Timing-Driven Interconnect Synthesis*

Jiang Hu† Gabriel Robins‡ and C. N. Sze§

1 Introduction

In this chapter we address performance-driven interconnect synthesis, which seeks to optimize circuit performance by minimizing signal delays to critical sinks. Timing-driven wiring geometries are in general quite different from optimal-area (i.e., Steiner) interconnect trees, especially as die sizes continue to grow while feature dimensions steadily shrink.[1] The exposition below focuses on selected approaches to performance-driven routing, and details key historical research developments that helped usher in the era of high-performance interconnect synthesis. For extensive surveys on this subject, see [19, 50]. For a general overview of computer-aided design (CAD) of very large scale integrated (VLSI) circuits, see some of the classical textbooks [40, 80, 90, 93, 95].

As transistor sizes continued to dramatically shrink while their switching speeds have increased into the multi-gigahertz range, the circuit performance bottlenecks migrated from the devices themselves to the wires that interconnect them. Indeed, it was observed in the late 1980's that given the VLSI scaling trends at that time, interconnection delay was already contributing up to 70% of the clock cycle in circuits [6, 30, 99]. Performance-driven layout design thus started to receive much research attention, especially timing-driven placement, which has a particularly significant effect on signal delays [30, 43, 56, 70, 73, 99]. However, during that early era in the evolution of VLSI CAD, routing solutions were typically not available during the placement phase.

*This work was supported by a Packard Foundation Fellowship, by National Science Foundation Young Investigator Award MIP-9457412, and by NSF grants CCR-9988331, CCF-0429737, and CNS-0716635.

†Department of Electrical and Computer Engineering, Texas A&M University, TX 77843.

‡Department of Computer Science, University of Virginia, Charlottesville, VA 22904.

§IBM Austin Research Laboratory, Austin, TX 78758.

[1] In routing *non-critical* nets (or sinks), rather than optimize delay we instead seek to minimize overall wirelength, an objective which gives rise to variants of the classical Steiner problem [12, 16, 17, 18, 31, 53, 55, 65, 82, 88]. On the other hand, modern ultra-deep-submicron VLSI CAD seeks to optimize and tradeoff various combinations of objectives and criteria, such as delay, skew, area, density, manufacturability, reliability, power, electromigration, parasitics, noise, and signal integrity [4, 9, 20, 54, 59, 62, 75, 96].

Performance-driven methods of the early 1990's therefore used simple (e.g., geometric or linear) estimates of interconnection delay to drive the placement process, sacrificing modeling accuracy in favor of computational tractability.

For a given timing-driven placement, a corresponding timing-driven routing seeks to minimize source-to-sink signal delays. In order to optimize circuit performance, early timing-driven routing methods relied on e.g., net priorities [80], static timing analysis [32], hierarchical approaches [57], and A* search [79]. Since the early 1990's there has been a steady shift from technology-independent routing methodologies to technology-dependent interconnect synthesis. Analyses of the Elmore delay formula [34] for distributed RC trees [72, 89, 102] motivated cost-radius tradeoffs that depended on the underlying technology [3, 5, 23, 24, 63]. Thus, routing tree constructions that were based on various technology parameters, net criticalities, and other timing or performance issues provided improvements over the previous static, technology-oblivious methods [62].

Several early works abandoned the algorithmic convenience and analytic simplicity of classical geometric objectives, and began to address the less tractable but more realistic "actual delay". For example, an early sequence of papers by Boese, Kahng, and Robins [7, 8, 9, 10] proposed new classes of delay objectives, along with improved-performance routing algorithms that directly optimized, e.g., the Elmore delay. These works also established the fidelity of Elmore-based constructions relative to accurate delay simulators (e.g., SPICE) [62]. That is, it was observed that optimizing the Elmore delay tends to also minimize real delay.

In parallel with these advances, sink-dependent delay objectives were recognized as more critical than net-dependent delay minimization. Because the timing-driven placement and routing design loop usually iterated tightly with static timing estimation, critical-path information was often available during routing. Thus, formulations which optimized delays with respect to a set of critical sinks proved more effective than ones that optimized delays in individual nets while ignoring the critical sinks [62]. The near-optimality of minimum-delay routing heuristics was also quantified empirically, showing e.g., that certain simple heuristics achieved almost optimal critical sink delays [9, 10, 62, 69]. Other advances in timing-driven interconnect synthesis for improving circuit performance included various approaches to wiresizing, non-Hanan routing, non-tree topologies, and arborescence trees. The remainder of this chapter will discuss some of these topics and techniques in greater detail.

2 Wirelength-Radius Tradeoffs

Researchers in interconnect synthesis observed that while low-wirelength routing trees have smaller capacitance-related delays, low-radius interconnects have shorter pathlength-related signal propagation delays [62].[2] However, there exists an inherent conflict between these two objectives (i.e., minimizing overall tree cost vs. minimizing source-to-sink pathlengths), and when one of these two objectives is optimized, the other objective typically suffers (Figure 1). Indeed, shortest-paths trees (i.e., those produced by Dijkstra's classical algorithm [29]), have the best possible source-to-sink pathlengths but usually induce high overall tree cost (Figure 1(a)). On the other hand, minimum spanning trees (i.e., those produced by Prim's classical algorithm [81]), have optimal tree cost but produce potentially high source-to-sink pathlengths (Figure 1(b)).

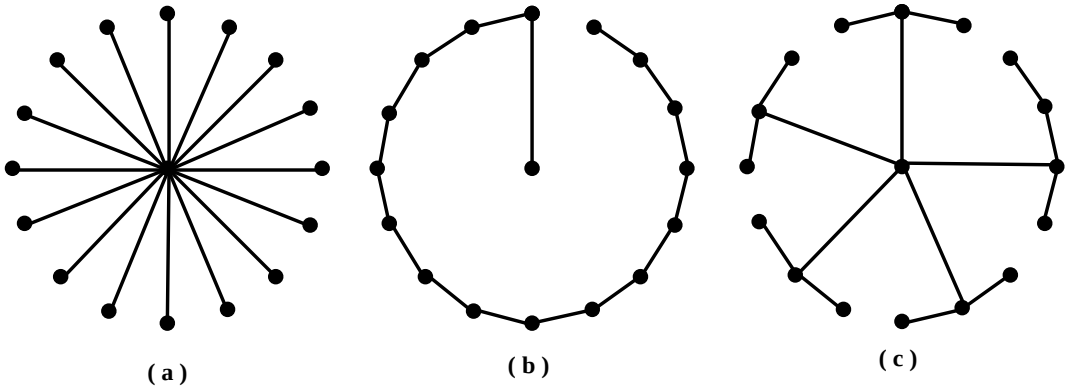

Figure 1: Candidate interconnection trees for the same net, where the signal source pin is located at the center and the sinks are located on the circumference of a circle: (a) a shortest paths tree; (b) a minimum spanning tree; and (c) a tradeoff low-cost low-radius hybrid tree.

In order to simultaneously optimize both the routing tree radius as well as its cost, the following formulation was proposed [22]:

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

[2]We define the *radius* of a routing tree/topology to be its maximum source-to-sink pathlength, and its *cost* to be its total wirelength. Similarly, the radius of a net is defined as its farthest source-to-sink distance. Distances and wirelengths are usually measured using the Manhattan / Rectilinear norm, although alternative interconnect architectures with more complicated underlying metrics have recently become popular, such as "preferred direction" routing and λ-geometries [14, 15, 16, 64, 68, 77, 92, 100, 104].

The user-specified parameter ϵ controls the tradeoff between the competing minimum-radius and minimum-cost objectives. Setting $\epsilon = 0$ induces a minimum-radius (i.e., shortest paths) tree, while increasing ϵ loosens the radius restriction, thus allowing further tree cost optimization. At the other extreme, setting $\epsilon = \infty$ results in a minimum-cost spanning tree. Note that these definitions and formulations easily generalize from spanning trees to Steiner trees (i.e., where new points/vias may be added to further optimize total wirelength). However, in performance-driven layout, where a fast delay estimator is employed in a tight iterative design loop, spanning trees are typically easier to compute than Steiner trees. Moreover, a spanning tree can usually be easily converted into a corresponding Steiner solution (e.g., by edge-overlapping), without disimproving its original radius.

The earliest heuristic to solve the BRMRT problem was the "Bounded-Prim" (BPRIM) approach of [22, 24], which follows the general structure of Prim's minimum spanning tree algorithm [81]. While simple to implement and effective in practice over typical inputs, this approach can produce trees with cost arbitrarily larger than optimal in the worst case. "Shallow-light" tree constructions avoid such worst-case scenarios by simultaneously bounding both the worst-case radius and the worst-case cost of the resulting routing tree [5, 23, 24, 63].

The basic approach of algorithms such as the Bounded-Radius Bounded-Cost (BRBC) method [24] is as follows: (a) traverse a minimum spanning tree in depth-first order; (b) insert additional edges whenever the prescribed radius bound is violated; and (c) return the shortest paths tree over the resulting graph (see Figure 2). The BRBC algorithm produces a tree with radius at most $(1 + \epsilon)$ times optimal, and cost at most $(1 + \frac{2}{\epsilon})$ times optimal [24, 62].

The BRMRT problem formulation and the BRBC algorithm generalize to regimes where we seek a low-radius tree that spans a vertex subset in an underlying graph, while using the remaining graph vertices as potential Steiner points to minimize the overall interconnection cost. Note that when $\epsilon = \infty$, the classical Graph Steiner problem is a special case of this generalization. A BRBC Steiner analog first constructs an approximate minimum-cost Steiner tree T that spans the target vertex subset, and then proceeds with the remaining radius-minimization optimization as before. This will yield a routing tree with radius bounded by $(1 + \epsilon)$ times optimal, and cost bounded by $(1 + \frac{2}{\epsilon})$ times the cost of T.

Note that the cost of the heuristic Steiner tree T can itself be bounded by a constant times

BRBC Algorithm [24, 62]
Input: Graph $G = (V, E)$ (with radius R, source $s_0 \in V$), $\epsilon \geq 0$ **Output:** Spanning tree T_{BRBC} with $r(T_{BRBC}) \leq (1+\epsilon) \cdot R$ and $cost(T_{BRBC}) \leq (1+\frac{2}{\epsilon}) \cdot cost(T_M)$

$Q = T_M$
$L =$ depth-first tour of T_M
$Sum = 0$
For $i = 1$ to $\|L\| - 1$
  $Sum = Sum + dist(L_i, L_{i+1})$
  If $Sum \geq \epsilon \cdot dist_G(s_0, L_{i+1})$ **Then**
  $Q = Q \cup \{$ edges in $minpath_G(s_0, L_{i+1})\}$
  $Sum = 0$
Output T_{BRBC} = shortest paths tree of Q

Figure 2: The bounded-radius bounded-cost (BRBC) spanning tree algorithm [24, 62] produces a tree T_{BRBC} with radius at most $(1+\epsilon) \cdot R$ and cost at most $(1+\frac{2}{\epsilon}) \cdot cost(T_M)$.

optimal. For example, if we use the best-known general graph Steiner heuristic of Robins and Zelikovsky [87, 88] which has an approximation bound of $1 + \frac{\ln 3}{2} \approx 1.5493$ times optimal for arbitrary weighted graphs, then the resulting Steiner-BRBC tree cost bound will be $(1+\frac{\ln 3}{2}) \cdot (1+\frac{2}{\epsilon})$ times optimal for general graphs. The underlying geometry can be exploited to further improve the cost bound of Steiner-BRBC to $2 \cdot (1+\frac{1}{\epsilon})$ times optimal for any metric. In particular, for the Manhattan and Euclidean geometries, this general bound can be further improved to $\frac{3}{2} \cdot (1+\frac{1}{\epsilon})$ times optimal and $\frac{2}{\sqrt{3}} \cdot (1+\frac{1}{\epsilon})$ times optimal, respectively. For λ-geometries (which allow wiring angles of $\frac{i\pi}{\lambda}$ [92]), a cost bound of $(\frac{2}{\sqrt{3}} \cos\frac{\pi}{\lambda}) \cdot (1+\frac{1}{\epsilon})$ times optimal can be shown for BRBC [62].

