Near-Optimal Critical Sink Routing Tree Constructions*

Kenneth D. Boese, Andrew B. Kahng, Bernard A. McCoy†, and Gabriel Robins†

CS Dept., University of California at Los Angeles, Los Angeles, CA 90024-1596
† CS Dept., University of Virginia, Charlottesville, VA 22903-2442

Abstract

We present *critical-sink routing tree* (CSRT) constructions which exploit available critical-path information to yield high-performance routing trees. Our CS-Steiner and "Global Slack Removal" algorithms together modify traditional Steiner tree constructions to optimize signal delay at identified critical sinks. We further propose an iterative *Elmore routing tree* (ERT) construction which optimizes Elmore delay *directly*, as opposed to heuristically abstracting linear or Elmore delay as in previous approaches. Extensive timing simulations on industry IC and MCM interconnect parameters show that our methods yield trees that significantly improve (by averages of up to 67%) over minimum Steiner routings in terms of delays to identified critical sinks. ERTs also serve as generic high-performance routing trees when no critical sink is specified: for 8-sink nets in standard IC (MCM) technology, we improve average sink delay by 19% (62%) and maximum sink delay by 22% (52%) over the minimum Steiner routing. These approaches provide simple, basic advances over existing performance-driven routing tree constructions, including the recent works of [1, 9]. Our results are complemented by a detailed analysis of the accuracy and *fidelity* of the Elmore delay approximation; we also *exactly* assess the suboptimality of our heuristic tree constructions. In achieving the latter result, we develop a new characterization of Elmore-optimal routing trees, as well as a decomposition theorem for optimal Steiner trees, which are of independent interest.

1 Introduction

Due to the scaling of VLSI technology, interconnection delay has become a dominant concern in the design of complex, high-performance circuits [12, 33]. Performance-driven layout design has thus become an active area of research over the past several years. In this paper, we develop a new *critical-sink* problem formulation and new solutions for performance-driven routing tree design.

For a given signal net, the typical goal of performance-driven routing is to minimize average or maximum source-sink delay. Much early work implicitly equates optimal routing with minimum-cost Steiner routing. For example, Dunlop et al. [13] use static timing analysis to yield net priorities, so that the highest-priority nets may be routed by minimum Steiner trees, leaving lower-priority nets to subsequently encounter blockages. Jackson, Kuh, and Marek-Sadowska [20] and Prastjutrakul and Kubitz [27] have given approaches which are tuned to building-block layout and allow prescribed upper bounds on individual source-sink delays;

*Partial support for this work was provided by a GTE 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. ABK was also supported by NSF MIP-9117328 during a Spring 1993 sabbatical visit to UC Berkeley.

the former work also incorporates a hierarchy-based net ordering. For minimum Steiner tree routing, the 1-Steiner method [21] is the best-performing heuristic, and we therefore use it as a basis for comparison below.

In [8], Cohoon and Randall proposed a heuristic which simultaneously considered both the *cost* (total edge length) and the *radius* (longest source-sink path length) of the routing tree. A more general formulation was given by Cong et al. [9], wherein a parameter ϵ guides the tradeoff between cost and radius minimization; the same authors in [9] proposed the "provably good" BRBC (bounded-radius, bounded-cost) algorithm, which affords both cost and radius simultaneously within *constant* factors of optimal. The BRBC method and works of Awerbuch et al. [3] and Khuller et al. [22] all achieve a smooth cost-radius tradeoff via the same basic idea: (i) make a depth-first traversal of the minimum spanning tree over the signal net, and (ii) if the accumulated path length from the source to some sink becomes too large, modify the tree to reduce that particular source-sink path length. The cost-radius tradeoff may also be viewed as one between competing minimum spanning tree (MST) (or minimum-cost Steiner tree) and shortest-path tree (SPT) constructions. Using this perspective, Alpert et al. [1] recently proposed the AHHK algorithm, which achieves a direct MST-SPT tradeoff. Finally, Cong et al. [10] have recently proposed the use of rectilinear Steiner arborescences [29], or A-trees; these are essentially minimum-cost SPTs with Steiner points allowed. Since the delay performance of the AHHK algorithm is considerably superior to that of the BRBC or A-tree constructions [1], below we shall use AHHK as another basis of comparison with our new methods.

1.1 Motivations For Critical-Sink Routing

In performance-driven layout for cell-based designs, timing-critical paths are determined by static timing analysis, and modules in these paths are then placed close together (see, e.g., [12, 17, 19, 25, 26, 33]). The static timing analysis thus *iteratively* drives changes within both the module placement and the global routing phases. Our contribution stems from carefully considering routing tree constructions within this overall performance-driven layout process.

In general, existing performance-driven placement algorithms may be classified as either *net-oriented* or *path-oriented*. *Net-oriented* placement typically uses centroid-connected star cost [32], probabilistic estimates of Steiner tree cost [19], minimum spanning tree cost [12] or the bounding box semiperimeter [26] to estimate wire capacitance and signal delay for a multi-terminal net. From this information, critical timing paths between primary inputs and primary outputs are computed, after which module placements are updated to reduce these "net-based" objectives for signal nets along the critical paths. By contrast, *path-oriented* placement considers delay between the source and a particular *critical sink* of a multi-terminal net. The critical sink is typically determined via timing analysis using known module delays and estimated path delays. For example, Lin and Du [25] use a linear delay approximation so that their method updates the module placement to reduce the rectilinear distance between sources and critical sinks. Other path-oriented methodologies include those of Hauge et al. [17] and Teig et al. [34].

If a timing-critical path passes through a given net, the path-oriented approach can provide an explicit bound on delay at that net's critical sink. While the net-oriented approach may arguably provide only implicit routing constraints, it is still easy to identify critical sinks after the timing analysis has been performed, or *a priori* by finding paths in the design that contain more module delays. This reveals a "placement-routing mismatch": the performance-driven routing constructions reviewed above generally address *net-specific* objectives (min cost, min radius, cost-radius tradeoffs, etc.) and do not exploit the critical-path information that is available during iterative performance-driven layout. As a consequence, designers cannot realize the full benefit of high-quality timing-driven module placements. With this in mind, our work develops new high-performance routing tree constructions which *directly* exploit available critical-path timing information.

1.2 The Critical-Sink Routing Tree Problem

A *signal net* N consists of a set of pin locations $\{n_0, n_1, ..., n_k\}$ in the Manhattan plane, which are to be connected by a *routing tree* $T(N)$. We use n_0 to denote the *source*, with the n_i $(1 \leq i \leq k)$ denoting *sinks.* The *cost* of an edge e_{ij} in $T(N)$, denoted by d_{ij}, is the Manhattan distance between the endpoints n_i and n_j of the edge. The cost of the tree $T(N)$ is simply the sum of its edge costs. In a given routing tree $T(N)$, the signal delay between two terminals n_i and n_j is denoted by $t(n_i, n_j)$; the shorthand notation $t(n_i)$ indicates the delay from the source to the sink n_i. Finally, we allow each n_i to have an associated *criticality,* α_i, reflecting the timing information obtained during the performance-driven placement phase. Our goal is to construct a routing tree $T(N)$ which minimizes the weighted sum of sink delays:

Critical-Sink Routing Tree (CSRT) Problem: Given a signal net $N = \{n_0, n_1, ..., n_k\} \subset \Re^2$ with source n_0 and possibly varying sink criticalities $\alpha_i \geq 0$, $i = 1, \ldots, k$, construct a routing tree $T(N)$ such that $\sum_{i=1}^{k} \alpha_i \cdot t(n_i)$ is minimized.

This CSRT problem formulation is quite general, and easily captures traditional performance-driven routing tree objectives: (i) *average delay* to all sinks is minimized by using all α_i = some positive constant, then taking the L_1 sum of the weighted delays; and (ii) *maximum delay* to any sink is minimized by using all α_i = some positive constant, then taking the L_∞ sum of the weighted delays. In the discussion below, we will concentrate on the simple yet realistic case where *exactly one* critical sink, denoted by n_c, has been identified. In other words, we assume that $\alpha_c > 0$ and that all other $\alpha_i = 0$. Our methods may be generalized to the case where a small number of critical sinks is specified.

The remaining discussion is organized as follows. Section 2 discusses the appropriate choice of a delay measure to guide the routing tree design, and derives motivating observations from analysis of the Elmore approximation for signal delay in distributed RC trees. Section 3 then presents our two main classes of CSRT algorithms. We first describe the *CS-Steiner* method, which perturbs an existing Steiner tree construction to account for the presence of identified critical sinks. We then propose an efficient class of *Elmore routing tree* (ERT) constructions which not only yield good CSRT solutions, but are also the first methods to optimize Elmore delay *directly* without any of the abstractions implicit in previous routing objectives. Section 3 also describes the extension of the ERT approach to net-dependent routing objectives. Experimental results are presented in Section 4, where we compare delays at critical sinks in our heuristic tree topologies with analogous delays obtained using the best-performing minimum Steiner tree heuristic [21] and the AHHK routing [1]. Our methods prove extremely effective, obtaining up to an *average* 69% reduction in signal delay to identified critical sinks in 8-sink nets. The ERT approach also yields *generic* high-performance routing trees when all sinks are equally critical: for 9-pin nets in 1.2μ CMOS IC (MCM) technology, we improve average sink delay by 19% (62%) and maximum delay by 22% (52%) over the minimum Steiner routing. We thus obtain a significant advance over every existing performance-driven routing tree construction in the literature, including such recent works as [1] [9] [27]. Our results are complemented by a detailed analysis of the accuracy and *fidelity* of the Elmore delay approximation, and we furthermore provide *exact assessments versus optimal* for our heuristic tree constructions. To determine the latter data, we have developed a new theoretical characterization of Elmore-optimal routing trees, as well as a decomposition theorem for (Elmore-) optimal Steiner trees, which are of independent interest.

2 On Delay Approximations and Tree Design Objectives

For arbitrary signal nets N, the appropriate objective to use in *efficiently* constructing "high-performance routing trees" has not yet been established. In this section, we first consider necessary qualities for a delay approximation that is to be used in routing tree design. By studying both the relative accuracies and the relative *fidelities* of linear, distributed RC, distributed RCL, and SPICE-computed delay approximations, we demonstrate that Elmore's distributed RC delay approximation is of surprisingly high fidelity with respect to SPICE3e2. From Elmore's simple formula for the first-order moment of the impulse response in a distributed RC tree, we then develop revealing intuitions regarding the "correct" objective for critical-sink routing tree design.

2.1 Accuracy and Fidelity of Delay Approximations

Ideally, a routing algorithm will compute and optimize signal delays according to a detailed circuit simulation, such as that provided by SPICE. Since the computation times required by SPICE are prohibitive for routing tree construction, simpler delay approximations must be used. For example, the traditional minimum-cost Steiner tree objective, in addition to minimizing wiring area, conforms to a *lumped-capacitance* model (i.e., signal delay is proportional to total tree capacitance, which is proportional to tree cost). In [8, 9, 33], the *linear* delay approximation is used; sink delays are thus proportional to source-sink path lengths, and a minimum-radius criterion is obtained.

Name	IC1	IC2	IC3	MCM
Technology	2.0 μm CMOS	1.2 μm CMOS	0.5 μm CMOS	MCM
r_d	164.0 Ω	212.1 Ω	270.0 Ω	25.0 Ω
unit wire resistance	0.033 $\Omega/\mu m$	0.073 $\Omega/\mu m$	0.112 $\Omega/\mu m$	0.008 $\Omega/\mu m$
unit wire capacitance	0.234 $fF/\mu m$	0.083 $fF/\mu m$	0.039 $fF/\mu m$	0.06 $fF/\mu m$
unit wire inductance	1x10^{-5} $fH/\mu m$	1x10^{-5}$fH/\mu m$	1x10^{-5}$fH/\mu m$	380 $fH/\mu m$
loading capacitance	5.7 fF	7.06 fF	1.0 fF	1000 fF
resistance ratio (x $10^6 \mu m$)	0.0050	0.0029	0.0024	0.0031
chip size	1x1 cm^2	1x1 cm^2	1x1 cm^2	10x10 cm^2

Table 1: Technology parameters for three MOSIS CMOS IC technologies and an MCM technology. Parasitics and SPICE simulation decks for the IC1 and IC2 technologies are provided by MOSIS; IC3 parasitics are courtesy of MCNC; MCM interconnect parasitics are courtesy of Professor Wayne W.-M. Dai of UC Santa Cruz, and correspond to data provided by AT&T Microelectronics Division. The driver resistances (r_d) and sink loading capacitances are derived for minimum-size transistors.

Such simple delay approximations are known to be inaccurate as technology scales, e.g., smaller wire geometries imply that resistive effects of the interconnect become more dominant, particularly in relation to driver on-resistance (see the discussion below of "resistance ratio" effects, and note the four technology characterizations in Table 1). Furthermore, higher system speeds and packing densities may expose inductive effects on delay. Given these considerations, distributed RC delay approximations (e.g., that of Elmore [14]) or distributed RCL delay approximations (e.g., the "Two-Pole" simulator of Zhou et al. [37]) are of interest, since they are more accurate than linear or lumped-capacitance approximations while requiring less computation time than SPICE.

Elmore delay [14] [31] is defined as follows. Given routing tree $T(N)$ rooted at n_0, let e_i denote the edge from n_i to its parent. The resistance and capacitance of edge e_i are denoted by r_{e_i} and c_{e_i}, respectively. Let T_i denote the subtree of T rooted at n_i, and let c_i denote the sink capacitance of n_i. We use C_i to denote the *tree capacitance* of T_i, namely the sum of sink and edge capacitances in T_i. Using this notation, the Elmore delay along edge e_i is equal to $r_{e_i}(c_{e_i}/2 + C_i)$. Let r_d denote the output driver resistance at the net's source. The Elmore delay $t_{ED}(n_i)$ at sink n_i is:

$$t_{ED}(n_i) = r_d C_{n_0} \quad + \sum_{e_j \in path(n_0,n_i)} r_{e_j}(c_{e_j}/2 + C_j) \tag{1}$$

Although Elmore delay has a compact definition and can be quickly computed[1], it does not capture all of the factors that account for delay. For example, the "Two-Pole" distributed RCL simulator [37] considers inductive effects; according to [4] and [37], its moment-based methodology is intermediate between SPICE and Elmore delay in both accuracy and computational efficiency.

[1]Elmore delay can be evaluated at *all* sinks in $O(k)$ time, as noted by Rubinstein et al. [31]. The calculation uses two depth-first traversals: (1) to compute the delay along each edge and (2) to sum up the delays along each source-sink path. This fact is enabling to the efficient ERT methodology that we propose in Section 3.2.

2.1.1 Accuracy

In choosing a delay simulator, one traditionally measures *accuracy*, which may vary with the circuit technology and the specifics of a net (for instance, the number of pins it contains, or the size of its bounding box). Table 2 indicates the accuracy of the linear, Elmore and Two-Pole models for each of the interconnect technologies described in Table 1. The Table gives the average ratio between SPICE delay and each of the two estimators[2], and also shows the consistency of this ratio in terms of its standard deviation. For each net size, the results are computed from 100 random nets connected using the minimum cost spanning tree (MST) construction. We use MSTs rather than random tree topologies so that our comparisons will be for relatively good (although not necessarily optimal) routing solutions; note that for these test sets, it is not feasible to determine optimal-delay topologies by exhaustive enumeration using SPICE.

