Computer Science Department Technical Report
University of California
Los Angeles, CA 90024-1596

# A DIRECT COMBINATION OF THE PRIM AND DIJKSTRA CONSTRUCTIONS FOR IMPROVED PERFORMANCE-DRIVEN GLOBAL ROUTING

C. J. Alpert
T. C. Hu
J. H. Huang
A. B. Kahng

December 1992
CSD-920051

# A Direct Combination of the Prim and Dijkstra Constructions for Improved Performance-Driven Global Routing*

C. J. Alpert, T. C. Hu†, J. H. Huang and A. B. Kahng

UCLA CS Dept., Los Angeles, CA 90024-1596
† CSE Dept., University of California at San Diego, La Jolla, CA 92093

**Abstract**

Motivated by analysis of distributed RC delay in routing trees, we propose a *direct* tradeoff between Prim's minimum spanning tree algorithm and Dijkstra's shortest path tree algorithm which yields a new tree construction for performance-driven global routing. This direct combination of two objective functions and their corresponding optimal algorithms contrasts with the more indirect "shallow-light" methods of [1, 5, 12]. Our method achieves routing trees which satisfy a given bound on the routing tree radius, using less wire than previous methods. Moreover, detailed simulation experiments show that this wirelength savings translates into significantly improved delay in both IC and multi-chip module (MCM) interconnect technologies. For example, average and worst-case source-sink delays in signal nets with 8 sinks (MCM technology) are reduced by 8.5% and 14.2%, respectively, over the method of [5]. These figures also represent 32.6% and 33.8% improvements, respectively, over the standard minimum spanning tree global routing.

## 1 Introduction

With scaling of VLSI device technologies, interconnection delay has become increasingly significant in determining circuit speed, and can contribute up to 50% to 70% of the clock cycle in dense, high performance circuits [7, 20]. Due to this trend, performance-driven layout design has been actively studied in the past several years. Initial work in this area centered on timing-driven *placement*, where modules in timing-critical paths are placed close together, with critical paths being determined by static timing analysis (see, e.g., [7, 10, 14, 15, 11, 20]). More recently, timing-driven *interconnection* algorithms have been developed which afford routing trees compatible with timing-driven placements: for a given signal net, the goal of the timing-driven interconnection is to minimize the average (or maximum) signal delay from the source pin to (any of) the sink pins. To this end, Dunlop et al. [8] used the static timing analysis performed within the iterative place/route paradigm to prioritize signal nets; the

*Partial support for this work was provided by a Department of Defense Graduate Fellowship and by NSF MIP-9110696, NSF Young Investigator Award MIP-9257982, ARO DAAK-70-92-K-0001, and ARO DAAL-03-92-G-0050. Address correspondence to: Prof. A. B. Kahng, UCLA Computer Science Department, 3732 Boelter Hall, Los Angeles, CA 90024-1596

assumption was that nets which are routed earlier will enjoy fewer detours and fewer feedthroughs, thus enhancing performance. Subsequently, Kuh, Jackson and Marek-Sadowska [13] gave an approach tuned to hierarchical building-block layouts and Prastjutrakul and Kubitz [16] used A* heuristic search for a similar problem domain.

In 1991, Cohoon and Randall [3] proposed a heuristic which simultaneously tried to reduce both the longest source-sink pathlength of the tree (i.e., the tree *radius*) as well as the tree cost. A more powerful approach was given by Cong, Kahng, Robins, Sarrafzadeh and Wong [4], wherein a maximum permissible routing tree radius could be specified by the user. Later, [5] proposed the "provably good" BRBC (bounded-radius, bounded-cost) algorithm, which afforded both radius and cost simultaneously within constant factors of optimal. The BRBC method, along with the work of Awerbuch et al. [1] and Khuller et al. [12], belongs to a class of "shallow-light" tree constructions. These methods embody a goal similar to the one we consider below, namely, achieving a *smooth tradeoff* between the competing "minimum radius" and "minimum cost" interconnection objectives. We will discuss this class of "shallow-light" tree constructions in more detail in Section 1.2 below.

## 1.1 Definitions and Motivation

In our discussion, a *signal net* $V = \{v_0, v_1, \ldots, v_n\}$ is a set of $n+1$ terminals, or nodes, in the Manhattan plane. We say that $v_0 \in V$ is the *source* terminal, and that the remaining terminals are *sinks*. The locations of the terminals in $V$ induce a weighted complete graph $G = (V, E)$ where each edge $e_{ij} \in E$ has weight, or *cost*, equal to the Manhattan distance $d_{ij}$ between $v_i$ and $v_j$. A *routing tree* is a (connected) tree $T = (V, E')$ with $E' \subset E$ and $|E'| = n$, i.e., $T$ is a spanning subgraph of $G$ with $n$ edges. The cost of a routing tree, $c(T)$, is the sum of the costs of its edges; the radius of the tree, denoted $r(T)$, is the cost of the longest source-sink path in $T$. Note that these definitions may be easily extended to the *general graph* case, where the edge weights do not necessarily correspond to any embedding of $V$ in the geometric plane.

