Provably Good Performance-Driven Global Routing

Jingsheng Cong, *Member, IEEE*, Andrew B. Kahng, *Associate Member, IEEE*, Gabriel Robins, Majid Sarrafzadeh, *Member, IEEE*, and C. K. Wong, *Fellow, IEEE*

***Abstract*—We propose a provably good performance-driven global routing algorithm for both cell-based and building-block design. The approach is based on a new bounded-radius minimum routing tree formulation. We first present several heuristics with good performance, based on an analog of Prim's minimum spanning tree construction. Next, we give an algorithm which simultaneously minimizes both routing cost and the longest interconnection path, so that both are bounded by small *constant* factors away from optimal. This method is based on the following two results. First, for any given value of a parameter ϵ, we can construct a routing tree with longest interconnection path length at most $(1 + \epsilon) \cdot R$, and with cost at most $(1 + (2/\epsilon))$ times the minimum spanning tree weight. Moreover, for Steiner global routing in arbitrary weighted graphs, we achieve longest path length at most $(1 + \epsilon) \cdot R$, with wiring cost within a factor $2 \cdot (1 + (2/\epsilon))$ of the *optimal Steiner tree* cost. In both cases, R is the minimum possible length from the source to the furthest sink. We also show that geometry helps in routing: in the Manhattan plane, the total wire length for Steiner routing improves to $3/2 \cdot (1 + (1/\epsilon))$ times the optimal Steiner tree cost, while in the Euclidean plane, the total cost is further reduced to $(2/\sqrt{3}) \cdot (1 + (1/\epsilon))$ times optimal. Furthermore, our method generalizes to the case where varying wire length bounds are prescribed for different source–sink paths. Extensive simulations confirm that this approach works well, using a large set of examples which reflect both cell-based and building-block layout styles.**

I. Introduction

With progress in VLSI fabrication technology, interconnection delay has become increasingly significant in determining circuit speed. Recently, it has been reported that interconnection delay contributes up to 50% to 70% of the clock cycle in the design of dense, high-performance circuits [5], [25]. Thus, with submicron device dimensions and up to a million transistors integrated on a single processor, on-chip and chip-to-chip interconnections play a major role in determining the performance of digital systems.

Because of this trend, performance-driven layout design has received increased attention in the past several years. Most of the work in this area has been on the timing-driven placement problem, where a number of methods have been developed for placing blocks or cells in timing critical paths close together. The so-called zero-slack algorithm was proposed by Hauge, Nair, and Yoffa [9]; fictitious facilities and floating anchors methods were used by Marek-Sadowska and Lin [19], and a linear programming approach was used by Jackson, Srinivasan, and Kuh [13], [14]. Several other approaches, including simulated annealing, have also been studied [5], [18], [25]. Since no global routing solution is generally available at the placement step, most of these placement algorithms use the net bounding box semiperimeter to estimate the interconnection delay of a net.

While such techniques have been developed for timing-driven placement, only limited progress has been reported for the timing-driven interconnection problem. In [6], net priorities are determined based on static timing analysis; nets with high priorities are processed earlier using fewer feedthroughs. In [15], a hierarchical approach to timing-driven routing was outlined. In [21], a timing-driven global router based on the A* heuristic search algorithm was proposed for building-block design. However, these results do not provide a general formulation of the timing-driven global routing problem. Moreover, these solutions are not flexible enough to provide a *trade-off* between interconnection delay and routing cost.

In this paper, we begin by proposing a new model of timing-driven global routing for cell-based design, based on the idea of finding minimum spanning trees with bounded radius. Our method constructs a spanning tree with radius $(1 + \epsilon) \cdot R$ by using an analog of the classical Prim minimum spanning tree (MST) construction, where R is the minimum possible tree radius and ϵ is a nonnegative user-specified parameter. Such an approach offers a very natural, smooth trade-off between the tree radius (maximum signal delay) and the tree cost (total interconnection length). This gives the circuit designer a great deal of algorithmic flexibility, as the parameter ϵ can be varied depending on performance constraints. The method is easy to describe and implement, and empirical performance results are very good; e.g., we obtain an average of 25% reduction in longest source–sink path for 10-pin nets. However, the total wire cost using this method can be an unbounded factor worse than optimal.

With this in mind, we also propose a second method for timing-driven global routing, which is based on a provably good algorithm that *simultaneously* minimizes both total wire length and maximum delay. More specifically, given a positive real parameter ϵ and a set of terminals, our method produces a routing tree with radius at most $(1 + \epsilon) \cdot R$, and with total cost at most $(1 + (2/\epsilon))$ times the MST cost. In other words, both the total wire

Manuscript received June 13, 1991; revised October 24, 1991. This work was supported in part by the National Science Foundation under Grants MIP-9110511 and MIP-9110696. This paper was recommended by Associate Editor R. H. J. M. Otten.

J. Cong, A. B. Kahng, and G. Robins are with the Department of Computer Science, University of California, Los Angeles, CA 90024-1596.

M. Sarrafzadeh is with the EECS Department, Northwestern University, IL, 60208-3118.

C. K. Wong is with the IBM T. J. Watson Research Center, Yorktown Heights, NY, 10598.

IEEE Log Number 9107109.

length and the maximum delay of the routing are simultaneously bounded by *constant* factors away from their optimal values. The method applies to building-block layout styles in addition to the more geometric cell-based designs. In fact, our method generalizes to arbitrary weighted graphs, and also to Steiner routing formulations, where we achieve a wire length bound of $2 \cdot (1 + (2/\epsilon))$ times the optimal *Steiner* tree cost, while still observing the $(1 + \epsilon) \cdot R$ radius limit.

We then show that geometry helps in routing: in the Manhattan plane, our wire length bound for Steiner routing can be improved to $(3/2) \cdot (1 + (1/\epsilon))$ times optimal, and in the Euclidean plane, the Steiner routing bound improves further to $(2/\sqrt{3}) \cdot (1 + (1/\epsilon))$ times optimal. This series of results is especially surprising since construction of a minimum spanning tree with bounded diameter in a general graph is NP-complete [10], as is the Steiner problem in graphs [12].

Our construction can minimize either total wire length (a minimum spanning tree) or the longest source–sink path (a minimum delay, or minimum radius, tree), depending respectively on whether we set $\epsilon = \infty$ or $\epsilon = 0$. Between these two extremes, the method offers a continuous, smooth trade-off. In practice, our algorithm exhibits very good empirical performance, which confirms this smooth trade-off between the competing requirements of minimum delay and minimum total wire length.

Note that in VLSI circuit design, the timing is actually path-dependent, rather than net-dependent. In other words, the timing constraint is specified by the delay from primary inputs to primary outputs. Thus, we may wish to use varying wire length constraints on the different source–sink paths within a given signal net; for example, a source–sink connection on a timing-critical path will require a small value of ϵ, while a connection not on any critical path can allow large ϵ. We therefore extend our method to handle this case, and establish analogous constant-factor bounds on both wire length cost and the radius of the routing solution.

The remainder of this paper is organized as follows. In Section II, we present the general formulation of the performance-driven global routing problem. In Section III, we give a very natural heuristic construction (as well as several simple variants) with good empirical performance for computing bounded-radius routing trees. In Section IV, we present a second effective algorithm for computing bounded-radius routing trees, and show that the algorithm is provably good with constant-factor performance bounds with respect to both delay and routing cost. Section V generalizes our method to Steiner tree global routing. Section VI generalizes our approach to the case where different values of ϵ are allowed within a given signal net, and experimental results are reported in Section VII.

II. The Problem Formulation

A signal net N is a set of terminals, with one terminal $s \in N$ a designated source and the remaining terminals sinks. Because terminals of a signal net can be embedded in the Manhattan plane (for standard-cell or sea-of-gates design) or within a channel intersection graph (for macrocell or block design), the global routing problem can have two distinct flavors. In the former case, the cost of routing between two nodes is given by geometric distance, while in the second case the cost is the total edge cost of the shortest path between the nodes.[1] With this in mind, we define the underlying *routing graph* to be a connected weighted graph $G = (V, E)$. A net is a subset of the nodes in this graph. A routing solution of a net N is a tree in G, which we call the routing tree of the net, connecting all the terminals/nodes in N.

Since the routing tree may be treated as a distributed *RC* tree, we may use the first-order moment of the impulse response (also called Elmore's delay) to approximate interconnection delay [8], [23]. A more accurate approximation can be obtained using the upper and lower bounds on delay in an *RC* tree derived in [23]. However, although both the formula for Elmore's delay and those in [23] are very useful for simulation or timing verification, they involve sums of quadratic terms and are difficult to compute and optimize during the layout design process.

Thus, a linear *RC* model (where interconnection delay between a source and a sink is proportional to the wire length between the two terminals) is often used to derive a simpler approximation for interconnection delay (e.g., [18], [22]). In this paper, we shall also use wire length to approximate interconnection delay in the construction of routing solutions. In practice, a subsequent iterative improvement step, based on a more accurate *RC* delay model, may be used to enhance the routing solutions.

We say that the *cost* of a path in G is the sum of the edge weights in the path. A *shortest path* in G between two terminals $x, y \in N$, denoted by $minpath_G(x, y)$, is a path connecting x and y with minimum cost. In a routing tree, T, $minpath_T(x, y)$ is simply the unique path between x and y. Note that in the geometric case, the cost of $minpath_G(x, y)$ is simply geometric distance, and we use the notation $dist(x, y)$ for clarity. For a weighted graph G, we use $dist_G(x, y)$ to denote the cost of $minpath_G(x, y)$.

Definition: The *radius* R of a signal net is the cost of a shortest path in G from the source to the farthest sink, i.e., $\max_{x \in N} dist_G(s, x)$.

Definition: The *radius* of a routing tree T denoted by $r(T)$, is the cost of a (shortest) path in T from the source s to the furthest sink. Clearly, $r(T) \geq R$ for any routing tree T.

According to the linear *RC* delay model, we minimize the interconnection delay of a net by minimizing the radius of the routing tree, which measures the maximum interconnection delay between the source and any sink.

[1] For simplicity of presentation, our definition of the cost of an edge reflects only the wire length. It is straightforward to extend the cost definition to be an increasing function of wire length, channel capacity, and current channel density. The results presented in the next four sections will still hold.