Accuracy of Linear, Elmore and Two-Pole Delay Estimates					
		$\lvert N\rvert = 4$		$\lvert N\rvert = 7$	
	Delay Ratio	average	std dev	average	std dev
IC1	SPICE/Linear†	1.0	28.4%	1.0	32.7%
	SPICE/Elmore	0.72	13.5%	0.69	15.4%
	SPICE/2-Pole	1.27	13.5%	1.23	15.4%
IC2	SPICE/Linear†	1.0	33.9%	1.0	38.8%
	SPICE/Elmore	0.74	16.1%	0.70	17.8%
	SPICE/2-Pole	1.30	15.9%	1.23	17.8%
IC3	SPICE/Linear†	1.0	34.9%	1.0	40.3%
	SPICE/Elmore	0.78	16.0%	0.72	17.8%
	SPICE/2-Pole	1.36	15.7%	1.27	17.9%
MCM	SPICE/Linear†	1.0	57.1%	1.0	61.6%
	SPICE/Elmore	0.69	20.5%	0.65	25.1%
	SPICE/2-Pole	1.20	20.8%	1.14	25.2%

Table 2: Accuracy of the Linear, Elmore and Two-Pole estimators. The table gives the average ratio and standard deviation of the ratios between SPICE3e2 delay and estimated delay computed over 100 random nets. All nets are connected using MST constructions. Standard deviations are reported as a percent of the average ratio. (†) Linear delay is scaled to give an average ratio of 1.00; hence, only the reported standard deviations are meaningful for linear delay.

The inaccuracy of the linear approximation is expected. It is also reasonable to expect "poor" accuracy of the Elmore and Two-Pole approximations with respect to SPICE, in light of the somewhat ill-defined state of delay modeling and analysis (cf. the many modeling options described in Footnote 2). For instance, Elmore delay intrinsically corresponds to 50% rise time [14], since it is the first moment of an impulse response. While we therefore use delay time $\equiv$ 50% rise time for all simulators, the nature of the Two-Pole

[2] Again, we equate SPICE3e2 results with "actual delay". Our range of modeling methodologies has been quite comprehensive, and may be summarized as follows. SPICE delay modeling uses constant unit resistance and capacitance values which vary with each interconnect technology. The root of the routing tree is driven by a resistor connected to the source; we thus remove some driver attributes since we are concentrating on measuring delay within the interconnect. For the Two-Pole and SPICE simulators, every interconnect segment is broken into uniform segments, each at most 100th the length of the layout dimension, connected in series. To model sink loads, we have used both uniformly-sized CMOS inverters and pure capacitive loads; these are derived using minimum-size transistors. For all simulators, we have used both the 50% and 90% rise time delay criteria, and we have measured both average sink delay and maximum sink delay. The reported data correspond to typical results (we generally report data for 50% rise times and pure capacitive sink loads), with the specific choices of methodology described in the accompanying table captions and text.

approximation makes it more suited to a 90% rise time criterion [37]. While SPICE can model active devices as loads, the Two-Pole simulator can only handle "equivalent" sink capacitances; while SPICE and Two-Pole can model series inductance (for MCM interconnect), Elmore delay is solely a distributed RC model (and extracted inductance parameters are often not distributed with IC technology files) – indeed, the list of incomparable variables seems endless. However, despite their seeming accuracy, each choice of modeling methodology shows that the Elmore and Two-Pole delay estimators are highly consistent, with typically small standard deviations and 95% confidence intervals. Thus, it is possible that precomputed "correction factors" can compensate for inaccuracy in these estimates.

2.1.2 Fidelity

A key observation is that precise accuracy is *not* really required of delay estimates used to construct routing trees. In practice, we only require that an estimator have a high degree of *fidelity* – i.e, an an optimal or near-optimal solution according to the estimator should also be nearly optimal according to actual delay. To this end, we have defined a measure of fidelity vis-a-vis an exhaustive enumeration of all possible routing solutions: we first rank *all* tree topologies[3] by the given delay model, then rank the topologies again by SPICE delay, and then find the average difference between the two rankings for each topology. This measure of fidelity corresponds to a standard rank-ordering technique used in the social sciences [2]. We have run simulations to estimate this measure of fidelity for nets of size 4 and 5 using the various delay estimators and each of the four technologies.

Table 3 shows the fidelity to SPICE of the linear, Elmore, and Two-Pole delay estimators; the delay criterion is the 50% delay time to a given randomly-chosen critical sink in the net. We report the average difference in ranking over all topologies; the average rank difference for the topology which has lowest delay according to the estimator; and the average difference for the five topologies which have lowest delay according to the estimator. Our results show that Elmore delay has high fidelity, particularly when we compare the SPICE ranking of the optimal topology for Elmore delay with the optimal topology for linear delay. For example, with nets of size 5 using technology IC3, optimal topologies under Elmore delay averaged only 2.3 rank positions (out of 125) away from optimal according to SPICE. In comparison, the best topology under linear delay averaged distance 24.7 from its correct SPICE ranking. For 5-pin nets under the IC1 and IC2 technologies, the best topology under Elmore delay also has a near-optimal SPICE ranking: on average the distance from its SPICE ranking is 3.5 for IC1 (versus 4.6 under linear delay) and 1.5 for IC2 (versus 6.3 under linear delay). Table 4 gives similar results when the delay criterion is the maximum 50% delay time to any sink in the net. (The choice of source location in the experimental methodology can significantly color the results. To illustrate this phenomenon, we show IC1 and IC2 data for a randomly chosen source location in each instance, while IC3 and IC4 data are for source location fixed

[3]An early theorem of Cayley [15] implies that there are $|N|^{|N|-2}$ distinct spanning tree topologies for any given net N.

		Linear vs SPICE		Elmore vs SPICE		2-Pole vs SPICE	
	Topologies	$\|N\| = 4$	$\|N\| = 5$	$\|N\| = 4$	$\|N\| = 5$	$\|N\| = 4$	$\|N\| = 5$
IC1	Best	0.50	2.06	0.40	0.10	0.32	0.08
	5 Best	0.66	2.86	0.72	0.47	0.48	0.38
	All	0.94	7.89	0.66	1.42	0.43	1.29
IC2	Best	0.40	2.26	0.18	0.20	0.18	0.12
	5 Best	0.68	2.69	0.52	0.53	0.47	0.42
	All	0.88	7.17	0.43	1.27	0.39	1.11
IC3	Best	3.36	24.2	0.58	4.6	0.43	5.0
	5 Best	2.49	21.8	0.78	5.2	0.61	4.9
	All	2.42	21.5	0.69	6.2	0.58	5.9
MCM	Best	4.00	30.0	0.44	4.6	0.36	6.1
	5 Best	2.62	24.3	0.57	4.8	0.43	3.9
	All	2.63	22.9	0.62	5.9	0.56	5.7

Table 3: Average difference in rankings of topologies, in terms of 50% delay to a given random **critical sink** in each net, according to different delay estimates. The sample consists of 50 random nets of each cardinality. The total number of topologies for each net is $4^{(4-2)} = 16$ for $|N| = 4$, and $5^{(5-2)} = 125$ for $|N| = 5$. To show the effect of source location in the experimental design, IC1 and IC2 runs correspond to *random* source locations, while the IC2 and MCM runs have the source pin located always in the lower-left corner of the layout.

		Linear vs SPICE		Elmore vs SPICE		2-Pole vs SPICE	
	Topologies	$\|N\| = 4$	$\|N\| = 5$	$\|N\| = 4$	$\|N\| = 5$	$\|N\| = 4$	$\|N\| = 5$
IC1	Best	0.50	2.06	0.40	0.10	0.32	0.08
	5 Best	0.67	2.86	0.72	0.47	0.47	0.38
	All	0.95	7.89	0.65	1.42	0.42	1.29
IC2	Best	0.40	2.48	0.18	0.28	0.18	0.20
	5 Best	0.68	3.00	0.52	0.73	0.47	0.60
	All	0.88	7.19	0.43	1.37	0.39	1.22
IC3	Best	1.60	9.16	0.20	0.46	0.14	0.26
	5 Best	1.21	7.68	0.30	0.77	0.12	0.38
	All	1.07	7.90	0.27	1.39	0.14	1.02
MCM	Best	2.72	15.26	0.14	0.38	0.02	0.18
	5 Best	1.61	10.64	0.11	0.51	0.04	0.20
	All	1.31	8.50	0.10	1.00	0.07	0.84

Table 4: Average difference in rankings of topologies in terms of **maximum sink delay** according to different delay estimates. The sample consists of 50 random nets of each cardinality. The total number of topologies for each net is $4^{(4-2)} = 16$ for $|N| = 4$, and $5^{(5-2)} = 125$ for $|N| = 5$. To show the effect of source location in the experimental design, IC1 and IC2 runs correspond to *random* source locations, while the IC2 and MCM runs have the source pin located always in the lower-left corner of the layout.

in the lower left corner of the layout, i.e., at a highly skewed position. Note that all further results are for random source locations.) Kim, Owens and Irwin [23] have similarly established the fidelity of Elmore delay for circuit design: they plotted Elmore- versus SPICE-computed delays for a suite of 209 different place/route solutions of the same ripple-carry adder circuit, and also found a very high correlation between the two delay measures.

IC1			IC2			IC3			MCM		
1-10	11-20	21-25	1-10	11-20	21-25	1-10	11-20	21-25	1-10	11-20	21-25
1.000	1.324	1.619	1.000	1.361	1.732	1.000	1.092	1.390	1.000	1.044	1.517
1.056	1.350	1.645	1.051	1.396	1.761	1.003	1.135	1.428	1.000	1.049	1.549
1.093	1.372	1.667	1.094	1.424	1.783	1.005	1.140	1.447	1.000	1.049	1.604
1.123	1.403	1.688	1.136	1.458	1.808	1.006	1.170	1.474	1.001	1.053	1.648
1.158	1.434	1.708	1.160	1.494	1.841	1.006	1.180	1.527	1.002	1.055	1.700
1.186	1.453		1.194	1.522		1.006	1.222		1.003	1.085	
1.209	1.514		1.226	1.607		1.009	1.259		1.009	1.306	
1.237	1.540		1.256	1.638		1.011	1.292		1.012	1.307	
1.266	1.577		1.289	1.680		1.018	1.341		1.014	1.348	
1.297	1.597		1.333	1.707		1.083	1.365		1.042	1.507	

Table 5: Average SPICE3e2 delay ratios for the best 25 topologies according to delay at a randomly chosen **critical sink** in each net, for $|N| = 5$. Values are normalized to the delay of the best topology and averaged over 20 random nets. For the very worst topology (rank 125), the average ratios are 5.51 (IC1), 6.61 (IC2), 9.51 (IC3), and 20.20(MCM).

IC1			IC2			IC3			MCM		
1-10	11-20	21-25	1-10	11-20	21-25	1-10	11-20	21-25	1-10	11-20	21-25
1.000	1.324	1.619	1.000	1.359	1.747	1.000	1.198	1.409	1.000	1.288	1.676
1.053	1.350	1.645	1.054	1.396	1.773	1.023	1.212	1.428	1.013	1.322	1.707
1.093	1.372	1.667	1.093	1.425	1.795	1.051	1.230	1.455	1.031	1.348	1.744
1.123	1.403	1.688	1.139	1.456	1.824	1.063	1.243	1.481	1.061	1.393	1.783
1.158	1.434	1.708	1.167	1.493	1.855	1.082	1.259	1.504	1.097	1.425	1.811
1.186	1.453		1.199	1.523		1.103	1.277		1.128	1.461	
1.209	1.514		1.234	1.609		1.133	1.326		1.172	1.528	
1.237	1.540		1.261	1.640		1.156	1.347		1.202	1.557	
1.266	1.577		1.293	1.689		1.171	1.374		1.235	1.601	
1.297	1.597		1.332	1.720		1.180	1.393		1.252	1.643	

Table 6: Average SPICE3e2 delay ratios for the best 25 topologies according to **maximum** sink delay, for $|N| = 5$. Values are normalized to the delay of the best topology and averaged over 20 random nets. For the very worst topology (rank 125), the average ratios are 5.51 (IC1), 6.69 (IC2), 5.30 (IC3), and 7.41 (MCM).

For IC1, the average difference of 0.10 positions for 5-pin nets with optimal Elmore delay implies an approximate average suboptimality of 0.6% in terms of SPICE-computed delay. This can be seen from Table 5, which shows the average increase in SPICE delay from optimal for the 25 top-ranking topologies, i.e., the 25 lowest SPICE delays. (The exact delay criterion is 50% delay time to a randomly chosen *critical sink* in each net). For IC2, the average distance of 0.20 rank positions implies a difference of approximately 1.0% in actual SPICE-computed delay; for IC3 a distance of 4.6 rank positions implies 0.6% delay suboptimality; and for MCM a difference of 4.6 rank positions implies 0.3% delay suboptimality. Similar data is given in Table 6 for the maximum sink delay criterion.

From these results, we see that the Two-Pole simulator has somewhat better fidelity to SPICE than Elmore delay, as would be expected. However, the relatively small improvement in fidelity does not seem to justify the significantly greater computation required to search over topologies using Two-Pole as opposed to the linear-time Elmore delay computation.

2.2 Intuitions from Elmore Delay

Because of its fidelity to SPICE-computed delay, Elmore delay is a good performance objective for constructing high-performance routing trees. Moreover, the simplicity of the Elmore delay formula (1) allows us to intuit heuristics which effectively minimize delay.

Since r_{e_v} and c_{e_v} are usually proportional to the length of edge e_v, we see that $t_{ED}(n_i)$ has a quadratic relationship to the length of the n_0-n_i path, suggesting a min-radius criterion. However, the C_j term implies that Elmore delay is also linear in the total edge length of the tree which lies outside the n_0-n_i path, suggesting a min-cost criterion. The relative size of the driver resistance r_d heavily influences the optimal routing topology: if r_d is large, the optimal routing tree (ORT) is a minimum cost tree; as r_d decreases, the ORT tends to a "star" topology. The size of r_d relative to unit wire resistance is a "resistance ratio" [4] that captures the technology vis-a-vis routing tree design. Relative values of the resistance ratio are larger for current-generation CMOS, but tend to decrease in MCM substrate and some submicron CMOS IC interconnects (Table 1).

In Figure 1, we show a signal net N with identified critical sink n_c, along with three routing trees: (a) the 1-Steiner tree, (b) a minimum-cost SPT, and (c) the optimal CSRT with respect to critical sink n_c. Based on this example, the example of Figure 1(d), and Equation (1), we make the following observations.

- The minimum cost solution (a) has large delay to the critical sink n_c due to the long source-sink path.
- However, requiring a monotone path to *every* sink, as in the SPT (b) or a Steiner arborescence [10, 29], can result in large tree capacitance which again leads to large delay at n_c.
- The optimal CSRT construction (c) shows the dependence of routing topology on the choice of critical sink, and reflects both the minimum-cost and the SPT solutions.
- Finally, Equation (1) implies that the number of Steiner points in the n_0-n_c path should be minimized, and the Steiner points "shifted" toward n_0 (i.e., branches off of the n_0-n_c path should occur as close to the source as possible). Figure 1(d) shows two trees which are both shortest-path trees and minimum Steiner trees, yet the rightmost tree has less signal delay at n_c.

