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

CS Dept., University of California at Los Angeles, 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 new tree construction for performance-driven global routing which directly *trades off between Prim's minimum spanning tree algorithm and Dijkstra's shortest path tree algorithm. This direct combination of two objective functions and their corresponding optimal algorithms contrasts with the more indirect "shallow-light" methods of [2, 4, 10]. Our method achieves routing trees which satisfy a given routing tree radius bound while using less wire than previous methods. Detailed simulations show that this wirelength savings translates into significantly improved delay over both the method of [4] and standard MST routing in both IC and multi-chip module (MCM) interconnect technologies.*

1 Introduction

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 [6, 16]. Thus, 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 (e.g., [6, 9, 12, 13]). More recently, timing-driven *interconnection* algorithms have been developed [7, 11, 14], where the goal is to minimize the average (maximum) signal delay from the source pin to (any of) the sink pins.

For a given signal net, the proper criterion to use in efficiently constructing a "performance-driven routing tree" is not yet well-established. However, scaling relationships in IC and other technologies (e.g., for multi-chip module (MCM) packaging) provide insight into the "proper" objective for performance-driven routing. Consider the Elmore delay model [8], which uses the first-order moment of the impulse response when the routing tree is treated as a distributed RC tree. If we root the interconnection tree at source v_0, each sink v_i may be identified with the edge e_i leading to its parent. We use r_{e_i} and c_{e_i} to denote the resistance and capacitance of edge e_i. Let T_i denote the subtree of T rooted at v_i, and let C_i denote the tree capacitance of T_i, i.e., the sum of edge capacitances in T_i. The Elmore delay at v_i is

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

The Elmore delay equation reveals the effect of *technology* on the optimality of a given routing topology. For example, we can capture driver resistance relative to interconnect resistance as a "resistance ratio" [3], and then introduce 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) to model a given technology via this resistance ratio.

A small v_0'-v_0 wirelength corresponds to a small resistance ratio; this arises, e.g., with MCM interconnects. In this regime, monotonicity of all source-sink paths remains more important than tree cost, and star-like topologies yield lower delays. When we increase the length of the v_0'-v_0 wire (i.e., increase the driver resistance), Elmore delay depends more on tree cost than on the directness of source-sink connections. This model reflects the previous generation of IC technologies and validates the minimum Steiner and spanning tree constructions of existing global routers. Because the interconnect objective depends on technology, the experiments described in Section 3 below use actual IC and MCM interconnect parameters.

The following sections develop 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 constructions of Prim [15] and Dijkstra [5], and is in some sense the most direct tradeoff possible.

2 Problem Formulation

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 $G =$

*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.

(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[1]. A *routing tree* is a tree $T = (V, E')$ with $E' \subset E$ and $|E'| = n$. The cost of a routing tree, $c(T)$, is the sum of the costs of its edges. For a given T, we use l_i to denote the $v_0 - v_i$ path weight in T; the maximum l_i value is the tree radius, $r(T)$.

From the above discussion, we see that minimizing source-sink Elmore delay involves conflicting min-cost and min-radius goals, and that technology determines the dominant objective. Although an MST will minimize cost, its radius may be unbounded factor larger than optimal. On the other hand, an SPT minimizes radius but may have cost unboundedly higher than the MST cost (see Fig. 1). This observation motivates the following:

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

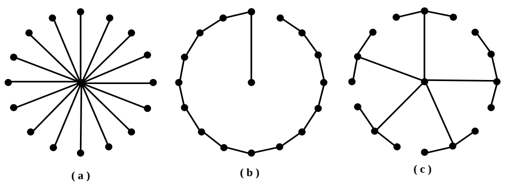

Figure 1: Various routing trees for the same net with v_0 at the center: (a) the shortest path tree; (b) the minimum spanning tree; and (c) a routing tree which "trades off" between the two constructions.

