4.6% average maximum delay reduction compared to the 1-Steiner algorithm. Using MCM parameters, the two algorithms produced as much as a 20% maximum delay reduction, and a 3.1% average maximum delay reduction.

For these experiments, it is assumed that the weight between all pairs is 1, indicating that every path is critical.

Surprisingly, the *MD A-Tree* algorithm produced slightly better maximum delay values for the MCM examples than the *MC MD A-Tree* algorithm. In these cases, the "shifted" center location of the tree root results in some branches having disproportionally high capacitance, and the tree is "unbalanced." As has been observed in other high performance routing studies, the location of branching has an effect on the total delay.

In the Table 8 we provide results for test sets of 8 nodes in the 0.5μm CMOS IC technology, with 1, 2, 3, and 10 randomly selected pairs of critical nodes. For this table, we include the *weighted path length* (WPL) and *required path length* (RPL); the weighted path length is $\sum W(p_i, p_j) \times d_T(p_i, p_j)$ for all critical pairs, while the required path length is $\sum W(p_i, p_j) \times d(p_i, p_j)$. The required path length provides a lower bound (which in some instances cannot be achieved with a tree topology).

When only a subset of paths are critical, the *MC MD A-Tree* algorithm produced maximum delay improvements ranging from 4% to 10%. Trees rooted within the critical feasible region $FR_c(P)$ had slightly lower delay than those rooted in the feasible region $FR(P)$. Tree lengths were comparable.

In all cases, the time required for topology construction was much smaller than the time required to perform the HSPICE simulations, and the bulk of topology construction time was consumed by the *A-Tree* algorithm. The most complex variation, *MC MD A-Tree*, had run times ranging from 0.6 seconds to 6 seconds for examples with 16 nodes and all pairs critical. The run time of the algorithm was strongly influenced by the size of the feasible region, as this impacts the number of candidate root locations; a larger feasible region (which can occur when fewer points are critical) resulted in a larger run time.

5 Conclusion

We have addressed a new problem in performance driven routing, shown that it is as hard as the minimum Steiner tree problem, and have presented a heuristic solution for the problem through the construction of *Minimum Cost Minimum Diameter A-Trees.*

n	Topology	MD (ns)	AMD (ns)	AD (ns)	Length	Diameter
4	1-Steiner	11.60	8.842	7.538	12.168	10.528
4	MD A-Tree	11.17	8.922	7.689	12.359	9.987
4	MC MD A-Tree	11.19	8.748	7.522	12.171	9.988
8	1-Steiner	28.81	18.90	14.80	19.945	15.491
8	MD A-Tree	25.01	18.24	15.20	20.996	12.837
8	MC MD A-Tree	25.05	17.74	14.64	20.497	12.840
16	1-Steiner	61.36	36.97	27.81	29.693	19.997
16	MD A-Tree	48.89	32.55	26.96	32.052	14.873
16	MC MD A-Tree	49.15	32.03	26.25	31.291	14.877

Table 7: Comparison of 1-Steiner, MD A-Tree, and MC MD A-Tree algorithms for maximum delay (MD), average maximum delay (AMD), and average delay (AD) through HSPICE simulation with MCM technology parameters. Average tree lengths and diameters are in centimeters.

pairs	Topology	MD (ns)	AMD (ns)	AD (ns)	Length	WPL	WD	RPL
1	1-Steiner	1.89	1.89	1.88	1.994	.836	.836	.709
1	MD A-Tree, $FR(P)$	1.92	1.92	1.91	2.099	.802	.802	.709
1	MD A-Tree, $FR_C(P)$	1.87	1.87	1.86	2.117	.709	.709	.709
1	MC MD A-Tree, $FR(P)$	1.86	1.86	1.85	2.049	.780	.780	.709
1	MC MD A-Tree, $FR_C(P)$	1.79	1.79	1.78	2.016	.709	.709	.709
2	1-Steiner	2.05	1.87	1.85	1.994	1.550	1.034	1.334
2	MD A-Tree, $FR(P)$	2.05	1.91	1.89	2.099	1.523	.963	1.334
2	MD A-Tree, $FR_C(P)$	2.01	1.88	1.86	2.106	1.440	.897	1.334
2	MC MD A-Tree, $FR(P)$	1.99	1.84	1.83	2.049	1.456	.943	1.334
2	MC MD A-Tree, $FR_C(P)$	1.95	1.80	1.78	2.016	1.383	.896	1.334
3	1-Steiner	2.14	1.86	1.84	1.994	2.297	1.131	1.983
3	MD A-Tree, $FR(P)$	2.13	1.90	1.88	2.099	2.268	1.033	1.983
3	MD A-Tree, $FR_C(P)$	2.10	1.89	1.87	2.114	2.222	.984	1.983
3	MC MD A-Tree, $FR(P)$	2.07	1.84	1.82	2.049	2.178	1.013	1.983
3	MC MD A-Tree, $FR_C(P)$	2.04	1.81	1.80	2.021	2.118	.989	1.983
10	1-Steiner	2.36	1.82	1.80	1.994	7.550	1.353	6.643
10	MD A-Tree, $FR(P)$	2.28	1.89	1.87	2.099	7.602	1.170	6.643
10	MD A-Tree, $FR_C(P)$	2.28	1.89	1.87	2.099	7.585	1.163	6.643
10	MC MD A-Tree, $FR(P)$	2.24	1.83	1.82	2.049	7.360	1.173	6.643
10	MC MD A-Tree, $FR_C(P)$	2.26	1.80	1.79	2.020	7.250	1.184	6.643