For a given signal net, the proper criterion to use in efficiently constructing a "performance-driven routing tree" has not yet been well-established. However, the scaling of IC technology, as well as the properties of alternate interconnect and device technologies (e.g., for multi-chip module (MCM) packaging), provide insight into the "proper" objective for performance-driven routing. Consider the Elmore delay model [9], which uses the first-order moment of the impulse response when the routing tree is treated as a distributed RC tree. The Elmore delay is computed as follows: Given an interconnection tree $T$ over source $v_0$ and sinks $v_i$, we use the shorthand notation $e_i$ to denote the edge from node $v_i$ to

its parent when we root the tree at $v_0$. The resistance and capacitance of $e_i$ are respectively given by $r_{e_i}$ and $c_{e_i}$. Let $T_i$ denote the subtree of $T$ rooted at $v_i$, and let $c_i$ denote the node capacitance of $v_i$. The *tree capacitance* $C_i$ of $T_i$ is the sum of node and edge capacitances in $T_i$ [9, 19]. The Elmore delay is then given by

$$t_{Elmore}(v_0, v_i) = \sum_{e_j \in path(v_0, v_i)} r_{e_v}(c_{e_j}/2 + C_j). \quad (1)$$

The Elmore delay equation reveals the effect of *technology* in the choice of optimal routing topologies. Observe that the formula seems to imply that to minimize the maximum source-sink delay, we can simply use a separate wire of minimum possible length (that is, a monotone path) from the source to each sink, resulting in the "star-like" topology of Figure 1(a). However, this fails to capture the effect of driver size; indeed, close examination of Equation 1 reveals that it contains an implicit assumption of zero resistance and capacitance at the source (i.e., driver) node. We may capture the reality of non-zero driver resistance by introducing a wire from a new "virtual source" $v_0'$ to the original source $v_0$, with the original routing tree topology still incident to $v_0$.

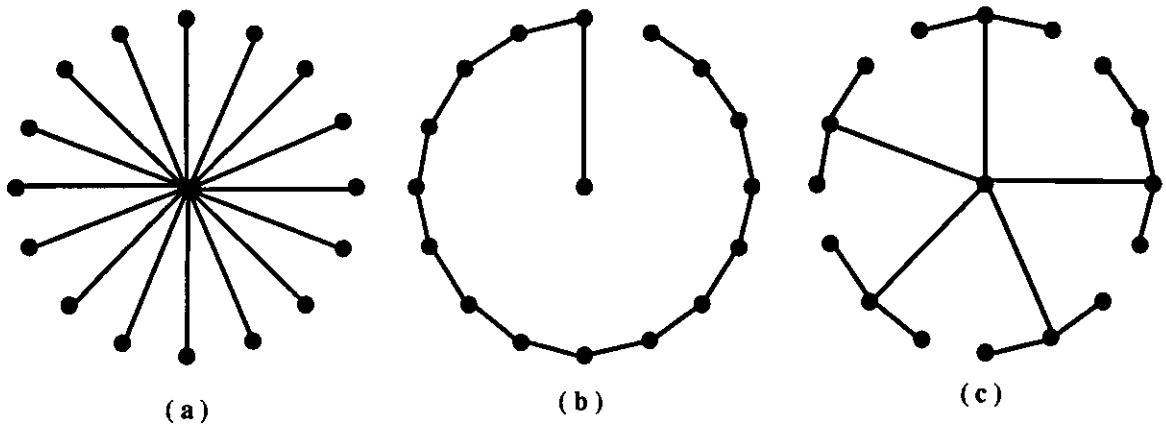


Figure 1: Various routing trees for the same net with $v_0$ at the center: (a) the shortest path tree; (b) the minimum spanning tree; and (c) a routing tree which "trades off" between the two constructions.

A small $v_0'$-$v_0$ wirelength corresponds to small driver resistance (relative to wire resistance), and this arises, e.g., with multi-chip module interconnects. In this regime, monotonicity of all source-sink paths remains more important than overall tree cost, and star-like topologies give better performance. When we increase the length of the $v_0'$-$v_0$ wire (i.e., increase the driver resistance), the Elmore delay becomes more dependent on total tree cost, and less dependent on the directness of source-sink connections. In other words, large driver resistance means that a minimum-cost routing is increasingly preferable; this model reflects the previous generation of IC technologies and confirms the utility of the minimum Steiner and spanning tree constructions used by existing global routers (see Figure 1(b)). Because the

interconnect objective is technology-dependent, our experiments described in Section 3 below consider interconnect parameters for both IC and MCM technologies.

## 1.2 Previous Work: The Shallow-Light Approach

We have seen that minimizing the source-sink Elmore delay involves conflicting min-cost and min-radius goals, and that the prevailing technology determines whcih objective dominates. For a given net $V$ and the implicit underlying graph $G = (V, E)$, a *shortest path tree* (SPT), $T_{SPT}$, provides a routing where every $v_0$-$v_i$ pathlength is equal to the shortest $v_0$-$v_i$ pathlength in $G$ (Figure 1(a)), while at the other extreme a *minimum spanning tree* (MST), $T_{MST}$, provides a min-cost routing tree as shown in Figure 1(b). The tradeoff between these extremes (see Figure 1(c)) may be captured by the following formulation.

**MST-SPT Tradeoff:** Given a net $V$ and a desired tree radius $R^*$, construct a routing tree $T$ that has radius $r(T) = R^*$ and minimum cost.

In addressing this problem, the class of *shallow-light* tree constructions [1, 5, 12] is of particular interest. Each of these algorithms accepts an input parameter $\alpha$ and constructs a routing tree $T$ with radius *at most* $\alpha$ times optimal, i.e., $r(T) \leq \alpha \cdot r(T_{SPT})$ (Note that $r(T_{SPT}) = d_{0j}$ where $v_j$ is the furthest sink from the source $v_0$). Thus, these algorithms actually solve a *bounded-radius* variant of the MST-SPT Tradeoff problem:

**MST-SPT Bounded-Radius Tradeoff:** Given a net $V$, a radius parameter $\alpha$, and a cost parameter $\beta$, construct a routing tree $T$ which has radius at most $\alpha \cdot r(T_{SPT})$ and cost at most $\beta \cdot c(T_{MST})$. (We call such a tree an $(\alpha, \beta)$-*tree*.)

