On the Bounded-Skew Clock and Steiner Routing Problems*

Dennis J.-H. Huang, Andrew B. Kahng and Chung-Wen Albert Tsao

UCLA Computer Science Dept., Los Angeles, CA 90024-1596 USA

Abstract

We study the minimum-cost bounded-skew routing tree (BST) problem under the linear delay model. This problem captures several engineering tradeoffs in the design of routing topologies with controlled skew. We propose three tradeoff heuristics. (1) For a fixed tree topology the *Extended-DME* (Ex-DME) approach extends the DME algorithm for exact zero-skew trees [4] via the concept of a *merging region*. Ex-DME is optimal for the infinite-skew limiting case. (2) For arbitrary topology and arbitrary embedding, we develop the *Extended Greedy-DME* (ExG-DME) algorithm, which very closely matches the best-known heuristic for the zero-skew case [6, 7], and for the infinite-skew case (i.e., the Steiner minimal tree problem) [2, 13]. (3) For arbitrary topology and single-layer (planar) embedding, the *Extended Planar-DME* (ExP-DME) algorithm exactly matches the best-known heuristic for zero-skew planar routing [15], and again closely approaches the best-known performance for the infinite-skew case. Our work provides unifications of the clock routing and Steiner tree heuristic literatures, and gives a smooth cost-skew tradeoff that allows good engineering solutions.

1 Introduction

In routing design for high-performance VLSI systems, control of signal delay has become a dominant objective. For clock distribution, approaches to "zero-skew" routing have spanned the range from highly geometric perspectives (e.g., [4, 6]) to system architecture perspectives (e.g., [10]). Controlling the skew of signal arrival times is also of increasing interest for large global routes on chip or on MCM substrates. At the same time, other objectives also require attention. Certainly, the routing design should have low wiring area to reduce die size and capacitive effects on performance. Increasingly, planar-embeddability (i.e., routability on a single layer) is also required for better process variation independence, reduced delay and attenuation through vias, and other reasons [23].

In this paper, we present the first *unified* approach to minimum-cost, skew-bounded, and planar/non-planar routing tree construction. Addressing these co-existing objectives is both motivated and enabled by several recent works. Pillage and coauthors [18, 19, 17], Edahiro [8], and Zhu and Dai [22, 24, 25] show how to use wiresizing to optimize source-sink signal delays in clock distribution. These works apply various continuous or discrete wiresizing optimizations to existing clock routing trees, in order to achieve improved phase delay and skew according to Elmore delay or higher-order approximations. (The work of [18] also performs buffer optimization to minimize power dissipation; see also the many works in the clock buffer placement and sizing literature, as surveyed in [10].)

*Partial support for this work was provided by NSF MIP-9257982 and MIP-922370.

The works of Zhu and Dai [22, 24] and Pullela, Menezes and Pillage [19] explicitly propose construction of initial *non-zero skew* clock routing solutions which can be sized to achieve a prescribed skew bound. Our work addresses the core of this approach, namely, the underlying *bounded-skew* routing construction. As in the work of [22, 24], we solve the bounded-skew problem under the linear delay model; this enables application of geometric heuristics to derive high-quality topologies which can then be sized using any of several existing methods.[1] The papers [22, 24, 19], as well as the wiresizing work of Sapetnekar for minimum-delay global routing [20], also propose the notion of an *engineering tradeoff* in routing tree design: since exact zero skew is never an actual design requirement (cf. the discussion in [14]), a bounded-skew, rather than exact zero-skew, formulation is appropriate. Before formally stating the bounded-skew routing problem, we review one prototypical setting for this optimization.

In [22], Zhu and Dai propose a two-level scheme for clock distribution in multi-chip module (MCM) designs with area-array (flip-chip with solder bumping) pads. Since a given design can accommodate some non-zero skew at the synchronizing elements, the overall formulation is that of minimum-cost bounded-skew routing. On the MCM substrate, Zhu and Dai construct a "global" planar clock tree with equal source-sink pathlengths (i.e., zero skew under the linear delay model) using their earlier method in [23]. Wiresizing is then used to achieve exact zero skew on the substrate according to higher-order delay models. To take advantage of the available skew for area/power reduction, the "global" clock tree is constructed only until its leaves are sufficiently close to the positions of all synchronizing elements. Then, "local" clock subtrees are used to distribute the clock signal from these leaves to the actual synchronizing elements. Zhu and Dai route each local clock tree as a delay-bounded Steiner minimal tree, with the delay bound equal to the skew limit of the design. Finally, the global and local levels of the clock tree are interfaced by clock buffers.[2] Thus, this particular work approaches the bounded-skew routing with a mix of top-level zero-skew and bottom-level minimum-delay constructions. Other methods merit investigation in light of the possible infeasibility of the local clock tree construction.

We now define the *bounded-skew routing tree* (BST) problem under the linear delay model.

The Bounded-Skew Routing Tree (BST) Problem: *Given a set $S = \{s_1, s_2, \ldots, s_n\} \subset \Re^2$ of clock sink locations, a clock source location s_0, a skew bound B, and a connection topology G (a rooted binary tree with n leaves corresponding to the sinks in S), we seek a tree $T(S)$ which embeds G in the Manhattan plane, and for which the maximum difference between any two source-sink pathlengths is $\leq B$.*

Our discussion below will also consider the BST variant where no topology G has been prescribed. This variant

[1] As noted below, the methods that we develop can be extended to achieve bounded skew under the Elmore model.

[2] The method depends on the ability to construct a "delay-bounded Steiner minimal tree". This was addressed separately by the same authors in [24], which gives a heuristic to "mix" a minimum-cost Steiner minimal tree (T_c) and a minimum-Elmore delay Steiner tree (T_d). The minimum-Elmore delay Steiner tree T_d is assumed to have maximum source-sink delay less than the delay bound (i.e., the skew limit in the original two-level formulation); edges of T_c are iteratively added and the resulting cycles broken as the "delay-bounded Steiner minimal tree" evolves. It is not clear how to determine T_d with acceptable maximum source-sink Elmore delay; whether such a tree exists for a given instance (set of terminals, and delay bound) is NP-complete.

is particularly useful in achieving a planar routing, or in taking full advantage of the available skew bound B. Note that for this variant, we typically assume that the source location s_0 is also unspecified.