Table 8: Comparison of 1-Steiner, MD A-Tree, and MC MD A-Tree algorithms for maximum delay (MD), average maximum delay (AMD), and average delay (AD) through HSPICE simulation with 0.5μm CMOS IC technology parameters. All test sets consisted of 8 nodes, with only a given number of node pairs being critical. Also included in this table are the average weighted path lengths (WPL), weighted diameter (WD), and required path length (RPL). The required path length provides a lower bound for the weighted path length, and in some cases cannot be obtained. Experiments considering both the feasible region ($FR(P)$) and the critical feasible ($FR_c(P)$) were performed.

We have also given a linear time algorithm to determine the set of possible root locations for a minimum diameter shortest path tree, and shown that any minimum diameter tree must pass through this region.

When compared with the 1-Steiner algorithm, minimum diameter A-trees produced lower maximum and average delays, with slightly higher tree length. Consideration of the feasible center region resulted in reductions in tree length, and also in delay for most cases.

When only a subset of pairs are critical, the methods still provide improvement, although approaches which do not restrict the diameter of the tree may provide greater improvement. These cases require more study; topologies which do not have minimum diameter may have dramatically better performance than those that do.

Bounded diameter constructions may also be of some interest; a bounded diameter tree can be constructed easily by rooting a bounded radius tree within $FR(P)$ or $FR_c(P)$. Construction of this type may also be effective for the performance-driven multiple source routing problem.

References

[1] C. J. Alpert, T. C. Hu, J. H. Huang, and A. B. Kahng, "A Direct Combination of the Prim and Dijkstra Constructions for Improved Performance-Driven Routing," *Proc. IEEE Int'l Symp. on Circuits and Systems*, pp. 1869-1872, 1993.

[2] K. D. Boese, A. B. Kahng, B. A. McCoy, and G. Robins, "Rectilinear Steiner Trees with Minimum Elmore Delay," *Proc. 31st ACM/IEEE DAC*, pp. 381-386, 1994.

[3] K. D. Boese, A. B. Kahng, and G. Robins, "High-Performance Routing Trees With Identified Critical Sinks," *Proc. ACM/IEEE DAC*, pp. 182-187, 1993.

[4] T.-H. Chao, Y.-C. Hsu, J.-M. Ho, K. D. Boese, and A. B. Kahng, "Zero Skew Clock Routing with Minimum Wirelength," *IEEE Trans. on Circuits and Systems*, Vol. 39, No. 11, pp. 799-814, November 1992.

[5] J. P. Cohoon and L. J. Randall, "Critical Net Routing," *Proc. IEEE Int'l. Conf. on Computer Design*, pp. 174-177, 1991.

[6] J. Cong, A. B. Kahng, G. Robins, and M. Sarrafzadeh, "Provably Good Performance-Driven Global Routing," *IEEE Trans. Computer Aided Design*, Vol. 11, No. 6, pp. 739-752, June 1992.

[7] J. Cong and K.-S. Leung, "Optimal Wiresizing Under the Distributed Elmore Delay Model," *Proc. Int'l Conf. on Computer-Aided Design*, pp. 110-114, 1993.

[8] J. Cong, K.-S. Leung, and D. Zhou, "Performance-Driven Interconnect Design Based on Distributed RC Delay Model," *Proc. 30th ACM/IEEE DAC*, pp. 606-611, 1993.

[9] W. C. Elmore, "The Transient Response of Damped Linear Network with Particular Regard to Wideband Amplifier", *J. Applied Physicas*, 19(1948), pp. 55-63.

[10] M. Hanan, "On Steiner's Problem With Rectilinear Distance," *SIAM Journal of Applied Mathematics*, pp. 255-265, February 1966, Vol. 14, No. 2.

[11] J.-M. Ho, D. T. Lee, C.-H. Chang, and C. K. Wong "Bounded-Diameter Minimum Spanning Trees and Related Problems," *Computational Geometry Conference*, pp. 276-282, 1989.

[12] X. Hong, T. Xue, E. S. Kuh, C. K. Cheng, and J. Huang, "Performance-Driven Steiner Tree Algorithms for Global Routing," *Proc. ACM/IEEE DAC*, pp. 177-181, 1993.

[13] A. B. Kahng and G. Robins, "A New Family of Steiner Tree Heuristics with Good Performance: The Iterated 1-Steiner Approach," *Proc. IEEE Int. Conf Computer Aided Design*, pp. 428-431, 1990.

[14] S. Khuller, B. Raghavachari, and N. Young, "Balancing Minimum Spanning Trees and Shortest-Path Trees," Proc. ACM/SIAM Symp. Discrete Algorithms, January 1993, pp. 243-250.

[15] D. Zhou, S. Su, F. Fsui, D. S Gao, and J. Cong, "A Simplified Synthesis of Transmission Lines with a Tree Structure," *Journal of Analog Integrated Circuits and Signal Processing (Special Issue on High-Speed Interconnects)*, pp. 19-30, January 1994, Vol. 5, No. 1.