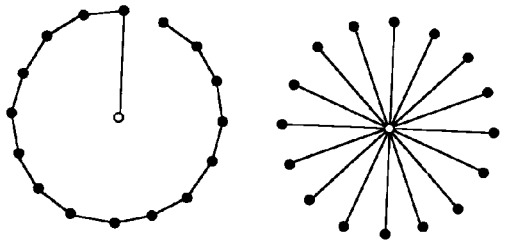
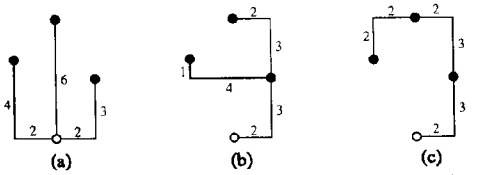


Fig. 1. An example where the cost of a shortest path tree (*right*) is $\Omega(|N|)$ times larger than the cost of a minimum spanning tree (*left*).



(a) (b) (c)

Fig. 2. An example in the Manhattan plane of how increasing the value of ϵ may result in decreased tree cost, but increased radius $r(T)$: (a) $\epsilon = 0$, $\text{cost}(T) = 17$, $n(T) = 6$; (b) $\epsilon = 1$, $\text{cost}(T) = 15$, $n(T) = 10$; (c) $\epsilon = \infty$, $\text{cost}(T) = 14$, $r(T) = 14$.

On the other hand, we also want a routing tree with small total wire length. Without this latter consideration, we could simply use the *shortest path tree* (SPT) of the net, i.e., the union of all the shortest source–sink paths computed by Dijkstra's single-source shortest-path algorithm [20]. Although the SPT has the smallest possible radius $r(SPT)$ of any routing tree, the SPT cost might be very high. Fig. 1 shows a case where the cost of the shortest path tree can be $\Omega(|N|)$ times greater than the cost of the minimum spanning tree.

A routing tree with high cost may increase the overall routing area. Moreover, high cost also contributes to the interconnection delay, which is not captured in the linear *RC* model. Therefore, neither tree shown in Fig. 1 is particularly desirable. In order to consider both the radius and the cost in the routing tree construction, we formulate the timing-driven global routing problem as follows:

The Bounded Radius Minimum Routing Tree (BRMRT) Problem: Given a parameter $\epsilon \geq 0$ and a signal net with radius R, find a minimum-cost routing tree T with radius $r(T) \leq (1 + \epsilon) \cdot R$.

The parameter ϵ controls the trade-off between the radius and the cost of the tree. When $\epsilon = 0$, we minimize the radius of the routing tree and thus obtain a shortest-path tree for the signal net; on the other hand, when $\epsilon = \infty$ we minimize the total cost of the tree and obtain a minimum spanning tree. In general, as ϵ grows, there is less restriction on the radius, allowing further reduction in tree cost. Fig. 2 shows an example where three distinct spanning trees are obtained using different values of ϵ: Fig. 2(a) shows the minimum radius spanning tree corresponding to the case $\epsilon = 0$, with maximum path length $r(T) = 6$; Fig. 2(b) shows a solution with $r(T) = 10$, corresponding to the case $\epsilon = 1$; and Fig. 2(c) shows the minimum spanning tree corresponding to the case $\epsilon = \infty$, with $r(T) = 14$.

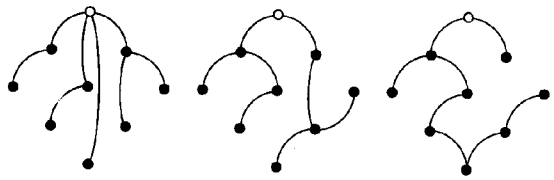


Fig. 3. An example in the Manhattan plane where a SPICE simulation indicated that the routing produced by our algorithm (*middle*) outperforms an MST routing (*right*) by 81 ps, and outperforms the SPT routing (*left*) by 414 ps. The coordinates of the terminals are {(102, 98), (147, 153), (202, 202), (153, 249), (53, 147), (253, 153), (153, 52), (100, 203), (200, 103)}, and $\epsilon = 1.5$. The SPICE simulation assumes a generic CMOS design: MOSIS 2.0 μm CMOS technology, layout normalized to a 1 cm die, and 0.3 pF gate loading capacitance.

Because the circuit delay is determined by critical paths between primary inputs and primary outputs, Section VI below will generalize the BRMRT formulation to allow different ϵ_i parameter values to be associated with each sink $x_i \in N$ in a given net. To validate the use of the linear delay model, SPICE simulations for a number of routing examples were examined. As an example, Fig. 3 shows how the optimal delay routing indeed embodies a trade-off between the shortest path tree routing and the minimum-weight spanning tree routing.

III. A Bounded-Radius Minimum Spanning Tree Heuristic

In global routing for cell-based design, the distances between nodes are given by geometric distance, and the underlying routing graph is $G = (V, E)$ with $V = N$. For this case, many global routing methods are based on constructing a spanning tree for each net (e.g., see [3]). Therefore, the BRMRT problem becomes the *bounded radius minimum spanning tree* (BRMST) problem.

We now give a very natural and simple heuristic that finds a routing solution by growing a single component, following the general scheme of Prim's classical minimum spanning tree construction.

A. The Basic Algorithm

Our basic algorithm grows a tree $T = (V', E')$ which initially contains only the source s. At each step, we choose $x \in V'$ and $y \in N - V'$ such that $dist(x, y)$ is minimum. If adding (x, y) to T would not violate the radius constraint, i.e., $dist_T(s, x) + dist(x, y) \leq (1 + \epsilon) \cdot R$, we include the edge (x, y) in T. Otherwise, we "backtrace" along the path from x to s to find the first terminal x' such that (x', y) is *appropriate* (i.e., $dist_T(s, x') + dist(x', y) \leq R$), and add (x', y) to the tree. In the worst case, the backtracing will terminate with $x' = s$, since the edge (s, y) is always appropriate.

Note that in backtracing we could choose x' such that $dist_T(s, x') + dist(x', y) \leq (1 + \epsilon) \cdot R$. However, our choice of appropriate edges leads to fewer backtracing operations, while guaranteeing that backtracing is still always possible. In other words, we intentionally introduce some "slack" at y so that terminals within an ϵR neighborhood of y will not cause additional backtracing. Lim-

iting the amount of backtracing in this way will keep the cost of the resulting tree close to that of the minimum spanning tree.

We call this algorithm the Bounded Prim (BPRIM) construction. The high-level description is given in Fig. 4. This algorithm has several advantages. First, we can show that the radius of the resulting tree is never greater than the radius of the MST whenever the MST is unique.

Property 1: If the MST is unique, then $r(T_{BPRIM}) \leq r(T_{MST})$.

Proof: If $r(T_{MST}) \leq (1 + \epsilon) \cdot R$, then $r(T_{BPRIM}) = r(T_{MST})$ since the two trees will be identical. Otherwise, $r(T_{BPRIM}) \leq (1 + \epsilon) \cdot R < r(T_{MST})$ by construction. □

If the MST is *not* unique, then the radius of different minimum spanning trees can vary by an unbounded amount, and $r(T_{BPRIM})$ may be greater than $r(T_{MST})$; i.e., Property 1 will not hold for some choice of the MST. Fig. 5 shows a point set where a Prim-like minimum spanning tree algorithm may choose a connection to point y_1 instead of point x_1; or y_2 instead of x_2, etc., so that the tree radius is much greater than optimum. In this way, for some MST it may be possible for an unfortunate sequence of choices by BPRIM to yield $r(T_{BPRIM}) > r(MST)$ even though the two trees have identical cost. However, $r(T_{BPRIM})$ cannot be greater than the maximum possible MST radius.

With regard to total tree cost, we note that the difference between BPRIM and MST tree cost will depend on the parameter ϵ. In practice, most nets will have between two and four pins. Furthermore, it is unlikely that a single gate will be used to drive more than six gates in CMOS design. In this case, we can show that the cost of the resulting tree is within a small constant factor of the cost of the MST for nets of practical size. Table I gives the worst-case ratio of BPRIM cost over MST length for small values of $|N|$, as a function of ϵ.

Property 2: Let $B(\epsilon)$ be the worst-case ratio of the cost of BPRIM output to the MST cost. Then the bounds listed in Table I hold.

Proof: These results are obtained by studying the number of backtracings that can occur. We show the proof for $|N| = 5$. Other cases are similar.

Assume that the coordinates of the set of terminals have been scaled so that the set has unit radius, and let $cost(MST)$ be the cost of a minimum spanning tree. If backtracing occurs, then $cost(MST) \geq 1 + \epsilon$. Suppose that there is only one backtracing. Let $cost(e)$ be the cost of the edge which caused the backtracing. Then

$$B(\epsilon) \leq \frac{cost(MST) - cost(e) + 1}{cost(MST)}$$

$$\leq 1 + \frac{1}{cost(MST)} \leq 1 + \frac{1}{1 + \epsilon} = \frac{2 + \epsilon}{1 + \epsilon}.$$

If backtracing occurs twice, let $cost(x)$ and $cost(y)$ be the costs of the edges which cause the backtracings. Then,

$T = (V', E') = (\{s\}, \emptyset)$
while $|V'| < |N|$
  Select two terminals $x \in V'$ and $y \in N - V'$ minimizing $dist(x, y)$
  if $dist_T(s, x) + dist(x, y) \leq (1 + \epsilon) \cdot R$ **then** $x' = x$
  else find the first terminal x' along the path in T from x to s
    such that $dist_T(s, x') + dist(x', y) \leq R$
  $V' = V' \cup \{x'\}$
  $E' = E' \cup \{(x', y)\}$

Fig. 4. Algorithm BPRIM: computing a bounded-radius spanning tree, T, for a given set of terminals, N, with source, $s \in N$, and radius, R, using parameter ϵ.

