Figure 6: SERT construction on 9-pin net using IC parameters; sinks numbered in order of distance from source (pin 1).

Figure 7: Examples of SERT-C constructions.
parameters and by 69% for MCM parameters. The ERT algorithm also improves upon AHHK, with reductions of 10% (IC) and 43% (MCM).\footnote{These results are particularly impressive because the implementation of AHHK in \cite{1} simulates delays for output trees for 21 different values of the $c$ parameter, while our contributions provide the best value of $c$ for each instance.}

The Steiner tree version of our ERT method also performs well as a generic high-performance router. For signal nets with 16 sinks, the two-pole simulations show that SERT improves average sink delay versus the 1-Steiner routing by 21% and 62% for the IC and MCM parameters, respectively. For 8-sink nets, average delays are reduced by 10% for IC and 42% for MCM. The percentage reductions in maximum delay are even greater. Again, with the MCM technology the ERT and SERT constructions tend to be more star-like, with increased tree cost; when delay is not the overriding concern, this may be compensated by simulating a larger $r_d$ value in the $t_{cd}$ computation.

Even more significant average reductions in delay are achieved when a critical sink has been identified. The SERT-C algorithm improves upon the SERT results by an additional reduction in delay at the critical sink of 10% for ICs and 7% for MCMs. Identification of a critical sink has key advantages in terms of 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. Finally, we note that SERT-C is more practical than HB, in that it does not use the circuit simulator during construction of the tree.\footnote{With respect to practicality: the average CPU times in seconds for SERT-C and the two-pole distributedRCM delay simulator \cite{21} on a Sun Sparc IPC were respectively 0.002 and 0.001 seconds for $|N| = 5$; 0.017 and 0.049 for $|N| = 9$; 0.31 and 0.089 for $|N| = 17$. This reflects an “all-purpose” implementation of our package, which has $O(k^2)$ complexity for each of ERT, SERT and SERT-C, although SERT-C and ERT can be implemented with time complexities $O(k^2)$ and $O(k^3)$, respectively, as discussed above.}

Figures 6 and 7 illustrate the SERT and SERT-C algorithms for one of the random signal nets used in our simulations, with the IC technology parameters. Figure 6 shows the progressive growth of the SERT construction. Figure 7 contains the trees produced by SERT-C for each choice of critical node. Note that the tree constructed when node 3 or node 7 is critical is also the 1-Steiner tree, and the tree constructed when node 8 is critical is the same as the generic SERT output.

\section{Extensions}

Our current work addresses integration of the ERT construction into an existing global router. We are also considering the CSRT problem in 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. The CS-Steiner and ERT approaches also extend to general-cell layout with arbitrary routing region costs. Finally, we believe that the SERT and ERT time complexities can be reduced by at least a factor of $k$; we leave this as an open problem.

\section{Acknowledgements}

We are grateful to the authors of \cite{21} for use of their simulator code.

\begin{thebibliography}{99}
\end{thebibliography}
t(n_c), our SERT-C algorithm takes a virtually opposite approach: it starts with the required n_o-n_c connection and grows the routing tree while keeping t_E(n_c) as small as possible.

**Time Complexities**

Surprisingly, the SERT-C algorithm can be implemented in $O(k^2)$ time. This is shown as follows. For any edge from sink $u \notin V$ of $E(v, w) \in E$, there is only one possible connecting point $w$ (the closest) that we consider. The effect of inserting edge $(u, w)$ into $T$ on the delay $t_E(n_c)$ arises only in the $C_j$ terms in equation (1), and is therefore 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 $(n_o, n_c)$. 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 the same and need not be recomputed). Thus, updating the effect on $t_E(n_c)$ of connecting each $u \notin V$ requires linear time in each pass through the while loop of Figure 5.

The ERT spanning tree algorithm can be implemented in $O(k^2)$ time, assuming constant unit resistance, unit capacitance, and sink capacitances. This fact follows from a simple observation: if a new tree edge incident to sink $u \notin V$ (Step 3 of Figure 5) minimizes the maximum Elmore delay $\max_i t_E(n_i)$, it must connect $u$ to the sink $v \notin V$ that is closest to $u$. At each pass through the while loop, we update the shortest “outside connections” for every $u \notin V$ (a total of $O(k^2)$ time), and then add each of these $O(k)$ shortest outside connections to $T$ in turn, evaluating the Elmore delays to all sinks of the resulting trees in $O(k)$ time. Hence, each pass through the while loop requires $O(k^2)$ time, yielding the $O(k^2)$ complexity result. In practice, this complexity will be transparent to the user since $k$ is typically small.