The remainder of our paper is organized as follows. In Section 2, we review the "DME" (Deferred-Merge Embedding) approach to zero-skew clock tree construction; the DME approach is central to our proposed methodology. Specifically, we summarize the original DME method, the "Greedy-DME" method of Edahiro [6], and the "Planar-DME" method of Kahng and Tsao [15]. In Section 3, we develop our first tradeoff heuristic, for the minimum-cost BST with *fixed topology*. This construction is optimal for the extreme cases of $B = 0$ and $B = \infty$ (i.e., the SMT problem with fixed topology). Section 4 develops our second heuristic, for the case of unrestricted topology and unrestricted embedding. Our tradeoff matches the best known heuristic (Greedy-DME) for the extreme cost of $B = 0$; we also essentially match the best known SMT heuristic [2, 13] for $B = \infty$. Section 5 gives our third heuristic, for the case of unrestricted topology but planar (single-layer) embedding. Again, we essentially match the best known heuristics in the literature for $B = 0$ and $B = \infty$, and achieve a smooth tradeoff in between. Finally, Section 6 concludes with experimental results and directions for future work.

2 A Review of Three DME Variants

The Deferred-Merge Embedding (DME) algorithm, proposed independently by three groups [5, 3, 1, 4], achieves exact zero skew given any delay model for which sink delays are monotone in the length of each edge of the clock tree (e.g., linear delay and Elmore delay have this property). For the linear delay model, DME is *optimal*: it returns a tree with minimum cost and minimum source-sink pathlength for any input sink set S and topology G.

Because the properties of the DME construction are central to our present work, we review key details of DME and its Greedy-DME and Planar-DME variants, following the developments in [1, 15]. In the following discussion, we identify each node v of the rooted topology G with the edge e_v to its parent. Once a node v of the topology has been embedded in the Manhattan plane, we often identify v with its location in the plane, denoted $l(v)$. The cost of a routing tree T is defined as $cost(T) = \sum_{v \in T} |e_v|$, i.e., the sum of edgelengths in T. We use $d(s,t)$ to denote the Manhattan distance between points s and t; the distance between two pointsets P and Q is $d(P,Q) = min\{d(p,q) | p \in P, q \in Q\}$.

The DME Algorithm

Given a set of sinks S and a topology G, DME embeds internal nodes of G via: (i) a bottom-up phase that constructs a tree of *merging segments* which represent loci of possible placements of internal nodes in a zero-skew tree (ZST) T; and (ii) a top-down embedding phase that determines exact locations for the internal nodes in T.

In the **bottom-up phase**, each node $v \in G$ is associated with a merging segment, denoted $ms(v)$, which represents a set of possible placements of v in a minimum-cost ZST. Let a and b be the children of node v, and let

TS_a and TS_b denote the subtrees of merging segments rooted at a and b. The construction of $ms(v)$ depends on $ms(a)$ and $ms(b)$, hence the bottom-up processing order. We seek placements of v which allow TS_a and TS_b to be merged with *minimum* added wire while preserving zero skew. This means that we want to minimize $|e_a|+|e_b|$ in the output tree $T(S)$, while balancing sink delays from $l(v)$. The values of $|e_a|$ and $|e_b|$ which achieve this are unique, and are retained for use in the subsequent top-down embedding phase of DME.

We use the following terminology. A *Manhattan arc* is a line segment, possibly of zero length, with slope +1 or -1. The collection of points within a fixed distance of a Manhattan arc is called a *tilted rectangular region*, or *TRR*; the boundary of a TRR consists of Manhattan arcs. The Manhattan arc at the center of a TRR is its *core*; the *radius* of a TRR is the distance between its core and its boundary.

We can now recursively define $ms(v)$, starting from the leaves of the topology G. If v is a sink s_i, then $ms(v) = \{s_i\}$. If v is an internal node of G, then $ms(v)$ is the set of all placements $l(v)$ which merge TS_a and TS_b with minimum wire cost, i.e., all points within distance $|e_a|$ of $ms(a)$ and within distance $|e_b|$ of $ms(b)$. More precisely, $ms(v)$ is obtained by intersecting two TRRs, trr_a with core $ms(a)$ and radius $|e_a|$, and trr_b with core $ms(b)$ and radius $|e_b|$; i.e., $ms(v) = trr_a \cap trr_b$. A lemma in [1] shows that all merging segments so constructed are Manhattan arcs.

Given the tree of merging segments, or *merging tree*, the **top-down phase** embeds each internal node v of G as follows: (i) if v is the root node, then DME selects any point in $ms(v)$ to be $l(v)$[3]; or (ii) if v is an internal node other than the root, DME chooses $l(v)$ to be any point on $ms(v)$ that is at distance $|e_v|$ or less from the embedding $l(p)$ of v's parent[4]. With (ii), DME creates a square TRR trr_p with radius $|e_v|$ and core $\{l(p)\}$; $l(v)$ can be any point in $ms(v) \cap trr_p$.

Note that DME requires an input topology. Several works [1, 3, 4, 6] have thus studied topology constructions that lead to low-cost routing solutions when DME is applied. The most successful work to date is the "Greedy-DME" method of Edahiro [6], which we now describe.

The Greedy-DME Algorithm

The Greedy-DME algorithm of Edahiro determines the topology of the merging tree in a greedy bottom-up fashion. More precisely, let K denote a set of merging segments which initially consists of all the sink locations, i.e., $K = \{ms(s_i)\}$. Greedy-DME iteratively finds the pair of nearest neighbors in K, i.e., $ms(v_i)$ and $ms(v_j)$ such that $d(ms(v_i), ms(v_j))$ is minimum. A new parent merging segment $ms(v)$ is computed for node v, based on a zero-skew merge of $ms(v_i)$ and $ms(v_j)$; then, K is updated by adding $ms(v)$ and deleting both $ms(v_i)$ and $ms(v_j)$. After $n-1$ operations, K has has only one merging segment for the root of the merging tree.

[3] If a clock source location clk has been specified, DME simply chooses $l(s_0) \in ms(s_0)$ with minimum distance from clk and connects a wire directly from clk to $l(s_0)$.

[4] Because $ms(p)$ was constructed such that $d(ms(v), ms(p)) \leq |e_v|$, there must exist feasible embedding points for node v.

In [6], an $O(n^2 \log n)$ implementation based on Delaunay triangulation was proposed. In [7], the time complexity was improved to $O(n \log n)$ by a clustering method in which several nearest-neighbor pairs are selected from within the Delaunay triangulation at the same time. In other words, the algorithm first constructs a "nearest-neighbor graph" which maintains the nearest neighbor of each merging segment in K. Via zero-skew merges, $|K|/k$ nearest-neighbor pairs are taken from the graph in non-decreasing order of distance, where k is a constant typically between 2 and 4. As each pair is processed, if either of its merging segments has already been merged, then the pair is discarded. The solution can be significantly improved by a post-processing step that entails a local exhaustive search to adjust the topology of the resulting tree (cf. "CL + I6" in [7]). Greedy-DME achieves 20% reduction in wiring cost compared with the methods of [3, 4]. Recently, a bucket decomposition technique was proposed in [9] which further reduces the Greedy-DME time complexity to linear on average, but this is achieved at the cost of some performance degradation.