Notice that while each of these shallow-light algorithms guarantees a routing tree that satisfies the radius bound, $\alpha \cdot r(T_{SPT})$, the tree may have radius significantly less than this value. Since we can lower cost by increasing radius, this output tree will probably not be a good solution to the original MST-SPT Tradeoff formulation. However, if we view each algorithm as inducing a *family* of output trees over the entire range of $\alpha$ values for a given instance, we may address the desired MST-SPT Tradeoff more accurately by choosing the family member with radius closest to $R^*$. This technique will find a tree with lower cost than would be obtained by simply applying the given radius bound, but which yet satisfies $r(T) \leq \alpha \cdot r(T_{SPT})$). With this observation in mind, the experimental results of Section 3 will compare heuristics by juxtaposing the best results over the *families* of outputs that can be obtained via each of the heuristics' respective versions of the MST-SPT tradeoff.

The "shallow-light" tree construction of Awerbuch, Baratz and Peleg [1], yields an $(2q+1, 2(1+1/q))$-tree, for $q \geq 0$, in the sense of the above formulation. Using ideas from [1], Cong et al. [5] and Khuller et al. [12] developed similar constructions which more effectively solve the MST-SPT Bounded-Radius tradeoff problem.[1] The BRBC algorithm of Cong et al., shown in Figure 2, constructs an $(1+\epsilon, 1+2/\epsilon)$-tree for any non-negative value of the $\epsilon$ parameter.[2] Khuller et al. [12] construct an $(\alpha, 1+2/(\alpha-1))$-tree when given the MST and the SPT; these bounds are very similar to those of [5]. Khuller et al. go on to show that their algorithm is in some sense optimal, since the problem of determining whether a given graph contains an $(\alpha, \beta)$-tree for $1 \leq \beta \leq 1 + 2/(\alpha - 1)$ is NP-complete.

**BRBC Algorithm** [5] for computing a $(1 + \epsilon, 1 + \frac{2}{\epsilon})$-tree

**Input:** $V \equiv$ signal net
$G \equiv$ complete graph of distances over $V$
$R \equiv$ maximum distance from source $v_0$ to any sink

**Compute** $MST_G$ and $SPT_G$
/* e.g., using Prim's and Dijkstra's algorithms, respectively */
$Q = MST_G$
$L =$ depth-first tour of $MST_G$
$S = 0$
**for** $i = 1$ to $|L| - 1$
  $S = S + d(L_i, L_{i+1})$
  **if** $S \geq \epsilon \cdot d(v_0, L_{i+1})$ **then**
    $Q = Q \cup minpath_G(v_0, L_{i+1})$
    $S = 0$
**Output** $T =$ shortest-path tree of $Q$

Figure 2: BRBC algorithm of [5] for computing an $(\alpha, \beta)$-tree $T$ using parameter $\epsilon$. $T$ satisfies $\alpha = (1 + \epsilon)$ and $\beta = 1 + \frac{2}{\epsilon}$.

In what follows, we present the main contribution of this paper, namely, a new algorithm that elegantly trades off between the MST and SPT constructions. Our method is based on *directly* combining the well-known constructions of Prim [17] and Dijkstra [6], and in some sense our tradeoff is the most direct that is possible.

[1] The work of Awerbuch et al. [1] actually addressed the closely related problem of computing a bounded-diameter, rather than bounded-radius, tree. Note that the MST-SPT tradeoff can be more generally stated as follows [12]: Given a graph $G = (V, E)$ with non-negative edge weights and identified source $v_0$, construct a spanning tree $T$ with the path cost from $v_0$ to any other node $v_i$ being less than $\alpha$ times the cost of the shortest shortest path from $v_0$ to $v_i$ in $G$, and with total tree cost $c(T)$ at most $\beta \cdot c(T_{MST})$. Here, for each sink the source-sink pathlength is bounded by $\alpha$ times the optimum distance for that particular sink, rather than by $\alpha$ times the "generic", maximum source-sink distance. Khuller et al. solve this more node-dependent formulation, and the Ph.D. thesis of G. Robins [18] gives exactly the same construction as that of Khuller et al., in also addressing this extension.

[2] In the BRBC template, recall that $G$ is the complete graph of distances among the terminals. We use $MST_G$ and $SPT_G$ to respectively denote the minimum spanning tree over $G$ and the shortest-path tree over $G$, with the additional term $minpath$ defined in the obvious manner. Notice that $\epsilon = \infty$ will allow BRBC to return a minimum spanning tree, and $\epsilon = 0$ in some sense leads to a shortest-path tree.

## 2 The AHHK Algorithm

The shallow-light heuristics [1, 5, 12] described above all use the same basic idea: make a depth first traversal of the minimum spanning tree, and if the pathlength from the source to some sink becomes too large, modify the tree to reduce this source-sink pathlength. In contrast, our new approach unifies the minimum-cost and minimum-radius objectives by directly combining the MST algorithm of Prim [17] with the SPT algorithm of Dijkstra [6].[3]

Prim's algorithm begins initially with the trivial tree consisting only of the source $v_0$. The algorithm iteratively adds the edge $e_{ij}$ and the node $v_i$ to $T$, where $v_i$ and $v_j$ are chosen to minimize

$$d_{ij} \quad s.t. \quad v_j \in T,\ v_i \in V - T \tag{2}$$

Dijkstra's algorithm also begins with the trivial tree consisting only of the source $v_0$. The algorithm iteratively adds the edge $e_{ij}$ and the node $v_i$ to $T$, where $v_i$ and $v_j$ are chosen to minimize

$$l_j + d_{ij} \quad s.t. \quad v_j \in T,\ v_i \in V - T \tag{3}$$

where $l_j$ is the cost of the path from $v_0$ to $v_j$ in $T$.