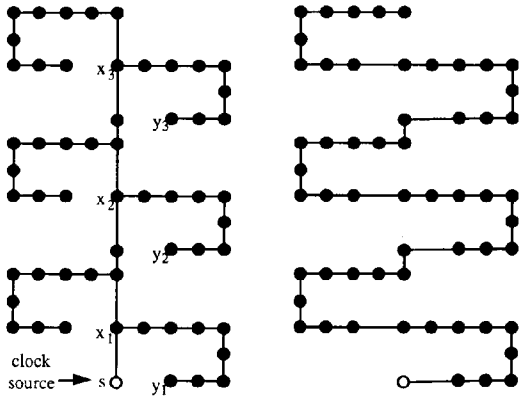


Fig. 5. An example where the radius of the routing tree (MST) produced by a Prim-like construction (*right*) is arbitrarily larger than a minimum-radius MST (*left*).

TABLE I
ANALYSIS FOR SMALL NETS IN THE MANHATTAN PLANE

Net size	Bound $B(\epsilon)$	$\epsilon = 0$	$\epsilon = \frac{1}{2}$	$\epsilon = 1$
$\|N\| = 2$	1	1	1	1
$\|N\| = 3$	$\frac{2}{1+\epsilon}$	2	$\frac{4}{3}$	1
$\|N\| = 4$	$\max(\frac{2+\epsilon}{1+\epsilon}, \frac{3}{1+2\epsilon})$	3	$\frac{5}{3}$	$\frac{3}{2}$
$\|N\| = 5$	$\max(\frac{3+\epsilon}{1+\epsilon}, \frac{4}{1+3\epsilon})$	4	$\frac{7}{3}$	2
$\|N\| = 6$	$\max(\frac{4+\epsilon}{1+\epsilon}, \frac{5}{1+4\epsilon})$	5	3	$\frac{5}{2}$

$$B(\epsilon) \leq \frac{cost(MST) - cost(x) - cost(y) + 2}{cost(MST)}$$

$$\leq 1 + \frac{2}{cost(MST)} \leq 1 + \frac{2}{1 + \epsilon} = \frac{3 + \epsilon}{1 + \epsilon}.$$

If backtracing occurs three times, the tree produced by BPRIM is a star graph. Moreover, in this case, it is easy to see that $cost(MST) \geq 1 + 3\epsilon$. Thus,

$$B(\epsilon) \leq \frac{4}{cost(MST)} \leq \frac{4}{1 + 3\epsilon}.$$

Therefore,

$$B(\epsilon) \leq \max\left(\frac{2+\epsilon}{1+\epsilon}, \frac{3+\epsilon}{1+\epsilon}, \frac{4}{1+3\epsilon}\right)$$
$$= \max\left(\frac{3+\epsilon}{1+\epsilon}, \frac{4}{1+3\epsilon}\right). \quad \square$$

In fact, the experimental results of Section V show that $B(\epsilon)$ is still bounded by a small constant even for very large nets (i.e., see the tables of Appendix I). However, examples exist which show that the worst-case performance ratio of BPRIM is not bounded by any constant for any value of ϵ.

Theorem 1: For any ϵ there exists a net for which BPRIM will have an arbitrarily large performance ratio.

Proof: On the net shown in Fig. 6, BPRIM will have an unbounded performance ratio. The optimal solution is shown on the left, where all source–leaf path lengths are equal to R. Terminal y is situated so that the path length from the source to any leaf via y is slightly greater than $(1 + \epsilon) \cdot R$. This will cause the BPRIM construction to backtrace from *all* of the leaves back to the source, yielding an unbounded performance ratio. If ϵ is large, y can be replaced by a long path of many closely spaced terminals so that BPRIM creates a long path between s and x; this yields the arbitrarily large performance ratio for *any* value of ϵ. $\square$

The time complexity of BPRIM is $O(n^2)$, and there are instances where this bound is tight, since each new terminal can force examination of most of the terminals that have already been added to the tree.

B. Extensions of BPRIM

As it turns out, the bounded-radius construction can also be applied to minimum spanning tree methods other than Prim's algorithm. A more general algorithm template is given in Fig. 7. This general template gives rise to a number of distinct variants, depending upon how the pair of terminals x and y are selected inside the inner loop. Several variants give significant performance improvements over the BPRIM algorithm:

- H1—Find x and y as in BPRIM, and select the terminal x' along the path in T from x to s which yields a *minimum-length* appropriate edge (x', y); add (x', y) to T.
- H2—Find a terminal $y \in N - V'$ minimizing $dist(x, y)$ for any $x \in V'$, and select the terminal $x' \in V'$ which yields a minimum-length appropriate edge (x', y); add (x', y) to T.
- H3—Find a pair of terminals $x \in V'$ and $y \in N - V'$ that yield a minimum-length appropriate edge (x, y); add (x, y) to T.

Property 1 also holds for each of the variants H1, H2, and H3. The time complexity of variants H1 and H2 is $O(n^2)$, while variant H3 can be easily implemented within time $O(n^3)$. Empirical results of the BPRIM method are very promising, as can be seen in Appendix I. However, Fig. 6 shows that BPRIM will also have unbounded worst-case performance ratio. Thus, the next section develops a new, provably good approach to performance-driven global routing based on a combination of minimum spanning tree and shortest path tree constructions.

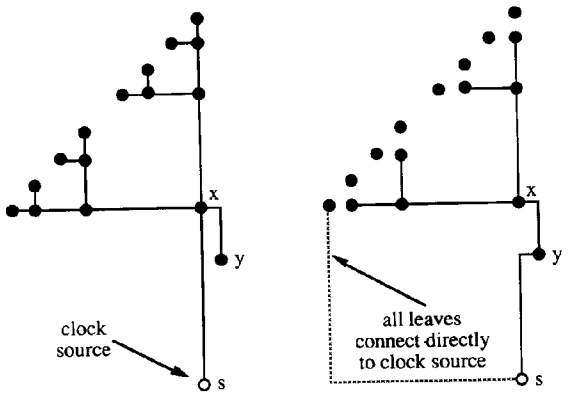


Fig. 6. Example where the performance ratio of the algorithm is not bounded by any constant for any ϵ. The optimal solution is shown on the left, while the BPRIM output is shown on the right.

$T = (V', E') = (\{s\}, \emptyset)$
while $|V'| < |N|$
  Select two terminals $x \in V'$ and $y \in N - V'$,
    with $dist_T(s, x) + dist(x, y) \leq (1 + \epsilon) \cdot R$
  $V' = V' \cup \{x\}$
  $E' = E' \cup \{(x, y)\}$

Fig. 7. A more general BPRIM template: computing a bounded-radius spanning tree T for a given set of terminals N with source $s \in N$ and radius R, using parameter ϵ.

IV. Bounded-Radius Spanning Tree Global Routing

The basic idea of our provably good bounded-radius minimum spanning tree algorithm is to construct a subgraph Q which spans N and has both small total cost and small radius. Then the shortest path tree of Q will also have small cost and radius, and will correspond to a good routing solution. We again use the routing graph $G = (V, E)$, with $V = N$. Our algorithm is as follows.

- Compute the shortest path tree SPT_G of G, and compute the minimum spanning tree MST_G of G. Also, initialize the graph Q to be equal to MST_G.
- Let L be the sequence of vertices corresponding to a depth-first tour of MST_G, where each edge of MST_G is traversed exactly twice (see Fig. 8). The total edge cost of this tour is twice that of MST_G.
- Traverse L while keeping a running total S of traversed edge costs. As this traversal reaches each node L_i, check whether S is greater than $\epsilon \cdot dist_G(s, L_i)$.

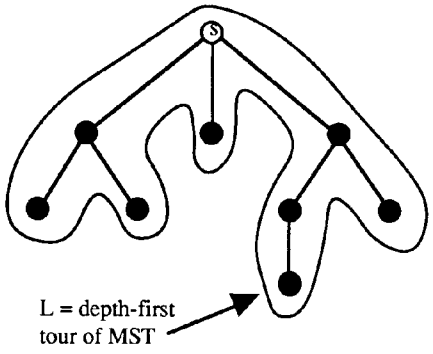


Fig. 8. A spanning tree and a depth-first tour.

If so, reset S to 0 and merge $minpath_G(s, L_i)$ into Q. Continue traversing L while repeating this process.

- Our final routing tree is SPT_Q, the shortest path tree over Q.

A formal description of the algorithm is given in Fig. 9.

We now prove that for any fixed ϵ this algorithm produces a routing tree with radius and total cost each simultaneously bounded by a constant times optimum:

Theorem 2: For any weighted graph G and parameter ϵ, the routing tree T constructed by our algorithm has radius $r(T) \leq (1 + \epsilon) \cdot R$.

Proof: For any $v \in V$, let v_{i-1} be the last node before v on the MST traversal L for which we added $minpath_G(s, v_{i-1})$ to Q in the algorithm, as shown in Fig. 10. By the construction of the algorithm, we know that $dist_L(v_{i-1}, v) \leq \epsilon \cdot R$. We then have

$$\begin{aligned} dist_T(s, v) &\leq dist_T(s, v_{i-1}) + dist_L(v_{i-1}, v) \\ &\leq dist_G(s, v_{i-1}) + \epsilon \cdot R \\ &\leq R + \epsilon \cdot R = (1 + \epsilon) \cdot R. \end{aligned}$$ □

Theorem 3: For any weighted graph G and parameter ϵ, the routing tree T constructed by our algorithm has $cost(T) \leq (1 + (2/\epsilon)) \cdot cost(MST_G)$.

Proof: Let $v_1, v_2, \cdots, v_m$ be the set of nodes to which the algorithm added shortest paths from the source node, and let $v_0 = s$. We have

$$cost(T) \leq cost(MST_G) + \sum_{i=1}^{m} dist_G(s, v_i).$$

since T is a subtree of the union of the MST with all of the added shortest paths. By the algorithm construction $dist_L(v_{i-1}, v_i) \geq \epsilon \cdot dist_G(s, v_i)$, and so we obtain

$$\begin{aligned} cost(T) &\leq cost(MST_G) + \sum_{i=1}^{m} \frac{1}{\epsilon} \cdot dist_L(v_{i-1}, v_i) \\ &\leq cost(MST_G) + \frac{1}{\epsilon} \cdot cost(L). \end{aligned}$$

compute MST_G and SPT_G
$Q = MST_G$
L = depth-first tour of MST_G
$S = 0$
for $i = 1$ to $|L| - 1$
  $S = S + cost(L_i, L_{i+1})$
  if $S \geq \epsilon \cdot dist_G(s, L_{i+1})$ **then**
    $Q = Q \cup minpath_G(s, L_{i+1})$
    $S = 0$
T = shortest path tree of Q

Fig. 9. Computing a bounded-radius spanning tree T for $G = (V, E)$, with source $s \in V$ and radius R, using parameter ϵ. T will have radius at most $(1 + \epsilon) \cdot R$, and cost at most $(1 + (2/\epsilon)) \cdot cost(MST_G)$.