3 Two Classes of CSRT Heuristics

3.1 The CS-Steiner Approach

Given the observations above, we may characterize the optimal CSRT solution in Figure 1(c) as one which minimizes total tree cost, *subject to the path from n_0 to n_c being monotone.* This simultaneous consideration

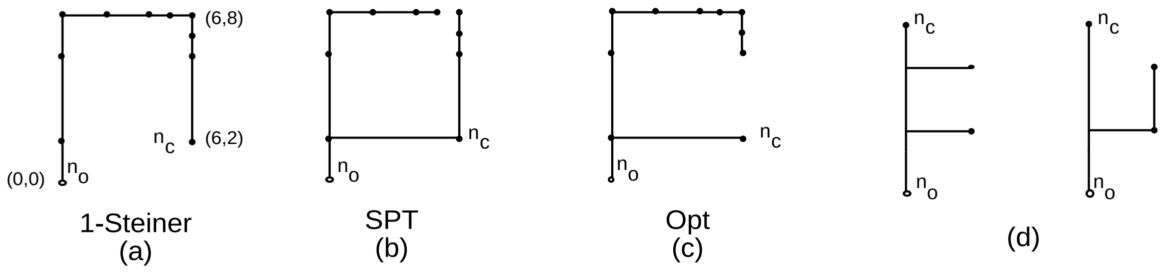


Figure 1: Parts (a)-(c): optimal Steiner tree (cost 2.0 cm, $t(n_c) = 3.34$ ns); minimum cost shortest-paths tree (cost 2.5 cm, $t(n_c) = 2.26$ ns); and optimal-delay tree (cost 2.2 cm, $t(n_c) = 1.67$ ns) for the same sink set. Coordinates shown are in mm, and the 1.2μ IC2 technology (Table 1) were used with the Two-Pole simulator with a 90% delay time criterion. Part (d): two distinct minimum-cost SPT solutions for a set of three sinks.

of radius and cost parameters recalls the motivations in [1] [8] [9], but here the tradeoff is formulated with respect to the critical sink n_c. We thus obtain our *CS-Steiner* heuristic for the CSRT problem (Figure 2).

```
CS-Steiner Algorithm
Input: signal net $N$; source $n_0 \in N$; identified critical sink $n_c \in N$
Output: heuristic CSRT solution $T$
1.  Construct heuristic minimum-cost tree $T_0$ over $N - n_c$.
2.  Form $T$ by adding a direct connection from $n_c$ to $T_0$,
      i.e., such that the $n_0$-$n_c$ path in $T$ is monotone.
```

Figure 2: The CS-Steiner heuristic.

The idea behind CS-Steiner is simple: construct a minimum-cost Steiner routing tree as usual, then "fix" the tree to reflect an identified critical sink. Since the algorithm template is quite general, we have examined a number of CS-Steiner variants. All of our variants use the 1-Steiner heuristic of Kahng and Robins [21] to construct the initial tree T_0 in Line 1. Section 4 reports results for the following three variants:[4]

H0: The direct connection in Line 2 consists of a single wire from n_c to n_0.

H1: The direct connection in Line 2 consists of the shortest possible wire that can join n_c to T_0, subject to the monotone path constraint.

HBest: Accomplish Line 2 by trying all shortest connections from n_c to edges in T_0, as well as from n_c to n_0; perform timing analysis on each of these routing trees, and return the tree with lowest delay at n_c.

[4]We also studied two additional variants. **Variant H2** modifies Line 1 of CS-Steiner so that the initial heuristic tree T_0 is constructed over the entire net N. H2 then deletes the edge which lies directly above n_c when we root T_0 at n_0, and rejoins (the component containing) n_c to (the component containing) n_0 using a shortest possible wire from n_c, as in variant H1. **Variant H3** performs Lines 1 and 2 simultaneously by executing the 1-Steiner algorithm subject to a "maintaining monotone feasibility" constraint. In other words, we iteratively choose a Steiner point which minimizes the sum of the tree cost and the cost of any needed direct connection from n_c to n_0. The direct connection from n_c requires that there exist a monotone path through the "bounding boxes" of the edges in the path to n_0. Intuitively, this favors initial choice of Steiner nodes along some monotone path from n_0 and n_c, since such nodes will most rapidly reduce the marginal cost of adding the direct n_c-n_0 connection. The H2 and H3 variants were inferior to H0, H1 and HBest.

The complexity of these variants is dominated by the construction of T_0 in Line 1 (or possibly by the simulator calls in HBest).

We enhance the CS-Steiner construction via an efficient *Global Slack Removal* (GSR) postprocessing algorithm. GSR [5] is similar to the method developed independently by Chen and Sarrafzadeh in [7], which also removes "U's" from interconnections. However, the objective of GSR is not to reduce tree cost (which is already effectively minimized by the 1-Steiner algorithm) but rather to maximize the monotonicity of all source-sink paths and reduce Elmore delay to all sinks. GSR accomplishes this without increasing overall tree cost. For expository reasons, we defer formal description of GSR, along with its proofs of correctness, to Appendix A.

3.2 Elmore Routing Trees

From the discussion of Section 2.2, we see that current routing objectives such as minimum tree cost, bounded tree radius, or prescribed cost-radius balance have often been *motivated* by the Elmore model. However, such objectives are abstractions: they do not actually optimize Elmore delay. Thus, the effectiveness of a given objective often depends on the prevailing technology, on the particular distribution of sink locations for a given signal net, and on the user's ability to find the parameter value (e.g., ϵ in the BRBC algorithm [9], or c in the AHHK algorithm [1]) which will yield a good solution for the particular input.

In this subsection, we depart from the abstraction inherent in "minimum cost" or "bounded radius" objectives, and propose a new class of greedy *Elmore routing tree* (ERT) algorithms which optimize Elmore delay *directly* as the routing tree is constructed. The ERT approach is efficient, since Elmore delay at all nodes of a routing tree can be evaluated in linear time (see Footnote 1 above). Based on the performance results in Section 4 for both critical-sink and "generic" performance-driven routing formulations, we believe that the ERT approach, along with its SERT and SERT-C extensions, offers a basic new tool for VLSI routing.

The Elmore routing tree (ERT) algorithm (Figure 3) is analogous to Prim's minimum spanning tree construction [28]: starting with a trivial tree containing only the source, we iteratively find a pin n_i in the tree and a sink n_j outside the tree so that adding edge e_{ij} yields a tree with minimum Elmore delay. The construction terminates when the entire net is spanned by the growing tree.[5] The ERT algorithm can be generalized to any delay model by using the appropriate estimator in Line 3.

[5]Our approach should be distinguished from the method of Prasitjutrakul and Kubitz [27] cited above, wherein A* heuristic search and the actual Elmore delay formula are used in a performance-driven routing tree construction. Like our method, [27] grows a routing tree over a net N starting from the source n_0; they perform A* search of a routing graph (e.g., in building-block design) to find the Elmore delay-optimal Steiner connection from the existing tree to a new sink. However, *the choice of this new sink is forced:* the algorithm always adds the sink that is closest (by Manhattan distance) to the existing tree, and thus falls into the standard pitfall of ignoring the underlying delay criterion. The effect of this difference is apparent in the ERT ordering of added nodes in Figure 4 of Section 4 below. Indeed, the method of [27] can yield Elmore delays substantially larger than those of ERT: given a very tall, "hairpin"-like version of Figure 1a with many sinks very closely spaced along the entire hairpin path, [27] forces the sinks to be added into the tree according to the path order (starting from the source n_0 at the lower left), yielding an obviously poor solution.

```
ERT Algorithm
Input: signal net $N$ with source $n_0 \in N$
Output: routing tree $T$ over $N$
1. $T = (V, E) = (\{n_0\}, \emptyset)$
2. While $|V| < |N|$ do
3.     Find $u \in V$ and $v \notin V$ which minimize the maximum Elmore delay
           from $n_0$ to any sink in the tree $(V \cup \{v\}, E \cup \{(u, v)\})$
4.     $V = V \cup \{v\}$
5.     $E = E \cup \{(u, v)\}$
6. Output resulting spanning tree $T = (V, E)$
```

Figure 3: The ERT Algorithm: direct incorporation of the Elmore delay formula into a heuristic routing tree construction.

The ERT algorithm is generalized to Steiner routing by allowing the new pin to connect to an *edge* of the existing tree, inducing a Steiner node at the point in the edge closest to the new pin.

- For *generic* performance-driven routing, our *Steiner Elmore routing tree* (SERT) algorithm iteratively finds $u \notin V$, $(v, v') \in E$, and a new point w on edge (v, v') to minimize the maximum source-sink Elmore delay in the tree $(V \cup \{u, w\}, (E - \{(v, v')\}) \cup \{(v, w), (w, v'), (u, w)\})$. We then add u and w to V, and replace E by $(E - \{(v, v')\}) \cup \{(v, w), (w, v'), (u, w)\}$.
- To address *critical-sink* routing, our *Steiner Elmore routing tree with identified critical sink* (SERT-C) algorithm begins with a tree containing the single edge (n_0, n_c) in Line 1 of Figure 3, then continues as in the SERT algorithm, except that we minimize $t_{ED}(n_c)$ rather than the maximum delay to all sinks.

While CS-Steiner began with a minimum-cost Steiner tree and heuristically perturbed it to improve $t(n_c)$, SERT-C uses the opposite approach of starting with the required n_0-n_c connection and growing the routing tree while keeping $t_{ED}(n_c)$ as small as possible. Again, we note that SERT-C offers a consistent, *direct* incorporation of Elmore delay within its construction, in contrast to heuristics whose objectives or strategies are only motivated by Elmore delay and whose solution quality may therefore be more sensitive to the input instance and choice of parameters.

The time complexities for our ERT variants are analyzed as follows.

Observation 1: The SERT-C algorithm can be implemented in $O(k^2 \log k)$ time.

Proof: The effect on delay $t_{ED}(n_c)$ of inserting a new edge (u, w) into T arises only in the C_j terms in Equation (1), and is an *additive constant* no matter when (u, w) is added into the tree. Initially, we compute the best connection from each non-critical sink to the tree containing only edge (n_0, n_c). For each new sink added, at most three new edges will be inserted into the tree. In constant time, we can calculate the effects of connections from a given sink outside T to these three new edges (all previously computed effects remain unchanged and need not be recomputed). We can insert the new delay effects into a priority queue for each $u \notin V$ in $O(\log k)$ time and also retrieve the current minimum-cost connection for v in $O(\log k)$ time. Thus, each pass through the **while** loop of Figure 3 can be accomplished in $O(k \log k)$ time, giving an overall time complexity of $O(k^2 \log k)$. □

Observation 2: The ERT spanning tree algorithm can be implemented in $O(k^3)$ time, assuming constant unit wire resistance, unit wire capacitance, and sink capacitances.

Proof: The result follows from a simple observation: if a new tree edge incident to sink $u \in V$ (Line 3 of Figure 3) minimizes the maximum Elmore delay $max_i t_{ED}(n_i)$, it must connect u to the sink $v \notin V$ that is closest to u. Thus, at each pass through the **while** loop, we simply compute the shortest "outside connection" for each node in V, i.e., every possible u, in $O(k^2)$ time. We then add each of the $O(k)$ shortest outside connections to T in turn. Evaluating the Elmore delays at all sinks in each of the resulting trees requires $O(k)$ time per tree. Hence, each pass through the **while** loop requires $O(k^2)$ time, and this yields the $O(k^3)$ complexity result.[6] □

In practice, the complexity of the ERT algorithm will be transparent to the user, since k is typically small (e.g., our runtimes for the problem sizes discussed here are $O(10^{-2})$ seconds on Sun SPARC1 hardware). We know of no implementation of the SERT algorithm that is faster than $O(k^4)$. Intuitively, the difficulty seems to be that (i) in Line 3 we must always consider $\Theta(k^2)$ Steiner connections, and (ii) the connection which minimizes $max_i t_{ED}(n_i)$ in Line 3 may not be the best one from the "perspective" of any individual sink in N or edge in T. Thus, we currently have a rather interesting situation where the CSRT problem formulation leads to an algorithm (SERT-C) that enjoys nearly quadratic speedup over the generic Steiner computation (SERT).

4 Experimental Results

4.1 CS-Steiner Trees

We implemented each of the CS-Steiner variants H0, H1 and HBest, along with the 1-Steiner algorithm [21], using C in the UNIX Sun environment, and ran these algorithms on random 4- and 8-sink inputs.[7] We also applied our GSR post-processing algorithm (denoted as +GSR) to 1-Steiner and each of the CS-Steiner variants. Our inputs correspond to the four distinct technologies described in Table 1.

Table 7 gives delay and tree cost (WL) results and comparisons. The delays at all sink nodes correspond to 50% rise times estimated using the Two-Pole simulator [36] [37]. Each entry in Table 7 represents an

[6]Again, we note the fundamental difference between the ERT approach and the method of [27]: while [27] must add the single sink that is closest to the existing tree, the ERT algorithm identifies both a new sink and its connection such that Elmore delay is minimized.

[7]Results for 16-sink inputs have been reported in preliminary form, e.g., [5]. While such large inputs magnify the effect of our new methods, most signal nets in practice are within the size range that we now discuss.

		IC1		IC2	
		$\lvert N\rvert = 5$	$\lvert N\rvert = 9$	$\lvert N\rvert = 5$	$\lvert N\rvert = 9$
Critical Sink Delay	1Stein	0.549 ns	0.848 ns	0.331 ns	0.520 ns
	1Stein+GSR	.978	.964	.970	.954
	H0+GSR	.980	.824	.876	.700
	H1+GSR	.960	.883	.934	.827
	HBest+GSR	.929	.817	.867	.721
Ave WL	1Stein	1.48 cm	2.18 cm	1.48 cm	2.18 cm
	H0+GSR	1.29	1.22	1.29	1.22
	H1+GSR	1.04	1.06	1.04	1.06
	HBest+GSR	1.07	1.11	1.10	1.12

		IC3		MCM	
		$\lvert N\rvert = 5$	$\lvert N\rvert = 9$	$\lvert N\rvert = 5$	$\lvert N\rvert = 9$
Critical Sink Delay (ns)	1Stein	0.218 ns	0.342 ns	2.31 ns	4.09 ns
	1Stein+GSR	.968	.950	.952	.927
	H0+GSR	.849	.664	.550	.333
	H1+GSR	.922	.810	.857	.665
	HBest+GSR	.844	.693	.593	.340
Ave WL (cm)	1Stein	1.48 cm	2.18 cm	14.8 cm	21.8 cm
	H0+GSR	1.29	1.22	1.29	1.22
	H1+GSR	1.04	1.06	1.04	1.06
	HBest+GSR	1.11	1.12	1.22	1.21

Table 7: Two-Pole simulation results comparing CS-Steiner trees with the 1-Steiner heuristic. Each entry corresponds to an average over delay computations for random critical sinks in each of 100 different random signal nets. 1-Steiner results are reported in the physical units (nanoseconds or centimeters) while other results are reported as ratios to the corresponding 1-Steiner results. Note that 1-Steiner and 1-Steiner plus GSR always produced nearly identical average costs.

average taken over every sink node in 50 random point sets. We emphasize that the 1-Steiner algorithm (or the BRBC, AHHK, etc. methods), being net-oriented, will return the same tree for a given sink set no matter which sink happens to be critical; the delays at the sinks n_i are in some sense "generic". In contrast, each of the three CS-Steiner variants can return a different tree for each choice of critical sink in the same net. Thus, for each variant we report the delay at n_i in the *specific* tree corresponding to identification of n_i as the critical sink.

Variants H0 and HBest significantly reduce delay to the critical sink, particularly in larger nets and for MCM interconnect technology where output driver and wire resistances are low. In other words, the simple strategy of connecting the critical node via a path with low branching factor is very successful for these cases. Of course, this strategy will produce larger net cost.[8]

4.2 Elmore Routing Trees

We constructed Elmore routing trees for the same sets of random inputs used in the CS-Steiner experiments. Delay simulation results, again obtained using the Two-Pole simulator, are presented in the upper

[8]Highly "star-like" topologies can possibly introduce other difficulties such as crossing wires, nodes with degree > 4, and capacitive coupling effects; these are not modeled by either SPICE or the Two-Pole simulator.