The key observation is that these two constructions are extremely similar: Prim's algorithm adds the node $v_i$ that minimizes $d_{ij}$, while Dijkstra's algorithm adds the node that minimizes $l_j + d_{ij}$. This naturally suggests that a middle ground can be reached by compromising between these objectives, namely, we propose to iteratively add the edge $e_{ij}$ and the node $v_i$ to $T$, where $v_i$ and $v_j$ are chosen to minimize

$$(c \cdot l_j) + d_{ij} \quad s.t. \quad v_j \in T,\ v_i \in V - T \tag{4}$$

for some choice of $0 \leq c \leq 1$ (see the template of Figure 2).

The complexity of the AHHK algorithm can be no worse than that of Dijkstra's algorithm, i.e., $O(n^2)$. In practice, the value for $c$ should be chosen based on the relative importance of minimizing cost versus minimizing the longest source-sink path. When $c = 0$, the algorithm constructs a tree with minimum cost. As $c$ increases, AHHK constructs a tree with increasingly higher cost but with lower radius. When $c = 1$, AHHK is identical to Dijkstra's algorithm and constructs a tree with minimum radius but possibly unbounded cost.

For a fixed value of $c$, AHHK is guaranteed to construct a spanning tree with radius less than or equal to $1/c$ times the optimal radius:

[3] Our tradeoff of two competing objectives, via a tradeoff of algorithms that are respectively optimal for these objectives, is unusual in the literature.

**AHHK Algorithm** to trade off MST and SPT constructions

**Input:** $V \equiv$ signal net
$G = (V, E) \equiv$ complete graph of distances over $V$
$c \equiv$ tradeoff parameter, $0 \leq c \leq 1$

$T = (V', E') = (v_0, \emptyset)$
**while** $|V'| < |V|$
  **Select** $v_j \in V'$ and $v_i \in V - V'$ minimizing $(c \cdot l_j) + d_{ij}$
  $V' = V' \bigcup \{v_i\}$;  $E' = E' \bigcup \{e_{ij}\}$
**Output** $T$

Figure 3: The AHHK algorithm outputs a routing tree with MST-SPT tradeoff governed by parameter $c$, with $0 \leq c \leq 1$. Choosing $c = 0$ yields an MST, while choosing $c = 1$ yields an SPT.

**Theorem 1** *The AHHK algorithm will construct a tree $T$ with radius at most $r(T) \leq \frac{r(T_{SPT})}{c}$.*

**Proof:** After the tree $T$ has been constructed, let $v_j$ be the terminal such that $l_j$ is maximum, i.e., $r(T) = l_j$. Let $v_i$ be the parent of $v_j$. Clearly, $d_{0j} \leq r(T_{SPT})$. Moreover, we may rewrite the cost of the $v_0$-$v_j$ path as $l_j = l_i + d_{ij}$. Since we always have the option of making a direct source-sink connection, we have $c \cdot l_i + d_{ij} \leq d_{0j}$. Combining these inequalities and using the fact that $c \leq 1$, we have:

$$c \cdot l_j \leq c \cdot (l_i + d_{ij}) \leq c \cdot l_i + d_{ij} \leq d_{0j} \leq r(T_{SPT})$$

Hence, $c \cdot l_j \leq r(T_{SPT})$ or equivalently, $r(T) \leq \frac{r(T_{SPT})}{c}$. □

We note that for any fixed value of $c$, there exists a general graph instance on which the AHHK algorithm will yield a routing tree with unbounded cost; this is shown in the example of Figure 4. However, we conjecture that the cost is bounded when the input net $V$ is embedded in the Manhattan plane.

# 3 Experimental Results

The MST-SPT Tradeoff problem formulation requires that an algorithm accepts an input parameter $\alpha$ and produces a tree with radius at most $\alpha$ times optimal. Our AHHK algorithm uses the parameter $0 \leq c \leq 1$ to trade off between the Prim and Dijkstra algorithms, and returns a tree with radius at most $1/c$ times optimal. Thus, we may view $c$ as equivalent to $1/\alpha$. Similarly, the BRBC algorithm [5] takes a parameter $\epsilon \geq 0$ and produces a routing tree with radius at most $1 + \epsilon$ times optimal; thus, $\epsilon$ is equivalent to $\alpha - 1$. By rewriting these relationships, we see that for a fixed $\alpha$, our parameter $c$ is