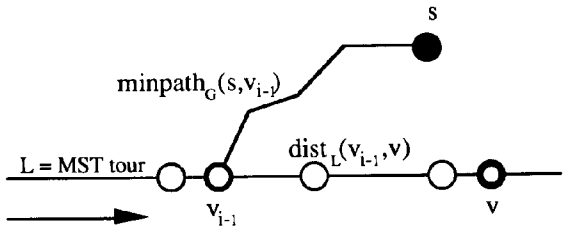


Fig. 10. Depiction of the bounded-radius construction.

Since $cost(L) \leq 2 \cdot cost(MST_G)$, we have

$$\begin{aligned} cost(T) &\leq \text{cost}(MST_G) + \frac{2}{\epsilon} \cdot cost(MST_G) \\ &= \left(1 + \frac{2}{\epsilon}\right) \cdot cost(MST_G). \end{aligned}$$ □

Because our method yields a bounded-radius, bounded-cost routing tree, we call this the BRBC algorithm. Theorem 3 suggests that for $\epsilon = 0$, the ratio $cost(T)/cost(MST_G)$ is not bounded by any constant; indeed, this is true for the example of Fig. 1 above, where $cost(T)/cost(MST_G)$ is $\Omega(|N|)$. A similar idea was recently used in the distributed computation literature by Awerbuch, Baratz, and Peleg [1] for constructing spanning trees with small diameter and small weight. However, our algorithm treats the bounded-radius minimum spanning tree problem, while they treat the tree diameter instead. Moreover, our method involves a simpler construction with tighter performance bounds.

V. Bounded-Radius Steiner Tree Global Routing

In the previous section, we treated the bounded-radius minimum spanning tree problem, where each net N is routed in an underlying routing graph $G = (V, E)$, with $V = N$. As noted earlier, the spanning tree routing has proved useful in cell-based design. In this section we treat the more general version of the problem, where Steiner points are allowed.

A. An Algorithm for Arbitrary Weighted Graphs With Steiner Points

For building-block design, the underlying routing graph is based on the channel intersection graph [4], and a net N is a subset of the vertices of G. In this case, the BRMRT problem is actually the *bounded radius optimal Steiner tree* (BROST) problem, and the channel intersection points (i.e., the nodes in $V - N$) are potential Steiner points. The following is immediate:

Lemma 1: The BROST problem in NP-complete.

Proof: Setting $\epsilon = \infty$ yields the graph Steiner problem, which is known to be NP-complete [12]. □

Hence, in the BROST problem, even the construction of a "minimum spanning tree" for N in G is equivalent to the Steiner problem in graphs. This means that if we are to apply our graph version of the BPRIM algorithm to the BROST problem and still maintain polynomial complexity, we have to be satisfied with an approximation to the minimum-cost tree spanning N (i.e., a Steiner tree) within G.

Recall that in applying the BRBC algorithm to general graphs, the only reason we use the MST is to obtain a reasonably short tour of the vertices. Toward this end, *any* tour of the vertices will suffice (e.g., traveling salesman, Chinese postman). In constructing this tour we are not even restricted to visiting each node at most once, just as long as every node is visited at least once, and the total cost of the tour is still reasonably small.

Our approximation algorithm for the bounded-radius optimal Steiner tree problem is similar to the algorithm presented in Section IV. Note that given any approximate Steiner tree $\hat{T}$, we can use the approach of Section IV to construct a routing tree with radius within $(1 + \epsilon) \cdot r(\hat{T})$ and cost within $(1 + (2/\epsilon)) \cdot cost(\hat{T})$. Our algorithm uses a heuristic from Kou, Markovsky and Berman (KMB) [17], [26] to build a Steiner tree $\hat{T} = T_{KMB}$ in the underlying routing graph, with T_{KMB} having cost within a factor 2 of optimal.[2]

We construct a depth-first tour of the heuristic Steiner tree T_{KMB}. Next, we traverse the tour, adding to T_{KMB} the shortest paths from the source to the appropriate vertices of the tour, as in Section IV. Finally, we compute the shortest path tree in the resulting graph and output the union of the shortest paths from the source to all terminals in N (which includes intermediate nonterminal nodes on the shortest paths as Steiner points). Note that the cost of the tour will be at most four times the *optimal* Steiner tree (T_{opt}) cost. Thus, the resulting routing tree cost is at most $2 \cdot (1 + (2/\epsilon))$ times optimal.

[2]Given a graph $G = (V, E)$ and a net of terminals $N \subseteq V$, the method of Kou, Markovsky, and Berman is as follows. First, construct the complete graph over N with each edge weight equal to the cost of the corresponding shortest path in G. Compute T, the minimum spanning tree of this complete graph, and expand each edge of T into the corresponding shortest path, yielding a subgraph G' that spans N. Finally, compute the minimum spanning tree T' of G', and delete edges from T' until all leaves are nodes of N. Output the resulting tree.

Theorem 4: For any weighted graph $G = (V, E)$, node subset $N \subseteq V$, and parameter ϵ, the routing tree T constructed by our algorithm has radius $r(T) \leq (1 + \epsilon) \cdot R$, and $cost(T) \leq 2 \cdot (1 + (2/\epsilon)) \cdot cost(T_{opt})$.

Proof: By our previous arguments, $r(T) \leq (1 + \epsilon) \cdot R$. In addition, $cost(T) \leq (1 + (2/\epsilon)) \cdot cost(T_{KMB})$, where T_{KMB} is the approximate Steiner tree produced by the method of [17]. Since $cost(T_{KMB}) \leq 2 \cdot cost(T_{opt})$, we have $cost(T) \leq 2 \cdot (1 + (2/\epsilon)) \cdot cost(T_{opt})$. □

B. Geometry Helps in Routing

If we are routing in a metric space and are allowed to introduce arbitrary Steiner points to reduce the routing cost/diameter, we can slightly modify the basic algorithm (of Fig. 9) to introduce Steiner points on the tour L whenever $S = 2\epsilon \cdot R$. From each of these Steiner points we construct shortest paths to the source and add them to Q as in the original algorithm. Thus, each node in the traversal of L will be within $\epsilon \cdot R$ of a Steiner point, i.e., within $(1 + \epsilon) \cdot R$ of the source. In this case, we can show that the same radius bound is maintained

Theorem 2': In the geometric plane, for a given parameter ϵ the routing tree T constructed by our algorithm has radius $r(T) \leq (1 + \epsilon) \cdot R$. □

At the same time, we can show that the cost of the routing tree will be reduced to the following:

Theorem 3': In the geometric plane, for a given parameter ϵ the routing tree T constructed by our algorithm has $cost(T) \leq 2 \cdot (1 + (1/\epsilon)) \cdot cost(T_{opt})$. □

The proofs of these two results are similar to those of Theorems 2 and 3.

In addition, well-known results which bound the MST/Steiner ratio in various geometries can be used with Theorem 4 and the above scheme to yield even better bounds whenever the edge weights correspond to a metric (e.g., Manhattan or Euclidean). To illustrate how these observations can be combined to yield improved bounds for Steiner routing in metric spaces, we give two immediate examples.

Corollary 1: Given a set of terminals N in the Manhattan plane and a real parameter ϵ, our algorithm will produce a routing tree T with $r(T)$ bounded by $(1 + \epsilon)$ times optimal and with cost bounded by $(3/2) \cdot (1 + (1/\epsilon))$ times optimal.

Proof: By a result of Hwang [11], the rectilinear minimum spanning tree gives a $3/2$ approximation to the rectilinear optimal Steiner tree. We then apply arguments similar to those of Theorems 2 and 3. □

Corollary 2: Given a set of terminals N in the Euclidean plane and a real parameter ϵ, our algorithm will produce a routing tree T with $r(T)$ bounded by $(1 + \epsilon)$ times optimal and with cost bounded by $(2/\sqrt{3}) \cdot (1 + (1/\epsilon))$ times optimal.

Proof: By a recent result of Du and Hwang [7], the Euclidean minimum spanning tree gives a $2/\sqrt{3}$ approx-

imation to the Euclidean optimal Steiner tree. We again apply the arguments of Theorems 2 and 3. □

Note that this result generalizes when we have increased flexibility in the wiring geometry, e.g., 30°–60°–90° wiring instead of rectilinear. By applying a recent result [24] for λ geometries (allowing angles $i\pi/\lambda$), a cost bound of $(2/\sqrt{3}) \cos(\pi/\lambda) \cdot (1 + (1/\epsilon))$ may be established. When λ approaches ∞, this bound approaches the bound of the corollary above.

VI. Generalization to Nonuniform Values of ε

Often we may wish to use varying wire length constraints on the different source–sink paths within a given signal net, since timing in VLSI circuits is actually path-dependent rather than net-dependent. For example, a source–sink connection on a timing-critical path will require a small value of ε, whereas for a connection not on any critical path, we may allow large ε in order to reduce total wire length. This yields the following generalization of the BRMRT formulation:

The Nonuniform Bounded Radius Minimum Routing Tree (NBRMRT) Problem: Given parameters $\epsilon_i \geq 0$ associated with each sink terminal t_i of a signal net having source s and radius R, find a minimum-cost routing tree T such that $dist_T(s, t_i) \leq (1 + \epsilon_i) \cdot R$ for each t_i.

In this section, we extend our method to handle this case, and establish constant-factor bounds on both wire length cost and radius of the routing solution. Although we restrict the discussion to spanning tree routing, extensions to (geometric) Steiner routing are straightforward, using the techniques of Sections IV and V.