The Planar-DME Algorithm

Finally, we review the Planar-DME algorithm of Kahng and Tsao [15], which creates a planar (single-layer routable) zero-skew tree. Before describing the method, some more terminology is necessary. For any sink subset $S' \subseteq S$, we define the *diameter* of S', written $diam(S')$, to be the maximum distance between two points of S. The *radius* of S', written $radius(S')$, is $diam(S')/2$. A *Manhattan disk* is a TRR with a core consisting of a single point; we use $MD(s_i, r)$ to denote the Manhattan disk with core $\{s_i\}$ and radius $r \geq 0$. The center of a sink set $S' \subseteq S$ is denoted as $center(S')$. In the Manhattan plane, [15] shows that $center(S') = core(\bigcap_{u \in S'} MD(u, radius(S)))$, so that $center(S')$ is always a Manhattan arc. We use $P_{S'}$ to denote the (Euclidean) *convex polygon* containing S'.

The main result of [15] is that under the linear delay model, the two-phase DME algorithm can be emulated by a top-down "Single-Pass DME". Specifically, the tree of merging segments constructed in the bottom-up DME phase can be generated *during* the top-down phase. The enabling result is that the root of the minimum-pathlength zero-skew subtree over any sink set $S' \subseteq S$ must always be located on the Manhattan arc $center(S')$ – independent of the subtree connection topology. Thus, Single-Pass DME allows the connection topology to be determined dynamically in a top-down fashion, and at the same time still finds a minimum-pathlength, minimum-cost embedding of whatever topology is eventually determined.

The Planar-DME algorithm [15] is essentially Single-Pass DME, with the top-down determination of node embeddings and connection topology guided by *embedding rules* and *partitioning rules*. The main idea is that (Euclidean) convex polygons can guide the top-down partitioning of both the routing area and the set of sinks. Given $S' \subseteq S$ and a convex polygon $P_{S'}$ containing S', the Planar-DME algorithm recursively divides $P_{S'}$ into two smaller convex polygons, such that routing inside one convex polygon cannot interfere with routing inside the other convex polygon or on the boundary between the polygons. This noninterfering property of the wiring immediately implies a planar solution.

Planar-DME reduces tree cost by an average of 20% from the previous planar clock routing method of [23], and has minimum possible source-sink pathlength and minimum tree cost for the generated topology.

3 A First Tradeoff: the Fixed-Topology Case

To address the BST problem for the case of a fixed tree topology, we propose the *Extended-DME* (Ex-DME) approach, which extends the DME algorithm by incorporating the concept of a *merging region*.

3.1 Definition of Merging Region

We will use the following terminology. An *octilinear polygon* is a convex polygon formed by Manhattan arcs, or *rectilinear* (i.e., horizontal or vertical) line segments. Such a polygon, along with its interior, defines an *octilinear polygonal region*. The *joining region* of two disjoint octilinear polygonal regions P and Q, is the set of points $JR(P,Q) = \{p | d(p,P) + d(p,Q) = d(P,Q)\}$. The *joining segments* of P and Q, denoted $JS(P)$ and $JS(Q)$, are the closest portions (to each other) of P and Q, i.e., $JS(P) = P \bigcap JR(P,Q)$ and $JS(Q) = Q \bigcap JR(P,Q)$. If P and Q overlap, then we define $JR(P,Q) = JS(P) = JS(Q) = l$, where l is any longest rectilinear line segment in the intersection of P and Q. Note that $JS(P)$ and $JS(Q)$ will be either (i) a pair of parallel Manhattan arcs, or (ii) a pair of parallel rectilinear line segments[5] (see Figure 1).

We use $t_{max}(p)$ and $t_{min}(p)$ to denote the maximum and minimum pathlength delay (or max-delay and min-delay for short) from point p, taken over all leaves in the subtree rooted at p. If all points of P have identical max-delay and min-delay, we use $t_{max}(P)$ and $t_{min}(P)$ to represent these values. The skew of point p, denoted $skew(p)$, is defined to be $t_{max}(p) - t_{min}(p)$. Similarly, $skew(P)$ denotes the common skew value of a pointset P. If B is the specified skew bound, then p is a *feasible merging point* if $skew(p) \leq B$. The *feasible merging section* of a pointset P, denoted $FMS(P)$, is the set of feasible merging points in P. The *minimum skew section* of P, denoted $MSS(P)$, is the set of points in P with the minimum skew.

A rectilinear line segment l is *well-behaved* if (i) $t_{max}(p)$ and $t_{min}(p)$ are both piecewise (with slope $+1$ or -1) linear functions of the position of p on l, and (ii) $t_{max}(p)$ ($t_{min}(p)$) is a concave (convex) function with at most one *turning point*, i.e., the value of $t_{max}(p)$ ($t_{min}(p)$) is minimum (maximum) at the turning point, and increases (decreases) toward both endpoints of l. A Manhattan arc l is well-behaved if all its points have the same max-delay and min-delay. A line segment l is well-behaved if l is a well-behaved rectilinear line segment or a well-behaved Manhattan arc. An octilinear polygonal region is well-behaved if its boundary segments are well-behaved. Given a well-behaved octilinear polygonal region P, we can compute the max-delay and min-delay

[5] Case (ii) subsumes the case where $JS(P)$ and $JS(Q)$ are a Manhattan arc and a rectilinear line segment.

of any interior point from the max-delay and min-delay values of P's vertices.

From the above definition, $skew(p)$ defined over a well-behaved rectilinear line segment l will be a piecewise linear concave function with up to three linear regions, depending on the values of the max-delay and min-delay turning points. Thus, $FMS(l)$ and $MSS(l)$ are both single intervals of l and can each be computed in $O(1)$ time.

Given a connection topology G, the *merging region* of each node $v \in G$, denoted $mr(v)$, and the delays for each point in $mr(v)$ are recursively defined as follows:

1. If v is a sink s_i, then $mr(v) = \{s_i\}$, and $t_{max}(v) = t_{min}(v) = 0$.

2. If v is an internal node whose children a and b have merging regions $mr(a)$ and $mr(b)$, then $mr(v)$ is constructed as follows.[6]

   (a) We first define the delays of any point $p \in JR(v) = JR(mr(a), mr(b))$ as follows: $t_{max}(p) = \max_{k=a,b}\{t_{max}(p_k) + d(p, p_k)\}$, $t_{min}(p) = \min_{k=a,b}\{t_{min}(p_k) + d(p, p_k)\}$, where p_k is the point in $mr(k)$ which is closest to p.