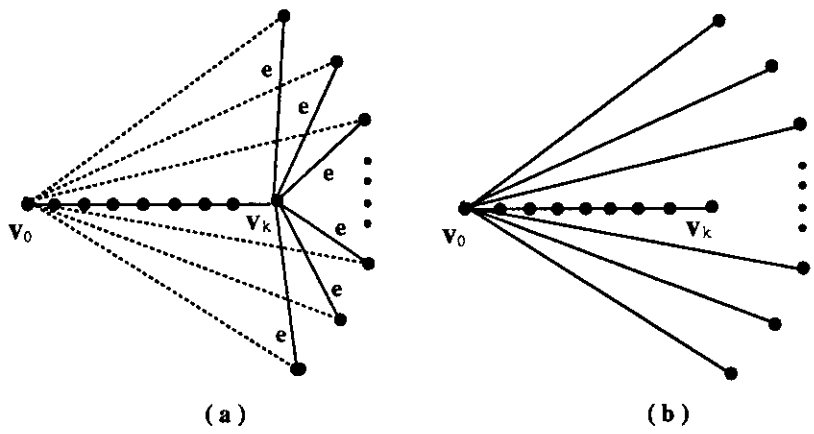


Figure 4: For any given value of $c$, we construct a pathological general graph input $G$ for the AHHK algorithm as follows. $G$ has a path of closely-spaced sinks, of length $\frac{1}{c} - e$, between $v_0$ and $v_k$ (for some $e$ small). $G$ also contains edges of length $e$ from $v_k$ to all other sinks that are not in this path. Edges in $G$ shown as dotted lines are of length 1, and remaining edges (not shown) have prohibitively large cost. The MST in (a) will contain the path from $v_0$ to $v_k$ of length $\frac{1}{c} - e$, along with edges of length $e$. The AHHK tree in (b) has cost $c(T_{AHHK})$ an unbounded factor greater than $c(T_{MST})$.

equivalent to $1/(1+\epsilon)$.

In our experiments, we computed AHHK trees for 21 values of $c$ ranging from 0 to 1 at intervals of 0.05; we also computed BRBC trees with the corresponding $\epsilon$ values, which ranged from $\infty$ to 0. As noted in the discussion above, these trees represent a *family* of output trees that may be generated by each algorithm over the range of $c$ or $\epsilon$ values.[4] In the experimental results of Tables 1-3 (see Appendix for all tables), AHHK and BRBC may be compared with respect to any given radius value (rather than by a radius *bound*) by determining which algorithm returns the tree with lesser cost and/or lesser delay.

We ran the AHHK and BRBC algorithms on $n$ random sets of sink locations with $n$ = 4, 8, 16 and 32 sinks; the source location was also random, and all locations were chosen from a uniform distribution in the unit square. Due to limited space, we only show results for $n$ = 16 (Tables 1-3); however, we have included summary results for all four net sizes in Table 4. All numbers represent averages over 50 point sets.

In Table 1, we report the average cost (normalized to MST cost) and radius (normalized to SPT radius) of the routing trees generated. In Tables 2-3, we report the results of delay simulations. Our inputs correspond to two distinct technologies: (i) *IC* is a representative 0.8$\mu$ CMOS process, and (ii) *MCM* is typical of current MCM technologies, with lower driver resistance and unit wire resistance.[5]

[4] For $c$ = 1.00, we actually used $c$ = 0.9999: this will also yield an SPT, but reduces total tree cost by breaking ties to take advantage of the fact that staircase paths are also shortest paths in the Manhattan geometry.

[5] Specifics of the technology files (IC,MCM): driver resistance = (100,25) $\Omega$; wire resistance = (0.03,0.008) $\Omega/\mu m$; wire inductance = (492,380) $fH/\mu m$; sink loading capacitance = (15.3,1000) $fF$; wire capacitance = (0.352,0.06) $fF/\mu m$;

The delays at all sink nodes were measured using the two-pole circuit simulator proposed by Zhou et al. [21] and discussed in [2]. This simulator is a computationally efficient code which has produced very accurate results (within a few percent) when tested against the commonly used circuit simulator SPICE. Table 2 gives results for the IC parameters, and Table 3 gives results for the MCM parameters. We consider both the average delay (taken over all sinks) and the worst-case delay (i.e., the latest arrival time of the signal to any sink); all results are normalized to the corresponding values for the MST routing.

We make the following observations.

1. BRBC tends to yield lower tree cost any fixed $\alpha$, but this may be explained by observing the AHHK radius does not approach the $1/c$ bound as closely as the BRBC radius approaches its $1+\epsilon$ bound. Indeed, if we consider the family of trees over each instance, we see that for any given tree radius, the AHHK solution will almost always have lower cost.

2. In view of the relationship between tree cost and delay, we would expect that the AHHK solutions would have lower signal delay than the BRBC solutions, and this is clearly the case (Tables 2 - 4) with respect to both the worst-case and average-case criteria. For example, the summary results of Table 4 show that AHHK solutions entail 16.0% less signal delay in the worst case, and 20.9% less in the average case, when compared against BRBC for nets with 16 sinks in the MCM technology. The same data also shows that AHHK yields improvements of 46.0% and 45.0% in terms of worst-case delay and average-case delay when compared with the standard MST global routing.

3. With respect to the two sets of interconnect parameters, note that the Elmore delay has greater dependence on tree cost in the IC technology; hence, any tree that approximates the minimum spanning tree should have reasonably good delay. Therefore, although AHHK outperforms BRBC, the differences between the algorithms are small since both will closely approximate the MST for those values of $\alpha$ that minimize delay. On the other hand, tree radius is more dominant in the Elmore delay model for the MCM interconnects, and thus the AHHK delay savings over the MST routing are larger than those obtained by BRBC.

4. As the nets become larger, the MST radius similarly becomes much greater than the BRBC or AHHK radius; in particular, the delay improvements from the AHHK solution become more obvious.

layout area = $(10^2, 100^2)$ $mm^2$.

5. Finally, we obtain a great deal of intuition as to the "proper" values of $c$ to use, by analyzing the $c$ values that afford lowest signal delay over the entire family of 21 trees generated by each algorithm for each instance. The "best" column in Tables 2 - 3 records the number of times that a particular $c$ or $\epsilon$ value resulted in the tree with lowest delay; the average "best" parameterization is reported in Table 4. The "best" parameter value clearly shifts with the interconnect technology and the net size; if one wishes to avoid generating the entire family of routing trees, we believe that $c$ values should be chosen according to the results of these studies.