To handle a different path length constraint ϵ_i for each terminal t_i in the net N, we modify the original algorithm of Fig. 9 by changing the conditional inside the loop from $S \geq \epsilon \cdot dist_G(s, L_{i+1})$ to $S \geq \epsilon_{i+1} \cdot dist_G(s, L_{i+1})$. An argument identical to that in the proof of Theorem 2 yields the following bound on the pathlengths:

Lemma 2: For an arbitrary weighted graph G with source s and radius R, and a set of terminal radius parameters $\epsilon_1, \epsilon_2, \cdots, \epsilon_{|N|}$, our modified algorithm constructs a routing tree T such that $dist_T(s, t_i) \leq (1 + \epsilon_i) \cdot R$ for each terminal t_i. □

Clearly, by arguments similar to those used earlier, our modified algorithm constructs a routing tree T with $cost(T) \leq (1 + 2/\min(\epsilon_1, \epsilon_2, \cdots, \epsilon_{|N|})) \cdot cost(MST_G)$. However, it is possible to improve this bound, as follows. Without loss of generality, we can assume that all of the ϵ_i's are sorted in nondecreasing order: $\epsilon_1 \leq \epsilon_2 \leq \cdots \leq \epsilon_{|N|}$. Let $\epsilon = \max_{i=1}^{|N|} \epsilon_i$, and define $k = \lceil 2 \cdot cost(MST_G)/((1 + \epsilon) \cdot R) \rceil$.

Lemma 3: For any weighted graph G and a set of terminal radius parameters $\epsilon_1 \leq \epsilon_2 \leq \cdots \leq \epsilon_{|N|}$, our modified algorithm constructs a routing tree T with $cost(T) \leq (1 + (k/(k-1)) \cdot 2/HM(\epsilon_1, \epsilon_2, \cdots, \epsilon_k)) \cdot cost(MST_G)$, where HM denotes harmonic mean.

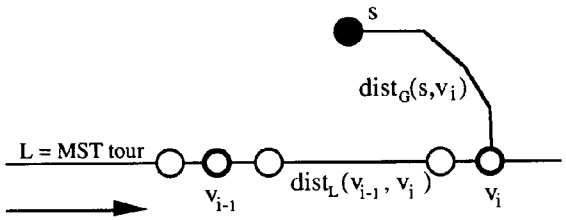


Fig. 11. Tree construction using nonuniform values of ε.

Proof: Let $v_1, v_2, \cdots, v_m$ be the set of nodes to which the algorithm added shortest paths from the source node, as shown in Fig. 11. As usual, the routing tree produced by our modified algorithm is a subtree of Q, the union of MST_G and the added shortest paths. The routing tree cost is therefore bounded by $cost(Q) = cost(MST) + \sum_{i=1}^{m} dist_G(s, v_i) \leq cost(MST) + \sum_{i=1}^{m} 1/\epsilon_i \, dist_L(v_{i-1}, v_i)$. Let l_i denote $dist_L(v_{i-1}, v_i)$. By the construction, we have $l_i \geq \epsilon_i \cdot dist_G(s, v_i)$. Because no edge length is greater than R, $l_i \leq (1 + \epsilon) \cdot R$, and $\sum_{i=1}^{m} l_i = 2 \cdot cost(MST_G) \leq k \cdot (1 + \epsilon) \cdot R$. Therefore,

$$\sum_{i=1}^{m} \frac{1}{\epsilon_i} dist_L(v_{i-1}, v_i) = \sum_{i=1}^{m} \frac{l_i}{\epsilon_i} \leq \sum_{i=1}^{k} \frac{(1 + \epsilon) \cdot R}{\epsilon_i}$$

since $l_i \leq (1 + \epsilon) \cdot R$, $\sum_{i=1}^{m} l_i \leq k \cdot (1 + \epsilon) \cdot R$, and the ϵ_i's are in sorted order. Factoring out $(1 + \epsilon)$ and using the definition of k, we obtain

$$= (1 + \epsilon) \cdot \sum_{i=1}^{k} \frac{1}{\epsilon_i} \cdot R$$

$$\leq (1 + \epsilon) \cdot \sum_{i=1}^{k} \frac{1}{\epsilon_i} \cdot \frac{2 \cdot cost(MST_G)}{(1 + \epsilon) \cdot (k - 1)}.$$

Canceling $(1 + \epsilon)$, multiplying by k/k, and regrouping, we get

$$\leq \frac{k}{k-1} \cdot \frac{\sum_{i=1}^{k} \frac{1}{\epsilon_i}}{k} \cdot 2 \cdot cost(MST_G)$$

$$= \frac{k}{k-1} \cdot \frac{1}{HM(\epsilon_1, \epsilon_2, \cdots, \epsilon_k)} \cdot 2 \cdot cost(MST_G).$$

It follows that

$$cost(Q) \leq cost(MST) + \frac{k}{k-1} \cdot \frac{1}{HM(\epsilon_1, \epsilon_2, \cdots, \epsilon_k)} \cdot 2 \cdot cost(MST_G)$$

$$= \left(1 + \frac{k}{k-1} \cdot \frac{2}{HM(\epsilon_1, \epsilon_2, \cdots, \epsilon_k)}\right) \cdot cost(MST_G)$$

□

These results are summarized as follows:

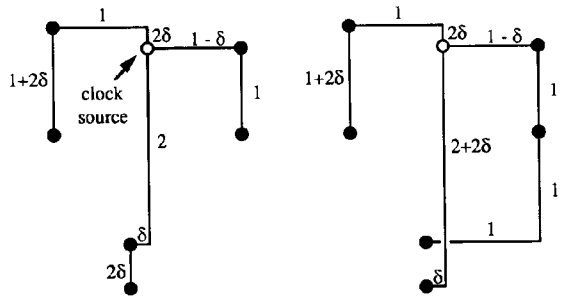


Fig. 12. Example where BPRIM outperforms variants H2 and H3; here δ is a very small real number and $\epsilon = (2 - 3\delta)/(2 + 3\delta)$.

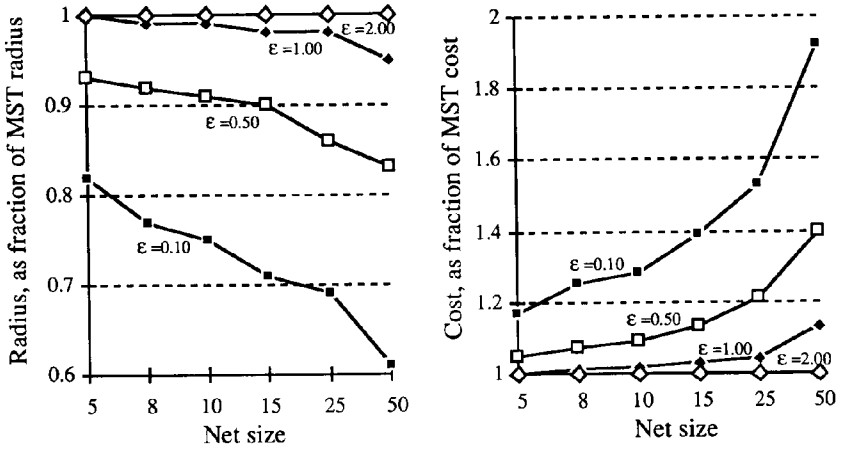


Fig. 13. These charts illustrate the smooth trade-off between total routing cost and maximum signal delay produced by the BPRIM algorithm. The parameter ϵ determines the trade-off between the shortest path tree and the minimum spanning tree.

Theorem 5: For any weighted graph, G, and a set of terminal radius parameters $\epsilon_1 \leq \epsilon_2 \leq \cdots \leq \epsilon_{|N|}$, our modified algorithm constructs a routing tree T with each terminal t_i having $dist_T(s, t_i) \leq (1 + \epsilon_i) \cdot R$, and with $cost(T) \leq (1 + (k/(k-1)) \cdot 2/HM(\epsilon_1, \epsilon_2, \cdots, \epsilon_k)) \cdot cost(MST_G)$, where $k = \lceil 2 \cdot cost(MST_G)/((1 + \epsilon) \cdot R) \rceil$ and HM denotes harmonic mean. □

VII. Experimental Results

The BPRIM algorithm and variants H1, H2, and H3, as well as the approximation algorithms for bounded-radius minimum spanning tree routing and for bounded-radius optimal Steiner tree routing, were implemented in ANSI C for the Sun-4, Macintosh, and IBM environments; code is available from the authors upon request.

The BPRIM algorithm and variants H1, H2, and H3 were tested on a large number of random nets of up to 50 terminals, generated from a uniform distribution in the 1000×1000 grid. These are standard testbeds which capture the statistical properties of signal nets in actual layouts. As noted in, e.g., [16], any set of approximation heuristics induces a *meta-heuristic* which returns the best solution found by any heuristic in the set and has asymptotic complexity equal to that of the slowest heuristic; we implemented the meta-heuristic over BPRIM, H1, H2 and H3, denoted by Meta (BPRIM, H1, H2, H3). Here, Meta (BPRIM, H1, H2, H3) returns the routing tree with minimum cost.

Although there exist examples where the BPRIM algorithm outperforms the more complicated variants (e.g., see Fig. 12), the data shown in Table III in Appendix I indicate that, *on average*, variant H1 dominates BPRIM, H2 dominates H1, and H3 dominates H2. Fig. 13 shows that the BPRIM approach produces a very smooth trade-off between routing cost and tree radius.

The BRBC algorithm for spanning tree routing was tested on a large number of random nets generated from a uniform distribution in the grid. Results are summarized in Fig. 14, which clearly shows the trade-off between routing cost and maximum delay. As ϵ decreases, both the cost and radius curves shift monotonically from that of the minimum spanning tree to that of the shortest path

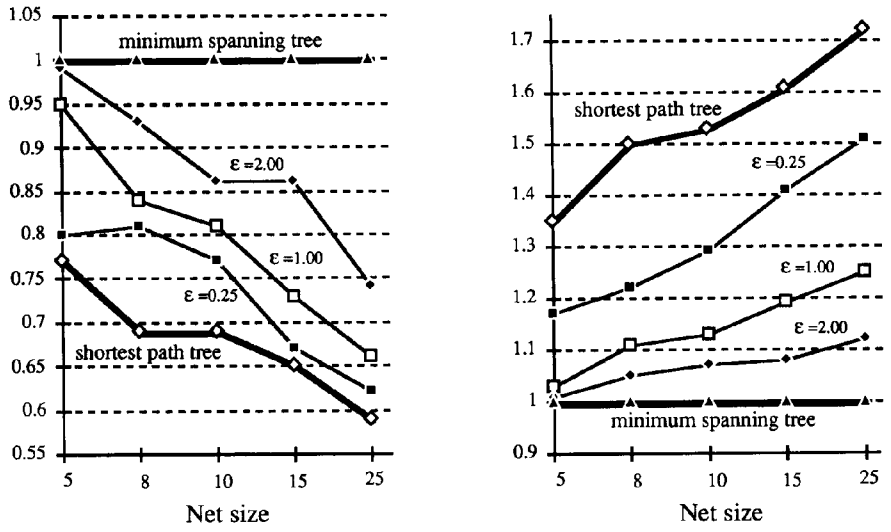


Fig. 14. These charts illustrate the smooth trade-off produced by our algorithm between total routing cost and maximum signal delay. In both cases the envelope of performance lies between the shortest path tree and the minimum spanning tree, and the parameter ϵ determines the exact trade-off.

tree. More detailed data is given in Table IV in Appendix II.