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 Step 3 we must always consider $O(k^2)$ Steiner connections, and (ii) the connection which minimizes $\max_i t_E(n_i)$ in Step 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 quadratic speedup over the generic Steiner computation (SERT).

4 **Experimental Results**

4.1 **CS-Steiner Trees**

We have implemented each of the CS-Steiner variants $H_0$, $H_1$ and $H_{Best}$ along with the $ISteiner$ algorithm [12] using C in the UNIX Sun environment, and have run the algorithms on random 4-, 8- and 16-sink inputs. We also applied our GSR post-processing algorithm (denoted as $+U$ to $ISteiner$ and each of the CS-Steiner variants. Our inputs correspond to two distinct technologies: (i) IC a representative 0.8 μm CMOS process, and (ii) MCM: a typical example of current MCM technologies.

Table 1 gives delay and tree cost results and comparisons. Delays at all sink nodes were estimated using the computationally efficient two-pole circuit simulator developed by Zhou et al. [21]. Each delay figure gives the average over all sink nodes in 50 random point sets. Since the $ISteiner$ algorithm is net-oriented, it will return the same solution no matter which sink is critical. In contrast, CS-Steiner can return a different tree for each choice of critical sink. Thus, we use the CS-Steiner delay at $n_v$ in the specific tree corresponding to the identification of $n_v$ as the critical sink.

<table>
<thead>
<tr>
<th></th>
<th>IC</th>
<th>MCM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$</td>
<td>N</td>
</tr>
<tr>
<td>Ave</td>
<td>ISteiner</td>
<td>2.44</td>
</tr>
<tr>
<td>Delay</td>
<td>H0+U</td>
<td>2.26</td>
</tr>
<tr>
<td>(ms)</td>
<td>H1+U</td>
<td>2.37</td>
</tr>
<tr>
<td></td>
<td>HBest+U</td>
<td>2.12</td>
</tr>
<tr>
<td>Ave</td>
<td>ISteiner+U</td>
<td>1.51</td>
</tr>
<tr>
<td>Cost</td>
<td>H0+U</td>
<td>1.95</td>
</tr>
<tr>
<td>(cm)</td>
<td>H1+U</td>
<td>1.58</td>
</tr>
<tr>
<td></td>
<td>HBest+U</td>
<td>1.67</td>
</tr>
</tbody>
</table>

Table 1: Routing tree simulation results using IC and MCM technology parameters and the two-pole simulator [21]. Notes: (i) all source and sink locations are chosen randomly in a layout region with grid resolution 25 μm; (ii) each value in a given column represents an average over the same 50 random signal nets.

Variants $H_0$ and $H_{Best}$ significantly reduce delay to the critical sink, particularly in nets with a large number of sinks and in the MCM technology where output driver and wire resistances are low. For 9-pin nets and the IC parameters, $H_{0+U}$ improves upon $ISteiner+U$ by 12% while $H_{Best+U}$ has an average improvement of 16%. With MCM parameters, the improvement is much greater: 48% for both $H_{0+U}$ and $H_{Best+U}$. The two-pole simulation results for 12-pin nets show even greater reductions in delay: 29% and 30% respectively for $H_{0+U}$ and $H_{Best+U}$ under IC; 70% for both $H_{0+U}$ and $H_{Best+U}$ under MCM. Thus, the simple strategy of connecting the critical node via a path with low branching factor is very successful for these cases (at the expense of larger net cost).

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 are presented in Table 2. For purposes of comparison, the table includes data from the minimum spanning tree, $AIHK$ tree (quoted from [1]), and $ISteiner$ tree constructions.

Even as generic net-dependent routers, our ERT methods are highly effective. For nets with 16 sinks, the spanning tree ERT construction reduces average sink delay versus the MST construction by 33% for IC.
**HBest:** Accomplish Step 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 \( n_c \).