Experimental benchmarks indicate that both the BPRIM and BRBC algorithms run quickly and indeed yield a smooth tradeoff between tree cost and tree radius [24, 62]. In fact, on typical nets, the cost-radius tradeoff is on average significantly more favorable than suggested by the theoretical bounds. For example, for 10 pins and $\epsilon = 1$, BRBC offers an average of 21% savings in tree radius over optimal, at the expense of only 13% average rise in tree cost over optimal. Moreover, the interconnects produced by BPRIM and BRBC have significantly better delay characteristics than classical Steiner trees, as verified by accurate timing simulators (e.g., SPICE) [24, 62].

An alternative approach to the wirelength-radius tradeoffs is the AHHK algorithm [3], which integrates Prim's minimum spanning tree algorithm [81] and Dijkstra's shortest path tree algorithm

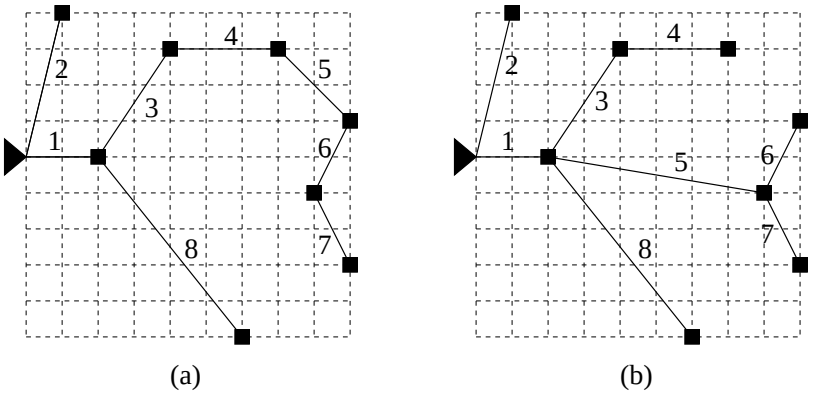

Figure 3: Examples of AHHK tree in the Euclidean plane, with $c = \frac{1}{3}$ (radius 15.9 and cost 26.4) in (a) and $c = \frac{2}{3}$ (radius 10.3 and cost 29.7) in (b). The edge labels indicate the order of adding the edges in the algorithm.

[29]. Prim's algorithm minimizes the total wirelength, while Dijkstra's algorithm minimizes the tree radius (i.e., the source-to-sink pathlengths). Thus, these two classic algorithms address, albeit separately, two major concerns in performance-driven interconnect synthesis. On the other hand, these two algorithms can be implemented similarly, by starting from the source node and adding one edge at a time until all the specified vertices in V are spanned.

The main difference between these two algorithms is the criterion for selecting which edge to be added at each iteration. Prim's algorithm [81] selects the edge with the minimum length. In particular, Prim's algorithm iteratively adds to the growing tree T a new node v_j and edge e_{ij}, where $v_i \in T$ and $v_j \in V - T$ are chosen to minimize the edge length $|e_{ij}|$. In contrast, Dijkstra's algorithm [29] attempts to minimize the pathlength from the source node when selecting an edge. Specifically, Dijkstra's algorithm iteratively adds to the growing tree T a new node v_j and edge e_{ij}, where $v_i \in T$ and $v_j \in V - T$ are chosen to minimize the the sum of the edge length $|e_{ij}|$ and the pathlength l_i from the source node to vertex v_i in T.

Generalizing this similarity between the two traditional methods of Prim and Dijkstra, the AHHK algorithm iteratively adds to the growing tree T a new node v_j and edge e_{ij}, where $v_i \in T$ and $v_j \in V - T$ are chosen to minimize the the sum of the edge length $|e_{ij}|$ and the pathlength l_i from the source node to vertex v_i in T *times a fixed constant* c. In this hybrid scheme, the chosen constant $0 \leq c \leq 1$ serves to smoothly *trade off* total wirelength againt tree radis (i.e., source-to-sink pathlengths). In particular, when $c = 0$, the resulting AHHK tree is identical to Prim's

minimum psanning tree, and when $c = 1$, the resulting AHHK tree it is the same as Dijkstra's shortest paths tree. Varying the value of c between 0 and 1 results in intermediate tradeoff trees between the two extremes of Prim's and Dijkstra's constructions. Figure 3 gives examples of AHHK trees for different values of the tradeoff parameter c.

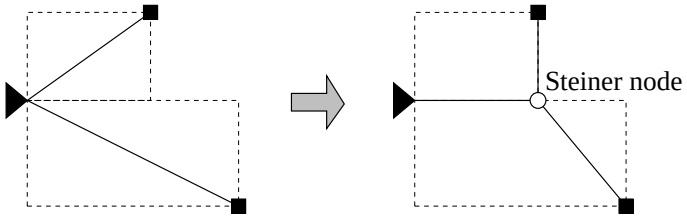

Figure 4: Examples of converting a spanning tree into a rectilinear Steiner tree through edge overlapping.

Once an AHHK spanning tree is obtained, it can be converted to a rectilinear Steiner tree using edge overlapping. That is, if the bounding boxes of two tree edges overlap, the overlapping portions can form a new edge with one end being a Steiner node, as illustrated in Figure 4. Such edge overlappings can usually reduce wirelength with respect to the original spanning tree.[3] If there are multiple options for edge overlapping at a given step, we can break ties by giving priority to overlapping edges that yield the greatest wirelength reduction.

3 Steiner Arborescences

Historically, the primary application of rectilinear Steiner minimum trees in VLSI CAD has been in global routing, since older physical design paradigms did not require the modeling of wires in the placement and floorplanning stages. However, the last several generations of technology have made it necessary to model the impact of wiring much earlier in the design process. For example, during placement, physical synthesis, and even floorplanning, we commonly wish to perform static timing analysis in order to evaluate the performance of the current design iteration. To predict achieve this with reasonable accuracy, a model of the wiring of each net must be available. Since

[3]While edge overlapping is a practical technique that reduces wirelength in typical scenarios, there are known pathological pointset instances where edge overlapping over any minimum spanning tree does not yield any wirelength savings whatsoever [61], whereas other Steiner-point inducing methods can still yield substantial savings [41, 60].

blocks and cells may move quite often during these earlier phases of the physical design process, it is imperative to be able to efficiently and accurately estimate wiring delays.

Such interconnect estimation was traditionally formulated as the Steiner problem. However, given the scaling trends in VLSI technology, a Steiner tree often results in inaccurate timing estimates, which may in turn misguide the floorplanning, placement, and physical synthesis design phases. On the other hand, Elmore delay analyses and cost-radius tradeoffs have motivated research into routing constructions that simultaneously optimize interconnect length, source-sink paths, and a quadratic objective that optimizes the sum of source-sink path lengths [62]. In particular, it was discovered that a minimum-cost *rectilinear Steiner arborescence* (RSA) heuristically addresses all of these objectives reasonably well [26], and thus provide highly accurate (as well as efficient) timing estimates.

The Rectilinear Steiner Arborescence (RSA) problem: Given a signal net S in the Manhattan plane with source pin s_0, find a minimum-cost Steiner tree T that spans S, where the pathlengths in the tree T from s_0 to every sink is equal to the corresponding Manhattan distance.

The RSA problem seeks a "minimum-cost shortest-paths Steiner tree" (Figure 5), and is thus a special case of the Steiner version of the BRMRT problem discussed above (where $\epsilon = 0$). The RSA problem originated with early works such as [76] and [28]. Efforts were made to find a polynomial-time optimal arborescence algorithm, resulting in a proliferation of RSA heuristics [26, 44, 67, 85, 101], until it was finally proven that the RSA problem is NP-complete [97].

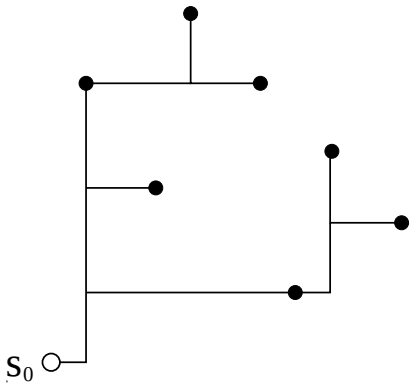

Figure 5: A minimum-cost rectilinear Steiner arborescence.

The first well-known effective rectilinear Steiner arborescence heuristic was proposed in [85]. Given a signal net in the Manhattan plane, the heuristic of [85] maintains a set of points, originally

being all of the pins of the net, and repeatedly merges (i.e., connects) in this set a pair of points/pins whose bounding box is farthest from the source pin. This process terminates when the resulting arborescence spans the entire net. Choosing a new merge point that is dominated by two existing points allows the greatest flexibility for subsequent merges to optimize wirelength while always maintaining the shortest paths property of partial solutions. Figure 6 describes this heuristic more formally, while Figure 7 gives an illustrative execution example. The running time of this method is $O(n \log n)$.

Algorithm: Rectilinear Steiner Arborescence (RSA) [85]

Input: A set of sink vertices $\{v_1, v_2, ...v_n\}$ in the first quadrant
Output: A rectilinear Steiner arborescence rooted at $(0, 0)$

Let Γ be the set of subtrees (Initially $\Gamma = \emptyset$)
For each sink v_i at location (x_i, y_i)
  Insert into Γ a subtree T_i rooted at (x_i, y_i) which contains only v_i
While $|\Gamma| > 1$ **Do**
  Find two subtrees T_j and T_k in Γ such that $x_r + y_r$ is maximum,
    where $x_r = min(x_j, x_k)$ and $y_r = min(y_j, y_k)$
  Create a new subtree T_r by creating a new root at (x_r, y_r)
  Connect the new root to (x_j, y_j) and (x_k, y_k) by a horizontal and/or a vertical edge
  Remove T_j and T_k from Γ
  Insert T_r into Γ
Construct a tree T by connecting $(0, 0)$ to (x_r, y_r) by a horizontal and/or a vertical edge
Return T

Figure 6: The rectilinear Steiner arborescence (RSA) algorithm [85].

Empirical studies indicate that for typical nets, the RSA heuristic of [85], as well as the "A-tree" construction of [26], both yield solutions with average cost within 4% of the optimal RSA cost. On the theoretical side, both of these approaches have been proven to produce rectilinear arborescence trees that are never worse than twice the optimal [85], and pathological examples were found where both methods meet this twice-optimal worst-case bound [62]. Whereas previous approaches typically handle cases where the sinks lie in the first quadrant (with respect to a net's source pin), an extension to all four quadrants, with running time $O(n \log n)$, was given in [27].

The RSA problem was generalized to arbitrary graphs as follows [1]. For an arbitrary weighted graph $G = (V, E)$ and two nodes $u, v \in V$, let $minpath_G(u, v)$ denote the *cost* of a shortest path between u and v in G. The graph Steiner arborescence problem can now be defined.