   (b) If $FMS(JR(v)) \neq \emptyset$, then $mr(v) = FMS(JR(v))$;

   (c) If $FMS(JR(v)) = \emptyset$, then $mr(v) = MSS(JR(v))$. In this case, $skew(mr(v)) > B$. To meet the skew bound constraint with least increase $\delta = skew(mr(v)) - B$ in merging cost, we set edge lengths as follows. We use L_a and L_b to respectively denote the joining segments of $mr(a)$ and $mr(b)$. If $skew(MSS(L_a)) \leq skew(MSS(L_b))$, then $|e_a| = 0$, and $|e_b| = d(mr(a), mr(b)) + \delta$. Otherwise, $|e_b| = 0$, and $|e_a| = d(mr(a), mr(b)) + \delta$. Then the delays for any point $p \in mr(v)$ are re-defined as $t_{max}(p) = \max_{k=a,b}\{t_{max}(p_k) + |e_k|\}$, $t_{min}(p) = \min_{k=a,b}\{t_{min}(p_k) + |e_k|\}$.

Notice that the merging cost at v is lower-bounded $d(mr(a), mr(b))$. If $FMS(JR(v)) \neq \emptyset$, then $mr(v)$ is the set of feasible merging points with minimum merging cost. Otherwise, $mr(v)$ does not have this property since some points outside $mr(v)$ may have lower merging cost than points in $JR(v)$.

3.2 Merging Region Construction

Given the merging regions $mr(a)$ and $mr(b)$ of v's children, the following rules can be used to construct the new merging region $mr(v)$ in constant time. We assume each merging region is a well-behaved octilinear polygonal region.

M1 Compute $FMS(l)$ for the rectilinear boundary segments l of $JR(v)$.

[6] Roughly speaking, $mr(v)$ is the region of possible placements of v within $JR(mr(a), mr(b))$ such that the merging cost $|e_a| + |e_b|$ is minimized and the skew bound B is satisfied.

M2 If $JS(mr(a)) = L_a$ and $JS(mr(b)) = L_b$ are parallel rectilinear line segments, then compute $FMS(l)$ for all rectilinear line segments $l = \overline{p_1 p_2}$ such that $p_1 \in L_a$, $p_2 \in L_b$, and either p_1 or p_2 is a skew turning point of L_a or L_b.

M3 Let F be the set of feasible merging sections computed above.

M4 If $F \neq \emptyset$, then $mr(v)$ is equal to the smallest octilinear polygonal region containing F.

M5 If $F = \emptyset$, then $mr(v) = MSS(L_a)$ if $skew(MSS(L_a)) \leq skew(MSS(L_b))$; otherwise, $mr(v) = MSS(L_b)$.

Lemma 1 in the Appendix shows that $JR(v)$ is a well-behaved octilinear polygonal region if the merging regions of v's children are well-behaved octilinear polygonal regions. Therefore, each of the feasible merging sections can be computed in $O(1)$ time. Since the construction rules compute at most 8 feasible merging sections and 2 minimum skew sections, $mr(v)$ can also be computed in constant time. The construction of merging regions is illustrated in Figure 1.

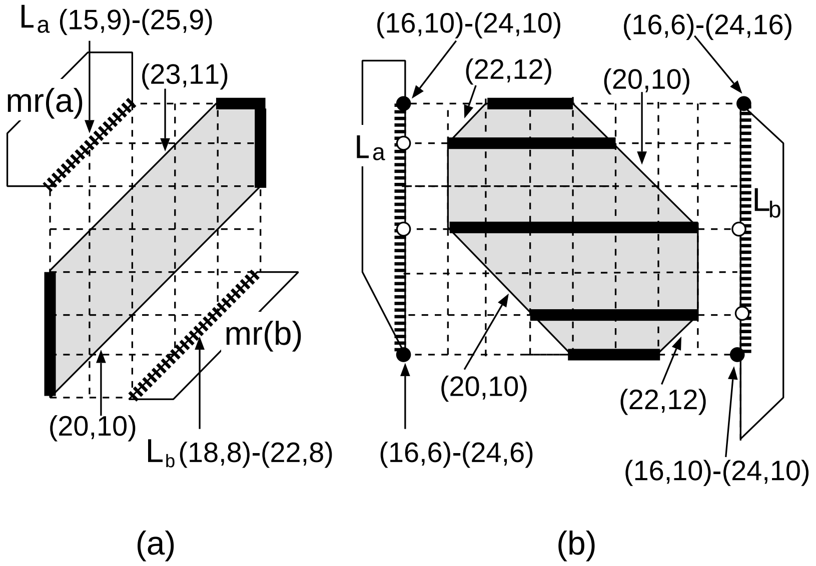


Figure 1: Construction of merging region $mr(v)$ given the merging regions for v's children a and b. The joining region $JR(mr(a), mr(b))$ is shown as the gridded area. The joining segments $JS(mr(a)) = L_a$ and $JS(mr(b)) = L_b$, shown as thick dotted lines, are parallel Manhattan arcs in (a) and parallel vertical line segments in (b). The hollow points in (b) are the skew turning points of L_a and L_b. The merging region for node v, shown as a shaded region, is the smallest octilinear polygonal region containing all feasible merging sections, shown as the thick bold line segments. The first and second coordinate pair associated with points on L_a and L_b represent (max-delay, min-delay) before and after merging, respectively. The coordinates of other points represent their (max-,min-) delays after merging.

In the Ex-DME algorithm, outlined in Figure 2, the merging region construction rules are analogous to the DME merging segment construction rules. The algorithm is similar to DME except for the following points.

- When two merged subtrees rooted at nodes a and b have merging cost greater than the lower bound $d(mr(a), mr(b))$, then the edge lengths $|e_a|$ and $|e_b|$ will be determined in the bottom-up phase. Otherwise, the edge length will be determined in the top-down phase.

- Each node v is embedded at the location in $mr(v)$ that is closest to the location of its parent p, even if $|e_v| > d(mr(v), l(p))$.

```
Procedure Build_Tree_of_Merging_Regions (G,S,B)
Input: Topology G; set of sink locations S;
          skew bound B
Output: Tree of merging regions TR containing
          mr(v) for each node v in G
 for each node v in G (bottom-up order)
   if v is a sink node,
     mr(v) ← {l(v)}
   else
     Let a and b be the children of v
     Let L_a = JS(mr(a)) and L_b = JS(mr(b))
     Calculate mr(v) by construction rules M1-M5
     if FMS(JR(v)) = ∅,
       if skew(MSS(L_a)) ≤ skew(MSS(L_b))
         |e_a| ← 0
         |e_b| ← d(mr(a), mr(b)) + skew(MSS(L_a)) − B
       else
         |e_b| ← 0
         |e_a| ← d(mr(a), mr(b)) + skew(MSS(L_b)) − B
```

(a) Bottom-up phase: Construction of the tree of merging regions TR.

```
Procedure Find_Exact_Placements(TR)
Input: Tree of merging regions TR
Output: BST T(S) with skew ≤ B; |e_v|, ∀v ∈ G
 for each internal node v in G (top-down order)
   if v is the root
     Choose any l(v) ∈ mr(v) closest to clock source
     location if given; otherwise, any l(v) ∈ mr(v)
   else
     Let p be the parent node of v
     if |e_v| not determined yet
       |e_v| ← d(mr(v), l(p))
     Choose any l(v) ∈ mr(v) closest to mr(v)
```

(b) Top-down phase: Construction of the BST by embedding internal nodes of G within TR.