# 4 Conclusion and Future Work

Analysis of distributed RC delay shows that performance-driven routing depends on tree constructions that can smoothly trade off cost and radius, depending on interconnect technology, the size of a signal net, etc. Previous approaches [1, 5, 12] have used "shallow-light" constructions to achieve such a tradeoff; these methods essentially rely on a depth first traversal of the MST and insert shortest paths as needed to maintain a prescribed radius bound. In contrast, our new AHHK approach *directly* combines the recurrences for Prim's MST algorithm and Dijkstra's SPT algorithm. The result is an elegant tradeoff between radius and cost which empirically yields lower-cost trees than the BRBC algorithm [5] for any given tree radius. Simulation results show that the AHHK algorithm constructs routing trees with significantly less maximum and average delay than the BRBC method, in both IC and MCM interconnect technologies. Thus, we will pursue the integration of the AHHK construction within existing performance-driven global routing methodologies.

For general graphs, AHHK can yield tree cost an unbounded factor greater than the MST cost. However, we conjecture that the AHHK tree has bounded cost for instances that are embedded in the Manhattan plane, i.e., there exists a constant $\beta$ such that AHHK will always produce a $(1/c, \beta)$-tree for any given value of $c$. We also leave open the question of determining the minimum possible value for $\beta$.

# Appendix: Tables and References

| AHHK | BRBC | Ave Cost vs. MST | | Ave Radius vs. SPT | |
|---|---|---|---|---|---|
| $c$ | $\epsilon$ | AHHK | BRBC | AHHK | BRBC |
| 0.00 | $\infty$ | 1.000 | 1.000 | 1.616 | 1.616 |
| 0.05 | 19.00 | 1.002 | 1.000 | 1.576 | 1.616 |
| 0.10 | 9.00 | 1.008 | 1.000 | 1.411 | 1.616 |
| 0.15 | 5.67 | 1.018 | 1.000 | 1.359 | 1.616 |
| 0.20 | 4.00 | 1.028 | 1.000 | 1.288 | 1.616 |
| 0.25 | 3.00 | 1.041 | 1.000 | 1.237 | 1.616 |
| 0.30 | 2.33 | 1.053 | 1.000 | 1.192 | 1.616 |
| 0.35 | 1.86 | 1.074 | 1.000 | 1.144 | 1.616 |
| 0.40 | 1.50 | 1.094 | 1.000 | 1.113 | 1.610 |
| 0.45 | 1.22 | 1.103 | 1.005 | 1.101 | 1.594 |
| 0.50 | 1.00 | 1.119 | 1.026 | 1.082 | 1.548 |
| 0.55 | 0.82 | 1.151 | 1.040 | 1.061 | 1.491 |
| 0.60 | 0.67 | 1.177 | 1.047 | 1.045 | 1.438 |
| 0.65 | 0.54 | 1.206 | 1.075 | 1.028 | 1.400 |
| 0.70 | 0.43 | 1.232 | 1.097 | 1.020 | 1.332 |
| 0.75 | 0.33 | 1.262 | 1.111 | 1.015 | 1.253 |
| 0.80 | 0.25 | 1.316 | 1.134 | 1.008 | 1.195 |
| 0.85 | 0.18 | 1.357 | 1.185 | 1.005 | 1.133 |
| 0.90 | 0.11 | 1.416 | 1.196 | 1.001 | 1.075 |
| 0.95 | 0.05 | 1.485 | 1.249 | 1.000 | 1.023 |
| 1.00 | 0.00 | 1.540 | 1.277 | 1.000 | 1.000 |

Table 1: Normalized cost (versus MST cost) and radius (versus SPT radius), for nets with 16 sinks. Data are averaged over 50 instances generated randomly in the unit square. We portray the entire families of routing trees generated by AHHK for $c$ at intervals of 0.05 and by BRBC for corresponding $\epsilon$ values. For a given average radius value, AHHK tends to yield lower tree cost than BRBC.

| Parameter | | Max Delay vs. MST | | | | Ave Delay vs. MST | | | |
|---|---|---|---|---|---|---|---|---|---|
| AHHK | BRBC | AHHK | | BRBC | | AHHK | | BRBC | |
| $c$ | $\epsilon$ | Delay | best $c$ | Delay | best $\epsilon$ | ave | best $c$ | ave | best $\epsilon$ |
| 0.00 | $\infty$ | 1.000 | 0 | 1.000 | 4 | 1.000 | 0 | 1.000 | 6 |
| 0.05 | 19.00 | 0.981 | 0 | 1.000 | 0 | 0.981 | 0 | 1.000 | 0 |
| 0.10 | 9.00 | 0.895 | 0 | 1.000 | 0 | 0.908 | 0 | 1.000 | 0 |
| 0.15 | 5.67 | 0.859 | 0 | 1.000 | 0 | 0.867 | 1 | 1.000 | 0 |
| 0.20 | 4.00 | 0.817 | 2 | 1.000 | 0 | 0.833 | 4 | 1.000 | 0 |
| 0.25 | 3.00 | 0.786 | 3 | 1.000 | 0 | 0.809 | 2 | 1.000 | 0 |
| 0.30 | 2.33 | 0.761 | 1 | 1.000 | 0 | 0.791 | 3 | 1.000 | 0 |
| 0.35 | 1.86 | 0.743 | 2 | 1.000 | 0 | 0.782 | 2 | 1.000 | 0 |
| 0.40 | 1.50 | 0.728 | 5 | 1.000 | 0 | 0.770 | 5 | 1.000 | 0 |
| 0.45 | 1.22 | 0.720 | 6 | 1.002 | 0 | 0.761 | 5 | 1.005 | 0 |
| 0.50 | 1.00 | 0.716 | 7 | 1.005 | 0 | 0.760 | 8 | 1.014 | 0 |
| 0.55 | 0.82 | 0.720 | 5 | 0.992 | 0 | 0.763 | 3 | 1.009 | 0 |
| 0.60 | 0.67 | 0.718 | 7 | 0.966 | 0 | 0.762 | 5 | 0.990 | 0 |
| 0.65 | 0.54 | 0.717 | 3 | 0.974 | 0 | 0.762 | 3 | 0.996 | 1 |
| 0.70 | 0.43 | 0.722 | 2 | 0.967 | 0 | 0.766 | 4 | 0.997 | 0 |
| 0.75 | 0.33 | 0.730 | 1 | 0.923 | 2 | 0.774 | 3 | 0.970 | 2 |
| 0.80 | 0.25 | 0.741 | 3 | 0.904 | 1 | 0.793 | 0 | 0.966 | 0 |
| 0.85 | 0.18 | 0.749 | 1 | 0.898 | 4 | 0.804 | 0 | 0.967 | 6 |
| 0.90 | 0.11 | 0.763 | 1 | 0.858 | 6 | 0.819 | 2 | 0.932 | 5 |
| 0.95 | 0.05 | 0.785 | 1 | 0.853 | 7 | 0.846 | 0 | 0.934 | 6 |
| 1.00 | 0.00 | 0.803 | 0 | 0.795 | 26 | 0.868 | 0 | 0.872 | 24 |