The BRBC algorithm for Steiner tree routing was tested on random block layouts in the grid; these were generated by adding a fixed number of nonoverlapping blocks, with length, width, and lower-left coordinates all uniformly distributed. Given a block design, nets with terminals on the block peripheries were routed within the corresponding channel intersection graph. An example of the output from our algorithm is shown in Fig. 15.

A detailed summary of experimental results for Steiner routing in block designs is contained in Table V in Appendix II. Once again, the simulations confirm the trade-offs inherent in the bounded-radius routing approach. Note that although our construction starts with the heuristic Steiner tree of Kou, Markovsky, and Berman, our routing solution may in some cases have smaller cost than the KMB tree. In all cases, the radius of our routing tree is no larger than that of the KMB tree. This too is reflected in the experimental data. Finally, SPICE was used to compare routings produced by our algorithm with MST routings for selected nets, as noted earlier in Fig. 3.

From Tables II, III, and IV, we observe the following. For any given value of ϵ, the BPRIM approach, being inherently greedy, will yield a routing solution with radius approaching $(1 + \epsilon) \cdot R$, but with small tree weight. On the other hand, the BRBC approach, being more conservative, will yield a routing solution with radius noticeably smaller than $(1 + \epsilon) \cdot R$, but at the expense of slightly larger tree cost. Therefore, the BRBC algorithm will have a slightly shifted cost–radius curve compared with the BPRIM algorithm. In practice, the asymptotic efficiency of implementation and the provably good output provide compelling reasons to adopt the BRBC algorithm, rather than the BPRIM approach.

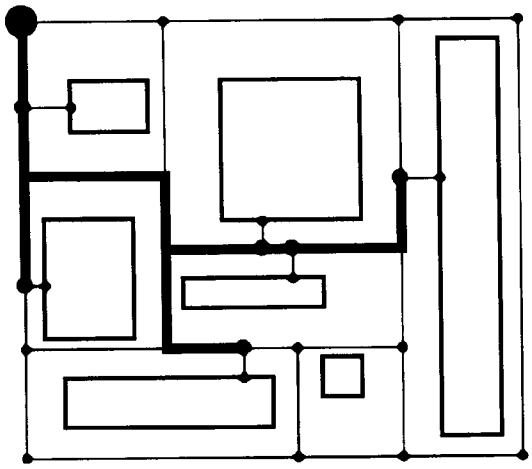


Fig. 15. A set of placed modules and their channel intersection graph. The highlighted tree is the routing produced by our algorithm.

VIII. Conclusions

We have proposed a new bounded-radius minimum spanning tree formulation for timing-driven global routing in both cell-based and building-block design. An effective method based on an analog of Prim's minimum spanning tree construction is given. Furthermore, we have also proposed a new, provably good general algorithm for timing-driven global routing. This method is based on a routing tree construction where *both* the total wire length and the maximum delay of the routing are bounded by

TABLE II
MINIMUM, AVERAGE, AND MAXIMUM RADIUS RATIOS FOR VARIOUS VALUES OF ϵ.

	net	BPRIM			H1			H2			H3			Meta		
ϵ	size	min	ave	max	min	ave	max	min	ave	max	min	ave	max	min	ave	max
0.10	5	0.42	0.82	1.00	0.42	0.82	1.00	0.42	0.82	1.00	0.42	0.82	1.00	0.42	0.82	1.00
0.10	8	0.26	0.77	1.00	0.26	0.77	1.00	0.28	0.77	1.00	0.28	0.77	1.00	0.28	0.77	1.00
0.10	10	0.36	0.75	1.00	0.36	0.74	1.00	0.36	0.75	1.00	0.36	0.75	1.00	0.36	0.75	1.00
0.10	15	0.34	0.71	1.00	0.34	0.71	1.00	0.34	0.71	1.00	0.34	0.71	1.00	0.34	0.71	1.00
0.10	25	0.33	0.69	1.00	0.33	0.68	1.00	0.33	0.69	1.00	0.32	0.69	1.00	0.32	0.69	1.00
0.10	50	0.30	0.61	0.99	0.30	0.61	0.99	0.31	0.61	0.99	0.30	0.61	0.99	0.30	0.61	0.99
0.50	5	0.41	0.93	1.00	0.41	0.92	1.00	0.41	0.92	1.00	0.41	0.92	1.00	0.41	0.92	1.00
0.50	8	0.48	0.92	1.00	0.33	0.92	1.00	0.33	0.92	1.00	0.44	0.92	1.00	0.44	0.92	1.00
0.50	10	0.46	0.91	1.00	0.45	0.90	1.00	0.45	0.90	1.00	0.48	0.90	1.00	0.48	0.90	1.00
0.50	15	0.44	0.90	1.00	0.44	0.89	1.00	0.44	0.89	1.00	0.41	0.89	1.31	0.41	0.89	1.31
0.50	25	0.38	0.86	1.00	0.37	0.86	1.00	0.37	0.86	1.00	0.37	0.86	1.07	0.37	0.86	1.07
0.50	50	0.39	0.83	1.00	0.39	0.83	1.00	0.38	0.82	1.00	0.39	0.82	1.04	0.39	0.82	1.04
1.00	5	0.58	1.00	1.00	0.58	0.99	1.00	0.58	0.99	1.00	0.58	0.99	1.00	0.58	0.99	1.00
1.00	8	0.67	0.99	1.00	0.56	0.99	1.00	0.56	0.99	1.00	0.53	0.99	1.00	0.53	0.99	1.00
1.00	10	0.65	0.99	1.00	0.57	0.98	1.00	0.57	0.98	1.00	0.57	0.98	1.00	0.57	0.98	1.00
1.00	15	0.65	0.98	1.00	0.54	0.98	1.00	0.54	0.97	1.00	0.54	0.97	1.06	0.54	0.97	1.06
1.00	25	0.48	0.98	1.00	0.48	0.97	1.00	0.48	0.97	1.00	0.46	0.97	1.10	0.46	0.97	1.10
1.00	50	0.53	0.95	1.00	0.53	0.94	1.00	0.53	0.94	1.00	0.53	0.94	1.06	0.53	0.94	1.06
2.00	5	1.00	1.00	1.00	1.00	1.00	1.00	0.69	1.00	1.00	0.69	1.00	1.00	0.69	1.00	1.00
2.00	8	0.86	1.00	1.00	0.80	1.00	1.00	0.67	1.00	1.00	0.67	1.00	1.00	0.67	1.00	1.00
2.00	10	0.96	1.00	1.00	0.96	1.00	1.00	0.78	1.00	1.00	0.78	1.00	1.18	0.78	1.00	1.18
2.00	15	1.00	1.00	1.00	1.00	1.00	1.00	0.85	1.00	1.00	0.85	1.00	1.10	0.85	1.00	1.10
2.00	25	0.84	1.00	1.00	0.84	1.00	1.00	0.81	1.00	1.00	0.69	1.00	1.26	0.69	1.00	1.26
2.00	50	0.82	1.00	1.00	0.82	1.00	1.00	0.78	1.00	1.00	0.58	0.99	1.16	0.58	0.99	1.16

constant factors away from optimal. Our approach readily extends to Steiner tree routing in arbitrary weighted graphs, where again the routing tree is only a small constant factor away from optimal in terms of both cost and radius. Extensive simulations over geometric routing graphs as well as channel intersection graphs derived from random block designs confirm that our approach gives very good performance. The results of Section VII indeed exhibit a smooth trade-off between the competing requirements of minimum delay and minimum total wire length.

Based on our methods for constructing bounded-radius routing trees, the global routing procedure will work as follows. We route all nets, one by one, according to their priorities. For each net, we construct a bounded-radius minimum spanning tree or bounded-radius minimum Steiner tree using the algorithms presented in Sections IV and V. The parameter ϵ is either given by the user or computed based on an estimation of the timing constraints for the net. As noted in Section VI, different values of ϵ_i can be used within a single net to reflect timing constraints in various input–output paths. The cost of each edge in the routing graph is a function of wire lengths, channel capacities, and the distribution of current channel densities. After routing each net, we update the edge costs in the routing graph. After all nets are routed, we may compute the timing-critical paths and, if necessary, further reduce the interconnection delay by rerouting some critical nets based on more accurate distributed *RC* delay models.

Our algorithms readily extend to other norms and to alternate geometries (e.g., 45° or 30°–60°–90° routing regimes). There are several remaining open problems, such as the complexity of computing the minimum cost bounded-radius spanning tree in the Manhattan plane or the complexity of choosing an MST with minimum radius when the MST is not unique.

APPENDIX I
BPRIM AND ITS VARIANTS

A. Experimental Data for Ratios of Heuristic Tree Radius to MST Radius

Table II gives the minimum, maximum, and average ratios of the heuristic tree radius to the MST radius, as computed by the BPRIM algorithm and its variants H1, H2, H3, and Meta (BPRIM, H1, H2, H3). The data shown represent averages of 500 cases generated from a uniform distribution in the unit square. The source node was selected to be one of the terminals at random. Note that the radius of the Meta (BPRIM, H1, H2, H3) solution may be larger than the radius produced by any single method, because the meta-heuristic is selecting the lowest-cost routing tree.