Figure 2: The Ex-DME Algorithm.

In the Appendix, we show that the merging regions described above possess the following properties.

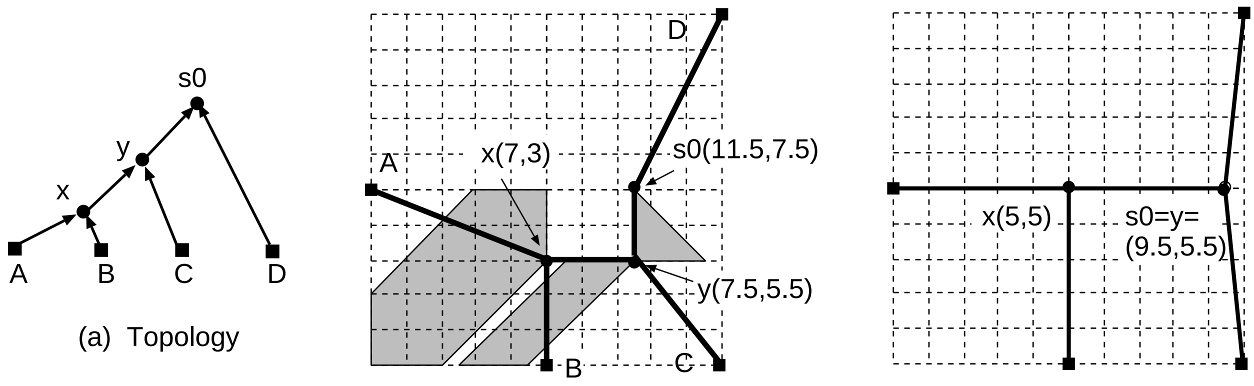
Theorem 1: Each merging region $mr(v)$ for a node $v \in G$ is a well-behaved octilinear polygonal region, and can be computed by construction rules M1-M5 in constant time.

It is obvious that given a sink set S, a connection topology G, and skew bound $B = 0$, if $FMS(JR(v)) \neq \emptyset$ for all internal nodes $v \in G$, Ex-DME is identical to DME and thus is optimal for any prescribed topology.[7] Furthermore,

Theorem 2: When $B = \infty$, for any sink set S and topology G, Ex-DME returns a Steiner tree over S with minimum cost for topology G. □

[7]Experimentally, the condition is true for most nodes in G as long as G is a "good" topology, e.g., the one generated by ExG-DME described below.

However, Ex-DME is not necessarily optimal for any intermediate value of B. A counterexample is given in Figure 3.



(b) Ex-DME (Cost = 27.5, Skew = 4)   (c) Optimal (Cost=25.5, Skew=4)

Figure 3: An example with 4 sinks (dark squares) showing the suboptimality of Ex-DME for finite B. The skew bound $B = 4$, and merging regions are indicated by shaded regions. Each internal node (dark circle) is embedded in its merging region. Each pair of coordinates associated with point p represents $(t_{max}(p), t_{min}(p))$.

4 A Second Tradeoff: the Unrestricted Case

We now consider the variant BST formulation where the topology is not fixed and can be determined dynamically, and where the embedding is unrestricted. We propose a new *Extended Greedy DME* (ExG-DME) algorithm which provides a good tradeoff; furthermore, ExG-DME matches the best known heuristic for the zero-skew limiting case, and very closely matches the performance of the best-known heuristic [2, 13] for the infinite-skew case (i.e., the Steiner minimal tree problem).

Basically, ExG-DME is an extension of Greedy-DME [6, 7] that exploits flexibility stemming from allowed skew during the topology construction. We now describe how ExG-DME determines the topology, while taking into consideration the allowed skew.

4.1 Flexible Generation of the Tree Topology

In the standard DME approach, two merging subtrees are always merged at their roots so as to maintain zero skew. However, the shortest connection between two trees may not be between their roots. The key observation is that subtrees may be merged at non-root nodes as long as the resulting skew is $\leq B$. Thus with non-zero allowed skew, we may change the position of roots to other locations in the subtrees, such that merging cost is reduced while the skew remains appropriately bounded. This ability is the key merit of the ExG-DME approach to BST construction.

Figure 4 shows how this strategy can improve solution quality when the skew bound B is large. In Figure 4a, the eight sinks are located in a horizontal line and are equally spaced. When B is near zero, the minimum tree cost can be obtained by merging subtrees T_1 and T_2 at their roots as shown. However, this topology is bad when B is large, even if the cost of two subtrees is still minimum. In Figure 4b, we show that a good topology can be obtained by appropriately changing the subtree topology before merging.

Based on this intuition, our method adjusts the tree topology by changing the position of the root, as shown in Figure 5. The candidate positions for the root are as the parent of nodes u and v, where u and v are the endpoints of some edge in the current tree. Note that when we move the root of the tree, the basic structure of the subtrees is kept the same. Thus, the costs of two subtrees will not increase because of the topology change. The overall ExG-DME algorithm, which is based on Greedy-DME, is given in Figure 6. One key difference between ExG-DME and Greedy-DME is the construction of the nearest-neighbor graph H. In Greedy-DME, each pair of nodes (v_1, v_2) of H represents a possible merging pair of subtrees rooted at v_1 and v_2; the weight of edge (v_1, v_2) is the merging cost of the respective subtrees, which is simply given by $d(ms(v_1), ms(v_2))$. In our ExG-DME method, each edge (v_1, v_2) in H represents a possible way of merging two subtrees T_1 (containing v_1) and T_2 (containing v_2). Before merging T_1 and T_2, ExG-DME will reposition their roots at edges e_{v_1} and e_{v_2}, then adjust the tree topologies as shown in Figure 5. Although the weight of edge (v_1, v_2) still represents the merging cost of T_1 and T_2, the computation becomes somewhat more complicated. If T_1 and T_2 are the same tree, then the weight of edge (v_1, v_2) is infinite (same trees cannot be merged). Otherwise, we first reposition the roots of T_1 and T_2 at the edges e_{v_1} and e_{v_2}, then construct the new subtrees T_1' and T_2'. We then merge T_1' and T_2' into a new tree T, and the the merging cost is given by $cost(T) - cost(T_1) - cost(T_2)$. To compute $cost(T)$ in constant time instead of $O(|T|)$ time, we maintain two data structures, $mr'(v_i)$ and $cost(v_i)$. Let T_i be the merging tree containing node v_i, and let T_i' be the resulting adjusted tree after the root of T_i is relocated to edge e_{v_i}. Then $mr'(v_i) =$ the root of T_i' and $cost'(v_i) = cost(T_i')$.[8] Then the value of $cost(T)$ (and thus the weight of edge (v_1, v_2)) can be computed in constant time as $cost(T) = d(mr'(v_1), mr'(v_2)) + cost(v_1) + cost(v_2)$. Once an edge (v_1, v_2) is selected, ExG-DME will adjust the topology of T_1 and T_2 before merging, and update $mr'(v)$ and $cost(v)$ for each node v in the new tree. By a DFS traversal of the tree, all $mr'(v)$ and $cost(v)$ information can be computed in linear time.