Table 2: Comparison of signal delays (computed using two-pole simulator of [21]) using IC technology interconnect parameters, for nets with 16 sinks. Data are averaged over 50 instances generated randomly in the unit square. Max data uses maximum delay to any sink; Ave data uses average delay over all sinks. Best column tabulates the number of instances for which a particular value of $c$ or $\epsilon$ yielded the lowest delay within the family of trees generated. Comparing the performance of AHHK versus BRBC at each value of $\alpha$, we find that AHHK yielded 18.0% improvement in max delay, and 16.5% improvement in average delay, over the BRBC results.

| Parameter | | Max Delay vs. MST | | | | Ave Delay vs. MST | | | |
|---|---|---|---|---|---|---|---|---|---|
| AHHK | BRBC | AHHK | | BRBC | | AHHK | | BRBC | |
| $c$ | $\epsilon$ | ave | best c | ave | best $\epsilon$ | ave | best c | ave | best $\epsilon$ |
| 0.00 | $\infty$ | 1.000 | 0 | 1.000 | 3 | 1.000 | 0 | 1.000 | 3 |
| 0.05 | 19.00 | 0.965 | 0 | 1.000 | 0 | 0.962 | 0 | 1.000 | 0 |
| 0.10 | 9.00 | 0.854 | 0 | 1.000 | 0 | 0.859 | 0 | 1.000 | 0 |
| 0.15 | 5.67 | 0.813 | 0 | 1.000 | 0 | 0.808 | 0 | 1.000 | 0 |
| 0.20 | 4.00 | 0.754 | 0 | 1.000 | 0 | 0.753 | 0 | 1.000 | 0 |
| 0.25 | 3.00 | 0.715 | 2 | 1.000 | 0 | 0.718 | 1 | 1.000 | 0 |
| 0.30 | 2.33 | 0.677 | 0 | 1.000 | 0 | 0.685 | 0 | 1.000 | 0 |
| 0.35 | 1.86 | 0.645 | 2 | 1.000 | 0 | 0.662 | 1 | 1.000 | 0 |
| 0.40 | 1.50 | 0.626 | 0 | 1.000 | 0 | 0.638 | 1 | 1.000 | 0 |
| 0.45 | 1.22 | 0.610 | 5 | 1.000 | 0 | 0.621 | 1 | 1.004 | 0 |
| 0.50 | 1.00 | 0.603 | 5 | 0.995 | 0 | 0.614 | 3 | 1.006 | 0 |
| 0.55 | 0.82 | 0.596 | 3 | 0.976 | 0 | 0.602 | 2 | 0.994 | 0 |
| 0.60 | 0.67 | 0.590 | 4 | 0.943 | 0 | 0.595 | 6 | 0.969 | 0 |
| 0.65 | 0.54 | 0.579 | 3 | 0.941 | 0 | 0.582 | 3 | 0.967 | 0 |
| 0.70 | 0.43 | 0.578 | 5 | 0.921 | 0 | 0.575 | 7 | 0.954 | 0 |
| 0.75 | 0.33 | 0.580 | 2 | 0.868 | 1 | 0.578 | 4 | 0.920 | 1 |
| 0.80 | 0.25 | 0.573 | 8 | 0.835 | 1 | 0.576 | 5 | 0.900 | 0 |
| 0.85 | 0.18 | 0.575 | 5 | 0.809 | 1 | 0.576 | 9 | 0.877 | 4 |
| 0.90 | 0.11 | 0.575 | 4 | 0.753 | 4 | 0.574 | 5 | 0.822 | 4 |
| 0.95 | 0.05 | 0.585 | 2 | 0.734 | 9 | 0.582 | 2 | 0.803 | 8 |
| 1.00 | 0.00 | 0.594 | 0 | 0.653 | 31 | 0.593 | 0 | 0.709 | 30 |

Table 3: Comparison of signal delays (computed using two-pole simulator of [21]) using MCM technology interconnect parameters, for nets with 16 sinks. Data are averaged over 50 instances generated randomly in the unit square. Max data uses maximum delay to any sink; Ave data uses average delay over all sinks. Best column tabulates the number of instances for which a particular value of $c$ or $\epsilon$ yielded the lowest delay within the family of trees generated. Comparing the performance of AHHK versus BRBC at each value of $\alpha$, we find that AHHK yielded 27.2% improvement in max delay, and 28.9% improvement in average delay, over the BRBC results.