		IC1		IC2	
		$\lvert N\rvert = 5$	$\lvert N\rvert = 9$	$\lvert N\rvert = 5$	$\lvert N\rvert = 9$
Crit. Sink Delay	MST	0.645 ns	0.984 ns	0.395 ns	0.609 ns
	AHHK	.904	.837	.863	.770
	ERT	.879	.837	.804	.741
	1Stein	0.549 ns	0.848 ns	0.331 ns	0.520 ns
	SERT	.967	.884	.921	.806
	SERT-C	.947	.847	.870	.735
Max Delay	MST	0.758 ns	1.213 ns	0.485 ns	0.792 ns
	AHHK	.876	.805	.835	.747
	ERT	.855	.790	.786	.699
	1Stein	0.627 ns	1.028 ns	0.393 ns	0.664 ns
	SERT	.955	.853	.919	.780
	SERT-C	.970	.914	.962	.892
Ave WL	MST	1.64 cm	2.43 cm	1.64 cm	2.43 cm
	AHHK	1.16	1.09	1.16	1.09
	ERT	1.10	1.15	1.18	1.25
	1Stein	1.48 cm	2.18 cm	1.48 cm	2.18 cm
	SERT	1.06	1.09	1.11	1.18
	SERT-C	1.06	1.06	1.15	1.11

		IC3		MCM	
		$\lvert N\rvert = 5$	$\lvert N\rvert = 9$	$\lvert N\rvert = 5$	$\lvert N\rvert = 9$
Crit. Sink Delay	MST	0.262 ns	0.403 ns	2.82 ns	4.80 ns
	AHHK	.885	.749	.777	.608
	ERT	.782	.702	.472	.329
	1Stein	0.218 ns	0.342 ns	2.31 ns	4.09 ns
	SERT	.908	.781	.584	.384
	SERT-C	.839	.693	.567	.340
Max Delay	MST	0.326 ns	0.533 ns	3.86 ns	7.05 ns
	AHHK	.822	.730	.759	.632
	ERT	.764	.668	.544	.399
	1Stein	0.262 ns	0.444 ns	3.06 ns	5.92 ns
	SERT	.908	.759	.699	.481
	SERT-C	.954	.892	.859	.846
Ave WL	MST	1.64 cm	2.43 cm	16.4 cm	24.3 cm
	AHHK	1.16	1.09	1.04	1.07
	ERT	1.19	1.27	1.61	2.15
	1Stein	1.48 cm	2.18 cm	14.8 cm	21.8 cm
	SERT	1.13	1.22	1.66	2.27
	SERT-C	1.16	1.14	1.28	1.22

Table 8: Two-Pole simulation results for Elmore routing tree variants. Spanning ERT constructions are compared with MST and AHHK; Steiner SERT and SERT-C constructions are compared with 1-Steiner. All choices of critical sink are random, and all results are averaged over 100 random nets. MST and 1-Steiner results are reported in the physical units (nanoseconds or centimeters) while other results are reported as ratios to the corresponding MST or 1-Steiner results.

parts of Table 8. For comparison, the table includes data for the minimum spanning tree and AHHK tree [1] constructions.

Our results show that even as generic net-dependent routers, the ERT methods we propose are highly effective, beyond their relative efficiency and ease of implementation. For nets with 9 sinks, the spanning tree ERT construction reduces critical sink delay versus the MST construction by 16%, 26%, and 30% in the respective IC technologies and by 67% in the MCM technology. ERT also improves upon AHHK for most of the technologies, with reductions of 0% (IC1), 4% (IC2), 6% (IC3), and 46% (MCM). These results are particularly impressive because our AHHK data follows the experimental methodology in [1], which generates output trees for 21 different values of the c parameter and then chooses the best tree found for each signal net instance. (Moreover, according to [1], AHHK already achieves strong improvements over such other recent methods as shallow-light routing [9] or Steiner arborescences [10] when measured by the same Two-Pole simulation methodology.) However, it must be noted that delay reductions in practice will probably not attain exactly these magnitudes (cf. the footnote above, in the discussion of CS-Steiner results), since our modeling methodology cannot capture all of the effects related to the geometric embedding of our topologies.

The Steiner ERT variant also performs well as a generic high-performance router. For 9-pin nets, SERT improves critical sink delay versus the 1-Steiner routing by 19% and 62% for the IC2 and MCM technologies, respectively. The percentage reductions in maximum delay are somewhat greater for the IC technologies, but somewhat smaller for MCM interconnects. It should be noted that for the MCM technology, the ERT and SERT constructions tend to be star-like, producing tree costs significantly higher than those of the 1-Steiner construction. In practice, when delay is not an overriding concern, the user may recapture wirelength by simulating a larger output driver resistance.

Finally, even more significant reductions in delay can be achieved when a critical sink has been identified per the original CSRT formulation. The SERT-C algorithm improves over the SERT results by an *additional* reduction in delay at the critical sink of 5%, 7% and 6% for the three IC technologies, and 8% for MCM. Identification of a critical sink has clear advantages in terms of tree cost, particularly for MCM routing: the SERT-C trees have much less cost than the SERT outputs, while still improving the delay to the critical sink. Since maximum sink delays still decrease, it is likely that overall skew in the routing tree will be reduced even when we treat the critical-sink formulation. Finally, we note that the SERT-C router produces very similar delays and costs compared to the HBest and H0 variants of CS-Steiner discussed in the previous subsection. However, SERT-C is more practical than HBest or H0 since it runs in $O(k^2 \log k)$ time (versus the $O(k^3)$ complexity of the best practical implementation of the 1-Steiner heuristic that is called by HBest and H0), and it does not require any simulator calls as does HBest.

Figures 4 and 5 illustrate the SERT and SERT-C algorithms for a 9-pin signal net using the IC2 technology parameters. Figure 4 shows the progressive growth of the SERT construction. Figure 5 contains the trees produced by SERT-C for the various choices of critical node. The tree constructed when n_c is node 3 or node 7 is also the 1-Steiner tree, and the tree constructed when n_c is node 8 is the same as the generic SERT result.

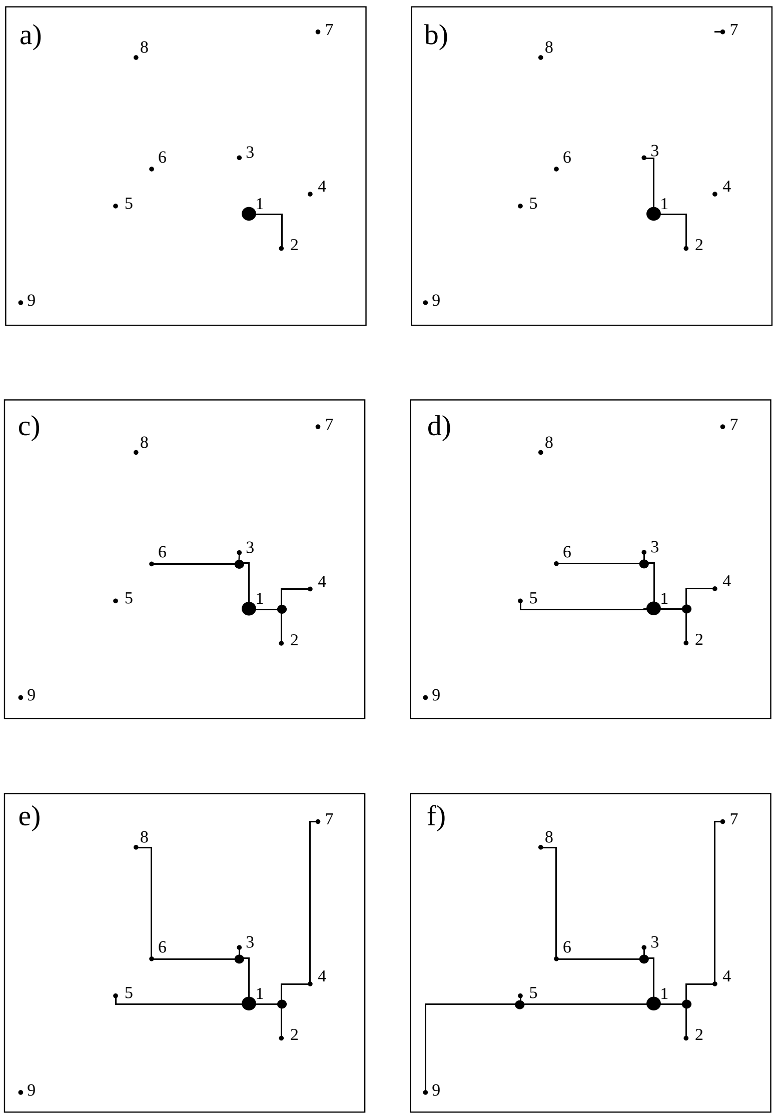
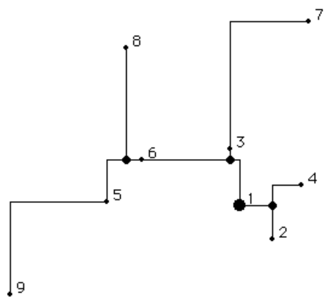


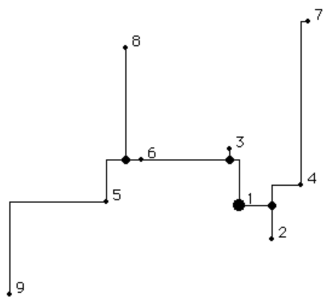
Figure 4: Example of the progressive SERT Steiner tree construction for a 9-terminal net using IC2 parameters. The source pin is labeled 1, and sinks are numbered in order of distance from the source.

4.3 Elmore-Optimality of Spanning Tree Constructions

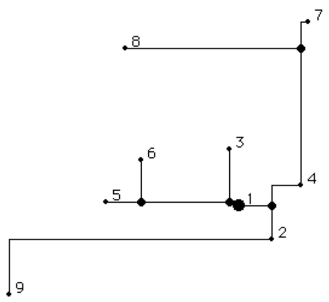
We have seen that the ERT constructions yield greatly improved signal delay when compared to previous methods. An obvious question is whether we still need to seek methods that better minimize Elmore delay. Thus, we have implemented a branch-and-bound algorithm which finds *optimal* generic routing trees according to Elmore delay. Starting with a trivial tree containing only the source pin, we incrementally add one edge at a time to the growing tree and evaluate the maximum sink delay. If this value exceeds the maximum sink delay in any *complete* candidate tree seen so far, we prune the search and backtrack to select a different edge at the previous step. A recursive implementation of this *Branch-and-Bound Optimal Routing Tree* (BBORT) search is shown in Figure 6. BBORT attempts to add sinks in all possible orders,



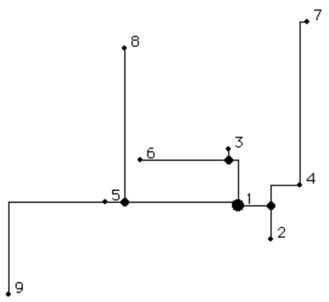
a) Node 2 (or 4) critical



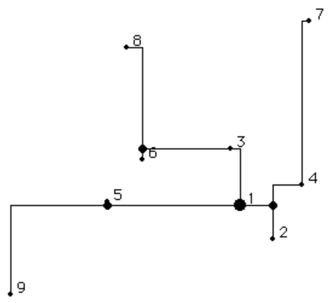
b) Node 3 (or 7) critical
(also 1-Steiner tree)



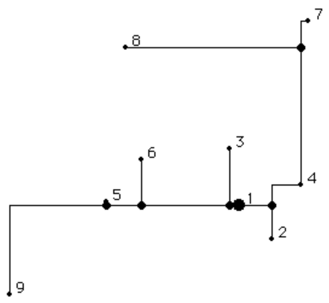
c) Node 5 critical



d) Node 6 critical



e) Node 8 critical
(also Steiner ERT)



f) Node 9 critical

Figure 5: SERT-C tree constructions for a single 9-pin net, showing variation of solution with choice of n_c.

but avoids testing any topology more than once by requiring that sinks be added in the order of a breadth-first traversal of the tree (if two sinks are connected to the same parent node, then the sink with smaller index must be added to the tree first). It is easy to verify that according to this convention, any tree topology will imply a unique ordering of the sinks.

To track all of the above simulation results, we have run BBORT trials on random sets of 200 nets for each of several net sizes. Our inputs are evaluated using the same four sets of technology parameters discussed previously. Table 9 compares Elmore delays of the BBORT and ERT constructions, as well as of the minimum spanning tree (MST) and shortest path tree (SPT) constructions, for each of the four

```
BBORT Algorithm
Input: signal net $N$ with source $n_0 \in N$
Output: optimal-delay tree $T_{opt}$ over $N$
1.  $T = (V, E) = (\{n_0\}, \emptyset)$
2.  $t_{\min} = \infty$
3.  Call Add_Edges($T$)
4.  Output $T_{opt}$
Procedure Add_Edges(Tree: $T = (V, E)$)
5.  While there exist $v \in V$ and $u \notin V$ such that
            $T' = (V \cup \{u\}, E \cup \{(u, v)\})$ is a new tree topology Do
6.      Compute tree delay $t(T')$
7.      If $t(T') \leq t_{\min}$ Then
8.          If $|T'| = |N|$ Then $T_{opt} = T'$ ; $t_{\min} = t(T')$
9.          Else Call Add_Edges($T'$)
```

Figure 6: Branch-and-Bound Optimal Routing Tree (BBORT) algorithm (recursive implementation).

technologies.[9] Delay for each tree is normalized to the ORT delay of the same net. Tree costs are similarly normalized to the MST cost of each net.

	IC1				IC2			
	$\|N\| = 5$		$\|N\| = 7$		$\|N\| = 5$		$\|N\| = 7$	
	delay	cost	delay	cost	delay	cost	delay	cost
ORT	1.0	1.103	1.0	1.133	1.0	1.140	1.0	1.175
ERT	1.007	1.104	1.017	1.142	1.010	1.159	1.022	1.215
(Std Err)	(.0015)		(.0021)		(.0017)		(.0022)	
SPT	1.085	1.290	1.130	1.395	1.058	1.290	1.096	1.395
MST	1.169	1.0	1.282	1.0	1.272	1.0	1.451	1.0

	IC3				MCM			
	$\|N\| = 5$		$\|N\| = 7$		$\|N\| = 5$		$\|N\| = 7$	
	delay	cost	delay	cost	delay	cost	delay	cost
ORT	1.0	1.146	1.0	1.190	1.0	1.432	1.0	1.547
ERT	1.011	1.172	1.027	1.252	1.009	1.585	1.024	1.892
(Std Err)	(.0018)		(.0025)		(.001)		(.0008)	
SPT	1.054	1.290	1.091	1.395	1.089	1.290	1.161	1.395
MST	1.311	1.0	1.499	1.0	1.894	1.0	2.457	1.0

Table 9: Elmore delays and wirelengths of various constructions using IC1, IC2, IC3 and MCM parameters. Simulations were run on 200 random nets for each net size. Tree cost is normalized to MST cost and delays are normalized to ORT delay. Standard errors for ERT delay are shown in parentheses.

In the table, we see that ERTs over seven pins in the IC1 technology have an average maximum Elmore delay only 1.7% greater than optimal, while MSTs have average Elmore delay 28.2% greater than optimal. For smaller nets, ERTs are even better: for nets with five pins, ERT delays are only 0.7% above optimal on average, while MSTs are 16.9% above optimal. Our confidence in the average difference computed between

[9]The SPT construction is the tree which minimizes cost subject to each source/sink path having minimum length, i.e., it is a Steiner arborescence, or A-tree [10, 29].

ERTs and ORTs is very high: for instance, the 1.7% difference obtained for 7 pins has a standard error[10] of 0.21%, indicating a 95% confidence interval between 1.3% and 2.1% (i.e., an interval of within two times the standard error of the average).