ExG-DME has higher time complexity than Greedy-DME. Recall that in Greedy-DME, the size of nearest-neighbor graph $|H|$ = the number of existing subtrees. Since ExG-DME allows a more flexible topology generation, $|H|$ is equal to the number of nodes in the existing subtrees, which is between the number of sinks n and $2n$. Therefore, straightforward computation of all edge weights takes (n^2) time. It was shown in [7] that the nearest-neighbor graph will be constructed and used to merge the remaining subtrees $O(\log n)$ times, if the parameter k is a constant[9]. Thus, the time complexity of ExG-DME is $O(n^2 \log n)$. By using the bucket decomposition

[8] If v_i is the root of T_i, then $mr'(v_i) = mr(v_i)$, and $cost(v_i) = cost(T_i)$

[9] More specifically, the while-loop in Figure 6 will be repeated $\log_{1+\frac{1}{c_e k-1}} n$ times, where $c_e \geq 1$ is a constant. Following [7], we

method of [9], the nearest-neighbor graph can be constructed in linear time, so that the total time complexity can be reduced to $O(n \log n)$.

When the skew bound B is small, significant detour wiring will be required to maintain near-zero skew whenever we merge two subtrees at positions in their lower levels. Thus, in practice we need only consider tree edges in the upper levels (near the original roots) as possible locations for the repositioned roots. In our implementation, this heuristically depends on the skew bound B. When $B = 0$, ExG-DME only merges two subtrees at their roots, and has the same linear time complexity as Greedy-DME.

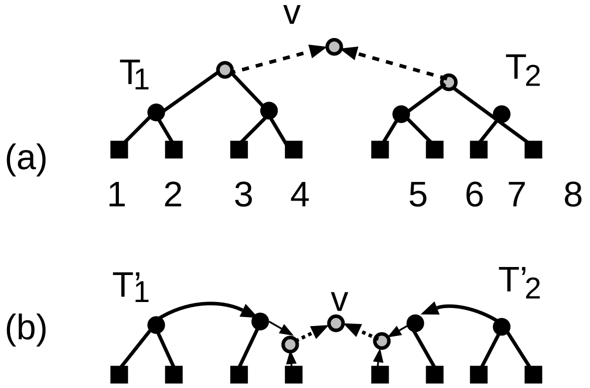


Figure 4: An example showing that given skew bound $B \gg 0$, changing the subtree topology before merging will reduce the merging cost. The eight sinks are located on a horizontal line and are equally spaced.

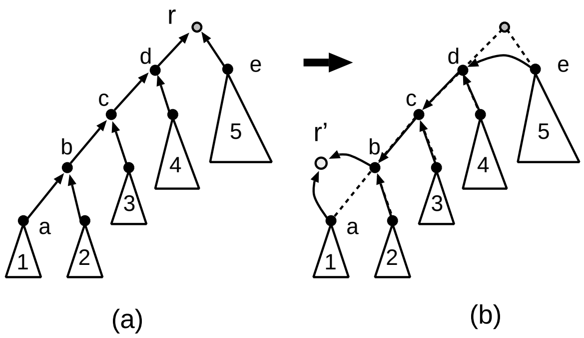


Figure 5: Rules for changing the topology.

5 The Third Tradeoff: Planar Routing

Our third and final heuristic addresses the BST variant where the topology can be arbitrary, but the output tree must be routable on a single layer. We propose an Extended Planar-DME (ExP-DME) algorithm which in the limit of $B = 0$ is identical to the Planar-DME algorithm, and which in the limit $B = \infty$ is identical to the standard SMT heuristic of edge-overlapping from a minimum spanning tree. This is the least imaginative of our three methods: we simply use Planar-DME to recursively partition the sink set until each sink cluster has

set $2 \leq k \leq 5$ in our experiments.

```
Algorithm ExG-DME(S, B)
Input: Set of sinks S; skew bound B
Output: BST T(S) with skew ≤ B
  n = |S| /* remaining unmerged subtrees */
  for all sinks v ∈ S
    cost(v) ← 0
    mr'(v) ← mr(v) ← {l(v)}
  endfor
  while (n > 1)
    Construct nearest-neighbor graph H for all nodes (in
    O(|H|^2) = O(n^2) time since edge weight can be computed
    in constant time with mr'(u), cost(u))
    A = set of sorted edges of H in non-decreasing
         order of edge weights
    m = MIN(MAX(1,n/k), n − 1));
    for i = 1 to m do
      Take edge e = (v_1, v_2) with smallest weight from A
      Delete all edges incident to v_1 or v_2
      Let T_1 (T_2) be the subtree containing node v_1(v_2)
      if v_1(v_2) is not the root of T_1 (T_2)
        Reposition roots of T_1 (T_2) at e_{v_1} (e_{v_2})
        Adjust tree topology accordingly
      endif
      Merge T_1 and T_2 as tree T'
      Construct mr(v) (in constant time)
      Update mr'(u),cost(u) ∀u ∈ T' (O(|T'|) time)
      n = n − 1 /* one less unmerged subtrees */
    endfor
  endwhile
  if clock source clk is given
    Reposition root of TR and adjust tree topology to
    minimize tree cost when clk connected to TR
  endif
  T(S) ← Find_Exact_Placements(TR)
```

Figure 6: The ExG-DME Algorithm.

radius less than B, then construct a planar radius-bounded tree over each cluster. A similar approach is used by Zhu and Dai [22], but we expect substantial cost savings (e.g., using Planar-DME instead of the method in [23] represents over 20% cost reduction).