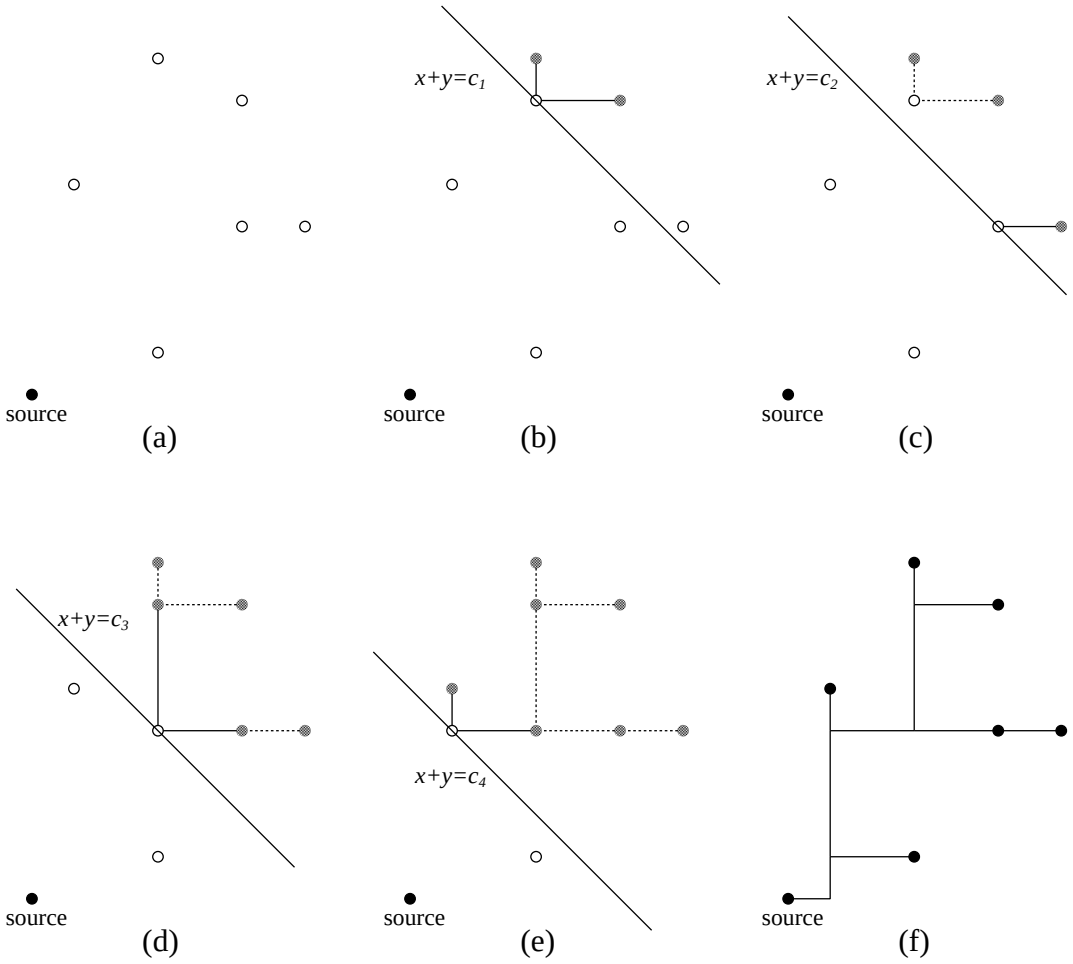

Figure 7: The rectilinear Steiner arborescence (RSA) heuristic of [85]. The solid circle in (a) is the source and the hollow circles are sinks. The first 4 iterations are shown in (b)-(e). At the beginning (a), there are 7 (1-node) subtrees, one per sink, plus the source itself. In (b) a pair of (distant-from-the-source) subtrees is merged to form a new subtree, resulting in 5 remaining subtrees. Trees continue to merge during subsequent iterations, resulting in the final RSA shown in (f).

The Graph Steiner Arborescence (GSA) Problem: Given a weighted graph $G = (V, E)$, and a specified net $N \subseteq V$ with source pin/node $n_0 \in N$ to be interconnected in G, construct a least-cost spanning tree $T = (V', E')$ with $N \subseteq V' \subseteq V$ and $E' \subseteq E$ such that $minpath_T(n_0, n_i) = minpath_G(n_0, n_i)$ for all $n_i \in N$.

As with the rectilinear arborescence problem, the GSA problem is NP-complete [1]. Constructing an arborescence can be viewed as folding or overlapping paths within a shortest paths tree,

so as to induce the maximum wirelength savings while maintaining shortest paths. Indeed, this is the operational principle of the RSA heuristic of [85], among others. In order to generalize this strategy to arbitrary graphs, we define *dominance* in weighted graphs as follows [1].

Definition 3.1 *Given a weighted graph* $G = (V, E)$*, and nodes* $\{n_0, p, s\} \subseteq V$*, we say that* p *dominates* s *if* $\text{minpath}_G(n_0, p) = \text{minpath}_G(n_0, s) + \text{minpath}_G(s, p)$

Thus, a node p dominates a node s if there exists a shortest path from the source n_0 to p that also passes through s (Figure 8(a)). Keeping in mind that the shortest path between a pair of nodes in a graph may not be unique, $MaxDom(p, q)$ is defined as a node in V dominated by both p and q, which maximizes the distance $minpath_G(n_0, MaxDom(p, q))$ to the source node n_0 (Figure 8(b)). The dominated vertex MaxDom is chosen to be as far from the source node as possible, so as to yield the greatest possible wirelength overlap between the two paths, while still maintaining the shortest-paths property with respect to the two target nodes.

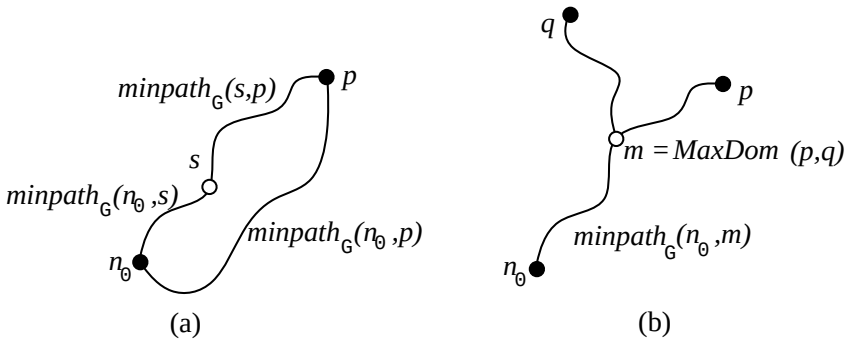

Figure 8: Defining dominance in graphs: (a) Graph node p dominates node s when $minpath_G(n_0, p) = minpath_G(n_0, s) + minpath_G(s, p)$; (b) shows $MaxDom(p, q)$ with respect to p and q. In order to maximize the wirelength savings, we seek the farthest point $m = MaxDom(p, q)$ from the source n_0, where q and p both dominate m.

The above definitions enable the following *Path-Folding Arborescence* (PFA) heuristic [1], as follows. Starting with the set of nodes N that initially contains the net (i.e., the source and all the sinks), we find a pair of nodes p and q in N such that $m = MaxDom(p, q)$ in G is farthest away from the source node n_0 among all such pairs. We then replace p and q in N with m, and iterate until only the source remains in N. The overall graph Steiner arborescence solution is formed by using shortest paths in G to connect each $MaxDom(p, q)$ to p and to q (Figure 9). Empirical

experiments indicate that the PFA method is effective in producing shortest-paths trees with low wirelength (i.e. PFA's average wirelength is close to that of the best existing graph Steiner heuristics) [1]. This observation was reconfirmed in [4], where it was demonstrated that using rectilinear arborescences during physical synthesis only induces an average of 2-4% wirelength penalty over rectilinear Steiner trees, while offering substantial accuracy gains in performance estimation.

Path-Folding Arborescence (PFA) algorithm [1]
Input: Weighted graph $G = (V, E)$ and net $N \subseteq V$ with source $n_0 \in N$ **Output:** A low-cost shortest-paths tree spanning N in G

$M = N$
While $N \neq \{n_0\}$ **Do**
  Find a pair $\{p, q\} \subseteq N$ such that $m = MaxDom(p, q)$
  has maximum $minpath(n_0, m)$ over all $\{p, q\} \subseteq N$
  $N = \{N - \{p, q\}\} \cup \{m\}$
  $M = M \cup \{m\}$
Output the tree formed by connecting each node $p \in M$
  (using a shortest path in G) to the nearest node in M that p dominates

Figure 9: The graph-based Path-Folding Arborescence (PFA) heuristic [1]: M initially holds all the nodes to be spanned, and is then augmented with the *MaxDom* Steiner points found during each iteration.

A different approach to the graph Steiner arborescence problem generalizes the Iterated 1-Steiner (I1S) approach of Kahng and Robins [60, 62] to yield an effective "Iterated Dominance" (IDOM) arborescence methodology for arbitrary weighted graphs [1]. The IDOM heuristic iteratively selects a single Steiner point that minimizes the cost of the *spanning* arborescence over all the sinks and Steiner points selected thus far. The reason that we iterate a *spanning* arborescence construction in order to produce a *Steiner* arborescence tree is that the former is easy to compute[4], while the latter is NP-complete. The IDOM heuristic thus repeatedly (and greedily) finds Steiner candidates that reduce the overall spanning arborescence cost, and includes them into the growing set of Steiner nodes (Figure 10).

In order to achieve an improved runtime for the IDOM approach, Alexander and Robins [1]

[4]Recall that a node p dominates a node s if there exists a shortest path from the root to p passing through s. An optimal *spanning* arborescences can be computed efficiently by using a shortest path to connect each sink to the closest sink/source that it dominates, and then computing Dijkstra's [29] shortest paths tree over the graph formed by the union of these paths.

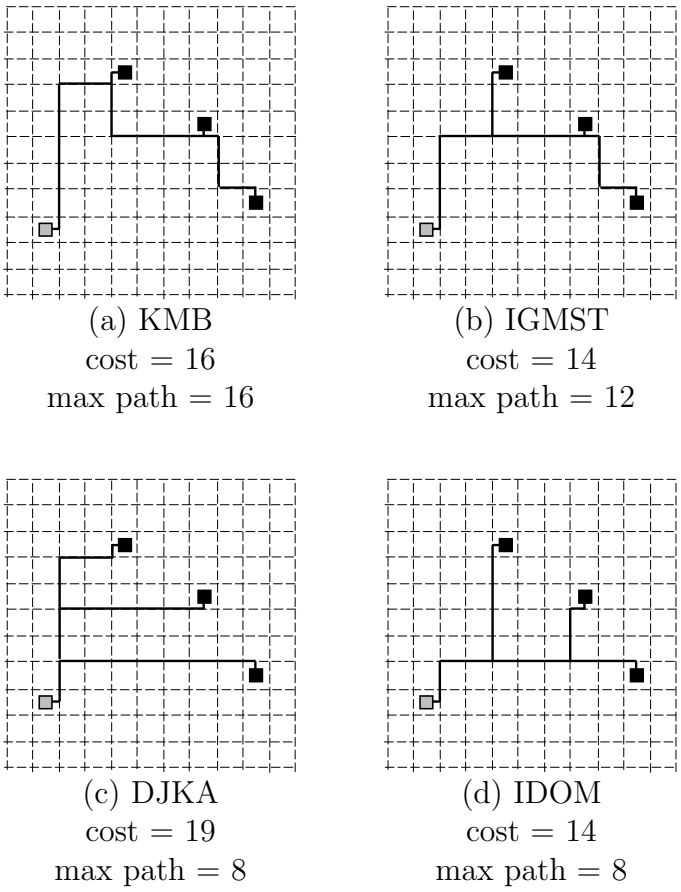

Figure 10: Four routing solutions for the same 4-pin net (the signal source is the gray-shaded square, and the solid squares are sinks): (a) the solution produced by the KMB graph Steiner heuristic of [66]; (b) the optimal Steiner tree, which is also the solution produced by the Graph Iterated 1-Steiner algorithm of [41, 62]; (c) Dijkstra's shortest paths tree [29]); (d) the optimal Steiner arborescence, which is also the solution produced by the IDOM algorithm of [1]. Note that the IDOM solution in (d) is optimal in terms of both total wirelength as well as maximum pathlength (although this double-optimal outcome is unusual).

defined the DOM heuristic, which is a restricted version of the PFA heuristic (Figure 9), except where $MaxDom(p, q)$ is selected only from N instead of allowed to be an arbitrary node in V. This substantially speeds up the search for $MaxDom(p, q)$ at each iteration, since N is typically much smaller than V. The DOM subroutine constructs an arborescence by using a shortest path to connect each sink in N to the closest sink/source in N that it dominates, and then computes a shortest paths tree over the graph formed by the union of these paths.

Given a set of Steiner candidate node $S \subseteq V - N$, the *cost savings* of S with respect to DOM is defined as $\Delta\text{DOM}(G, N, S) = \text{cost}(\text{DOM}(G, N)) - \text{cost}(\text{DOM}(G, N \cup S))$. The IDOM approach starts with an initially empty set of Steiner candidates $S = \emptyset$. It then finds a node $t \in V - N$ which maximizes $\Delta\text{DOM}(G, N, S \cup \{t\}) > 0$, and repeats this procedure with $S \leftarrow S \cup \{t\}$. The wirelength required by DOM to span $N \cup S$ will decreases with each added node t, and the overall construction terminates when there is no $t \in V - (N \cup S)$ such that $\Delta\text{DOM}(G, N, S \cup \{t\}) > 0$. The final overall solution is $\text{DOM}(G, N \cup S)$. This method is described in Figure 11, and a sample execution is gven in Figure 12.

Iterated Dominance (IDOM) Algorithm [1]
Input: A weighted graph $G = (V, E)$, a net $N \subseteq V$ with $n_0 \in N$ **Output:** A low-cost arborescence $T' = (V', E')$ spanning N, where $N \subseteq V' \subseteq V$ and $E' \subseteq E$
$S = \emptyset$ **Do Forever** $T = \{t \in V - N \mid \Delta\text{DOM}(G, N, S \cup \{t\}) > 0\}$ **If** $T = \emptyset$ **Then Return** $\text{DOM}(G, N \cup S)$ **Find** $t \in T$ with maximum $\Delta\text{DOM}(G, N, S \cup \{t\})$ $S = S \cup \{t\}$

Figure 11: The Iterated Dominance (IDOM) algorithm [1] for producing arborescences in arbitrary weighted graphs.