Technology IC3 gives our worst results in terms of the optimality of ERTs. For the IC3 parameters and 7-pin nets, ERT gives an average value within 2.7% of ORT with a 95% confidence interval of between 2.2% and 3.2%. For MCM parameters, the Elmore-based ERT constructions are also very close to optimal: on average, they are within 2.4% of ORT delay for 7-pin nets. Our tables also compare the delays of the SPT algorithm with ERT and MST; SPT outperforms MST but not ERT in terms of Elmore delay.

4.4 Elmore-Optimality of Steiner Tree Constructions

We have shown that our spanning tree constructions are nearly optimal when we optimize Elmore delay to a critical sink and when we optimize the maximum Elmore delay over all sinks in the net. Because Steiner constructions give lower delay values than spanning trees in general, we close this section with a similar comparison for our SERT-C and SERT Steiner constructions. At first, this comparison appears very complicated because there are infinitely many possible locations for Steiner nodes. Indeed, while it is well-known that the result of Hanan [16] restricts the choice of Steiner nodes to at most $k \cdot (k+1)$ points, no such characterization has been established for a Steiner tree with optimal *Elmore* delay. In Appendix B, we present new theoretical results which restrict the choice of Steiner nodes in Elmore-optimal trees to exactly the same finite "Hanan grid" that contains the Steiner nodes of minimum-cost trees. This allows a finite algorithm which determines *optimal* trees with respect to any given linear combination of Elmore delays to critical sinks. We also present an entirely new "peeling decomposition" of any optimal Elmore delay Steiner tree into a sequence of subtrees, each of which adds a sink by a "closest connection" to some edge in the previous tree.

When the driver resistance r_d is very large, the optimal Elmore delay tree is a minimum-cost Steiner tree (recall Equation (1); also see [4]). As a consequence, our results extend very naturally to the well-studied problem of minimum-cost Steiner tree construction, and the restriction of Elmore-optimal Steiner nodes to the Hanan grid both generalizes and extends Hanan's original results. (As Hanan did for minimum-cost Steiner trees, we prove that every Steiner node in an Elmore-optimal tree is connected to one sink by a horizontal segment of edges, and to another sink by a vertical segment of edges. However, our techniques (Lemmas B1 - B4 in Appendix B) are much more powerful in order to address optimality of the Steiner tree with respect to Elmore delay.) We also note that our peeling decomposition, and its extension to minimum-cost Steiner trees, is of independent interest since it provides both a new characterization of, and

[10] As used here, the term *standard error* is defined as follows. For a random variable X, let $\hat{X} = \sum_{i=1}^{n} X_i$ be an estimator for the expected value of X. The standard error of $\hat{X}$ is an estimate of its standard deviation over multiple sample sets, and is equal to the standard deviation of X divided by $\sqrt{n}$. Because delays are recorded as ratios to the ORT delay, the standard error of the average difference between ERT and ORT delays is equivalent to the standard error of average ERT delay.

a new means of generating, such trees.

Based on the results of Appendix B, we achieve a simple modification to our BBORT method which finds an optimal Steiner routing tree for any linear combination of Elmore delays to critical sinks. Rather than considering connections from each sink n_j outside the current tree to each sink n_i inside the tree as in BBORT, the *Branch-and-Bound* method for *Steiner Optimal Routing Trees with Critical Sinks* (BB-SORT-C) considers connections from n_j to each edge created when n_i was added to the tree. In other words, each node n_i already contained in T is replaced as a possible connection point by each of the edges created when n_i was added to the tree earlier. Again we use branch-and-bound pruning to reduce the complexity of the search.[11]

Table 10 compares Elmore delay for trees constructed by the SERT-C algorithm with optimal Elmore delay trees found by BB-SORT-C for each of our four technologies. The size of nets used in the comparison is limited to nets with six sinks (i.e., seven pins) because of the exponential time complexity of BB-SORT-C. For nets with seven pins, our results show that SERT-C achieves Elmore delay that is on average within 11.1% of optimal for the IC1 technology. The results for IC2, IC3, and MCM are quite similar. The table also gives average tree costs for our constructions and the standard error of our estimate for the ratio between SERT-C and SORT-C delays. Our SERT-C algorithm does not perform as well as the ERT algorithm in terms of its nearness to optimality for the types of delay functions we have considered. Thus, future work may improve the near-optimality of critical sink constructions – however, Table 10 shows that any future Elmore delay improvement will be limited to between 8% and 12% for nets with up to seven pins.

	IC1				IC2			
	$\|N\| = 5$		$\|N\| = 7$		$\|N\| = 5$		$\|N\| = 7$	
	delay	cost	delay	cost	delay	cost	delay	cost
SORT-C	1.0	1.111	1.0	1.112	1.0	1.161	1.0	1.158
SERT-C	1.042	1.046	1.083	1.047	1.049	1.120	1.114	1.106
(Std Err)	(.004)		(.006)		(.006)		(.009)	
1-Steiner	1.117	1.0	1.200	1.0	1.228	1.0	1.362	1.0
	IC3				MCM			
	$\|N\| = 5$		$\|N\| = 7$		$\|N\| = 5$		$\|N\| = 7$	
	delay	cost	delay	cost	delay	cost	delay	cost
SORT-C	1.0	1.175	1.0	1.165	1.0	1.296	1.0	1.262
SERT-C	1.046	1.140	1.112	1.112	1.000	1.296	1.001	1.256
(Std Err)	(.006)		(.010)		(.000)		(.0001)	
1-Steiner	1.275	1.0	1.429	1.0	1.455	1.0	1.634	1.0

Table 10: Comparison of Elmore delays and wirelengths of various Steiner tree constructions using IC1, IC2, IC3 and MCM parameters. Simulations were run on 200 random nets for each net size. Delay is normalized to BB-SORT-C delay and tree cost is normalized to 1-Steiner cost. Standard errors for average SERT-C delays are shown in parentheses.

[11] Because we consider connections to up to three edges for each sink in the growing tree, our BB-SORT-C will introduce some redundancies in the tree topologies; we check for possible redundancies and prune the search at each redundant tree that we find.

5 Conclusions

We have addressed a *critical-sink routing tree* (CSRT) formulation which arises when critical-path information becomes available during the timing-driven layout process. Two new classes of CSRT constructions are proposed: (i) the CS-Steiner method, which perturbs a minimum Steiner tree to accommodate an identified critical sink, and (ii) the SERT-C method, which begins with a connection from the source to the critical sink and then grows a tree so as to minimize the increase in Elmore delay to the critical sink. Each of these algorithms is efficient, and offers very significant performance improvements over existing performance-driven routing tree constructions. We note that the greedy "Elmore routing tree" (ERT) approach underlying the SERT-C algorithm seems quite powerful. In particular, ERT generalizes to a "generic" SERT Steiner router which outperforms all previous performance-driven routing algorithms in the literature. The ERT approach is also the first to *consistently*, and directly, optimize the Elmore delay formula itself, rather than an objective which heuristically abstracts Elmore delay. Since Elmore routing trees are efficiently computed, our approaches may lead to basic new utilities that can be integrated within existing performance-driven global routing codes. Assessments of the near-optimality of our Steiner constructions have led to a new characterization of Elmore-optimal Steiner trees, and to a new decomposition theorem for minimum-cost and minimum-Elmore delay Steiner trees; both of these results are of independent interest.

Current work addresses extensions of the critical path-dependent routing tree design to the general case of multiple critical sinks with varying criticalities. If a subset of the sinks are designated as critical, the SERT-C algorithm can be extended by first routing the critical sinks under the min-max delay objective of SERT, then connecting non-critical sinks as in SERT-C to minimize the weighted sum of the delays at the critical sinks. There are also interesting extensions of the CS-Steiner and ERT algorithms which involve wiresizing, which treat general-cell layout with arbitrary routing region costs, and which exploit the inherent parallelizability of our approaches. Finally, we leave as an open problem the reduction in time complexity of the ERT constructions.

6 Acknowledgements

We are grateful to the authors of [37], particularly Professors Jason Cong and Dian Zhou, for use of their Two-Pole simulator code. Professor Ernest S. Kuh and Dr. Kamal Chaudhary provided many helpful comments regarding modeling and experimental methodology. Professors Jim Aylor and Michael Shur provided invaluable assistance with SPICE simulation methodology. Part of this work (ABK) was performed during a Spring 1993 sabbatical visit to UC Berkeley; support from NSF MIP-9117328 and the hospitality of Professor Ernest S. Kuh and his research group is gratefully acknowledged.

Appendix A: Global Slack Removal

Recall from Section 3.1 that *Global Slack Removal* (GSR) is a linear-time postprocessing enhancement to the CS-Steiner approach. GSR shifts edges in the 1-Steiner output to maximize the monotonicity of all source-sink paths without any increase in total tree cost or Elmore delay to any sink. In what follows, we use the term *1-Steiner tree* to refer to any tree that can be output by the 1-Steiner algorithm.

Definition: A V is a subpath of *three* consecutive nodes on a root-leaf path in a routing tree such that the combined edge cost along the subpath is greater than the distance between its two end points (e.g., path v_1-v_3 in Figure 7(a)).

Definition: A U is a subpath of *four* consecutive nodes on a root-leaf path with edge cost greater than the distance between its end points (e.g., path v_1-v_4 in Figure 8(a)).

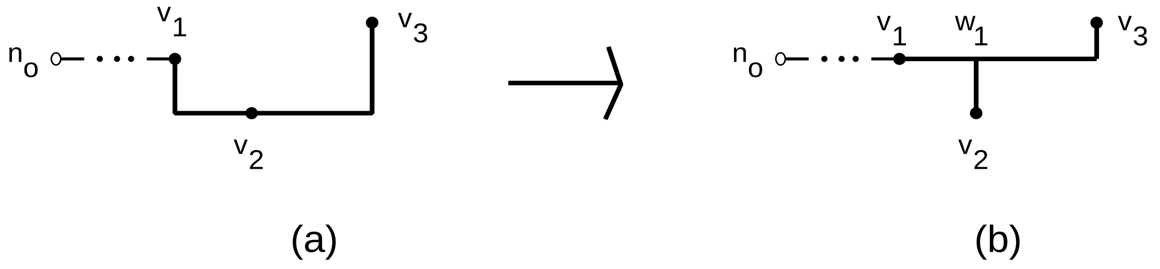


Figure 7: Removing a single "V" in the GSR Algorithm.

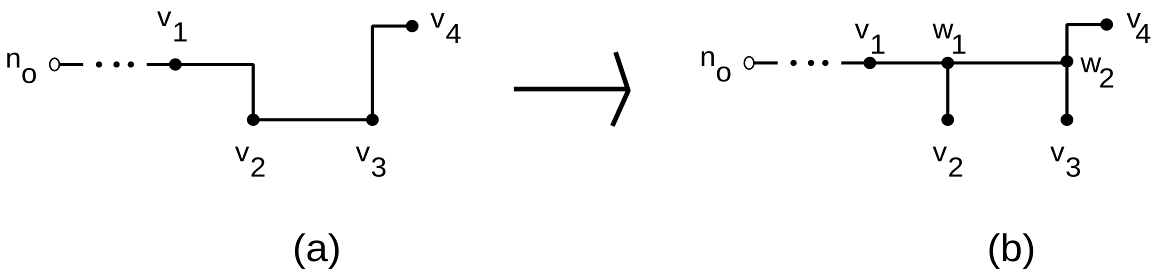


Figure 8: Removing a single "U" in the GSR Algorithm.

Note that the nodes in a V or a U can be either Steiner nodes or pins. A V can be removed from a routing tree by introducing a Steiner node which eliminates the overlap between the two adjacent edges, as in Figure 7(b). It is easy to see that, if a U (say $v_1v_2v_3v_4$) does not contain any V's, then its middle edge (v_2, v_3) must be either completely horizontal or vertical. Consequently, a U containing no V's can be removed by moving the middle edge and adding up to two new Steiner nodes as in Figure 8(b).

Figure 9 describes the GSR algorithm for removing V's and U's from any Steiner tree. We define a U (or V) to be *located at* node v if v is the node in U (or V) furthest topologically from the source.

Three clarifying points should be noted.

```
GSR Algorithm
Input: Steiner tree $T$ with source $n_0$
Output: Steiner tree $T$ with all $U$'s removed
1.    Remove all Steiner nodes of degree $\leq 2$ from $T$;
2.    $Q \leftarrow \{n_0\}$;
3.    While $Q \neq \emptyset$
4.        $v \leftarrow Dequeue(Q)$;
5.        For each node $v' \in children(v)$ do
6.            $Q \leftarrow Enqueue(v')$;
7.            If there is a $V$ located at $v'$
8.                Remove_V($v'$)
9.            If there is a $U$ located at $v'$
10.               Remove_U($v'$)
11.               Clean_Up($v'$)
12.    remove all Steiner nodes of degree $\leq 2$ from $T$;

Subroutine Clean_Up(node: $v'$)
C1.   If there is a $V$ located at $parent(v')$
C2.       Call Remove_V($parent(v')$)
C3.   If there is a $U$ located at $v'$
C4.       Call Remove_U($v'$)
C5.       Call Clean_Up($v'$)
C6.   Else
C7.       If there is a $U$ locaated at $parent(v')$
C8.           Call Remove_U($parent(v')$)
C9.           Call Clean_Up($parent(v')$)
```

Figure 9: Pseudo-code for the Global Slack Removal (GSR) algorithm. Local variables include a queue Q and nodes v and v'. We use $children(v)$ to denote the set nodes with children of v when the tree is rooted at n_0; $parent(v)$ denotes the parent of v in the rooted tree. The subroutine Remove_V(v) removes a V located at v as in Figure 7 and Remove_U(v) removes a U located at v as in Figure 8.

1. GSR utilizes a queue Q which can be implemented arbitrarily as long as each node in the tree is processed before its children. In practice, a simple depth-first ordering suffices.

2. The procedure Remove_U is invoked only for U's that do not contain any V's. Thus, it is executed as in Figure 8.

3. All *low-degree* Steiner nodes of degree ≤ 2 are clearly superfluous and are removed since more U's can be found if these low-degree Steiner nodes are removed at the outset. Because each removal of a U can introduce additional low-degree Steiner nodes, they are removed again at the end of the algorithm.

We now show that the tree returned by GSR dominates the input tree in terms of total tree cost, path length from the source to each sink, and Elmore delay at each sink. Let *cost*(T) denote the cost of routing tree T.

Theorem A1: Given any tree T as input, GSR will return a tree T' such that (i) *cost*(T') $\leq$ *cost*(T); (ii) for each $i > 0$, the n_0-n_i path length in T' is less than or equal to the n_0-n_i path length in T; and (iii) the Elmore delay $t_{ED}(n_i)$ at each n_i in T' is less than or equal to the Elmore delay $t_{ED}(n_i)$ in T.

Proof: (i) Removing a V reduces cost in the routing tree; removing a U as in Figure 8 leaves tree cost unchanged; and by the triangle inequality the removal of a low-degree Steiner point will either reduce cost or leave it unchanged. These are the only operations in GSR that change the structure of the tree.

(ii) Removing a V does not affect source-sink path lengths; removing a U reduces the source-sink path length to the fourth node in the U (v_4 in Figure 8) and all of its descendants, and leaves all other source-sink path lengths unchanged; removing low-degree Steiner nodes does not affect source-sink path lengths.