B. Experimental Data for Ratios of Heuristic Tree Cost to MST Cost

Table III gives the minimum, maximum, and average ratios of the heuristic tree cost to the MST cost, as computed by the BPRIM algorithm and its variants H1, H2, H3, and Meta (BPRIM, H1, H2, H3). The data shown represent averages of 500 cases generated from a uniform distribution in the unit square. The source node was selected to be one of the terminals at random.

TABLE III
MINIMUM, AVERAGE, AND MAXIMUM COST RATIOS FOR VARIOUS VALUES OF ϵ.

ϵ	net size	BPRIM			H1			H2			H3			Meta		
		min	ave	max	min	ave	max	min	ave	max	min	ave	max	min	ave	max
0.10	5	1.00	1.17	2.22	1.00	1.17	2.22	1.00	1.17	2.22	1.00	1.16	2.22	1.00	1.16	2.22
0.10	8	1.00	1.25	2.20	1.00	1.23	1.94	1.00	1.22	2.26	1.00	1.20	2.26	1.00	1.20	1.94
0.10	10	1.00	1.28	2.33	1.00	1.26	2.33	1.00	1.25	2.18	1.00	1.23	2.18	1.00	1.22	2.18
0.10	15	1.00	1.39	2.79	1.00	1.32	2.77	1.00	1.28	2.53	1.00	1.25	2.28	1.00	1.23	2.28
0.10	25	1.00	1.53	2.71	1.00	1.39	2.45	1.00	1.33	2.30	1.00	1.28	2.16	1.00	1.25	2.00
0.10	50	1.00	1.92	3.49	1.00	1.52	2.91	1.00	1.41	2.92	1.00	1.33	2.22	1.00	1.30	2.22
0.50	5	1.00	1.05	1.60	1.00	1.04	1.56	1.00	1.04	1.56	1.00	1.04	1.56	1.00	1.04	1.56
0.50	8	1.00	1.07	1.97	1.00	1.05	1.59	1.00	1.05	1.59	1.00	1.04	1.84	1.00	1.04	1.59
0.50	10	1.00	1.09	1.73	1.00	1.06	1.59	1.00	1.06	1.59	1.00	1.05	1.59	1.00	1.05	1.59
0.50	15	1.00	1.13	2.08	1.00	1.08	1.60	1.00	1.06	1.53	1.00	1.05	1.53	1.00	1.05	1.53
0.50	25	1.00	1.21	2.91	1.00	1.10	1.97	1.00	1.08	1.88	1.00	1.05	1.72	1.00	1.05	1.72
0.50	50	1.00	1.40	3.67	1.00	1.15	1.93	1.00	1.10	1.75	1.00	1.06	1.77	1.00	1.06	1.74
1.00	5	1.00	1.00	1.27	1.00	1.00	1.27	1.00	1.00	1.27	1.00	1.00	1.27	1.00	1.00	1.27
1.00	8	1.00	1.01	1.73	1.00	1.01	1.54	1.00	1.01	1.54	1.00	1.01	1.54	1.00	1.01	1.54
1.00	10	1.00	1.02	1.47	1.00	1.01	1.32	1.00	1.01	1.31	1.00	1.01	1.31	1.00	1.01	1.31
1.00	15	1.00	1.03	1.79	1.00	1.02	1.30	1.00	1.01	1.30	1.00	1.01	1.30	1.00	1.01	1.30
1.00	25	1.00	1.04	2.38	1.00	1.02	1.39	1.00	1.01	1.37	1.00	1.01	1.33	1.00	1.01	1.33
1.00	50	1.00	1.13	2.66	1.00	1.04	1.71	1.00	1.03	1.47	1.00	1.02	1.31	1.00	1.02	1.31
2.00	5	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
2.00	8	1.00	1.00	1.34	1.00	1.00	1.07	1.00	1.00	1.07	1.00	1.00	1.07	1.00	1.00	1.07
2.00	10	1.00	1.00	1.08	1.00	1.00	1.08	1.00	1.00	1.08	1.00	1.00	1.08	1.00	1.00	1.08
2.00	15	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
2.00	25	1.00	1.00	1.39	1.00	1.00	1.14	1.00	1.00	1.14	1.00	1.00	1.09	1.00	1.00	1.09
2.00	50	1.00	1.00	1.71	1.00	1.00	1.13	1.00	1.00	1.11	1.00	1.00	1.11	1.00	1.00	1.09

TABLE IV
BRBC TREE AND SHORTEST PATH TREE RADIUS AND COST STATISTICS FOR RANDOM NETS, EXPRESSED AS A FRACTION OF THE CORRESPONDING MINIMUM SPANNING TREE VALUES

ϵ	net size	tree radius of our algorithm			Shortest Path tree radius			tree cost of our algorithm			Shortest Path tree cost		
		min	ave	max	min	ave	max	min	ave	max	min	ave	max
0.10	5	0.44	0.82	1.00	0.44	0.81	1.00	1.00	1.25	1.84	1.00	1.30	1.96
0.10	8	0.43	0.74	1.00	0.43	0.74	1.00	1.03	1.35	1.99	1.03	1.41	1.99
0.10	10	0.38	0.71	1.00	0.38	0.70	1.00	1.00	1.39	1.96	1.05	1.45	2.25
0.10	15	0.27	0.65	1.00	0.27	0.65	1.00	1.20	1.53	2.66	1.20	1.60	2.71
0.10	25	0.34	0.64	0.94	0.34	0.63	0.93	1.25	1.57	2.03	1.25	1.66	2.16
0.50	5	0.57	0.90	1.00	0.47	0.85	1.00	1.00	1.15	1.60	1.00	1.25	2.04
0.50	8	0.48	0.74	0.99	0.46	0.69	0.99	1.00	1.22	1.66	1.02	1.37	1.94
0.50	10	0.42	0.81	1.00	0.37	0.75	1.00	1.00	1.23	1.57	1.02	1.45	2.05
0.50	15	0.44	0.72	0.99	0.42	0.69	0.99	1.11	1.29	1.53	1.13	1.54	1.94
0.50	25	0.33	0.66	0.97	0.31	0.63	0.97	1.17	1.34	1.73	1.32	1.60	2.14
1.00	5	0.66	0.95	1.00	0.56	0.83	1.00	1.00	1.03	1.30	1.00	1.22	1.90
1.00	8	0.56	0.84	1.00	0.44	0.74	0.95	1.00	1.11	1.31	1.12	1.37	1.96
1.00	10	0.51	0.81	1.00	0.50	0.73	1.00	1.00	1.13	1.47	1.03	1.45	1.96
1.00	15	0.44	0.73	1.00	0.30	0.66	0.97	1.00	1.19	1.41	1.14	1.61	2.28
1.00	25	0.32	0.66	0.99	0.31	0.61	0.93	1.11	1.25	1.43	1.37	1.71	2.38
2.00	5	0.84	0.99	1.00	0.50	0.78	1.00	1.00	1.01	1.15	1.00	1.30	2.03
2.00	8	0.47	0.93	1.00	0.40	0.72	0.94	1.00	1.05	1.22	1.03	1.48	2.06
2.00	10	0.51	0.86	1.00	0.38	0.69	1.00	1.00	1.07	1.23	1.10	1.50	2.28
2.00	15	0.48	0.86	1.00	0.35	0.69	1.00	1.00	1.08	1.19	1.18	1.49	1.95
2.00	25	0.36	0.74	1.00	0.23	0.60	1.00	1.01	1.12	1.27	1.25	1.68	2.31

APPENDIX II
BRBC ALGORITHM FOR SPANNING AND STEINER TREE ROUTING

A. Experimental Data for Random Nets

Table IV shows the cost and radius of the BRBC tree and the SPT, compared with the corresponding MST values, for bounded-radius minimum spanning tree routing. For each ϵ value and net size, 50 random test cases were generated from a uniform distribution in the unit square, and the minimum, average, and maximum values were computed. The source was selected to be one of the terminals at random.

B. Experimental Data for Channel Intersection Graphs of Random Block Designs

Table V shows the cost and radius of the BRBC tree and the SPT, compared with the corresponding KMB val-

TABLE V
HEURISTIC TREE AND SHORTEST PATH TREE RADIUS AND COST STATISTICS FOR RANDOM BLOCK DESIGNS, EXPRESSED AS FRACTIONS OF THE CORRESPONDING KMB TREE VALUES

ϵ	net size	tree radius of our algorithm			Shortest Path tree radius			tree cost of our algorithm			Shortest Path tree cost		
		min	ave	max	min	ave	max	min	ave	max	min	ave	max
0.10	3	0.63	0.93	1.00	0.63	0.93	1.00	0.91	1.12	1.42	0.91	1.12	1.42
0.10	4	0.50	0.90	1.00	0.50	0.90	1.00	0.96	1.14	1.69	0.96	1.14	1.69
0.10	5	0.43	0.84	1.00	0.43	0.84	1.00	0.99	1.17	1.51	0.99	1.18	1.57
0.10	7	0.42	0.82	1.00	0.42	0.82	1.00	0.99	1.14	1.45	0.99	1.15	1.47
0.10	10	0.51	0.82	1.00	0.51	0.82	1.00	1.00	1.22	1.58	1.00	1.22	1.58
0.10	15	0.31	0.81	1.00	0.31	0.80	1.00	1.02	1.21	1.53	1.02	1.22	1.53
0.50	3	0.60	0.94	1.00	0.60	0.94	1.00	0.89	1.09	1.57	1.00	1.14	1.61
0.50	4	0.55	0.88	1.00	0.52	0.86	1.00	0.98	1.11	1.43	1.00	1.16	1.51
0.50	5	0.43	0.89	1.00	0.43	0.87	1.00	0.97	1.15	1.68	0.97	1.23	1.61
0.50	7	0.48	0.86	1.00	0.45	0.82	1.00	0.96	1.11	1.41	0.96	1.20	1.61
0.50	10	0.42	0.80	1.00	0.42	0.77	1.00	0.98	1.17	1.50	1.00	1.27	1.58
0.50	15	0.40	0.78	1.00	0.40	0.75	1.00	0.96	1.15	1.41	0.96	1.19	1.51
1.00	3	0.65	0.99	1.00	0.57	0.93	1.00	0.89	1.02	1.27	0.89	1.14	1.72
1.00	4	0.64	0.99	1.00	0.54	0.91	1.00	0.97	1.02	1.19	1.00	1.15	1.71
1.00	5	0.67	0.95	1.00	0.48	0.86	1.00	1.00	1.09	1.38	1.00	1.23	1.87
1.00	7	0.55	0.92	1.00	0.55	0.84	1.00	1.00	1.09	1.37	1.00	1.21	1.58
1.00	10	0.53	0.90	1.00	0.47	0.81	1.00	0.96	1.10	1.32	1.01	1.22	1.69
1.00	15	0.47	0.85	1.00	0.47	0.78	1.00	0.98	1.11	1.30	0.97	1.21	1.71
2.00	3	1.00	1.00	1.00	0.62	0.92	1.00	1.00	1.00	1.00	1.00	1.13	1.51
2.00	4	0.71	0.99	1.00	0.55	0.89	1.00	0.92	1.01	1.26	0.92	1.15	1.59
2.00	5	0.61	0.99	1.00	0.59	0.85	1.00	1.00	1.02	1.23	1.00	1.19	1.64
2.00	7	0.49	0.97	1.00	0.43	0.80	1.00	0.95	1.03	1.22	0.97	1.22	1.59
2.00	10	0.49	0.93	1.00	0.45	0.81	1.00	1.00	1.05	1.25	1.02	1.26	1.75
2.00	15	0.46	0.88	1.00	0.45	0.76	1.00	0.99	1.06	1.21	1.06	1.25	1.49