Here, the class of *shallow-light* tree constructions [2, 4, 10] is of particular relevance. The original construction of Awerbuch, Baratz and Peleg [2], yields a $(1 + 2\epsilon, 2 + 2/\epsilon)$-tree, for parameter $\epsilon \geq 0$. Using ideas from [2], Cong et al. [4] and Khuller et al. [10] developed slightly more effective $(1 + \epsilon, 1 + 2/\epsilon)$-tree constructions. Both essentially rely on a depth-first traversal of an MST, adding extra edges as necessary when the tree radius is found to be too large.

3 The AHHK Tradeoff

Prim's MST algorithm [15] begins initially with the trivial tree consisting only of the source v_0. The algorithm iteratively adds the edge e_{ij} and the sink v_i to T, where v_i and v_j are chosen to minimize

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

Dijkstra's SPT algorithm [5] also begins with the trivial tree consisting only of the source v_0. The algorithm iteratively adds the edge e_{ij} and the sink v_i to T, where v_i and v_j are chosen to minimize

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

[1]Note that only the definition of d_{ij} is dependent on geometry. All other definitions presented may be applied to the *general graph* case, where the edge weights do not necessarily correspond to any embedding of V in the geometric plane.

Our AHHK algorithm iteratively adds the edge e_{ij} and the sink v_i to T, where v_i and v_j are chosen to minimize

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

for some choice of $0 \leq c \leq 1$. When $c = 0$, the algorithm constructs a tree with minimum weight. As c increases, AHHK constructs a tree with increasingly larger weight but with lower radius. When $c = 1$, AHHK is identical to Dijkstra's algorithm. The compexity of the AHHK algorithm is no worse than that of Dijkstra's algorithm, i.e., $O(n^2)$.

While previous work [2, 4, 10] guarantees the routing tree has radius no worse that a constant times optimal, the AHHK algorithm has the much stronger property that for *every* sink, l_i is within a constant factor of d_{0i}. This is particularly advantageous when the goal is to minimize average, rather than maximum, sink delay. So that the following theorem will apply to general graphs, we define the *sink radius* r_i as the sum of edge costs in the shortest path in G from v_0 to v_i. In the Manhattan plane, $r_i = d_{0i}$.

Theorem 1 *AHHK constructs a tree T with $c \cdot l_i \leq r_i$ for all sinks v_i.*

Proof: By strong induction. Assume that for every ancestor v_j of v_i in the SPT, $c \cdot l_j \leq r_j$. Consider a snapshot of T immediately before AHHK adds sink v_i to T. Let v_j be the sink in T which is the closest ancestor to v_i in the SPT (possibly $j = 0$), and let v_k be the sink lying immediately past v_j along the shortest $v_0 - v_i$ path (v_k is not yet in T, and possibly $k = i$). Also, let v_m denote the parent of v_i in T. Since AHHK adds v_i before v_k, $c \cdot l_i \leq c \cdot l_m + d_{mi} \leq c \cdot l_j + d_{jk}$. By the inductive hypothesis, $c \cdot l_j \leq r_j$, and by the principle of optimality of shortest paths, $r_j + d_{jk} = r_k \leq r_i$. Combining these inequalities yields $c \cdot l_i \leq r_i$. $\square$

For any fixed value of c, there exists a general graph instance on which AHHK will yield a routing tree with unbounded cost ratio; see Fig. 3. However, we conjecture that the cost ratio is bounded when the signal net V is embedded in the Manhattan plane.

4 Experimental Results

The MST-SPT Tradeoff formulation requires that an algorithm accept an input parameter α and produce a tree with radius at most α times optimal. AHHK 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. Similarly, BRBC [4] accepts a parameter $\epsilon \geq 0$ and produces a routing tree with radius at most $1 + \epsilon$ times optimal. Thus, $c \equiv 1/\alpha$ and $\epsilon \equiv \alpha - 1$. By fixing α we see that $c \equiv 1/(1 + \epsilon)$.