(iii) Assuming constant technology parameters[12], removing a U or a V can affect Elmore delay along a source-sink path in only three ways: a) changing the length of the path; b) changing tree capacitances along the path (i.e., increasing the wirelength of branches off from the path); and c) shifting tree capacitances along the path (changing where branches connect to the path). Removing a V will reduce some path lengths, reduce tree capacitances, and shift tree capacitances closer to the source, thereby reducing Elmore delay to all pins in the tree. Removing a U reduces path length to node v_4 in Figure 8 and shifts tree capacitance closer to the source for nodes v_2, v_3, and v_4. (For v_3, the capacitance that met the n_0-v_3 path at v_3 now meets the path at w_1 and w_2.) Removing 1-degree Steiner nodes reduces total wirelength, and thus reduces Elmore delay, while removal of 2-degree Steiner nodes leaves Elmore delay unchanged. Thus, GSR will not increase Elmore delay to any sink in the net. $\square$

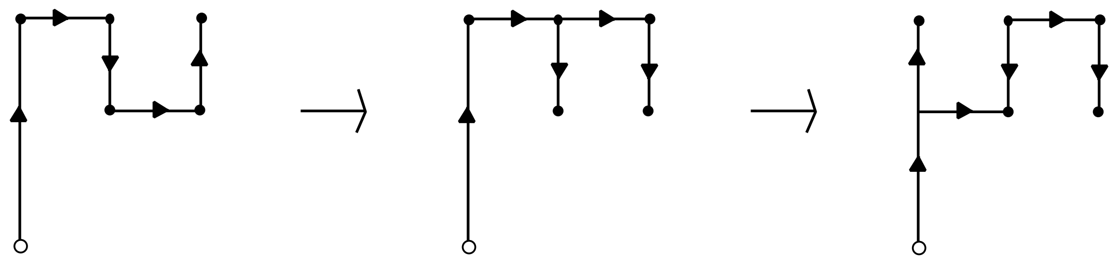


Figure 10: An example for which processing U's in a bottom-up order (b) returns a tree with one remaining U. Processing U's in a top-down order (a) is guaranteed to remove all U's.

We note that the order in which U's are removed from the tree is important. If the U's are processed in a bottom-up order instead of a top-down, then new U's can be introduced and the resulting tree may not have all of its U's removed, as can be seen in the example in Figure 10. Furthermore, two different top-down orderings can produce different output trees (although both will have no remaining U's). An example is shown in Figure 11.

We now prove that GSR will remove all V's and all U's from any input tree, and that its worst-case time complexity is quadratic. Note that we have constructed a class of nets for which the 1-Steiner heuristic will create an input tree that forces $\Omega(k^2)$ runtime [5] for GSR. However, GSR in practice seems to exhibit

[12] I.e., including unit wire resistance, unit wire capacitance, driver resistance, and sink loading capacitances.

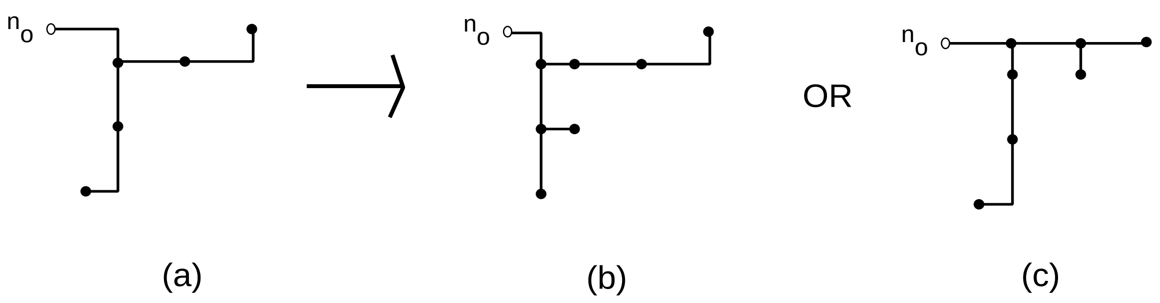


Figure 11: The GSR algorithm with input (a) can produce either tree (b) or tree (c), depending on the order in which the U's are processed.

close to linear-time complexity, since multiple calls to procedure **Clean_Up** occur for very few nodes.

Theorem A2: For any input tree, (i) GSR returns a tree containing no V's and no U's, and (ii) GSR runs in $O(n^2)$ time in the worst case.

Proof: (i) Since GSR checks for V's and U's at each node in the tree, the output tree will contain a V or U only if GSR creates one at a node that has already been traversed. A new V or U can be produced at a node v only if the $n_0 - v$ path length is increased (which is impossible by Theorem 1) or if nodes are removed from the $n_0 - v$ path.

Removing a V at Line 8 will not introduce a new V or U at v_2 (in Figure 7), because the $n_0 - v_2$ path length is unchanged and a new Steiner point w_1 is added to this path. Removing a V will not introduce a V at v_3 either, because $v_1 w_1 v_3$ is not a V. A U may remain at v_3 after removing the V, but this will be detected later at Line 9.

Removing a U at v_4 in Figure 8 can only introduce a new V or U at w_2, v_4, or one of their descendants, because all other nodes have unchanged source-sink path lengths and no fewer Steiner nodes on their source-sink paths. The subroutine **Clean_Up** checks for V's and U's at w_2 and v_4, and recursive calls to **Clean_Up** will eventually terminate because a new V or U can be introduced only by reducing the number of nodes on the n_0–v_4 path.

Figure 12 shows how **Clean_Up** can require several recursive calls before terminating. However, for any node v', a call to **Remove_U(v')** will introduce a new V or U at v' or $parent(v')$ only if it reduces the number of nodes on the n_0–v' path. Because any Steiner tree connecting $k+1$ points can contain at most $2k$ nodes in total, there are $O(k)$ nodes on the n_0–v' path. Hence, at most $O(k)$ calls can be made to **Clean_Up** for each node v' added to the queue in Line 6 of the template, and the total number of calls to **Clean_Up** is $O(k^2)$. □

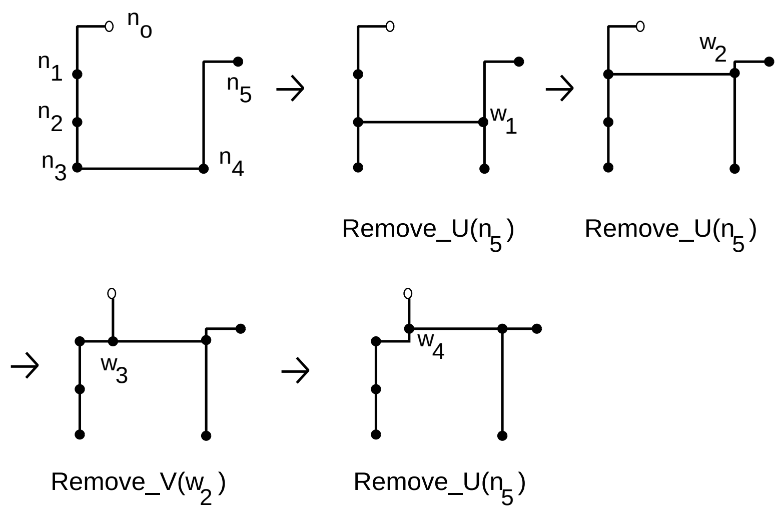


Figure 12: Example in which removing a U at n_5 requires two subsequent U-removals and a V-removal to complete the **Clean_Up** procedure.

Appendix B: Optimal Steiner ERTs

For minimum-cost Steiner trees, the classic result of Hanan [16] restricts the choice of Steiner nodes to at most $k \cdot (k+1)$ points (the "Hanan grid") and enables finite branch-and-bound methods to determine optimal solutions. Here, we prove an analogous result for trees which optimize any linear combination of sink Elmore delays. Like Hanan, we show that any tree containing a Steiner node which is not a vertex in the Hanan grid can have its edges and Steiner nodes shifted to lie on the Hanan grid. However, we do not shift edges in the same way as Hanan (the edge shifts he uses can be suboptimal in terms of Elmore delay). Indeed, the result of, e.g., Lemma B1 below is obvious when minimizing tree cost, but requires a fairly involved proof when minimizing Elmore delay. Our development of the Hanan grid result becomes complete with the proof of Lemma B4 below. In Lemma B5, we extend our result to show that the branch-and-bound SORT-C method described in Section 4.4 returns the optimal delay Steiner tree.

B.1. Definitions

We assume that all delays are defined in terms of Elmore delay. We seek to characterize the *optimal* Steiner tree over N, denoted by T^*, which minimizes the linear combination of sink delays $f = \sum_{i=1}^{k} \alpha_i \cdot t(n_i)$, with each $\alpha_i > 0$. (The case of some $\alpha_i = 0$ is effectively handled by setting these α_i to a small positive value.) We assume that T^* contains no Steiner nodes with degree < 3. For convenience, we normalize time and distance so that unit wire resistance and unit wire capacitance are both equal to one. We also consider a tree to be defined as a set of nodes and edges, so that the notations $v \in T$ for node v and $e \in T$ for edge e are well defined. An edge that is completely vertical or horizontal is called a *straight edge*; any other edge is called an *L-shaped edge*.

Assume that a Steiner tree T over N is rooted at n_0. We define $T \backslash v$ to be the tree induced by removing node v and all of its descendants from T, and then removing all degree-2 Steiner nodes from the resulting tree. The *closest connection* between three nodes is the location of the single Steiner node in a minimum-cost Steiner tree over the three nodes. This location is unique and has coordinates given by the medians of the x- and the y- coordinates of the three nodes (if the minimum-cost Steiner tree is a chain, then the closest connection is the middle node). The *closest connection* between a node v and edge e is the closest connection between v and the two endpoints of e. We say that node $v \in T$ is *incident to* an edge $e \in T \backslash v$ if its parent node in T is located on edge e. If $parent(v)$ is located at the closest connection between v and an edge $e \in T \backslash v$ incident to v, then v is said to make a *closest connection* to e in T. Lemma B1 will show that the each node a in the optimal delay tree T^* makes a closest connection to any incident edge in $T^* \backslash a$.

B.2. Proof of Closest Connections in T^*

Lemma B1: Let x be the parent of node $a \in T^*$, $a \neq n_0$. Then either $x = n_0$, or else x is located at the closest connection between a and each edge in $T^* \backslash a$ that is incident to a in T^*.

Proof: Consider a sink $n_i \in T^* \backslash a$ such that x is located on the n_0—n_i path in T^*. Assume that each L-shaped edge is oriented so that its interior points are as close to a as possible. We partition the set of points along the n_0—n_i path into three sets, I, D, and $\{n_0\}$. If starting from a point p and moving along the n_0—n_i path toward the source by a very small amount increases (decreases) the distance to a, then $p \in I$ ($p \in D$). If $x \in D$, then moving x closer to the source by a small amount along the n_0—n_i path will reduce both the overall tree cost and the path lengths to all sinks in T_a, i.e., all sink Elmore delays will improve. Since T^* is optimal, we must have $x \notin D$.

Suppose then that $x \in I$. Let p and c be the endpoints of the maximal contiguous subset of I which contains x, with p topologically closer than c to n_0 (see Figure 13; note that $p \notin I$, but that there are points in I arbitrarily close to p). We will show that x must be located at c. We assume that there is exactly one node q between p and c in $T^* \backslash a$, and that q has degree 3 as shown in Figure 13. Our argument is easily extended when more nodes are present between p and c, and so we restrict our attention to the case shown in Figure 13.

We will show that the delay function f is concave in terms of x for $0 \leq x \leq c$. For convenience, we overload x, a, b, c, and q to also represent the respective path lengths from p to these nodes or locations. Even though c is not necessarily a node in T^*, we will use T_c to represent the subtree of T^* below location c. Finally, recall that C_a, C_b, and C_c represent the tree capacitance in subtrees T_a, T_b, and T_c, respectively. We will show that the delay function f is concave in terms of x for $0 \leq x \leq c$. Our proof will invoke four facts recalled from elementary real analysis: (1) any concave function defined over an interval on the real line will be minimized at one of the boundary points of the interval; (2) multiplying any concave function by a positive constant produces another concave function; (3) the sum of any concave functions is also

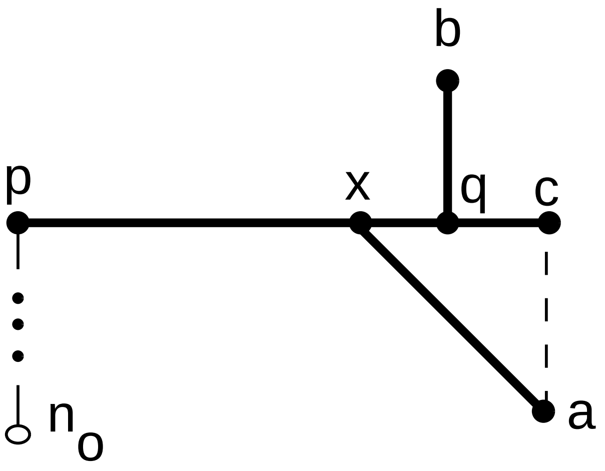


Figure 13: Proof of Lemma B1: we show that if x, the parent of a in T^*, is located on the maximal contiguous subset of I between locations p and c, then x must be located at c.

concave; and (4) any quadratic function of x having a negative coefficient for x^2 is concave in terms of x.

Consider the contribution made by the edge (x, a) to Elmore delay at various sinks n_j. First, consider the case of $n_j \in T_a$. Delay $t(n_j)$ is the sum of four functions: f_1 = delay from n_0 to p; f_2 = delay from p to x due to capacitance in $T^* \backslash b$; f_3 = delay from p to x due to capacitance in edge (b, q) and T_b; and f_4 = delay from x to n_j. Simple application of the Elmore formula for these four functions gives

$$\begin{aligned} f_1 &= K_0 + K_1(a - x) && (2) \\ f_2 &= x * (\frac{x}{2} + a - x + C_a + c - x + C_c) && (3) \\ f_3 &= x * (b - q + C_b) \qquad \text{if } x \le q && (4) \\ f_3 &= q * (b - q + C_b) \qquad \text{if } x \ge q && (5) \\ f_4 &= (a - x) * (\frac{a - x}{2} + C_a) + K_4 && (6) \end{aligned}$$

where K_0, K_1, and K_4 are constants. To be precise, K_0 is the sum of resistance/capacitance products along the n_0—p path; K_1 is the sum of resistances from n_0 to p; and K_4 is the delay from a to n_j. We see that function f_1 is linear in x, while f_2 and f_4 are quadratic in x with negative coefficients for x^2. Function f_3 is continuous, increasing for $x \le q$ and remaining constant for $x \ge q$. Consequently, f_1, f_2, f_3, and f_4 are all concave in x; this implies that $t(n_j)$ is concave in x.

If $n_j \in T_c$ then f_1, f_2, and f_3 are identical to the case of $n_j \in T_a$. Function f_4 equals $(c-x)*(\frac{c-x}{2}+C_c)$, which is concave in x, and so $t(n_j)$ is again concave in x.

If $n_j \in T_b$ or $n_j = q$, then we can express Elmore delay to n_j in terms of three functions f_1, f_2 and f_3. The definitions of f_1 and f_2 are the same as for $n_j \in T_a$, and f_3 gives the delay from p to n_j due to capacitance in $T^* \backslash a$. The equation for f_1 is identical to that for $n_j \in T_a$, while f_3 is a constant in terms of

x. Hence, f_1 and f_3 are concave. For f_2, we have

$$\begin{aligned} f_2 &= x * (a - x + C_a) \quad \text{if } x \le q & (7) \\ &= q * (a - x + C_a) \quad \text{if } x \ge q & (8) \end{aligned}$$

We recall that any continuous, piece-wise differential function of a real variable is concave as long as its first derivative is monotone decreasing. It is clear that this property holds for f_2, except possibly at $x = q$. Let f_2' be the derivative of f_2. Then for $x < q$, $f_2'(x) = a - 2x + C_a$, and for $x > q$, $f_2'(x) = -q$. Substituting q for x in these equations, we see that f_2' is indeed decreasing at $x = q$. Consequently, f_2 and $t(n_j)$ are concave in x.