| | | IC Parameters | | MCM Parameters | |
|---|---|---|---|---|---|
| #sinks | ALG. | Max Delay vs. MST | Ave Delay vs. MST | Max Delay vs. MST | Ave Delay vs. MST |
| 4 | AHHK | 0.881 (ave c = 0.28) | 0.903 (ave c = 0.29) | 0.803 (ave c = 0.38) | 0.801 (ave c = 0.47) |
| 4 | BRBC | 0.883 | 0.909 | 0.813 | 0.879 |
| 8 | AHHK | 0.787 (ave c = 0.49) | 0.811 (ave c = 0.48) | 0.674 (ave c = 0.59) | 0.662 (ave c = 0.71) |
| 8 | BRBC | 0.839 | 0.879 | 0.736 | 0.771 |
| 16 | AHHK | 0.690 (ave c = 0.53) | 0.739 (ave c = 0.49) | 0.540 (ave c = 0.66) | 0.550 (ave c = 0.71) |
| 16 | BRBC | 0.769 | 0.837 | 0.643 | 0.695 |
| 32 | AHHK | 0.602 (ave c = 0.46) | 0.644 (ave c = 0.45) | 0.429 (ave c = 0.64) | 0.451 (ave c = 0.69) |
| 32 | BRBC | 0.695 | 0.805 | 0.558 | 0.604 |

Table 4: Summary Table. Direct comparison of AHHK and BRBC algorithms using the data from the *best* tree for each problem instance (using the max or average criterion as appropriate). Results are averaged over 50 random instances, except for BRBC data for 32 sinks (average of 10 random instances). We also report the average value of the parameter $c$ that yielded the tree with best delay. We suggest using this information to determine an appropriate parameter $c$ if the user wishes to generate only one or two representatives from the entire family of routing trees. For nets with 8 sinks, AHHK trees offer delay improvements ranging from 6.2% to 14.2% over BRBC, depending on which technology and criterion are chosen.

# References


[1] B. Awerbuch, A. Baratz and D. Peleg, "Cost-Sensitive Analysis of Communication Protocols", *Proc. ACM Symp. on Principles of Distributed Computing*, 1990, pp. 177-187.

[2] K. D. Boese, J. Cong, A. B. Kahng, K. S. Leung and D. Zhou, "On High-Speed VLSI Interconnects: Analysis and Design", *Proc. Asia-Pacific Conf. on Circuits and Systems*, Dec. 1992, to appear.

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

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

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

[6] E. W. Dijkstra, "A Note on Two Problems in Connection With Graphs", *Numerische Mathematik* 1(1959), pp. 269-271.

[7] W. E. Donath, R. J. Norman, B. K. Agrawal, S. E. Bello, S. Y. Han, J. M. Kurtzberg, P. Lowy and R. I. McMillan, "Timing Driven Placement Using Complete Path Delays", *Proc. ACM/IEEE Design Automation Conf.*, 1990, pp. 84-89.

[8] A. E. Dunlop, V. D. Agrawal, D. N. Deutsh, M. F. Jukl, P. Kozak and M. Wiesel, "Chip Layout Optimization Using Critical Path Weighting", *Proc. ACM/IEEE Design Automation Conf.*, 1984, pp. 133-136.

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

[10] P. S. Hauge, R. Nair and E. J. Yoffa, "Circuit Placement for Predictable Performance", *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1987, pp. 88-91.

[11] M. A. B. Jackson and E. S. Kuh, "Estimating and Optimizing RC Interconnect Delay During Physical Design", *Proc. IEEE Intl. Conf. on Circuits and Systems*, 1990, pp. 869-871.

[12] S. Khuller, B. Raghavachari and N. Young, "Balancing Minimum Spanning and Shortest Path Trees", *Proc. ACM/SIAM Symp. on Discrete Algorithms*, January 1993, to appear.

[13] E. Kuh, M. A. B. Jackson and M. Marek-Sadowska, "Timing-Driven Routing for Building Block Layout", *Proc. IEEE International Symposium on Circuits and Systems*, pp. 518-519, 1987.

[14] I. Lin and D. H. C. Du, "Performance-Driven Constructive Placement", *Proc. ACM/IEEE Design Automation Conf.*, 1990, pp. 103-106.

[15] M. Marek-Sadowska and S. Lin, "Timing Driven Placement", *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1989, pp. 94-97.

[16] S. Prastjutrakul and W. J. Kubitz, "A Timing-Driven Global Router for Custom Chip Design", *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1990, pp. 48-51.

[17] R. C. Prim, "Shortest Connecting Networks and Some Generalizations", *Bell System Tech. J.* 36 (1957), pp. 1389-1401.

[18] G. Robins, "On Optimal Interconnections", Ph.D. Thesis, CS Dept., University of California, Los Angeles, June 1992.

[19] J. Rubinstein, P. Penfield, and M. A. Horowitz, "Signal Delay in RC Tree Networks", *IEEE Trans. on CAD* 2(3) (1983), pp. 202-211.

[20] S. Sutanthavibul and E. Shragowitz, "Adaptive Timing-Driven Layout for High Speed VLSI", *Proc. ACM/IEEE Design Automation Conf.*, 1990, pp. 90-95.

[21] D. Zhou, S. Su, F. Tsui, D. S. Gao and J. Cong, "Analysis of Trees of Transmission Lines", *technical report* UCLA CSD-920010.