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 using the corresponding ϵ values, which ranged from ∞ to 0. These trees represent the *families* of output trees that may be gener-

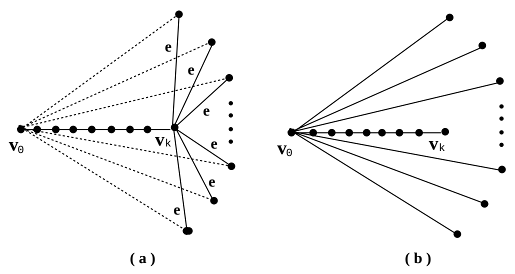

Figure 2: Pathological graph input G. G has a path of closely-spaced sinks between v_0 and v_k with path weight $\frac{1}{c} - e$ (for e small). G also contains edges of weight e from v_k to all other sinks that are not in this path. Edges shown as dotted lines are of weight 1, and remaining edges (not shown) have prohibitively large weight. AHHK result (b) has cost an unbounded factor larger than the MST (a).

AHHK	BRBC	Ave Cost vs. MST		Ave Radius vs. SPT	
c	ϵ	AHHK	BRBC	AHHK	BRBC
0.00	∞	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: Cost (versus MST cost) and radius (versus SPT radius), for nets with 16 sinks. We portray the entire families of routing trees generated by AHHK for c at intervals of 0.05 and by BRBC for corresponding ϵ values.

ated by each algorithm over the range of c or ϵ values.[2] In the experimental results of Tables 1-2, 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 random nets 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 show only results for n = 16 (Tables 1-2), but we include summary results for all four net sizes in Table 3. 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 Table 2, we report the results of delay simulations. Our inputs correspond to two distinct technologies: (i) *IC* is a representative 0.8μ CMOS process, and (ii) *MCM* is typical of current MCM technologies, with lower driver resis-

[2] For $c = 1.00$, we actually used $c = 0.9999$, in order to take advantage of staircase shortest paths in Manhattan geometry.

IC Technology					
Parameter		Max Delay vs. MST		Ave Delay vs. MST	
c	ϵ	AHHK	BRBC	AHHK	BRBC
0.00	∞	1.000	1.000	1.000	1.000
0.05	19.00	0.981	1.000	0.981	1.000
0.10	9.00	0.895	1.000	0.908	1.000
0.15	5.67	0.859	1.000	0.867	1.000
0.20	4.00	0.817	1.000	0.833	1.000
0.25	3.00	0.786	1.000	0.809	1.000
0.30	2.33	0.761	1.000	0.791	1.000
0.35	1.86	0.743	1.000	0.782	1.000
0.40	1.50	0.728	1.000	0.770	1.000
0.45	1.22	0.720	1.002	0.761	1.005
0.50	1.00	0.716	1.005	0.760	1.014
0.55	0.82	0.720	0.992	0.763	1.009
0.60	0.67	0.718	0.966	0.762	0.990
0.65	0.54	0.717	0.974	0.762	0.996
0.70	0.43	0.722	0.967	0.766	0.997
0.75	0.33	0.730	0.923	0.774	0.970
0.80	0.25	0.741	0.904	0.793	0.966
0.85	0.18	0.749	0.898	0.804	0.967
0.90	0.11	0.763	0.858	0.819	0.932
0.95	0.05	0.785	0.853	0.846	0.934
1.00	0.00	0.803	0.795	0.868	0.872
MCM Technology					
Parameter		Max Delay vs. MST		Ave Delay vs. MST	
c	ϵ	AHHK	BRBC	AHHK	BRBC
0.00	∞	1.000	1.000	1.000	1.000
0.05	19.00	0.965	1.000	0.962	1.000
0.10	9.00	0.854	1.000	0.859	1.000
0.15	5.67	0.813	1.000	0.808	1.000
0.20	4.00	0.754	1.000	0.753	1.000
0.25	3.00	0.715	1.000	0.718	1.000
0.30	2.33	0.677	1.000	0.685	1.000
0.35	1.86	0.645	1.000	0.662	1.000
0.40	1.50	0.626	1.000	0.638	1.000
0.45	1.22	0.610	1.000	0.621	1.004
0.50	1.00	0.603	0.995	0.614	1.006
0.55	0.82	0.596	0.976	0.602	0.994
0.60	0.67	0.590	0.943	0.595	0.969
0.65	0.54	0.579	0.941	0.582	0.967
0.70	0.43	0.578	0.921	0.575	0.954
0.75	0.33	0.580	0.868	0.578	0.920
0.80	0.25	0.573	0.835	0.576	0.900
0.85	0.18	0.575	0.809	0.576	0.877
0.90	0.11	0.575	0.753	0.574	0.822
0.95	0.05	0.585	0.734	0.582	0.803
1.00	0.00	0.594	0.653	0.593	0.709