Delay to any other sink in T^* is linear (and thus also concave) in x. Because f is a non-negative linear combination of concave functions, it is also concave. Therefore, over the interval $0 < x \le c$ we know that f can only be minimized when $x = c$. It is easy to see that if a is incident to edge $e \in T^*\backslash a$ on the n_0—n_i path, then a's parent x must be located at the closest connection between a and e. $\square$

Straightforward corollaries of Lemma B1 include: (i) that any non-source node in the optimal delay tree T^* must have degree ≤ 4, and (ii) that the possible configurations of edges incident to a Steiner node $q \in T^*$ are restricted to the five configurations shown in Figure 21. Note that Lemma B1 by itself is not sufficient to prove that BB-SORT-C will return the optimal delay tree. For example, if T^* connects a four-pin net into an "H" with two degree-3 Steiner nodes q_1 and q_2 (see Figure 14), then the parent of each non-source node v is connected by a closest connection to $T^*\backslash v$. However, T^* cannot be constructed by BB-SORT-C since the "H" cannot be formed by adding the three sinks sequentially by closest connections to the growing tree.

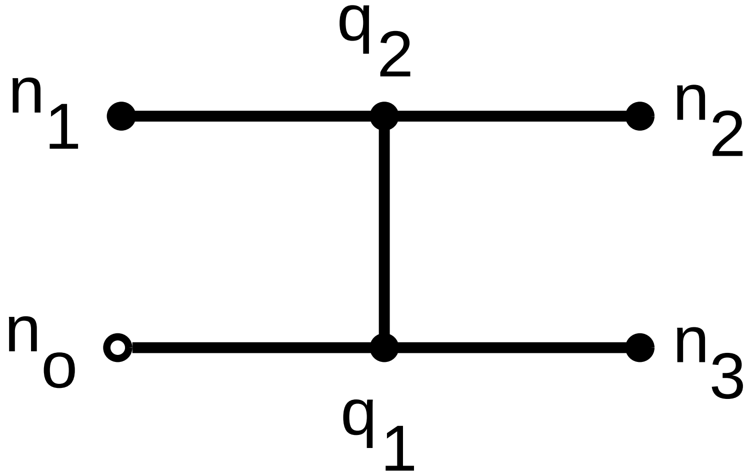


Figure 14: Example of a routing tree T which cannot be constructed by algorithm BB-SORT-C, but which satisfies the condition that each non-source node $v \in T$ makes a closest connection to each incident edge in $T\backslash v$.

B.3. Hanan Grid Proof for Steiner Nodes in T^*

We define a *segment* to be a contiguous set of straight edges in T which are either all horizontal or all vertical; a *maximal segment* (MS) is a segment which is not properly contained in any other segment. The node on an MS which is topologically closest to n_0 is called its *entry point*. A segment which contains all of an MS to one side of a node v on the MS is a *half segment* with respect to v, and a half segment with respect to the entry point of an MS is called a *branch*. A branch b is called a *branch off of* MS M if M contains b's entry point and b is perpendicular to M. Note that a given MS, M, will divide the plane into two half-planes. Suppose M does not contain n_0; then the half-plane containing the edge between M's entry point and its parent is called the *near side* of M (because it is "nearer" to the source), and the other half-plane is called the *far side* of M. Branches off of M that are located on its near (resp. far) side are called *near* (resp. *far*) *branches*. In addition, a *sink* located on M is defined to be a *far branch* off of M if none of its children are located on the far side of M (i.e., it is not the entry point to a larger far branch). For any segment M, we use $Near(M)$ (resp. $Far(M)$) to denote the set of near (resp. far) branches off of M. Figure 15 gives an example of an MS M with endpoints p_1 and p_2, entry point p_0, and four branches, including near branch b_1, far branch b_2, and a far branch consisting only of sink n_3.

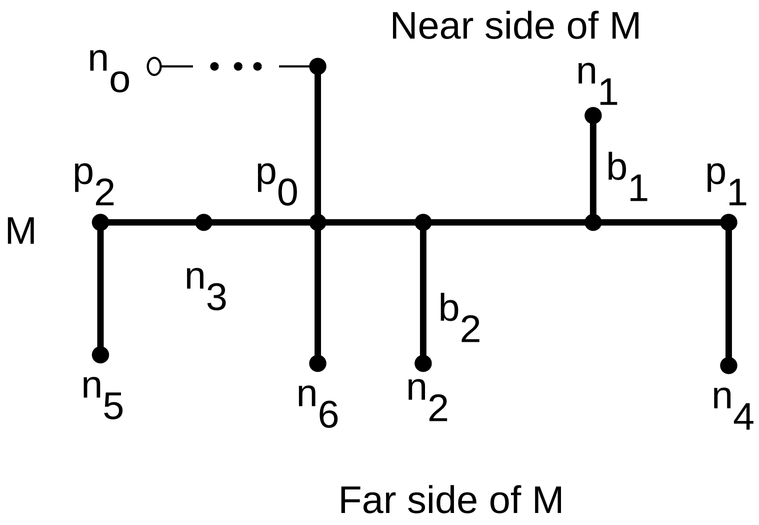


Figure 15: Example of a maximal segment M with entry point p_0, one near branch b_1, and four far branches, including b_2. Note that by definition, n_3 forms a far branch with no edges. Also, edge (p_0, n_6) does *not* form a far branch off of M because p_0 is not an entry point to the MS containing (p_0, n_6).

In the next two lemmas, we establish some properties that must hold for any maximal segment in T^*. Lemma B4 then uses these properties to show that each maximal segment in T^* will have a sink located on it. We thus generalize the classic result of Hanan [16] to trees that are optimal with respect to objective functions of form f (i.e., non-negative linear combinations of Elmore delays at sinks).

Lemma B2: In the optimal tree T^*, let q_0 be the entry point to maximal segment M, and let S be any segment contained in M and having q_0 as an endpoint. Then $|Far(S)| \geq |Near(S)|$.

Proof: By contradiction. Let S be the smallest segment in M with q_0 as an endpoint so that $Near(S) > Far(S)$. Then a portion of T^* between n_0 and q' looks like Figure 16(a). Label the branches $b_1, \ldots, b_j$ in order from entry point q_0. Figure 16(b) shows how we can shift segment S topologically toward the source; this effectively shifts wire from each near branch to a far branch with is topologically closer to the source (i.e., with a smaller label). Shifting S does not affect tree cost[13], and source-sink path lengths will be unchanged to all sinks except those connected to the tree through branches in $Near(S)$, which will have reduced source-sink path lengths. Consequently, the shift will decrease delay to all sinks in subtree $T^*_{q_0}$ and leave delay to all other sinks unchanged, contradicting the optimality of T^*. □

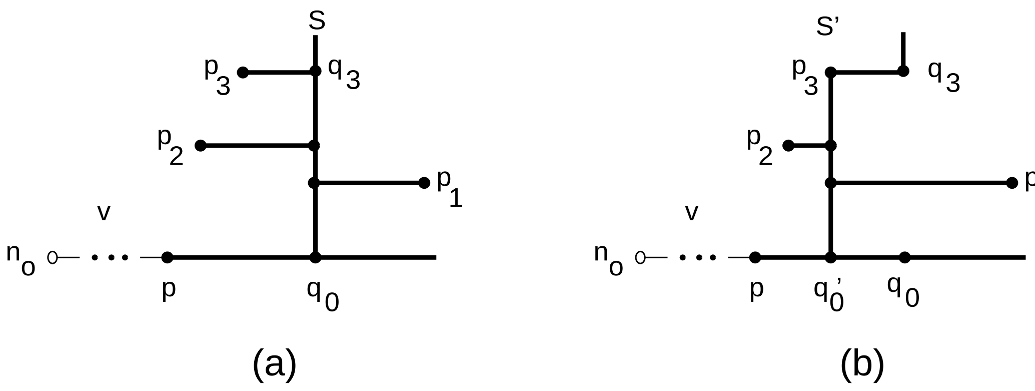


Figure 16: Example (a) with $|Near(S)| > |Far(S)|$ for a segment S between q_0 and q_3; (b) shows how S can be shifted to S' to reduce delay to all sinks in $T^*_{q_0}$ and leave delay unchanged at all other sinks.

Lemma B3: In the optimal tree T^*, let M be an MS not containing n_0. Then $|Far(M)| > |Near(M)|$.

Proof: By Lemma B2, $|Far(M)| \geq |Near(M)|$. Suppose that the exact equality $|Far(M)| = |Near(M)|$ holds. Lemma B2 then implies that each endpoint of M has a near branch incident to it as in Figure 17(a). In Figure 17(b) we show how M can be shifted toward the source to reduce delay to all sinks below q_0 in T^* and leave delay to all other sinks unchanged, thereby contradicting the optimality of T^*. □

Lemma B4: In the optimal tree T^*, any maximal segment must contain either the source or a sink.

Proof: (See Figure 18.) Let M be a lowest maximal segment in T^* which does not contain either the source or a sink, i.e., every MS that is topologically below M contains a sink. Let q_0 be the entry point to M and let p_0 be the parent node of q_0 in T^*. Consider the possibility of shifting M either toward the source or away from the source without passing over any node in $T^*_{q_0}$ which is not in M. Without loss of generality, assume that M is a vertical segment with x-coordinate x_0, with the near side of M having $x < x_0$. Let $x_1 < x_0$ be the be the closest value to x_0 on the near side of M such that shifting M to $x = x_1$ would cause M to intersect a node that is in $T^*_{q_0}$ but not in M. Similarly, let $x_2 > x_0$ be the closest value

[13] Unless q_0 is an endpoint for the MS containing edge (p, q_0), in which case tree cost will decrease.

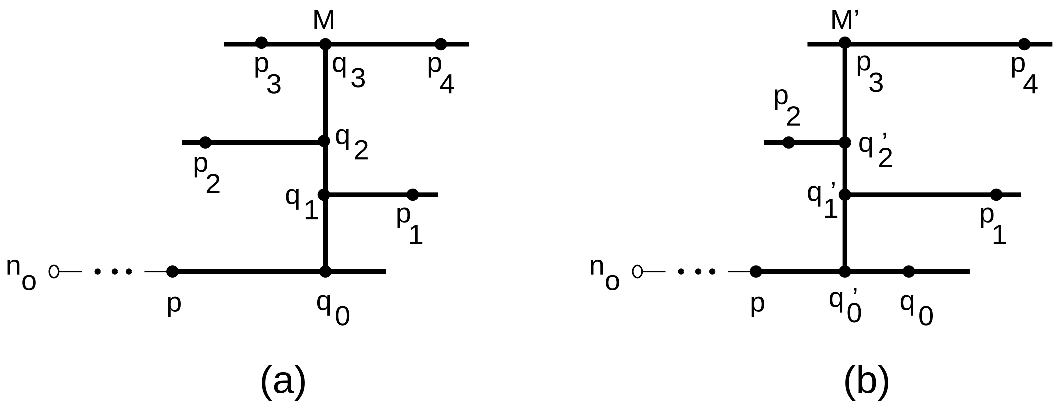


Figure 17: (a) Example where $|Near(M)| = |Far(M)|$ for maximal segment M. M can be shifted to M', as shown in (b), to reduce delay at all sinks in the subtree rooted at q_0'.

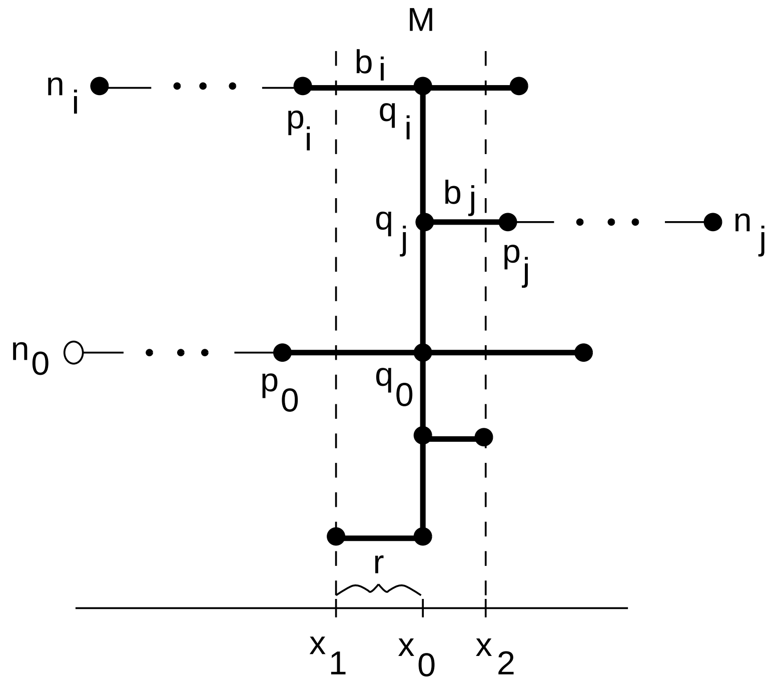


Figure 18: Proof of Lemma B4: because the objective function f is concave in r over the interval $0 \leq r \leq x_2 - x_1$, f will be minimized when the maximal segment M passes through either the gridline at x_1 or the gridline at x_2.

to x_0 on the far side of M such that shifting M to $x = x_2$ would cause M to intersect some node that is in $T^*_{q_0}$ but not in M. Let the variable r, $0 \leq r \leq x_2 - x_1$, denote the position of M between the x-coordinates x_1 and x_2. We will show that minimizing the delay function f implies that either $r = 0$ or $r = x_2 - x_1$.

Let $d = Far(M) - Near(M)$. Consider the delay to some sink n_i located along a near branch b_i off of M which has entry point q_i. (In general, we let q_j denote the entry point to branch b_j.) Delay $t(n_i)$ is quadratic in r only along the edge (p_0, q_0) and along the edge (q_i, p_i), where p_i is the child of q_i on b_i. To be precise, the delay due to (p_0, q_0) is equal to $r * (r/2 - d * r + K)$, where K is some constant; the delay due to (q_i, p_i) is equal to $r * (r/2 + K') + K''$, where K' and K'' are again constants. Therefore, the equation for $t(n_i)$ is

$$t(n_i) = (1 - d) * r^2 + K_1 * r + K_0 \tag{9}$$

where K_1 and K_0 are constants. From Lemma B3, we know that $d \geq 1$, implying that $t(n_i)$ is a concave function of r. Similarly, delay to a sink n_j along a far branch b_j off of M will be equal to

$$-d * r^2 + K_1' r + K_0' \tag{10}$$

where again K_1' and K_0' are again constants; this too is a concave function of r. Finally, delay to any sink whose source-sink path does not contain an edge in M will be linear in r, and thus also a concave function. Since any linear combination of functions that are each concave on a given interval will also be concave on that interval, f is concave in r and is minimized at one of its extreme values, i.e., at $r = 0$ or $r = x_2 - x_1$.

Thus, M may be moved so that it contains a new node, say p_i. If p_i is a sink, the lemma is proved. If p_i is a Steiner node, then because it has degree > 2, there must be a vertical MS incident to p_i, and this vertical MS must contain a sink since M is the lowest maximal segment not containing a sink. Hence, if p_i is a Steiner node, the shifted M will also contain a sink. $\square$

A direct corollary of Lemma B4 is that all Steiner nodes in the Elmore-optimal Steiner tree are contained in the Hanan grid:

Corollary: Let X be the set of x-coordinates for all pins in N, and let Y be the set of y-coordinates in N. Then if (x, y) is the location of a Steiner node in T^*, $x \in X$ and $y \in Y$. $\square$

Thus, only a finite number of possible Steiner point locations need to be considered. Hanan's original theorem may be viewed as a special case of this Corollary with the driver on-resistance $r_d \rightarrow \infty$.