The IDOM approach is a general template for producing arborescences for designated subgraphs (i.e., nets) in arbitrary weighted graphs (i.e., underlying routing grids) [1]. Moreover, the IDOM heuristic escapes the known twice-optimal worst-case examples of previous arborescence heuristics, both in the rectilinear plane as well as in arbitrary weighted graphs.[5] The IDOM approach outperforms the previous heuristics on empirical benchmarks [1], including in FPGA routing, which is inherently a graph-based regime. Subsequent graph arborescence algorithms, including fast polynomial-time heuristics as well as exponential-time optimal algorithms were introduced in [21].

[5]There exist very rare worst-case graphs which force IDOM to produce a tree with cost logarithmic factor times optimal, matching the best known non-approximability results for the graph Steiner arborescence problem [1].

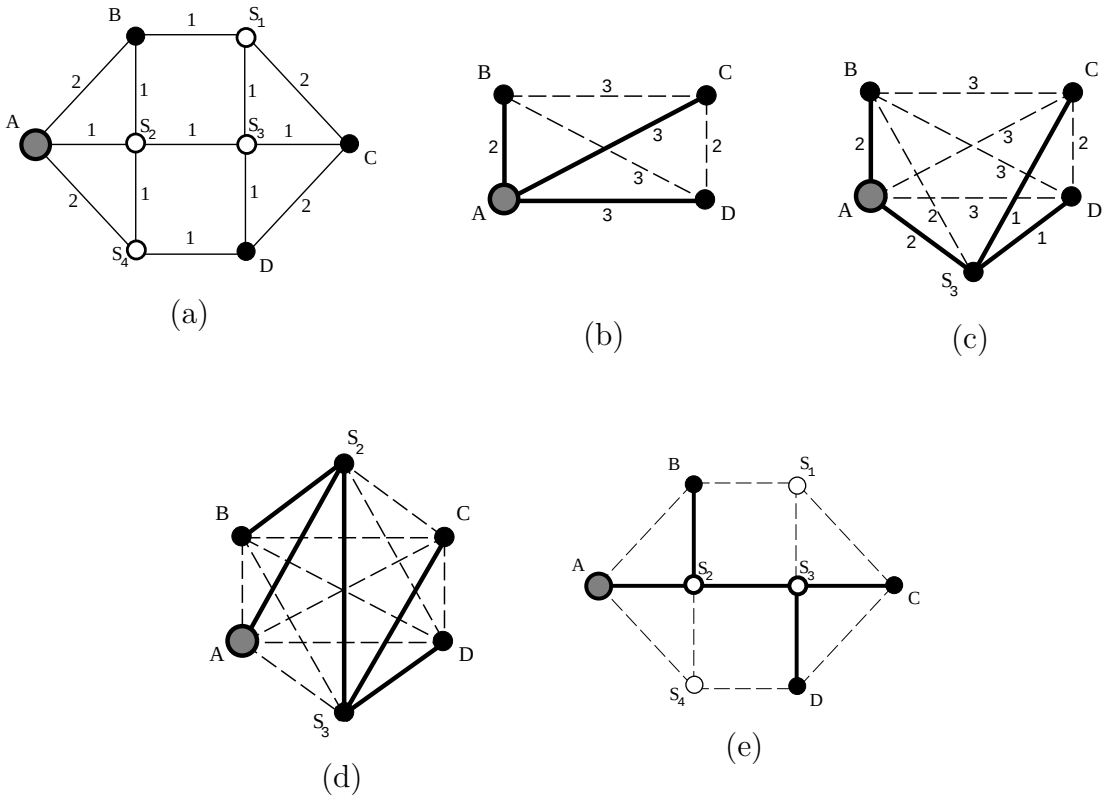

Figure 12: An execution example for the IDOM algorithm: (a) a Graph Steiner Arborescence (GSA) problem instance with source node A (gray), sink nodes $\{B, C, D\}$ (solid), and graph edge weights shown; (b) initial DOM solution over the contracted pathlengths distance graph (over the net $N\{A, B, C, D\}$), having cost 8; (c) Steiner candidate S_3 produces a savings of ΔDOM= 2, which reduces the overall tree cost from 8 to 6; thus S_3 is retained as a Steiner point; (d) Steiner candidate S_2 is the final Steiner point with positive ΔDOM, and further reduces the solution cost from 6 to 5; (e) the final IDOM solution (having cost 5), with paths re-expanded relative to the original input graph.

4 Elmore Delay -Based Routing Constructions

Objectives such as minimum tree cost, bounded radius, cost-radius tradeoffs, and even arborescences were all motivated by analyses of the Elmore delay approximation [34, 72, 89, 102]. However, these objectives are merely abstractions which do not directly optimize delay. This section describes approaches that optimize Elmore delay *directly* while synthesizing a routing tree.

The earliest Elmore-based routing approach is the *Elmore routing tree* (ERT) spanning construction of Boese, Kahng and Robins [10, 62, 86] (Figure 13). Similarly to Prim's MST algorithm [81], the ERT heuristic starts with a tree $T = (V, E)$ initially containing only the source s_0, and then repeatedly finds a terminal $s_i \in V$ and a sink $s_j \in S - V$ so that adding edge (s_i, s_j) to T minimizes the maximum Elmore delay to any sink in the growing tree. The greedy approach implicit in the ERT algorithm easily generalizes to any delay model by using the corresponding delay estimator in the inner loop of Figure 13. For example, [106] proposed the use of a Two-Pole simulator within a similar greedy construction, and [98] used this strategy for MCM routing under a second-order delay model.

Elmore Routing Tree (ERT) Algorithm [10]
Input: signal net S with source $s_0 \in S$ **Output:** routing tree T over S
1. $T = (V, E) = (\{s_0\}, \emptyset)$ 2. **While** $\|V\| < \|S\|$ **do** 3. **Find** $s_i \in V$ and $s_j \in S - V$ that minimize the maximum Elmore delay from s_0 to any sink in the tree $(V \cup \{s_j\}, E \cup \{(s_i, s_j)\})$ 4. $V = V \cup \{s_j\}$ 5. $E = E \cup \{(s_i, s_j)\}$ 6. **Output** resulting spanning tree $T = (V, E)$

Figure 13: The ERT Algorithm [10] directly uses the Elmore delay formula in a greedy routing tree construction.

The ERT algorithm template can produce a timing-driven Steiner Elmore routing tree (SERT) when new sinks are allowed to connect *anywhere* along an edge in the growing tree, inducing a Steiner node at that connection point [9]. Following the ERT approach, the SERT variant greedily minimizes the maximum source-to-sink Elmore delay at each tree-growing step. To allow additional optimization leeway, embeddings of L-shaped edges can remains indeterminate (within their bounding boxes) for as long as possible during the execution. The SERT variant produces a /em Steiner topology with low source-to-sink Elmore delays. Figure 14 depicts the execution of the SERT heuristic on a sample 8-sink net.

In performance-driven layout, timing-critical paths are determined using timing analysis, and then cells along these paths are placed closer together [30, 43, 56, 70, 73, 99]. Timing analysis thus iteratively drives changes within the placement as well as global routing phases. In order to avoid

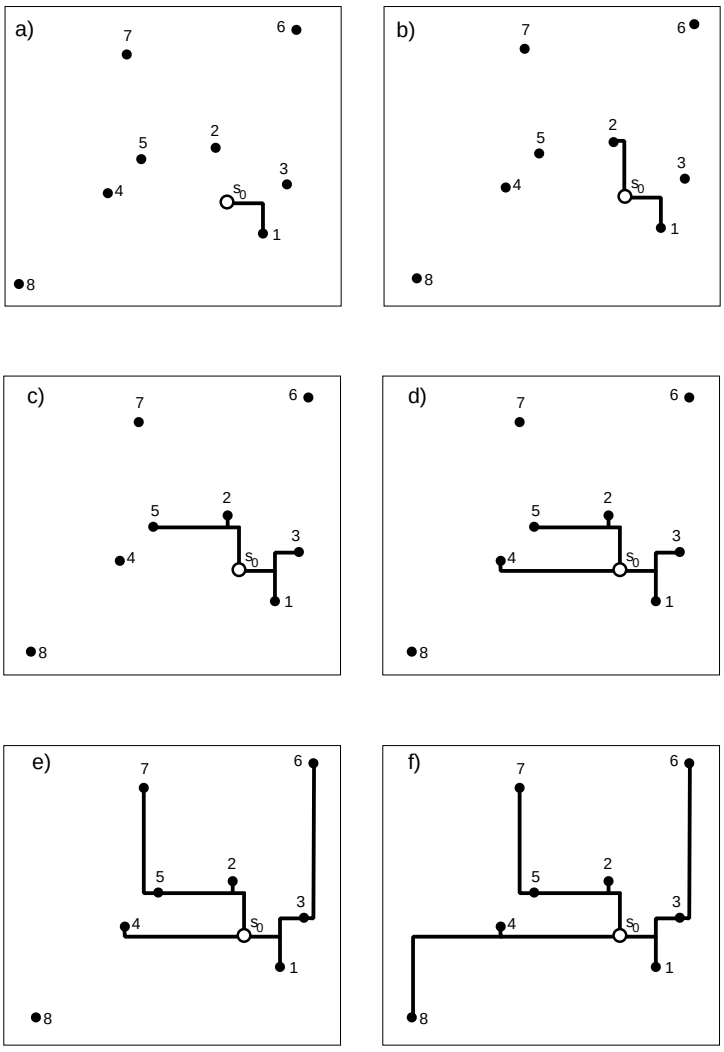

Figure 14: Execution of the SERT Steiner tree construction [10] for an 8-sink net. The source terminal is labeled 1, and the remaining sinks are numbered in the order of their distance from the source.

the "placement-routing mismatch" where inherently net-dependent methods fail to exploit the critical-path information available during iterative performance-driven layout, Boese, Kahng and Robins [10] proposed formulations that extend the basic (S)ERT scheme to accommodate critical sinks. They proved the NP-completeness of the Critical Sink Routing Tree problem (CSRT) problem [8], and provided efficient heuristics that combine Steiner construction, delay estimation, and global slack removal [10].

To address the CSRT formulation, Boese, Kahng and Robins generalized their SERT method to produce a *"Steiner Elmore Routing Tree with identified Critical sink"* (SERT-C) [10]. The SERT-

C heuristic begins with a tree containing a direct connection (s_0, s_c) between the source and the specified critical sink, and then grows the routing tree around it while minimizing the Elmore delay (or an alternate delay model) from the source to the critical sink (Figure 15). Figure 16 illustrates the execution of SERT-C for various choices of the critical sink (using the same 8-sink signal net as in Figure 14). The SERT-C algorithm can be implemented to run within time $O(n^2 \log n)$ for n-pin nets. Similarly to the ERT and SERT approaches, SERT-C's direct optimization of the Elmore delay allows considerable flexibility with respect to the underlying technology parameters, delay model, and specific input instance.

The methods described above easily extend to higher dimensions and alternate metrics and geometries, including to non-Manhattan interconnect architectures such as "preferred direction" routing and λ-geometries [14, 15, 16, 64, 68, 77, 92, 100, 104]. The Elmore-based routing tree construction methods of [9] influenced followup works on performance-driven routing trees, addressing additional issues such as buffer insertion, wirelength estimation, alternative delay models, timing constraints, and antenna effects [2, 39, 46, 49, 51, 78, 103].

SERT-C Algorithm [10]

Input: A signal net S with source $s_0 \in S$ and critical sink $s_c \in S$
Output: A critical-sink routing tree T over S

1. $T = (V, E) = (\{s_0, s_c\}, \{(s_0, s_c)\})$
2. **While** $|V| < |S|$ **do**
3. Find $s_j \in S - V$ and $(v, v') \in E$ such that connecting s_j to a point x on (v, v') minimizes the Elmore delay to s_c in the tree $(V \cup \{s_j, x\}, E \cup \{(v, x), (v', x), (x, s_j)\} - \{(v, v')\})$
4. $V = V \cup \{s_j, x\}$
5. $E = E \cup \{(v, x), (v', x), (x, s_j)\} - \{(v, v')\}$
6. **Output** resulting Steiner tree $T = (V, E)$

Figure 15: The SERT-C algorithm [10] directly incorporates the Elmore delay formula into a greedy critical-sink routing tree construction.