To achieve the planar radius-bounded tree construction over the sinks in any given cluster, we first modify the KRY method of Khuller et al. [16], which is shown to be the best "shallow-light" construction possible ("shallow-light" indicates a construction that returns a spanning tree within constant factors of optimal in terms of both tree radius (shallowness) and tree cost (lightness)).[10] We employ simple modifications to KRY so that it does not cross the clock tree edge that leads into the given sink cluster. The spanning tree output by KRY may be converted to a Steiner tree by overlapping the embeddings of tree edges within the union of their bounding boxes. This has the advantage of preserving the spanning tree radius within the eventual Steiner tree output. Ho et al. [11] provided an optimal edge-overlapping construction, but it cannot always be applied to the KRY spanning

[10] Given a shortest paths tree (SPT) and a minimum spanning tree (MST) over S, along with a parameter $\epsilon > 0$, the KRY algorithm returns a spanning tree where the distance between any leaf and the root of the SPT is $\leq 1 + \epsilon \times$ (the shortest possible pathlength from that leaf to the root), and yet the tree cost is $\leq 1 + \frac{2}{\epsilon} \times$ (MST cost). Like earlier methods of Awerbuch et al. and Cong et al., the KRY method performs a depth-first traversal of a minimum spanning tree and adds partial connections directly back to the source if any sink is found to be too distant from the source.

trees because high-degree nodes may occur.[11] We therefore obtain Steiner trees from KRY trees by applying a greedy edge-overlapping algorithm. Our greedy method considers each pair of adjacent edges in the tree, and calculates the cost reduction achievable by optimally overlapping these two edges (i.e., inducing a Steiner point). The candidate Steiner point (i.e., the overlapping of two edges) which yields maximum cost savings is iteratively added until no further cost reduction is possible.[12]

The final step in our ExP-DME implementation is to remove crossing edges, which further reduces the tree cost and still preserves the spanning tree radius. Figure 7 demonstrates the removal of crossing edges. Assume edges $\overline{v_1v_2}$ and $\overline{u_1u_2}$ intersect at point w and that $v_2(u_2)$ is the parent of $v_1(u_1)$. Assume further without loss of generality, that the pathlength from the root (or the lowest common ancestor of v_1 and u_1) to v_1 is shorter than that to u_1. By replacing $\overline{v_1v_2}$ and $\overline{u_1u_2}$ with edges $\overline{v_1w}$, $\overline{wv_2}$, $\overline{u_1w}$, the edge crossing can be removed without increasing the tree radius. Furthermore, the tree cost is reduced by $|\overline{wu_2}|$. This number of operations is bounded by the number of edge crossings.

The resulting ExP-DME heuristic is described in Figure 8. When $B = 0$, the time complexity of ExP-DME is $O(n \log n)$ since ExP-DME is identical is to Planar-DME. When the skew bound increase, the running time of ExP-DME becomes dominated by the construction of the radius-bounded KRY tree and edge-overlapping heuristic. Experimentally, ExP-DME takes no more than 43 minutes for all benchmark examples with any skew bound.

Theorem 3: The routing tree $T(S)$ constructed by ExP-DME is skew-bounded and planar. $\square$

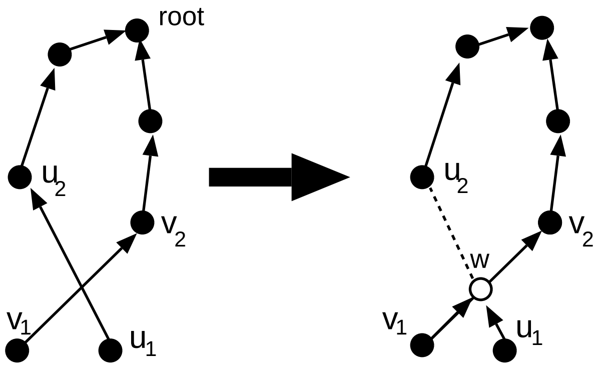


Figure 7: Removal of crossing edges.

[11] The method of Ho et al. [11] requires an input spanning tree to be *separable*, i.e., no pair of edge bounding boxes intersects except possibly at their boundaries. When the spanning tree has high-degree nodes, separability may fail to hold.

[12] The output of this heuristic is nearly identical to that of the optimal edge-overlapping algorithm of Ho et al. (called S-RST in [11]). For random 10-node instances, greedy edge-overlapping averages 8.8% cost reduction from an input minimum spanning tree, while S-RST is reported to average 9.0% reduction. For random 25-node instances, the greedy heuristic averages 9.3% percent cost reduction over the minimum spanning tree, while S-RST is reported to average 9.5% reduction.

Algorithm ExP-DME(S',$P_{S'}$,p, B)
Input: Set of sinks S'; convex polygon $P_{S'}$ containing S'; parent node p lying inside $P_{S'}$; skew bound B
Output: Planar BST T(S') with root v

if $radius(S') \leq B$
  $R \leftarrow \{w \mid d(w,u) \leq B,\ \forall u \in S', w \in P_{S'}\}$
  if $p = nil$
    $|e_v| = 0$
  else if p = clock source clk
    $|e_v| = d(R, clk)$
    $R = R \bigcap \{w \mid d(w, l(p)) \leq |e_v|\}$;
  else
    $|e_v| = radius(S_p) - B$, where S_p = set of sinks in T_p
    $R = R \bigcap \{w \mid d(w, l(p)) \leq |e_v|\}$;
  endif
  /* embed v – root of radius-bounded tree T' */
  /* R = possible placements of roots of T' */
  Choose any $l(v) \in R$
  Build MST T_1 & SPT T_2 over S' s.t. $\overline{pv}$ not crossed
  Build KRY tree T' with T_1 and T_2 as input
  Lower $cost(T')$ using edge-overlapping heuristic while not crossing clock edge $\overline{pv}$
  Remove crossing edges (Fig. 7)
else
  Call Planar-DME to embed node v and partition S' and $P_{S'}$ into S'_1,S'_2 and $P_{S'_1}$, $P_{S'_2}$
endif
ExP-DME(S'_1,$P_{S'_1}$,v)
ExP-DME(S'_2,$P_{S'_2}$,v)

Figure 8: The ExP-DME Algorithm.

6 Experimental Results

All of our algorithms are implemented in C on a Sun SPARC-10 workstation. We tested our methods on the seven benchmark examples prim1-prim2 [12], and r1-r5 [21].

Table 1 shows the total wirelengths of BSTs by ExG-DME/ExP-DME for different values of the skew bound. In the last row of the table are the results obtained by the Steiner heuristic in [2], which has only $O(n \log n)$ time complexity and closely matches the Iterated 1-Steiner solution quality [13]. The data for ExG-DME are the best results obtained using $k = 2$ to 5.