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

We enhance the CS-Steiner approach via a linear-time Global Stack Removal (GSR) postprocessing algorithm. GSR takes as input a 1-Steiner tree\(^2\) and shifts edges in the tree to maximize monotonicity of all source-sink paths. If we orient a 1-Steiner tree \( T \) by rooting it at the source \( n_0 \), a \( U \) is defined to be a subpath of three consecutive edges on a root-leaf path, such that the combined edge cost is greater than the distance between the endpoints (e.g., path \( v_1 \rightarrow v_4 \) in Figure 3(a)). GSR (Figure 4) removes \( U \)'s as shown in Figure 3(b). The input tree must be processed in top-down order to guarantee the following two results (see [4] for discussion and proofs):

![Figure 3: GSR: Removing a single “U”.](image)

**Theorem 1:** Given a 1-Steiner tree \( T(N) \), GSR returns \( T'(N) \) such that (i) \( cost(T'(N)) \leq cost(T(N)) \); (ii) all \( n_0-n_1 \) pathlengths in \( T'(N) \) are reduced or remain the same; and (iii) all Elmore delays \( t_{ED}(n_0, n_1) \) in Step 1 of Figure 5 and then continuing as in the SERT algorithm, except that we minimize Elmore delay at the critical sink \( t_{ED}(n_c) \) rather than the maximum delay to all sinks. This yields the SERT-C (“SERT with identified Critical sink”) algorithm.

While CS-Steiner began with a minimum-cost Steiner tree and perturbed it to heuristically improve

### 3.2 Elmore Routing Trees

We now propose a new class of greedy Elmore routing tree (ERT) algorithms which optimize Elmore delay directly during construction of the routing tree. The ERT approach is significant for its avoidance of such abstractions as “minimum cost” or “bounded radius”, while still maintaining computational efficiency.

The ERT algorithm starts with the trivial tree \( T \) containing only the source \( n_0 \), and iteratively adds a sink to \( T \). In each step, we seek terminals \( u \in T \) and \( v \notin T \), such that adding edge \((u, v)\) to \( T \) will minimize the maximum Elmore delay at all sinks in the new tree.\(^3\) A formal description of the ERT algorithm is given in Figure 5.

![ERT Algorithm](image)

**Figure 5:** The ERT algorithm.

The ERT algorithm generalizes to yield the Steiner Elmore routing tree (SERT) algorithm when we allow 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. In other words, we find \( u \notin V \), \((v, v') \in E \), and a new point \( w \) which minimizes the maximum Elmore delay from \( n_0 \) to any sink in the tree \((V \cup \{u, w\}), (E \cup \{(u, v')\}) \cup \{(v, w), (w, v'), (u, w)\})\). We then add \( u \) and \( w \) to \( V \), and replace \( E \) by \((E \cup \{(u, v')\}) \cup \{(v, w), (w, v'), (u, w)\})\). We address the CSRT problem by beginning with a tree containing the single edge \((n_0, n_1)\) in Step 1 of Figure 5 and then continuing as in the SERT algorithm, except that we minimize Elmore delay at the critical sink \( t_{ED}(n_c) \) rather than the maximum delay to all sinks. This yields the SERT-C (“SERT with identified Critical sink”) algorithm.

\(^2\) A 1-Steiner tree has the property that no single Steiner point can be added to reduce the tree cost. The 1-Steiner algorithm\([12]\) always produces such a tree.

\(^3\) Our approach should be distinguished from the method of Prasijutrapakul and Kubitz\([17]\) cited in Section 2.1. Like our method,\([17]\) 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 suffers from the standard pitfall of disregarding the true underlying delay criterion. Indeed, an example is easy to construct (similar to Figure 1a) for which their method yields a tree with Elmore delays at least twice as large as those of ERT. Practical considerations also separate the two methods, e.g.,\([17]\) cannot be easily modified to address our CSRT formulation.
layouts, and [17] uses A* heuristic search in a similar domain.