B.4. Decomposition Theorem for T^*

To prove that BB-SORT-C will return the optimal-delay tree T^*, we show that T^* can be constructed by starting with a tree T_0 containing only n_0, then successively adding a sequence of sinks n_i, $1 \leq i \leq k$, each of which yields a tree T_i by making a closest connection to some edge in the current tree T_{i-1}. We show that such a sequence of trees exists by starting with $T^* = T_k$ and $i = k$, then "peeling off" an n_i at each iteration such that n_i was joined by a closest connection in T_i to some edge in $T_{i-1} = T_i \backslash n_i$.

At each step, we find an interior node $q \in T_i$ whose children are all leaves. Each of these leaves must be a sink, because all low-degree Steiner nodes (i.e., with degree < 3) are removed from $T_{i+1} \backslash n_{i+1}$. We choose one of q's leaves to be the n_i that is peeled off, and set $T_{i-1} = T_i \backslash n_i$. The choice of which leaf should be peeled is guided by the function $Pin(q)$, which specifies one of q's children that should *not* be peeled off from q. Thus, when q is removed as a low-degree Steiner node, the edge between q and its parent is replace with an edge between $Pin(q)$ and q's parent. More formally, $Pin(v)$ is defined for each node $v \in T^*$ as follows: (i) if v is a sink, then $Pin(v) = v$; and (ii) if v is a Steiner node, then $Pin(v)$ is chosen according to the template given in Figure 19.

```
$Pin(q)$ **Assignment Procedure**
**Input:** Optimal delay tree $T^*$
        Steiner node $q \in T^*$ such that $Pin(w)$ has been assigned
        for each $w \in T_q^*$, $w \neq q$
**Output:** $Pin(q)$
1.   $p = parent(q)$ in $T^*$
2.   **If** edge $(p, q)$ is L-shaped
3.       Set $a$ arbitrarily to be one of $q$'s two children
           /* ($v$ has exactly two children, by Lemma B1) */
4.   **Else** /* $(p, q)$ is a straight edge */
5.       Let $M$ be the MS containing $(p, q)$
6.       **If** $T_q$ contains a sink on $M$
7.           Set $a$ to be the child of $q$ on $M$
8.       **Else if** $p$ is the entry point to $M$
9.           Set $B$ to be the far branch of $M$ at $q$
               /* (such a $B$ exists by Lemma B2) */
10.          Set $a$ to be the child of $q$ on $B$
11.      **Else if** there is a *near* branch of $M$ at $p$
12.              **If** there is a *far* branch $B$ of $M$ at $q$
13.                  Set $a$ to be the child of $q$ on $B$
14.              **Else** Set $a$ to be the child of $q$ on $M$
15.      **else** /* there is a *far* branch of $M$ at $p$ */
17.              **If** there is a *near* branch $B$ of $M$ at $q$
18.                  Set $a$ to be the child of $q$ on $B$
19.              **Else** Set $a$ to be the child of $q$ on $M$
20.  $Pin(q) = Pin(a)$
```

Figure 19: Criteria used to associate a sink $Pin(q)$ with each Steiner node q in the optimal-delay tree T^*. The assignment is used when determining the order in which sinks are "peeled off" from T^*.

Given the determination of $Pin(v)$, we now use the rules described in Figure 20 to peel off sinks, thus determining the correct sequence in which sinks should be added to construct T^*. Note that node q in Line 3 of Figure 20 must exist since T_i is finite and has no cycles. We now show that the procedure of Figure 20 gives a sequential decomposition of the optimal-delay tree T^*, such that each T_i is constructed by connecting sink n_i to tree T_{i-1} by a closest connection to some edge in T_{i-1}.

Lemma B5: There exists a sequence of subtrees $T_0 = \{n_0\}, T_1, T_2, \ldots, T_k = T^*$ such that for each i, $1 \leq i \leq k$, (i) there is a sink $n_i \in T_i$ such that $T_{i-1} = T_i \backslash n_i$, and (ii) either n_i is connected to n_0, or n_i makes a closest connection in T_i to some edge in T_{i-1}.

Proof: Part (i) of the Lemma is true since the construction of Figure 20 removes exactly one sink during each pass through Lines 3 to 9.

To show (ii), let p be the parent of the node q at Line 3 in Figure 20. The first case is when q is a sink

```
$T^*$ Decomposition Procedure
Input: Optimal delay tree $T^*$
Output: Sequence of sinks $n_1, \ldots n_k$ used to construct $T^*$
        using only closest connections of each $n_i$ to $T_{i-1}$
1.  $i = k$
2.  Repeat until $i == 0$
3.      Find a node $q \in T_i$ whose children are all leaves
4.      If $q$ has degree 4
5.          Set $c$ be the child of $q$ in $T^*$ such that
              edges $(q, c)$ and $(parent(q), q)$ are colinear
6.      Else Set $c$ to be a child of $q$ such that $Pin(c) \neq Pin(q)$
7.      $n_i = Pin(c)$
8.      $T_{i-1} = T_i \backslash n_i$
9.      $i = i - 1$
```

Figure 20: Procedure to determine a sequence of sinks $n_1, \ldots, n_k$ which can be used to construct T^* by a sequence of closest connections from n_i to tree T_{i-1}.

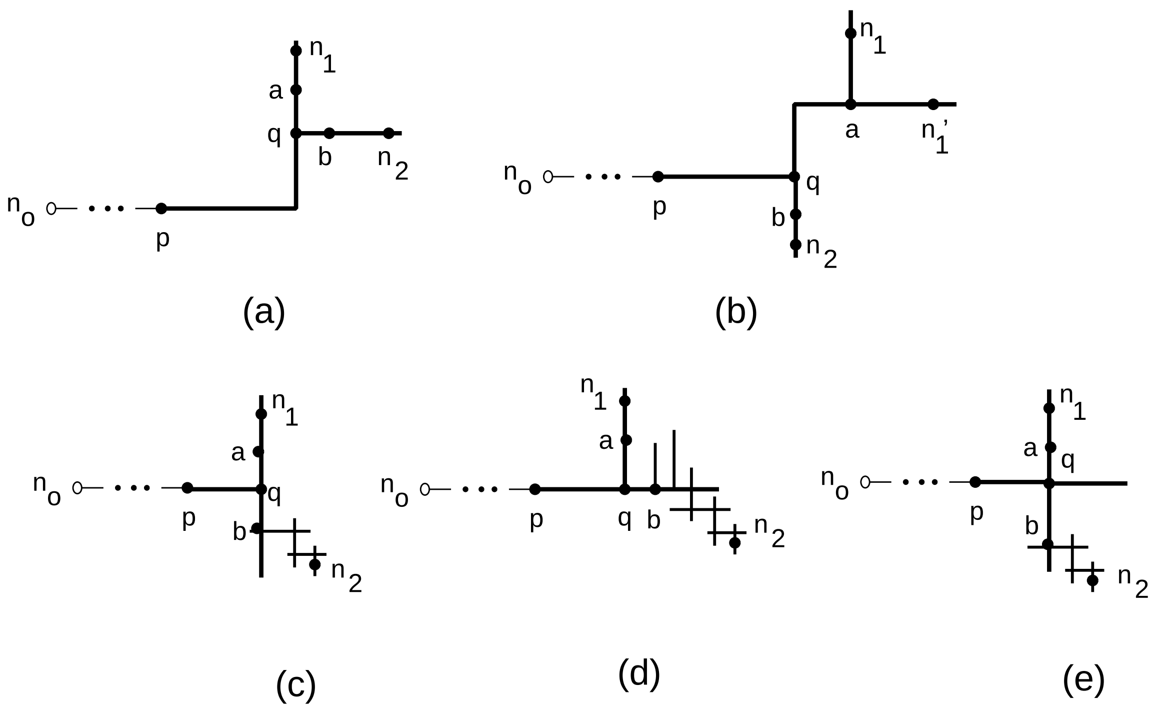


Figure 21: Five possible topologies at any Steiner node q in T^*. Each diagram shows two sinks n_1 and n_2 below node q in the tree, such that q is the closest connection between n_1, n_2 and q's parent p.

or a degree-4 Steiner node in T_i (as in Figure 21(e)). In this case, edge (p, q) will remain in tree T_{i-1}. If (p, q) is L-shaped, we must have a connection as in Figure 21(a), where the two children of q are eventually replaced by sinks on the maximal segments with entry point q (i.e., n_1 and n_2 in the Figure). Both of these sinks have closest connections to (p, q) at q. If (p, q) is a straight edge, let M be the MS containing (p, q), and let a be a child of q in T^*. The sink $Pin(a)$ is assigned in the Figure 19 template such that the q–$Pin(a)$

path in T^* will contain only edges in M, edges in branches off of M, or edges in a sequence of far branches off of branches of M. (For example, consider the paths from q to sinks n_1 and n_2 in Figure 21(c)-(e).) Thus, $Pin(a)$ and p cannot be on the same side of a line that passes through q and is perpendicular to M. Consequently, q will be the closest connection between edge (p,q) and $Pin(a)$.

The second case is when q is a degree-3 Steiner node in T_i. Let a and b be the children of q in T^* such that $Pin(a)$ and $Pin(b)$ are q's children in T_i. Without loss of generality, we assume that $Pin(q) = Pin(a)$ and $n_i = Pin(b)$. We must show that q is located at the closest connection between nodes p, $Pin(a)$, and $Pin(b)$. There are four possible configurations for connections at q, as shown in parts (a)-(d) of Figure 21.

- In Figure 21(a), edge (p,q) is L-shaped and both $Pin(a)$ and $Pin(b)$ (denoted by n_1 and n_2 in the figure) must be on maximal segments with entry point q; it is easy to see that q is the closest connection between p, $Pin(a)$, and $Pin(b)$.

In Figure 21(b)-(d), edge (p,q) is a a straight edge. Let M be the MS containing (p,q), and let M' be the MS perpendicular to M with entry point q.

- In Figure 21(b), edge (q,a) is L-shaped and edge (q,b) is on the MS M'. By Lemma B4, M' must contain a sink, which will be contained in subtree T_b. Thus, $Pin(b)$ (n_2 in the Figure) is located on M'. Node a is the entry point for two branches perpendicular branches containing sinks (by Lemma B4); $Pin(a)$ is chosen arbitrarily from one of these branches (Line 3 in Figure 19). In Figure 21, either $Pin(a) = n_1$ or $Pin(a) = n_1'$; thus, it can be seen from the Figure that q is the closest connection between p, $Pin(a)$, and $Pin(b)$.

- In Figure 21(c), M' is the union of two branches. One of these branches contains a sink (by Lemma B4); without loss of generality, let this be the branch containing edge (q,a), with $Pin(a) = n_1$ in M'. Let B be the branch containing edge (q,b). If $Pin(b)$ is on B, then q will be the closest connection between p, $Pin(a)$ and $Pin(b)$. Otherwise, according to Lemma B2 we must have that b is the entry point to a far branch off of M'. Hence, if $Pin(b)$ is not on B, the b–$Pin(b)$ path in T^* contains only edges on far branches (by the criteria in Lines 8-10 in Figure 19; see $n_2 = Pin(b)$ in Figure 21(c)). Thus, $Pin(b)$ is contained in the upper-right quadrant relative to q in the Figure, and q is the closest connection between p, $Pin(a)$, and $Pin(b)$.

- Finally, consider the configuration in Figure 21(d). Here, MS M' is a branch of M containing node a and sink $Pin(a)$. Suppose that M' is a far branch; if $Pin(b)$ is not on MS M, then there must be a near branch off of M somewhere below q in T^* (otherwise, we could reduce all delays by shifting the entire half segment of M below q toward a). Let B_j be the near branch below q closest to q. Either sink $Pin(b)$ is on B_j, or the q_j–$Pin(b)$ path in T^* consists only of edges in B_j or far branches. In either case, $Pin(b)$ ($= n_2$ in the Figure) is contained in the lower-right quadrant relative to q. If M'

is a near branch, an analogous argument again shows that $Pin(b)$ is q's lower-right quadrant. Thus, q is the closest connection between p, $Pin(a)$ and $Pin(b)$. $\square$

Except for redundancies and pruning of sub-optimal trees, BB-SORT-C searches over all possible ways to construct a Steiner tree sequentially, such that each sink is added by a closest connection to some edge in the current tree. Thus, we have:

Theorem B1: For any positive linear combination of sink delays, $f = \sum_{i=1}^{k} \alpha_i \cdot t(n_i)$, $\alpha_i > 0 \; \forall i$, algorithm BB-SORT-C returns a Steiner tree T^* which minimizes f. $\square$

References

[1] C. J. Alpert, T. C. Hu, J. H. Huang and A. B. Kahng, "A Direct Combination of the Prim and Dijkstra Constructions for Improved Performance-Driven Global Routing", *technical report* CSD-920051, UCLA Department of Computer Science, 1992.

[2] T. G. Andrews, ed., *Methods of Psychology*, New York, John Wiley, 1948.

[3] 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.

[4] 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*, Sept. 1992, pp. 35-40.

[5] K.D. Boese, A. B. Kahng and G. Robins, "High-Performance Routing Trees with Identified Critical Sinks", *Proc. ACM/IEEE Design Automation Conf.*, June 1993, pp. 182-187.

[6] K. D. Boese, A. B. Kahng, B. A. McCoy and G. Robins, "Fidelity and Near-Optimality of Elmore-Based Routing Constructions", University of Virginia TR-CS-93-14, 1993.

[7] D.-S. Chen and M. Sarrafzadeh, "A Wire-Length Minimization Algorithm for Single-Layer Layouts", *Proc. IEEE Intl. Conference on Computer-Aided Design*, 1992, pp. 390-393.

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

[9] 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.

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

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

[12] 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.

[13] 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.

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

[15] S. Even, *Graph Algorithms*, Potomac, MD, Computer Science Press, 1979.

[16] M. Hanan, "On Steiner's Problem with Rectilinear Distance", *SIAM J. Appl. Math.*, 14 (1966), pp. 255-265.

[17] 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.

[18] J.-M. Ho, G. Vijayan and C. K. Wong, "New Algorithms for the Rectilinear Steiner Tree Problem", *IEEE Transactions on Computer-Aided Design*, 9(2), 1990, pp. 185-193.

[19] 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.

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

[21] A. B. Kahng and G. Robins, "A New Class of Iterative Steiner Tree Heuristics with Good Performance", *IEEE Transactions on CAD* 11(7), July 1992, pp. 893-902.

[22] 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.

[23] S. Kim, R. M. Owens and M. J. Irwin, "Experiments with a Performance Driven Module Generator", *Proc. ACM/IEEE Design Automation Conf.*, 1992, pp. 687-690.

[24] T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout*, Berlin, Wiley-Teubner, 1990.

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

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

[27] S. Prasitjutrakul 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.

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

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

[30] G. Robins, "On Optimal Interconnections", Ph.D. thesis (*technical report* CSD TR-920024), CS Department, University of California, Los Angeles, June 1992.

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

[32] A. Srinivasan, K. Chaudhary and E. S. Kuh, "RITUAL: A Performance Driven Placement Algorithm for Small-Cell ICs", *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1991, pp. 48-51.

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

[34] S. Teig, R. L. Smith and J. Seaton, "Timing Driven Layout of Cell-Based ICs", *VLSI Systems Design*, May 1986, pp. 63-73.

[35] R. S. Tsay, "Exact Zero Skew", *Proc. IEEE Intl. Conference on Computer-Aided Design*, 1991, pp. 336-339.

[36] D. Zhou, F. P. Preparata and S. M. Kang, "Interconnection Delay in Very High-speed VLSI", *IEEE Trans. on Circuits and Systems* 38(7), 1991.

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