5 Non-Hanan Interconnect Synthesis

In older (pre 1990's) VLSI regimes, where interconnect delay was mostly capacitive, resistance-related delay components were negligible, and the objective of delay optimization therefore coincided with minimizing the total interconnect length. However, as discussed above, in more

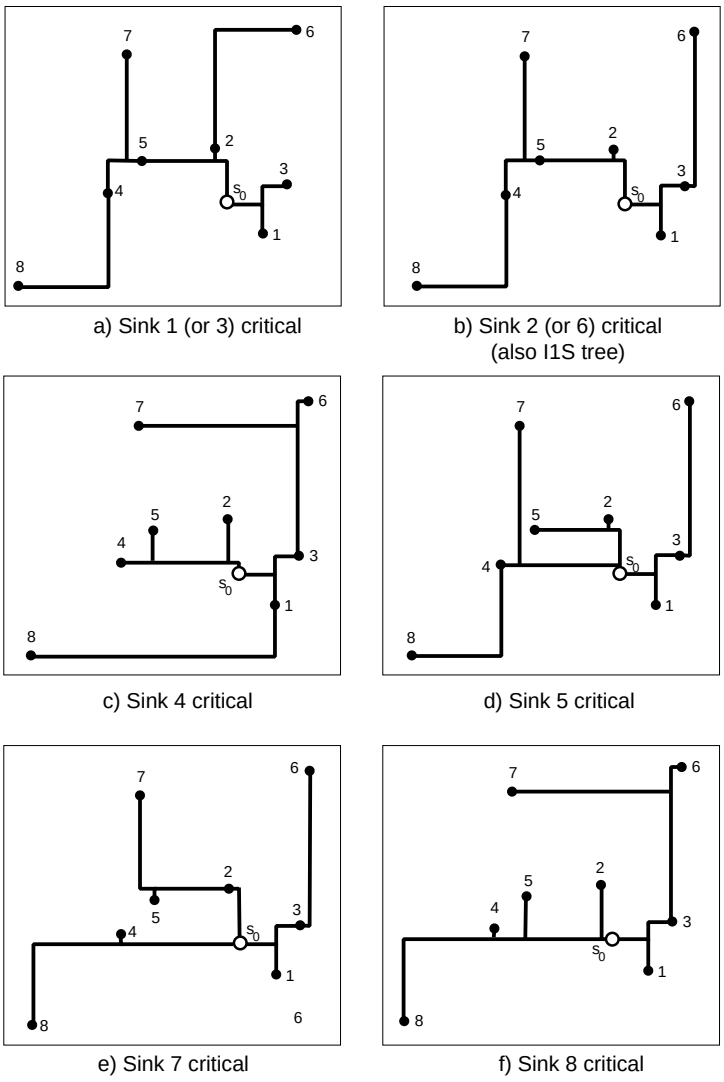

Figure 16: The SERT-C critical sink routing tree construction for an 8-sink net, showing solutions for different choices of critical sink. The tree constructed when the source s_c is node 2 or node 6 is also the I1S solution, and the tree constructed when s_c is node 7 is also the generic SERT result.

modern VLSI technologies interconnect resistance began to dominate circuit performance, causing optimized performance-driven interconnect to resemble minimum wirelength topologies less and less. Another modern deviation from classical constructions involves the *Hanan grid*, which is obtained by drawing horizontal and vertical lines through all the pins of a given net [42] (Figure 17). Hanan's theorem states that there always exists a rectilinear minimum Steiner tree embedded in the Hanan grid [42, 105].

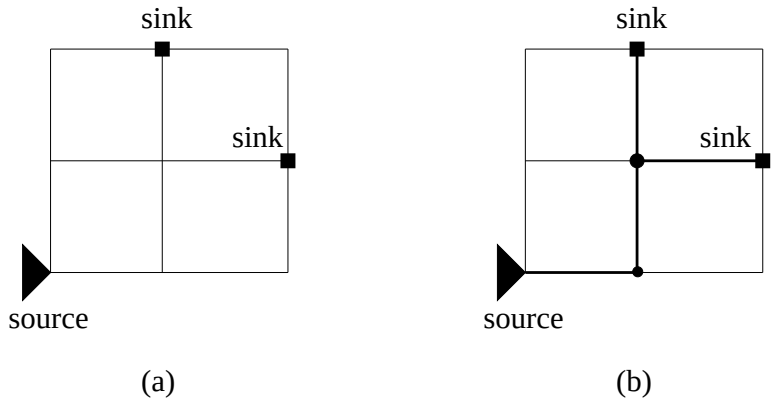

Figure 17: An example of (a) a Hanan grid induced by a net, and (b) a minimum Steiner tree embedded in the Hanan grid.

Boese, Kahng and Robins [9] proved that only points from the Hanan grid need be considered in minimizing the weighted sum of critical sink delays. On the other hand, for the "minmax" objective of minimizing the maximum sink delay, better routing solutions are possible when considering points that lie off the Hanan grid [9]. For example, in Figure 18 a non-Hanan point is required to minimize the maximum source-sink delay during tree construction. Such examples illustrate that the timing requirements at different sinks are often mutually-competing, and therefore good approaches must consider all the sinks simultaneously, and utilize every available degree of optimization to produce improved timing-driven interconnect solutions. In particular, the observation that restricting Steiner nodes to be Hanan grid points is suboptimal motivates the problem of non-Hanan interconnect synthesis.

Below we outline a general interconnect synthesis methodology that uses non-Hanan optimization to yield better-performing interconnect topologies [49]. In particular, we address two problem variants: (a) the minmax problem of minimizing the maximum source-to-sink delay, and (b) the critical sink problem that seeks a specified delay at each sink. The later problem can be transformed into a variant of the former problem, and optimal solutions may lie off the Hanan grid in either variant. We next describe a procedure for constructing low-cost routing trees that satisfy prescribed delay constraints at each sink.

Define the delay violation at each sink as its delay minus its required arrival time (RAT). A positive delay violation value therefore implies that the corresponding delay constraint was not met. On the other hand, a negative delay violation value indicates timing slack, and enables the

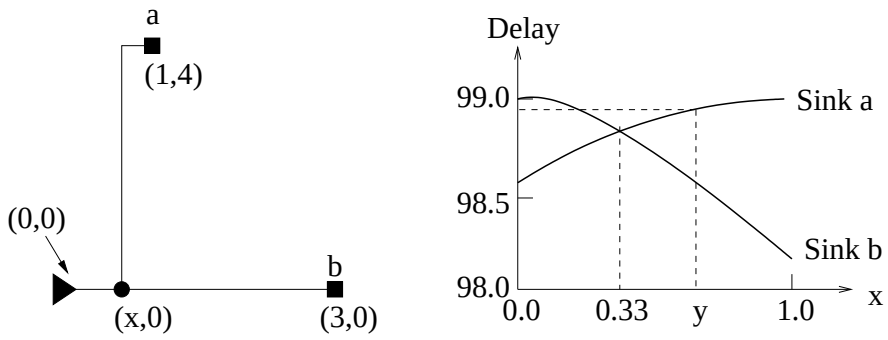

Figure 18: An example illustrating the efficacy of non-Hanan routing. We assume unit resistance and unit capacitance per unit wirelength. The driver has a source resistance of 6, and the sinks a and b have load capacitances of 1 and 4.5 units, respectively. The variation in the Elmore delay at each sink as the Steiner point x is moved from $(0, 0)$ to $(1, 0)$ is plotted on the right. The maximum sink delay for the tree is minimized at the non-Hanan point $x = 0.33$. The analyses of [9] can be used to show that a Steiner point to the right of $(1, 0)$ is suboptimal. Even more dramatic discrepancies between Hanan and non-Hanan routings are achievable in larger examples.

possibility of further optimizing the routing tree cost by reducing the timing slacks. This tradeoff motivates the Maximum delay Violation Elmore Routing Tree (MVERT) problem formulation, as follows.

The Maximum delay Violation Elmore Routing Tree (MVERT) problem: Given a signal net N with source v_0 and a set of sinks $V_{sink} = \{v_1, v_2, ..., v_n\}$, construct a Steiner routing tree with minimum total wirelength, so that the delay violation at each sink is non-positive (i.e., meets the corresponding timing constraints).

Since the routing tree topology is no longer restricted to the Hanan grid, the set of candidate Steiner points is unbounded (as opposed to corresponding to the set of Hanan points as in classical formulations). We must therefore find an efficient method for identifying the best (non-Hanan) Steiner points that produce a good routing tree. We now describe a framework that utilizes properties of the delay function in order to develop a simple and efficient algorithm to address this challenge.

Following [9], define a maximal segment to be a set of contiguous edges, being either all vertical or all horizontal. The work of [9] shows that the Elmore delay at each sink is a concave function

with respect to the location of a Steiner node moving along a maximal segment. This property also holds for a *soft* edge which is an edge connecting two nodes $v_i, v_j \in V$, $v_i = (x_i, y_i)$, $v_j = (x_j, y_j)$, such that: (1) $x_i \neq x_j$ and $y_i \neq y_j$; and (2) the precise edge route between v_i and v_j is not yet determined. The length l_{ij} of edge (v_i, v_j) is the Manhattan distance $|x_i - x_j| + |y_i - y_j|$. The use of soft edges avoids premature commitment to a specific geometric embedding of a wire in rectilinear space, which enables further wirelength optimization later on [49].

For a general routing tree topology (Figure 19), consider the process of determining an optimal connection between a new node v_k to be attached to an existing edge e_{ij}. The dashed lines in Figure 19 denote other nodes and edges of the existing routing tree, and CC represents the closest connection point between node v_k and edge e_{ij}. It can be shown that any connection downstream of CC cannot yield an optimal solution [9]. Specifically, we seek an optimal connection point within the bounding box defined by v_i and CC. Suppose we connect v_k to e_{ij} at point $v' = (x', y')$. Let $z = |x' - x_i| + |y' - y_i|$ be the Manhattan distance from v' to v_i. For convenience, we overload the term CC to also denote its Manhattan distance to v_i.

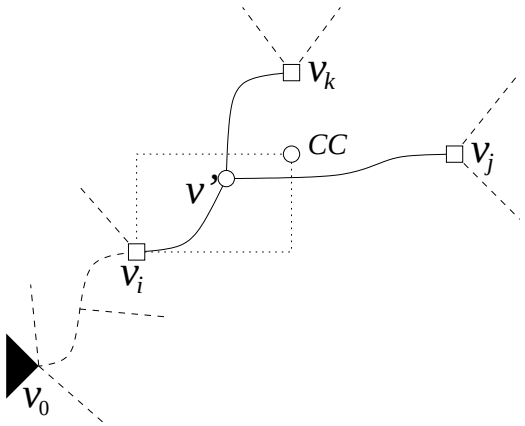

Figure 19: A general routing topology where a new node v_k is to be connected to an existing edge e_{ij}.

Following the work of [9], a delay function with respect to the connection locations for soft edges under the Elmore delay model can be derived as follows. If a node is not downstream from node v_i, its Elmore delay from the source is:

$$f_1 = R_d(C_t - cz) + \lambda_0 + \lambda_1(l_{ik} - z) \tag{1}$$

where λ_0 and λ_1 are constants, and C_t denotes the total capacitive load that would be seen from the last stage of the driver if v_k was connected to v_i. The Elmore delay from v_i to v' is given by:

$$f' = rcz(\frac{z}{2} + l_{ij} - z + l_{ik} - z) + rz(C_j + C_k). \tag{2}$$

The delay from v' to any node in the subtree T_j rooted at v_j, can be calculated as:

$$f_2 = r(l_{ij} - z)(\frac{c(l_{ij} - z)}{2} + C_j) + \lambda_2. \tag{3}$$

Similarly, the delay from v' to any node in subtree T_k is:

$$f_3 = r(l_{ik} - z)(\frac{c(l_{ik} - z)}{2} + C_k) + \lambda_3. \tag{4}$$

where λ_2 and λ_3 are constants. The Elmore delay of a sink in T_j is given by the sum of f_1, f' and f_2. The Elmore delay of a sink in T_k is the sum of f_1, f' and f_3. The Elmore delay of a sink not downstream of v_i, is simply f_1. In all these cases, the delay is either a linear or a quadratic function of the Manhattan distance z with non-positive coefficient for the second-order term. We can therefore conclude that the delay for any sink is a concave function with respect to z, as follows.