Cong et al. [6] proposed the BRBC (bounded-radius, bounded-cost) algorithm, which simultaneously constrains both radius (maximum source-sink pathlength) and cost (total tree wirelength) within constant factors of optimal. Both [6] and [13] belong to the class of “shallow-light” methods [2], which achieve a radius-cost tradeoff by a modified depth-first traversal of the minimum-cost spanning tree (MST) over $N$. Very recently, a radius-cost tradeoff was achieved by combining the Dijkstra (shortest-path tree, or SPT) and Prim (MST) recursions: the AHHK algorithm of [1] iteratively adds edge $e_{ij}$ and pin $n_i$ to $T$, choosing $n_i \in T$ and $n_i \in N - T$ to minimize $(c \cdot l_j + d_{ij})$ ($l_j$ is the pathlength from $n_0$ to $n_j$, and $c$ is a tradeoff parameter). AHHK sink delays outperform those of BRBC by anywhere from 6% to 28%, depending on net size, technology, and whether average or maximum sink delay is considered. Thus, Section 4 compares our methods against the AHHK algorithm. Because minimum Steiner trees also give bona-fide “performance-driven” routing in certain technologies, Section 4 also compares our methods against the 1-Steiner heuristic of [12].

### 2.2 Intuitions From the Elmore Model

To solve the CSRT problem, we first develop intuitions from Elmore’s distributed RC-tree model [9]. Elmore delay 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 $r_e$ and $c_e$, 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 $C_j$ to denote the tree capacitance of $T_j$, namely the sum of sink and edge capacitances in $T_j$. Using this notation, the Elmore delay along edge $e_i$ is equal to $r_{es}c_{es}(2 + C_j)$. 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:

$$t_{ED}(n_i) = r_dC_{n_0} + \sum_{e_j \in Path(n_0, n_i)} r_{e_j}(c_{e_j}/2 + C_j)$$  

1

Since $r_{es}$ and $c_{es}$ are usually proportional to the length of $e_i$, $t_{PD}(n_i)$ has quadratic dependence on 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 length edge of the tree, 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 is a minimum cost tree; when $r_d$ decreases, the ORT tends to a “star” topology. The size of $r_d$ relative to unit wire resistance is a “resistance ratio” [3] that captures the technology vis-à-vis routing tree design. Typical values of $r_d$ are relatively large for current-generation CMOS, but decrease in, for example, MCM substrate and submicron CMOS IC interconnects.

Figure 1 shows a signal net with identified critical sink $n_c$, along with three routing trees solutions. We make the following observations. (i) The minimum cost solution (a) has large delay to the critical sink $n_c$ due to the long source-sink path. (ii) However, requiring a monotone path to every sink as in the SPT (b) can result in large tree capacitance, again leading to large delay at $n_c$. (iii) The optimal CSRT construction (c) shows the dependence of routing topology on the choice of critical sink, and reflects both the SPT and MST solutions. (iv) 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 $n_0$. 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 $n_c$.

![Figure 1](image-url)

**Figure 1:** (a) Optimal Steiner tree (cost $2.0\times m$, $t(n_c) = 5.90\mu s$); (b) minimum cost SPT (cost $2.5\times m$, $t(n_c) = 4.11\mu s$); and (c) optimal-delay tree (cost $2.2\times m$, $t(n_c) = 3.07\mu s$) for the same sink set. Coordinates are in mm, and the simulation used $0.8\mu m$ CMOS IC parameters (see Section 4). (d) Two distinct minimum-cost SPT solutions for a net with three sinks.

### 3 Two Classes of CSRT Heuristics

#### 3.1 The CS-Steiner Approach

We may view the optimal CSRT solution in Figure 1(c) as minimizing total tree cost, subject to the path from $n_0$ to $n_c$ being monotone. This recalls the motivations in [1][6], but the tradeoff is with respect to only the critical sink $n_c$. We thus obtain the simple CS-Steiner heuristic:

**CS-Steiner Algorithm**

**Input:** Signal net $N$ with critical sink $n_c$  
**Output:** heuristic CSRT solution $T$

1. construct heuristic tree $T_0$ over $N - \{n_c\}$
2. connect $n_c$ to $T_0$ by a monotone $n_0-n_c$ path

![Figure 2](image-url)

**Figure 2:** CS-Steiner template.

Since the algorithm template in Figure 2 is quite general, we have examined a number of CS-Steiner variants. All of our variants use the 1-Steiner heuristic [12] to construct the initial tree $T_0$ in Step 1. Section 4 reports results for the following three variants:

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

**H1:** The direct connection in Step 2 consists of the shortest possible wire that can join $n_c$ to $T_0$, subject to the monotone path constraint.
High-Performance Routing Trees
With Identified Critical Sinks*