Table 2: Comparison of signal delays (computed using two-pole simulator of [17]) for nets with 16 sinks. We compare the performance of AHHK versus BRBC at each value of α. IC (top): AHHK yields 18.0% improvement over BRBC in max delay, and 16.5% improvement in average delay. MCM (bottom): AHHK yields 27.2% improvement over BRBC in max delay, and 28.9% improvement in average delay.

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

Table 3: Summary table. Direct comparison of the *best* trees in the AHHK and BRBC families of solutions for each instance (results averaged over 50 random instances). We also show the average value of the parameter c that yielded the tree with best delay.

tance and unit wire resistance.[3] The delays at all sink nodes were measured using the two-pole circuit simulator proposed by Zhou et al. [17] and discussed in [3]. This simulator is a computationally efficient code which has produced very accurate results (within a few percent) when tested against SPICE. 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. While BRBC tends to yield lower tree cost for any fixed α, if we consider the family of trees over each instance, we see that the AHHK solution will almost always have lower cost for any given tree radius.

2. AHHK yields lower worst-case and average-case signal delay than BRBC (Table 2). As nets become larger, MST radius becomes quite large, and the delay improvements from the AHHK solution become more obvious.

3. While tree cost closely reflects Elmore delay in the IC technology, differences between the algorithms are small since both will closely approximate the MST for those values of α that minimize delay. On the other hand, since tree radius is dominant for MCM interconnects, AHHK is superior in this technology since it gives less cost for a given radius.

4. For each instance of each net size, we recorded the c and ϵ values that afforded lowest signal delay over the entire family of 21 trees generated by each algorithm. The average "best" parameterization is reported in Table 3, and is useful in guiding the application of AHHK when it is too expensive to generate an entire family of trees.

5 Conclusions

Analysis of distributed RC delay shows that performance-driven routing requires tree constructions that can trade off cost and radius according to interconnect technology and net size. Previous approaches [2, 4, 10] 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 [4] 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 [4], in both IC and MCM interconnect technologies. We are pursuing integration of the AHHK construction within existing performance-driven global routers.

Although AHHK can yield tree cost an unbounded factor greater than the MST cost in general graphs, we conjecture that AHHK trees have bounded cost for instances that are embedded in the Manhattan plane, i.e., there exists a constant β such that AHHK will always produce a $(1/c, \beta)$-tree for any given value of c. We are also considering alternate Prim-Dijkstra tradeoffs, as described in [1].

[3]Specifics of the technology files (IC,MCM): driver resistance = (100,25) Ω; 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) $fF/\mu m$; layout area = $(10^2, 100^2)$ mm^2.

References

[1] C. J. Alpert, T. C. Hu, J. H. Huang, A. B. Kahng and D. Karger, "On the Prim-Dijkstra Tradeoff", *submitted for publication*, 1992.

[2] 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.

[3] 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*, (1992), pp. 35-40.

[4] 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.

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

[6] 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.

[7] 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.

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

[9] 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.

[10] 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.

[11] 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.

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

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

[14] 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.

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

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

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