Theorem 5.1 *Under the Elmore delay model, the delay at any sink in the routing tree is a concave function with respect to the Manhattan distance. [49].* □

Rewriting the constraints on the routing tree into the form $t(v_i) - q(v_i) \leq 0$ for all sinks $v_i \in V_{sink}$, we see that the maximum delay violation must always be non-positive. Since by Theorem 5.1, each of the $t(v_i)$'s is a concave function of the connection point z, and since any concave function shifted by a constant is a concave function, this implies that we must find a reconnection point z such that the maximum of the set of concave functions is non-positive. This is pictorially shown in Figure 20 for a net with four sinks u, v, w, and y, all of which have the same timing specification q. The maximum violation function (depicted by a thicker line) is a piecewise concave function composed of three concave *pieces*. Note that the graph shows that sink u is never critical in this case, for any value of z. The delay violation at each sink as a function of

z is a concave function and the objective is to find a value of z closest to CC (corresponding to a minimal increase in the net length) that satisfies all the timing constraints. In Figure 20, this point is found to be z^*, and in general this point will be a non-Hanan point.

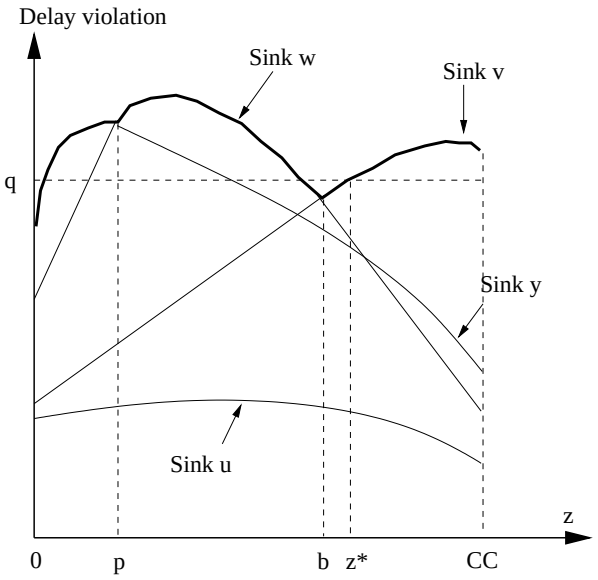

Figure 20: Finding the optimal value of z that satisfies all the timing constraints.

In searching for the point z^*, we observe that it is possible to perform a search on the value of z from 0 to CC, while taking advantage of the fact that the value on each concave piece is minimized at its intersection with the concave piece on either side (if such a piece exists), or at 0 or CC otherwise. In Figure 20, this translates to the fact that for the minmax problem, the only candidate solutions are 0, p, b and CC. This permits a dramatic reduction of the search space from the infinity of possible intermediate points between 0 and CC.

For the problem of meeting the timing constraints at each sink, several pruning strategies are possible during the search. Consider a binary search on a concave segment with end points x_1 and x_2 ($x_1 < x_2$) where the function values are $f(x_1)$ and $f(x_2)$, respectively. If $f(x_1) < T_{spec} < f(x_2)$ and $T_{spec} < f(\frac{x_1+x_2}{2})$, as illustrated in Figure 21, then the search can completely eliminate the interval $[\frac{x_1+x_2}{2}, x_2]$. This follows from the fact that any concave function over an interval is concave over any continuous subinterval. By a symmetric argument, if $T_{spec} \geq f(\frac{x_1+x_2}{2})$, then the search can be confined to the interval $[\frac{x_1+x_2}{2}, x_2]$.

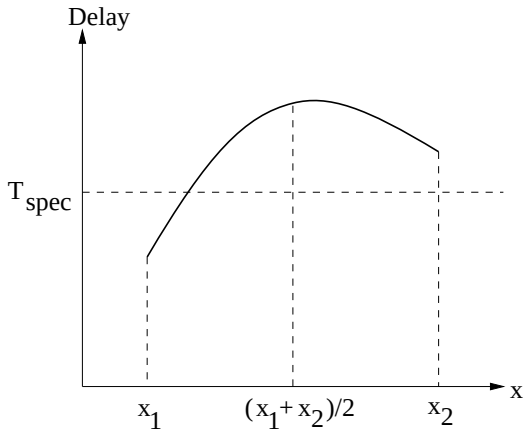

Figure 21: Using piecewise concavity to speed up the optimization procedure.

Optimal Connection Algorithm [49]
Input: Subtree T_k rooted at sink v_k, Partial routing tree $T \backslash T_k$, edge $e_{ij} \in T \backslash T_k$ **Output:** Optimal connection between v_k and e_{ij}
1. Tentatively join v_k to CC, $\Delta_{rit} \leftarrow \Delta_{max}$, $CS_{rit} \leftarrow$ sink with $\Delta_{max}, S_{rit} \leftarrow (CC, \Delta_{rit}, CS_{rit})$ 2. If $\Delta_{rit} \leq 0$, **Return** CC 3. Tentatively join v_k to v_i $CS_{lft} \leftarrow$ sink with $\Delta_{max}, S_{lft} \leftarrow (v_i, \Delta_{max}, CS_{lft})$ 4. **Return** $Search(S_{lft}, S_{rit})$
Function: $Search(S_{lft}, S_{rit})$
5. If $\Delta_{rit} \leq 0$, **Return** S_{rit} 6. If ($\Delta_{lft} > 0$ and $CS_{lft} == CS_{rit}$) or $dist(v_{lft}, v_{rit}) <$ resolution 7. If $\Delta_{lft} < \Delta_{rit}$, **Return** S_{lft} 8. Else **Return** S_{rit} 9. $v_{mid} \leftarrow ((x_{lft} + x_{rit})/2, (y_{lft} + y_{rit})/2)$ 10. Join v_k to e_{ij} at $v_{mid}, \Delta_{mid} \leftarrow \Delta_{max}$ $CS_{mid} \leftarrow$ sink with $\Delta_{max}, S_{mid} \leftarrow (v_{mid}, \Delta_{mid}, CS_{mid})$ 11. If $\Delta_{mid} \leq 0$, **Return** $Search(S_{mid}, S_{rit})$ 12. $S_r \leftarrow Search(S_{mid}, S_{rit})$ 13. If $\Delta_r \leq 0$, **Return** S_r 14. $S_l \leftarrow Search(S_{lft}, S_{mid})$ 15. If $\Delta_l < \Delta_r$, **Return** S_l 16. Else **Return** S_r

Figure 22: Algorithm for finding an optimal connection point between a sink and an edge [49].

The pseudocode corresponding to this search is shown in Figure 22. The routing tree without subtree T_k is denoted by $T \backslash T_k$. The efficiency of the search can be greatly enhanced by taking advantage of the piecewise-concave nature of the delay function. The search for z^* occurs between 0 and CC in a binary search fashion, and begins at CC. If the value of the delay violation at CC is negative, then we are done; otherwise we need to test the delay violation at 0. We use CS to represent the critical sink that has the maximum delay violation $\Delta_{max} = \max\{t(v_i) - q(v_i), \forall v_i \in V_{sink}\}$. If Δ_{max} is positive at both 0 and CC, and the critical sink at 0 is the same as at CC, then there is no solution satisfying the timing constraints. In this case, we choose the solution that yields the least delay violation between 0 and CC.

A more complicated situation occurs when Δ_{max} at 0 is negative, or Δ_{max} is positive at 0 but the corresponding critical sink is different from that at CC. Then, the search proceeds as a quasi-binary search, as encoded in the function $Search(S_{lft}, S_{rit})$ in Figure 22. The notation S indicates a solution which is a triple of the form (connection node, Δ_{max}, critical sink), and S_{lft} and S_{rit} denote the solutions at the left and right end of the search interval, respectively. If the size of the interval is less than a user specified resolution, then the search terminates (lines 6-10 in Figure 22). On the other hand, if the connection at the middle point of the interval yields a non-negative Δ_{max}, then the search continues only on the right half of the interval (line 11 in Figure 22); otherwise, the left half of the interval may be searched as well (lines 12-16 in Figure 22).

The MVERT algorithm [49] operates in two phases: (I) The initial tree construction phase, where an initial tree is heuristically built to minimize delay; and (II) The cost-improvement phase, where the tree is iteratively refined to reduce its cost while ensuring that it still meets all the timing constraints. The tree construction in Phase I is similar to the SERT construction procedure proposed in [9] (described above). Recall that the essential idea of the SERT method is based on greedily building a Steiner tree using a Prim-like method. Starting with a trivial tree T consisting of only the source v_0, the tree is iteratively built by joining a sink v_k outside the tree to an edge (or the source) already in the tree, so as to yield a resulting new tree with minimum Elmore delay. This process iterates until all the sinks are included in the tree.

The initial tree construction procedure above considers only Hanan grid points as candidate Steiner points. It therefore attempts to connect each point to either the closest connection (CC), the upstream end of a tree edge, or directly to the source node. If the delay associated with a CC

connection is larger than the delay associated with a connection to the upstream edge endpoint, then the algorithm will not choose the connection at CC. However, due to the interactions between paths, MVERT solutions may lie at different (and possibly non-Hanan) points, and a connection to the upstream end of an edge may result in a larger net length than is necessary. We therefore examine the tree constructed in Phase 1 and move node connections from the upstream end of an edge towards CC in order to reduce the tree length while still satisfying all the timing constraints. The idea is illustrated in the example of Figure 18 for the constraint of 98.8 units, where a connection to $(y, 0)$ is preferable over a connection to $(0.33, 0)$.

Non-Hanan Optimization Algorithm [49]
Input: Routing tree $T(V, E)$ **Output:** Optimized routing tree T'
1. $T' = T$ 2. Sort all the sinks in descending order of distance to source 3. **For** each $v_k \in V_{sink}$ 4. Disjoin v_k and its subtree T_k from T 5. **For** each edge $e_{ij} \in T \backslash T_k$ 6. Reconnect v_k to e_{ij} at FindOptimalConnection$(T_k, T \backslash T_k, e_{ij})$ 7. If $\exists$ improvement compared to T', Then $T' = T$ 8. **Return** T'

Figure 23: The non-Hanan optimization algorithm [49].

This non-Hanan interconnect synthesis algorithm (shown in Figure 23) can be implemented as follows [49]. We first sort all the sinks in descending order of distance from the source. We then disconnect each sink v_k (along with its downstream subtree T_k) and reconnect it back to the tree at a better reconnection point, if possible (as determined by the subroutine of Figure 22). Thus at each iteration we choose an edge that provides the largest wirelength improvement while still still respecting the timing constraints. The computational complexity of the MVERT algorithm is $O(n^4)$, where n is the number of sinks. The experimental results in [47] show that non-Hanan optimization can in some instances provide considerable wirelength reduction as compared to other timing-driven routing methods.

6 Wiresizing

The fundamental tradeoffs between interconnect capacitance and resistance in modern VLSI technology suggests that in order to maximize performance, some wire segments should be made wider than others. This motivates the technique of wiresizing, where every wire segment may have a different width, independently of all the other wires. This degree of freedom afforded by wiresizing can be leveraged throughout every phase of the performance-driven physical layout process. Historically, while early works wiresized mainly clock trees [37, 38, 83, 107] and power distribution networks [33], the wiresizing of general interconnect became viable in the early 1990's [25, 26, 45, 91] due to the confluence of VLSI scaling trends and algorithmic advances. Wiresizing considerations can be easily incorporated into all the routing constructions discussed above [62], and can even drive the routing process itself [45], as well as other layout phases higher in the design hierarchy. A more detailed discussion of wiresizing techniques may be found elsewhere in this book.

7 Non-Tree Routing

Historically, routing methodologies implicitly assumed that interconnections must have tree topologies. In retrospect, this was a natural constraint because a tree achieves electrical connectivity using minimum wire, and the VLSI technology trends of the 1980's were heavily skewed toward wirelength and area minimization as the primary objective. However, as feature sizes shrank dramatically and interconnect delays began to dominate circuit performance, researchers began to investigate "non-tree" (i.e., general graphs) routing topologies. Aside from improving performance, non-tree routing topologies offer other advantages, including the management of signal reflections, increased reliability, and reduced skew in sink delays. Thus non-tree topologies were used for power/ground distribution, where general graph topologies enhance reliability by lowering current densities and electromigration damage [33, 35, 36], as well as for clock distribution, where non-tree topologies can reduce skew and minimize the impact of manufacturing variation [71].