ues, for bounded-radius Steiner routing. For each ϵ value and net size, 50 test cases were generated, each with 15 randomly placed modules. Routing was performed in the channel intersection graph, and minimum, average, and maximum values were computed. The source was selected to be one of the terminals at random.

REFERENCES

[1] B. Awerbuch, A. Baratz, and D. Peleg, "Cost-sensitive analysis of communication protocols," in *Proc. ACM Symp. Principles of Distributed Computing*, 1990, pp. 177–187.

[2] J. Cong, A. B. Kahng, B. Robins, M. Sarrafzadeh, and C. K. Wong, "Performance-driven global routing for cell based IC's," in *Proc. IEEE Int. Conf. Computer Design*, 1991, pp. 170–173.

[3] J. Cong and B. Preas, "A new algorithm for standard cell global routing," in *Proc. IEEE Int. Conf. Computer Aided Design*, Nov. 1988, pp. 176–179.

[4] W. M. Dai, T. Asano, and E. S. Kuh, "Routing region definition and ordering scheme for building-block layout," *IEEE Trans. Computer-Aided Design*, vol. CAD-4, pp. 189–197, July 1985.

[5] W. E. Donath, R. J. Norman, B. K. Agrawal, and S. E. Bello, "Timing driven placement using complete path delays," in *Proc. IEEE Design Automat. Conf.*, 1990, pp. 84–89.

[6] A. E. Dunlop *et al.*, "Chip layout optimization using critical path weighting," in *Proc. IEEE Design Automat. Conf.*, 1984, pp. 133–136.

[7] D. Z. Du and F. K. Hwang, "A proof of Gilbert–Pollak's conjecture on the Steiner ratio," in *Proc. IEEE Symp. Foundations of Computer Science*, 1990.

[8] W. C. Elmore, "The transient response of damped linear networks with particular regard to wide-band amplifiers," *J. Appl. Phys.*, vol. 19, no. 1, pp. 55–63, 1948.

[9] P. S. Hauge, R. Nair, and E. J., Yoffa, "Circuit placement for predictable performance," in *Proc. IEEE Int. Conf. Computer Aided Design*, 1987, pp. 88–91.

[10] J. Ho, D. T. Lee, C. H. Chang, and C. K. Wong, "Bounded-diameter spanning trees and related problems," in *Proc. ACM Symp. Computational Geometry*, 1989, pp. 276–282.

[11] F. K. Hwang, "On Steiner minimal trees with rectilinear distance," *SIAM J. Appl. Math.*, vol. 30, no. 1, pp. 104–114, 1976.

[12] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP Completeness*. San Francisco: W. H. Freeman, 1979.

[13] M. A. B. Jackson and E. S. Kuh, "Performance-driven placement of cell-based IC's," in *Proc. IEEE Design Automat. Conf.*, 1989, pp. 370–375.

[14] M. Jackson, A. Srinivasan, and E. S. Kuh, "A fast algorithm for performance-driven placement," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1990, pp. 328–331.

[15] M. A. B. Jackson, E. S. Kuh, and M. Marek-Sadowska, "Timing-driven routing for building block layout," in *Proc. IEEE Int. Symp. Circuits Syst.*, 1987, pp. 518–519.

[16] A. Kahng and G. Robins, "A new family of Steiner tree heuristics with good performance: The iterated 1-Steiner approach," in *Proc. IEEE Int. Conf. Computer Aided Design*, 1990, pp. 428–431.

[17] L. Kou, G. Markowsky, and L. Berman, "A fast algorithm for Steiner trees," *Acta Informatica*, vol. 15, pp. 141–145, 1981.

[18] I. Lin and D. H. C. Du, "Performance-driven constructive placement," in *Proc. IEEE Design Automat. Conf.*, 1990, pp. 103–106.

[19] M. Marek-Sadowska and S. P. Lin, "Timing driven placement," in *Proc. IEEE Int. Conf. Computer Aided Design*, 1989, pp. 94–97.

[20] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization*. Englewood Cliffs, NJ: Prentice-Hall, 1982.

[21] S. Prasitjutrakul and W. J. Kubitz, "A timing-driven global router for custom chip design," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1990, pp. 48–51.

[22] P. Ramanathan and K. G. Shin, "A clock distribution scheme for non-symmetric VLSI circuits," in *Proc. IEEE Int. Conf. Computer Aided Design*, 1989, pp. 398–401.

[23] J. Rubinstein, P. Penfield and M. A. Horowitz, "Signal delay in *RC* tree networks," *IEEE Trans. Computer-Aided Design*, vol. CAD-2, no. 3, pp. 202–211, 1983.

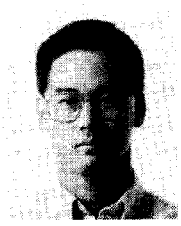
[24] M. Sarrafzadeh and C. K. Wong, "Hierarchical Steiner tree construction in uniform orientations," *IEEE Trans. Computer-Aided Design*, to be published.

[25] S. Sutanthavibul and E. Shragowitz, "An adaptive timing-driven layout for high speed VLSI," in *Proc. IEEE Design Automat. Conf.*, 1990, pp. 90-95.
[26] Y. F. Wu, P. Widmayer, and C. K. Wong, "A faster approximation algorithm for the Steiner problem in graphs," *Acta Informatica*, vol. 23, no. 2, pp. 223-229, 1986.

Jingsheng (Jason) Cong (S'88-M'90) received the B.S. degree in computer science from Peking University in 1985, and the M.S. and Ph.D. degrees in computer science from the University of Illinois at Urbana-Champaign in 1987 and 1990, respectively.

Currently, he is an Associate Professor in the Computer Science Department of the University of California at Los Angeles. From 1986 to 1990, he was a Research Assistant in the Computer Science Department of the University of Illinois. He worked at the Xerox Palo Alto Research Center in the summer of 1987 and was with the National Semiconductor Corporation in the summer of 1988. His research interests include computer-aided design of VLSI circuits, fault-tolerant design of VLSI systems, and design and analysis of efficient combinatorial and geometric algorithms.

Dr. Cong received the Best Graduate Award from Peking University in 1985. He was the recipient of the Ross J. Martin Award for excellence in research from the University of Illinois at Urbana-Champaign in 1989 and of the Student Research Paper Award from Sigma Xi, the scientific research society at the University of Illinois at Urbana-Champaign, in 1990. He is a member of ACM.



Andrew B. Kahng (A'89) was born in October 1963 in San Diego, CA. He holds the A.B. degree in applied mathematics and physics from Harvard College and the M.S. and Ph.D. degrees in computer science from the University of California at San Diego.

He has been an Assistant Professor in the Computer Science Department at UCLA since July 1989. His research interests include computer-aided design of VLSI circuits, combinatorial and parallel algorithms, global optimization theory, and computational geometry. He is a member of ACM and SIAM.

Gabriel Robins is currently completing work for the Ph.D. degree at the UCLA Computer Science Department, where he has won a Distinguished Teaching Award and currently holds an IBM Graduate Fellowship. His primary areas of research are VLSI CAD and geometric algorithms, with recent work focusing on performance-driven routing and heuristic Steiner tree constructions, as well as robot motion planning and pattern recognition. Mr. Robins is a member of ACM.

Majid Sarrafzadeh (M'87) received the B.S., M.S., and Ph.D. degrees in 1982, 1984, and 1986, respectively, all from the University of Illinois at Urbana-Champaign in electrical and computer engineering.

He is currently Assistant Professor of Electrical Engineering and Computer Science at Northwestern University. His research interests lie in the area of design and analysis of algorithms and computational complexity, with emphasis in VLSI.

Dr. Sarrafzadeh received a NSF Engineering Initiation Award in 1987, and a Design Automation Award in 1988.

C K. Wong (M'71-SM'78-F'85) received the B.A. degree (first class honors) in mathmatics from the University of Hong Kong in 1965 and the M.A. and Ph.D. degrees in mathematics from Columbia University in 1966 and 1970 respectively.

He joined the IBM T. J. Watson Research Center in 1969 as a Research Staff Member and is currently manager of the VLSI Design Algorithms group in the Computer Sciences Department. He was Visiting Professor of Computer Sciences at the University of Illinois, Urbana, in 1972-1973 and at Columbia University in 1978-1979. His most recent research is on VLSI design algorithms. He holds four U.S. patents and has published over 170 papers in these areas. He is the author of *Algorithmic Studies in Mass Storage Systems* (Computer Science Press, 1983). He has received an Outstanding Invention Award on Outstanding Technical Achievement Award, and four other Invention Awards from IBM.

Dr. Wong is a member of the Association for Computing Machinery. He was Editor of IEEE TRANSACTIONS ON COMPUTERS from 1982 to 1985 and Chair of the IEEE Computer Society Technical Committee on VLSI from 1990 to 1991. He is on the editorial board of the international journal *Fuzzy Sets and Systems* and is the founding editor in chief of the international journal *Algorithmica*.