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

Computer Science Department, University of California at Los Angeles, Los Angeles, CA 90024-1596
† Computer Science Department, University of Virginia, Charlottesville, VA 22903-2442

Abstract

We present two critical-sink routing tree (CSRT) constructions which exploit critical-path information that becomes available during timing-driven layout. Our CS-Steiner heuristics with “Global Slack Removal” modify traditional Steiner constructions and produce routing trees with significantly lower critical-sink delays compared with existing performance-driven methods. We also propose a new class of Elmore routing tree (ERT) constructions, which iteratively add tree edges to minimize Elmore delay. This direct optimization of Elmore delay yields trees that improve delays to identified critical sinks by up to 69% over minimum Steiner routings. ERTs also improve performance over such recent methods as [1] [6] when no critical sinks are specified.

1 Introduction

Prevailing approaches to performance-driven layout use static timing analysis to iteratively drive the module placement and global routing phases. For example, net-dependent placement algorithms typically use centroid-connected star cost [19], probabilistic estimates of Steiner tree cost [11], minimum spanning tree cost [7] or the bounding box semiperimeter [16] to estimate wire capacitance and signal delay for a multi-terminal net. From this information, critical timing paths can be computed and module placements updated to reduce these “net-based” objective functions. On the other hand, path-dependent methods consider delay between the source and a particular critical sink of a multi-terminal net. The critical sink is determined via timing analysis using known module delays and estimated path delays [10, 15, 20].

If a timing-critical path passes through a given net, the path-dependent approach will afford an explicit routing constraint which bounds delay at that net’s critical sink. The net-dependent approach seemingly imposes only implicit routing constraints; nevertheless, it is easy to identify critical sinks a priori or after timing analysis has been performed. Despite the availability of this critical-path information during iterative performance-driven layout, current routing methods do not fully exploit this information. With this in mind, our paper offers two contributions. First, we demonstrate that simple changes to existing methodologies can effectively exploit available critical-path timing information. Second, we propose a basic advance in routing tree construction: we give a class of greedy heuristics which directly optimize Elmore delay in the tree construction, thus avoiding such previous abstractions as “minimum wirelength” or “bounded radius”. Our use of the Elmore model has been validated by recent results in [5] which demonstrate the high fidelity of Elmore delay with respect to physical delay: optimizing Elmore delay in 4- and 5-pin nets produces routing tree topologies that are nearly optimal according to SPICE simulations.

2 High-Performance Routing Trees

A signal net N consists of fixed terminal locations \( N = \{n_0, n_1, \ldots, n_k\} \subset \mathbb{R}^2 \) which are to be connected by a routing tree \( T(N) \). \( N \) contains a source terminal \( n_0 \) with the remaining terminals sinks. We allow each \( n_i \) to have an associated criticality \( \alpha_i \), reflecting available timing constraints. The cost of an edge \( e_{ij} \) in \( T(N) \), denoted by \( d_{ij} \), is the Manhattan distance between its endpoints \( n_i \) and \( n_j \). The cost of \( T(N) \) is the sum of its edge costs. The signal delay between two terminals \( n_i \) and \( n_j \) in \( T(n) \) is denoted by \( t(n_i, n_j) \); we use \( t(n_i) \) to indicate the delay from the source to a sink \( n_i \). Our goal is to solve the

Critical-Sink Routing Tree (CSRT) Problem: Given signal net \( N \), construct \( T(N) \) which minimizes \[ \sum_{i=1}^{k} \alpha_i \cdot t(n_i). \]

The CSRT formulation captures traditional performance criteria: if we set all \( \alpha_i \) equal, (i) we minimize average sink delay by taking the weighted delay sum using the \( L_1 \) norm, and (ii) we minimize maximum sink delay by taking the weighted delay sum using the \( L_\infty \) norm. Our discussion will concentrate on the simple yet realistic case where exactly one critical sink, denoted by \( n_{c_i} \), has been identified, i.e., \( \alpha_i = 1 \) and all other \( \alpha_i = 0 \). Our methods generalize to the case where a small number of critical sinks is specified.

2.1 Existing Approaches

Early performance-driven routing methods relied on minimizing wirelength via minimum Steiner constructions, e.g., [8] uses static timing analysis to prioritize nets such that critical nets can be routed by minimum Steiner trees. Kuh et al. [14] give a performance-driven routing method for hierarchical building-block