Adding extra wires to an existing routing tree can improve certain source-sink delays. While additional wires will always increase the total tree capacitance, the creation of multiple source-sink paths can substantially lower certain internode resistances. Thus, as VLSI interconnect becomes thinner and more resistive, non-tree routing topologies become increasingly attractive. McCoy

and Robins [75] have studied the following *Optimal Routing Graph* (ORG) problem, which is a generalization of some of the routing problems discussed above.

The Optimal Routing Graph (ORG) Problem: Given a signal net $S = \{s_0, s_1, ..., s_n\}$ with source s_0, find a set N of Steiner points and routing graph $G = (S \cup N, E)$ such that G spans S and minimizes $t(G) = \max_{i=1}^{n} t(s_i)$.

The ORG problem extends to critical-sink formulations as well as lumped RC and Elmore delay models, which can be computed efficiently for general RC graph topologies [13, 74]. The ORG problem is addressed algorithmically in [75] by starting with a reasonable initial topology (e.g., a heuristic Steiner or spanning tree), and greedily adding new edges to this topology so as to keep improving the specified delay objectives in the growing routing graph. Steiner points may also be introduced during this process to further optimize both delay and wirelength. Using a fast delay estimator to drive this process yields an efficient technique for synthesizing non-tree routing topologies with significantly improved performance characteristics (in terms of skew as well as delay), as compared with the corresponding initial trees [62, 75]. Non-tree routing topologies can also be combined with wiresizing optimizations, as discussed above. More recently, non-tree routings were used for manufacturign yield improvement [58] and robust performance [52].

8 Discussion and Future Research Directions

Given the numerous existing algorithms for performance-driven Steiner tree construction, CAD practitioners are often faced with the question of which algorithm to choose for particular applications. In [4], a comparative study is performed for several Steiner tree algorithms [3, 9, 11, 21, 84]. One important result from [4] is that the wirelength of MRSA (Minimum Rectilinear Steiner Arborescence) is not prohibitively large, even though MRSA constructions provide shortest paths from the source to all sinks. Experiments with several industrial designs show that the average wirelengths of heuristic rectilinear Steiner arborescences are only around 2-4% larger than those of rectilinear Steiner minimum trees. Arborescence constructions (e.g., AHHK-based Steiner trees with $c = 1$) are therefore a good option for acheiving minimum tree radii with relatively small wirelength overhead.

As ultra-deep-submicron VLSI technology continues to evolve, new efficiently-computable models are needed to accurately capture the relationships and tradeoffs between high-performance

routing and actual delays, parasitics, noise, signal integrity, reliability, power, manufacturability, and yield. The techniques described in this chapter can be generalized to alternate metrics, geometries, and novel interconnect architectures such as "preferred direction" routing and λ-geometries. As VLSI engineering tolerances shrink, issues such as buffer insertion, wirelength estimation, and antenna effects will have to be revisited. In particular, extensive application of buffers [94] for performance improvement may drastically alter the landscape for interconnect topology construction. When buffers are inserted, the fanout size of subtrees between buffers are usually smaller than that of unbuffered nets. Moreover, the construction of global topology connecting the subtrees should be aware of the concerns in buffering algorithms [2, 48]. As always, tighter and more effective integration between timing-driven routing and other design phases will enable additional optimizations of various combinations of objectives and criteria. Finally, when feature sizes become small enough, entirely new issues such as quantum effects will have to be considered during interconnect synthesis, as well as elsewhere in the design process.

References

[1] Alexander, M. J., and Robins, G. New performance-driven fpga routing algorithms. *IEEE Transactions Computer-Aided Design 15*, 12 (December 1996), 1505–1517.

[2] Alpert, C. J., Gandham, G., Hrkic, M., Hu, J., Kahng, A. B., Lillis, J., Liu, B., Quay, S. T., Sapatnekar, S. S., and Sullivan, A. J. Buffered steiner trees for difficult instances. *IEEE Transactions Computer-Aided Design 21*, 1 (January 2002), 3–14.

[3] Alpert, C. J., Hu, T. C., Huang, J. H., Kahng, A. B., and Karger, D. Prim-Dijkstra tradeoffs for improved performance-driven routing tree design. *IEEE Transactions Computer-Aided Design 14*, 7 (July 1995), 890–896. (also appeared in ISCAS 1993).

[4] Alpert, C. J., Kahng, A. B., Sze, C. N., and Wang, Q. Timing-driven steiner trees are (practically) free. In *Proc. ACM/IEEE Design Automation Conf.* (San Francisco, CA, 2006), pp. 389–392.

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

[6] Bakoglu, H. *Circuits, Interconnections and Packaging for VLSI.* Addison-Wesley, Reading, MA, 1990.

[7] Boese, K. D., Kahng, A. B., McCoy, B. A., and Robins, G. Fidelity and near-optimality of elmore-based routing constructions. In *Proc. IEEE International Conf. Computer Design* (Cambridge, MA, October 1993), pp. 81–84.

[8] Boese, K. D., Kahng, A. B., McCoy, B. A., and Robins, G. Rectilinear steiner trees with minimum elmore delay. In *Proc. ACM/IEEE Design Automation Conf.* (San Diego, CA, June 1994), pp. 381–386.

[9] Boese, K. D., Kahng, A. B., McCoy, B. A., and Robins, G. Near-optimal critical sink routing tree constructions. *IEEE Transactions Computer-Aided Design 14*, 12 (December 1995), 1417–1436.

[10] Boese, K. D., Kahng, A. B., and Robins, G. High-performance routing trees with identified critical sinks. In *Proc. ACM/IEEE Design Automation Conf.* (Dallas, June 1993), pp. 182–187.

[11] Borah, M., Owens, R. M., and Irwin, M. J. An edge-based heuristic for steiner routing. *IEEE Transactions Computer-Aided Design 13* (1994), 1563–1568.

[12] Caldwell, A., Kahng, A. B., Mantik, S., Markov, I., and Zelikovsky, A. On wirelength estimations for row-based placement. In *Proc. International Symposium on Physical Design* (Monterey, California, April 1998), pp. 4–11.

[13] Chan, P. K., and Karplus, K. Computing signal delay in general rc networks by tree/link partitioning. *IEEE Transactions Computer-Aided Design 9*, 8 (August 1990), 898–902.

[14] Chen, H., Cheng, C.-K., Kahng, A., Măndoiu, I. I., Wang, Q., and Yao., B. The y-architecture for on-chip interconnect: Analysis and methodology. *IEEE Transactions Computer-Aided Design 24*, 4 (April 2005), 588–599.

[15] Chen, H., Cheng, C.-K., Kahng, A. B., Măndoiu, I., and Wang, Q. Estimation of wirelength reduction for λ-geometry vs. manhattan placement and routing. In *Proc. ACM International Workshop on System-Level Interconnect Prediction* (2003), pp. 71–76.

[16] Cheng, X., and Du, D.-Z. *Steiner Trees in Industry*. Kluwer Academic Publishers, The Netherlands, 2001.

[17] Cieslik, D. *Steiner Minimal Trees*. Kluwer Academic Publishers, The Netherlands, 1998.

[18] Cieslik, D. *The Steiner Ratio*. Kluwer Academic Publishers, The Netherlands, 2001.

[19] Cong, J., He, L., Koh, C.-K., and Madden, P. H. Performance optimization of vlsi interconnect layout. *Integration: the VLSI Journal 21* (November 1996), 1–94.

[20] Cong, J., Kahng, A. B., Koh, C. K., and Tsao, C.-W. A. Bounded-skew clock and steiner routing. *ACM Transactions on Design Automation of Electronic Systems 3* (October 1999), 341–388.

[21] Cong, J., Kahng, A. B., and Leung, K.-S. Efficient algorithms for the minimum shortest path steiner arborescence problem with applications to vlsi physical design. *IEEE Transactions Computer-Aided Design 17*, 1 (January 1998), 24–39.

[22] Cong, J., Kahng, A. B., Robins, G., Sarrafzadeh, M., and Wong, C. K. Performance-driven global routing for cell based ic's. In *Proc. IEEE International Conf. Computer Design* (Cambridge, MA, October 1991), pp. 170–173.

[23] Cong, J., Kahng, A. B., Robins, G., Sarrafzadeh, M., and Wong, C. K. Provably good algorithms for performance-driven global routing. In *Proc. IEEE International Symp. Circuits and Systems* (San Diego, CA, May 1992), pp. 2240–2243.

[24] Cong, J., Kahng, A. B., Robins, G., Sarrafzadeh, M., and Wong, C. K. Provably good performance-driven global routing. *IEEE Transactions Computer-Aided Design 11*, 6 (1992), 739–752.

[25] Cong, J., and Leung, K. S. Optimal wiresizing under the distributed elmore delay model. In *Proc. IEEE International Conf. Computer-Aided Design* (1993), pp. 634 – 639.

[26] Cong, J., Leung, K. S., and Zhou, D. Performance-driven interconnect design based on distributed rc delay model. In *Proc. ACM/IEEE Design Automation Conf.* (Dallas, June 1993), pp. 606–611.

[27] Cordova, J., and Lee, Y. H. A heuristic algorithm for the rectilinear steiner arborescence problem. Tech. Rep. TR-94-025, University of Florida, 1994.

[28] de Matos, R. R. L. *A Rectilinear Arborescence Problem.* PhD thesis, University of Alabama, 1979.

[29] Dijkstra, E. W. A note on two problems in connection with graphs. *Numerische Mathematik 1* (1959), 269–271.

[30] Donath, W. E., Norman, R. J., Agrawal, B. K., Bello, S. E., Han, S. Y., Kurtzberg, J. M., Lowy, P., and McMillan, R. I. Timing driven placement using complete path delays. In *Proc. ACM/IEEE Design Automation Conf.* (1990), pp. 84–89.

[31] Du, D.-Z., Smith, J. M., and Rubinstein, J. H. *Advances in Steiner Trees.* Kluwer Academic Publishers, The Netherlands, 2000.

[32] Dunlop, A. E., Agrawal, V. D., Deutsch, D., Jukl, M. F., Kozak, P., and Wiesel, M. Chip layout optimization using critical path weighting. In *Proc. ACM/IEEE Design Automation Conf.* (1984), pp. 133–136.

[33] Dutta, R., and Marek-Sadowska, M. Algorithm for wire sizing of power and ground networks in vlsi designs. *Journal of Circuits, Systems and Computers 2* (June 1992), 141–57.

[34] Elmore, W. C. The transient response of damped linear networks with particular regard to wide-band amplifiers. *J. Appl. Phys. 19*, 1 (1948), 55–63.

[35] Erhard, K. H., and Johannes, F. M. Power/ground networks in vlsi: are general graphs better than trees? *Integration: the VLSI Journal 14*, 1 (November 1992), 91–109.

[36] Erhard, K. H., Johannes, F. M., and Dachauer, R. Topology optimization techniques for power/ground networks in vlsi. In *Proc. European Design Automation Conf.* (Hamburg, Germany, September 1992), pp. 362–367.

[37] Fisher, A. L., and Kung, H. T. Synchronizing large systolic arrays. In *Proc. of SPIE* (May 1982), pp. 44–52.

[38] Friedman, E. G. Clock distribution design in vlsi circuits - an overview. In *Proc. IEEE International Symp. Circuits and Systems* (May 1993), pp. 1475–1478.

[39] Ganley, J. L. Accuracy and fidelity of fast net length estimates. *Integration: the VLSI Journal 23*, 2 (1997), 151–155.

[40] Gerez, S. H. *Algorithms for VLSI Design Automation*. John Wiley and Sons, Chichester, 1998.

[41] Griffith, J., Robins, G., Salowe, J. S., and Zhang, T. Closing the gap: Near-optimal steiner trees in polynomial time. *IEEE Transactions Computer-Aided Design 13*, 11 (November 1994), 1351–1365.

[42] Hanan, M. On steiner's problem with rectilinear distance. *SIAM J. Applied Math. 14* (1966), 255–265.

[43] Hauge, P. S., Nair, R., and Yoffa, E. J. Circuit placement for predictable performance. In *Proc. IEEE International Conf. Computer-Aided Design* (Santa Clara, CA, November 1987), pp. 88–91.

[44] Ho, J. M., Ko, M. T., Ma, T. H., and Sung, T. Y. Algorithms for rectilinear optimal multicast tree problem. In *Proc. International Symp. on Algorithms and Computation* (June 1992), pp. 106–15.

[45] Hodes, T. D., McCoy, B. A., and Robins, G. Dynamically-wiresized elmore-based routing constructions. In *Proc. IEEE International Symp. Circuits and Systems* (London, England, May 1994), pp. 463–466 (Vol. I).

[46] Hong, X., Xue, T., Kuh, E. S., Cheng, C. K., and Huang, J. Performance-driven steiner tree algorithms for global routing. In *Proc. ACM/IEEE Design Automation Conf.* (June 1993), pp. 177–181.

[47] Hou, H., Hu, J., and Sapatnekar, S. S. Non-hanan routing. *IEEE Transactions Computer-Aided Design 18*, 4 (April 1999), 436–444.

[48] Hrkic, M., and Lillis, J. Buffer tree synthesis with consideration of temporal locality, sink polarity requirements, solution cost, congestion and blockages. *IEEE Transactions Computer-Aided Design 22*, 4 (April 2003), 481–491.

[49] Hu, J., and Sapatnekar, S. S. Algorithms for non-hanan -based optimization for vlsi interconnect under a higher order awe model. *IEEE Transactions Computer-Aided Design 19*, 4 (April 2000), 446–458.

[50] Hu, J., and Sapatnekar, S. S. A survey on multi-net global routing for integrated circuits. *Integration: the VLSI Journal 11* (2001), 1–49.

[51] Hu, J., and Sapatnekar, S. S. A timing-constrained simultaneous global routing algorithm. *IEEE Transactions Computer-Aided Design 21*, 9 (September 2002), 1025–1036.

[52] Hu, S., Li, Q., Hu, J., and Li, P. Steiner network construction for timing critical nets. In *Proc. ACM/IEEE Design Automation Conf.* (2006), pp. 379–384.

[53] Hwang, F. K., Richards, D. S., and Winter, P. *The Steiner Tree Problem*. North-Holland, Annals of Discrete Mathematics 53, 1992.

[54] Ismail, Y. I., and Friedman, E. G. *On-Chip Inductance in High-Speed Integrated Circuits*. Kluwer Academic Publishers, Boston, 2001.

[55] Ivanov, A. O., and Tuzhilin, A. A. *Minimal Networks: The Steiner Problem and Its Generalizations*. CRC Press, Boca Raton, Florida, 1994.

[56] Jackson, M. A. B., and Kuh, E. S. Performance-driven placement of cell-based ic's. In *Proc. ACM/IEEE Design Automation Conf.* (1989), pp. 370–375.

[57] Jackson, M. A. B., Kuh, E. S., and Marek-Sadowska, M. Timing-driven routing for building block layout. In *Proc. IEEE International Symp. Circuits and Systems* (1987), pp. 518–519.

[58] Kahng, A. B., Liu, B., and Mandoiu, I. I. Non-tree routing for reliability and yield improvement. *IEEE Transactions Computer-Aided Design 23*, 1 (2004), 148–156.

[59] Kahng, A. B., Mantik, S., and Stroobandt, D. Towards accurate models of achievable routing. *IEEE Transactions Computer-Aided Design 20* (May 2001), 648–659.

[60] Kahng, A. B., and Robins, G. A new class of iterative steiner tree heuristics with good performance. *IEEE Transactions Computer-Aided Design 11*, 7 (July 1992), 893–902.

[61] Kahng, A. B., and Robins, G. On performance bounds for a class of rectilinear steiner tree heuristics in arbitrary dimension. *IEEE Transactions Computer-Aided Design 11*, 11 (November 1992), 1462–1465.

[62] Kahng, A. B., and Robins, G. *On Optimal Interconnections for VLSI*. Kluwer Academic Publishers, Boston, MA, 1995.

[63] Khuller, S., Raghavachari, B., and Young, N. Balancing minimum spanning and shortest path trees. In *Proc. ACM/SIAM Symp. Discrete Algorithms* (January 1993), pp. 243–250.

[64] Koh, C.-K., and Madden, P. H. Manhattan or non-manhattan?: A study of alternative vlsi routing architectures. In *Proc. Great Lakes Symp. VLSI* (Chicago, IL, 2000), pp. 47–52.

[65] Korte, B., Promel, H. J., and Steger, A. *Steiner Trees in VLSI-Layouts, in Paths, Flows and VLSI-Layout*. Springer-Verlag, New York, 1990.

[66] Kou, L., Markowsky, G., and Berman, L. A fast algorithm for steiner trees. *Acta Informatica 15* (1981), 141–145.

[67] Leung, K.-S., and Cong, J. Fast optimal algorithms for the minimum rectilinear steiner arborescence problem. In *Proc. IEEE International Symp. Circuits and Systems* (Hong Kong, 1997), vol. 3, pp. 1568–1571.

[68] Li, Y. Y., Cheung, S. K., Leung, K. S., and Wong, C. K. Steiner tree construction in λ_3-metric. *IEEE Transactions Circuits and Systems-II: Analog and Digital Signal Processing 45*, 5 (May 1998), 563–574.

[69] Lillis, J., Cheng, C. K., Lin, T.-T. Y., and Ho, C.-Y. New performance driven routing techniques with explicit area/delay tradeoff and simultaneous wire sizing. In *Proc. ACM/IEEE Design Automation Conf.* (1996), pp. 395–400.

[70] Lin, I., and Du, D. H. C. Performance-driven constructive placement. In *Proc. ACM/IEEE Design Automation Conf.* (1990), pp. 103–106.

[71] Lin, S., and Wong, C. K. Process-variation-tolerant clock skew minimization. In *Proc. IEEE International Conf. Computer-Aided Design* (San Jose, CA, November 1994), pp. 284–288.

[72] Lin, T. M., and Mead, C. A. Signal delay in general rc-networks. *IEEE Transactions Computer-Aided Design CAD-3*, 4 (October 1984), 331–349.

[73] Marek-Sadowska, M., and Lin, S. P. Timing driven placement. In *Proc. IEEE International Conf. Computer-Aided Design* (Santa Clara, CA, November 1989), pp. 94–97.

[74] Martin, D., and Rumin, N. C. Delay prediction from resistance-capacitance models of general mos circuits. *IEEE Transactions Computer-Aided Design 12*, 7 (July 1993), 997–1003.

[75] McCoy, B. A., and Robins, G. Non-tree routing. *IEEE Transactions Computer-Aided Design 14*, 6 (June 1995), 790–784.

[76] Nastansky, L., Selkow, S. M., and Stewart, N. F. Cost-minima trees in directed acyclic graphs. *Zeitschrift for Operations Research 18* (1974), 59–67.

[77] Nielsen, B. K., Winter, P., and Zachariasen, M. An exact algorithm for the uniformly-oriented steiner tree problem, springer verlag lecture notes in computer science 2461. In *Proc. European Symposium on Algorithms* (Rome, Italy, 2002), Springer-Verlag, pp. 760–771.

[78] Peyer, S., Zachariasen, M., and Grove, D. J. Delay-related secondary objectives for rectilinear steiner minimum trees. *Discrete and Applied Mathematics 136*, 2 (February 2004), 271–298.

[79] Prasitjutrakul, S., and Kubitz, W. J. A timing-driven global router for custom chip design. In *Proc. IEEE International Conf. Computer-Aided Design* (Santa Clara, CA, November 1990), pp. 48–51.

[80] Preas, B. T., and Lorenzetti, M. J. *Physical Design Automation of VLSI Systems.* Benjamin/Cummings, Menlo Park, CA, 1988.

[81] Prim, A. Shortest connecting networks and some generalizations. *Bell Syst. Tech J. 36* (1957), 1389–1401.

[82] Promel, H. J., and Steger, A. *The Steiner Tree Problem: A Tour Through Graphs, Algorithms, and Complexity.* Friedrich Vieweg and Son, Braunschweig, 2002.

[83] Pullela, S., Menezes, N., and Pillage, L. T. Reliable non-zero skew clock trees using wire width optimization. In *Proc. ACM/IEEE Design Automation Conf.* (San Diego, CA, 1993), pp. 165–170.

[84] Qiu, W., and Shi, W. Minimum moment Steiner trees. In *Proc. ACM/SIAM Symp. Discrete Algorithms* (2004), pp. 488–495.

[85] Rao, S. K., Sadayappan, P., Hwang, F. K., and Shor, P. W. The rectilinear steiner arborescence problem. *Algorithmica 7*, 1 (1992), 277–288.

[86] Robins, G. *On Optimal Interconnections.* PhD thesis, Department of Computer Science, UCLA, CSD-TR-920024, 1992.

[87] Robins, G., and Zelikovsky, A. Improved steiner tree approximation in graphs. In *Proc. ACM/SIAM Symp. Discrete Algorithms* (San Francisco, CA, January 2000), pp. 770–779.

[88] Robins, G., and Zelikovsky, A. Tighter bounds for graph steiner tree approximation. *SIAM J. on Discrete Mathematics 19*, 1 (2005), 122–134. (winner of the 2007 SIAM Outstanding Paper Prize).

[89] Rubinstein, J., Penfield, P., and Horowitz, M. A. Signal delay in rc tree networks. *IEEE Transactions Computer-Aided Design 2*, 3 (1983), 202–211.

[90] Sait, S. M., and Youssef, N. *VLSI Physical Design Automation - Theory and Practice.* World Scientific Publishing Company, Singapore, 1999.

[91] Sapetnekar, S. Rc interconnect optimization under the elmore delay model. In *Proc. ACM/IEEE Design Automation Conf.* (San Diego, CA, June 1994), pp. 387–391.

[92] Sarrafzadeh, M., and Wong, C. K. Hierarchical steiner tree construction in uniform orientations. *IEEE Transactions Computer-Aided Design 11*, 9 (September 1992), 1095–1103.

[93] Sarrafzadeh, M., and Wong, C. K. *An Introduction to VLSI Physical Design.* McGraw Hill, New York, NY, 1996.

[94] Saxena, P., Menezes, N., Cocchini, P., and Kirkpatrick, D. A. Repeater scaling and its impact on CAD. *IEEE Transactions Computer-Aided Design 23*, 4 (April 2004), 451–463.

[95] Sherwani, N. *Algorithms for VLSI Physical Design Automation, Third Edition.* Kluwer Academic Publishers, Boston, MA, 1998.

[96] Sherwani, N., Bhingarde, S., and Panyam, A. *Routing in the Third Dimension.* IEEE Press, New York, NY, 1995.

[97] Shi, W., and Su, C. The rectilinear steiner arborescence problem is np-complete. *SIAM J. Comput. 35*, 3 (2006), 729–740.

[98] Sriram, M., and Kang, S. M. Performance driven mcm routing using a second order rlc tree delay model. In *IEEE International Conf. on Wafer Scale Integration* (San Francisco, CA, USA, January 1993), pp. 262–267.

[99] Sutanthavibul, S., and Shragowitz, E. An adaptive timing-driven layout for high speed vlsi. In *Proc. ACM/IEEE Design Automation Conf.* (1990), pp. 90–95.

[100] Teig, S. The x architecture: Not your father's diagonal wiring. In *Proc. ACM International Workshop on System-Level Interconnect Prediction* (2002), pp. 33–37.

[101] Trubin, V. A. Subclass of the steiner problems on a plane with rectilinear metric. *Cybernetics and Systems Analysis 21*, 3 (1985), 320–322.

[102] Tsay, R. S. Exact zero skew. In *Proc. IEEE International Conf. Computer-Aided Design* (Santa Clara, CA, November 1991), pp. 336–339.

[103] Wu, D., Hu, J., and Mahapatra, R. Coupling aware timing optimization and antenna avoidance in layer assignment. In *Proc. International Symposium on Physical Design* (New York, NY, 2005), ACM Press, pp. 20–27.

[104] Yildiz, M. C., and Madden, P. H. Preferred direction steiner trees. In *Proc. Great Lakes Symp. VLSI* (West Lafayette, IN, 2001), pp. 56–61.

[105] Zachariasen, M. A catalog of hanan grid problems. *Networks – an International Journal 38*, 2 (2001), 76–83.

[106] Zhou, D., Tsui, F., and Gao, D. S. High performance multichip interconnection design. In *Proc. ACM/SIGDA Physical Design Workshop* (Lake Arrowhead, CA, April 1993), pp. 32–43.

[107] Zhu, Q., Dai, W. W. M., and Xi, J. G. Optimal sizing of high-speed clock networks based on distributed rc and lossy transmission line models. In *Proc. IEEE International Conf. Computer-Aided Design* (1993), pp. 628–633.