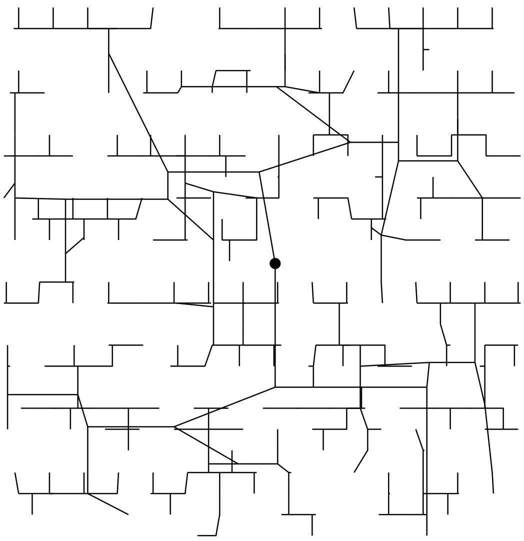
When $B = 0$, ExG-DME and ExP-DME are equivalent to the best-known non-planar and planar ZST algorithms [7, 15] in the literature. (Note that our numbers are slightly different from those reported in [7], since that paper uses the Elmore delay model.) When $B = \infty$, the Steiner trees constructed by ExG-DME have only average 0.21% higher cost than [2]; on the other hand, ExP-DME reduces an "MST + edge overlapping" heuristic [11] for the Steiner minimal tree problem.

When $0 < B < \infty$, both ExG-DME and ExP-DME obtain a smooth skew-cost tradeoff. The effectiveness of our methods is due to our novel topology generation strategy, which allows a very natural transition from zero-skew tree to minimum-Steiner tree.

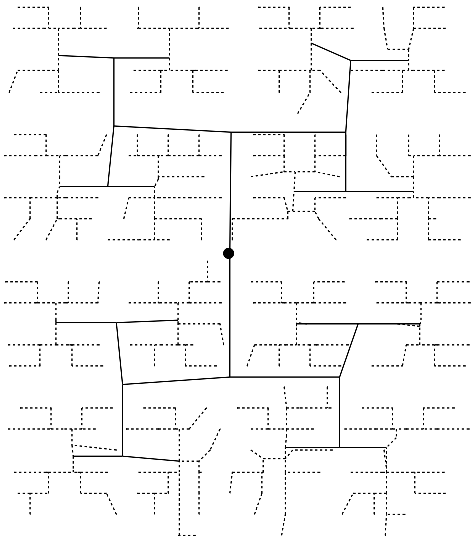
Finally, Figure 9 shows the routing solutions for benchmark prim1 constructed by ExG-DME and ExP-DME with skew bound equal to $100\mu m$.

Skew Bound	prim1	prim2	r1	r2	r3	r4	r5
	Total wirelengths of ExG-DME/ExP-DME						
0	130.8/134.6	311.3/347.7	1,323.9/1,511.8	2,581.1/3,284.3	3,316.1/3,943.9	6,690.0/7,810.6	9,871.5/11,491.1
.1	126.1/134.6	302.7/347.7	1,288.2/1,511.8	2,500.2/3,284.3	3,314.3/3,943.9	6,682.4/7,810.6	9,845.8/11,490.9
.2	123.6/133.3	293.2/338.8	1,295.9/1,511.7	2,551.7/3,283.9	3,307.0/3,943.6	6,608.1/7,809.9	9,815.5/11,488.7
.5	116.7/119.0	277.3/313.6	1,282.6/1,511.0	2,510.8/3,281.0	3283.8/3,940.0	6,523.9/7,800.5	9,693.7/11,468.6
1	109.6/111.2	255.0/283.3	1,265.9/1,505.7	2,466.5/3,267.7	3,211.3/3,917.5	6,391.0/7,744.2	9,550.3/11,370.4
2	96.9/104.5	232.1/255.7	1,237.7/1,484.2	2,423.3/3,221.1	3,177.4/3,830.3	6,239.5/7,527.3	9,201.6/11,009.6
5	88.5/94.8	201.1/229.1	1,176.0/1,360.7	2,314.8/3,007.7	2,959.5/3,510.6	5,858.5/6,841.6	8,613.6/10,066.9
10	81.6/91.9	184.9/208.1	1,093.5/1,226.0	2,164.4/2,663.7	2,70.0/3,200.6	5,407.2/6,316.2	7,964.9/9,354.9
20	78.8/85.5	176.5/191.4	1,014.9/1,124.0	1,962.1/2,395.5	2,473.6/2,977.0	4,842.6/5,995.6	7,245.5/8,760.1
50	78.8/82.2	171.2/182.0	875.3/1,044.1	1,780.1/2,179.6	2,233.4/2,786.3	4,399.5/5,635.1	6,538.4/8,430.6
100	78.8/82.2	171.2/178.2	829.3/888.6	1,643.8/1,876.2	2,042.6/2,655.7	4,195.4/5,380.7	6,146.1/8,085.9
200	78.8/82.2	171.2/175.0	772.3/820.2	1,512.1/1,591.4	1,945.5/2,062.1	3,985.7/4,549.1	5,982.0/6,615.8
500	78.8/82.2	171.2/174.2	772.3/780.4	1,500.0/1,537.1	1,906.7/1,948.4	3,793.3/4,004.9	5,595.4/5,849.8
1000	78.8/82.2	171.2/174.2	772.3/780.4	1,500.0/1,529.4	1,906.7/1,936.7	3,792.2/3,864.4	5,586.5/5,701.9
2000	78.8/82.2	171.2/174.2	772.3/780.4	1,500.0/1,529.4	1,906.7/1,932.7	3,792.2/3,849.8	5,584.3/5,676.6
∞	78.8/82.2	171.2/174.2	772.3/780.4	1,500.0/1,529.4	1,906.7/1,932.7	3,792.2/3,849.8	5,584.3/5,676.6
∞([2])	78.8	170.8	769.3	1,498.8	1,902.6	3,781.4	5,571.1

Table 1: Total wirelengths of ExG-DME and ExP-DME for different skew bound. The unit is $100\mu m$. When $B = \infty$, the results are competitive with those obtained by [2], shown in the last row.



(a) cost=109.6, non-planar



(b) cost=111.2, planar

Figure 9: Routing solutions by (a) ExG-DME and (b) ExP-DME for benchmark prim1 when the skew bound $B = 100\mu m$. In (b) the radius-bounded KRY trees are shown by the dotted line, the tree edges output by Planar-DME are shown by the solid line.

7 Conclusions

We have addressed the *bounded skew routing tree* (BST) problem, which has applications in the engineering design of clock distribution and global routing topologies. We believe that the tradeoffs we provide are effective, in that the limiting behaviors are essentially those of the best-known methods. Extensions of our methods to other monotone delay models such as Elmore delay are straightforward, given appropriate modification of the rules for building merging regions (e.g., under Elmore delay the boundary of merging regions may consist of rectilinear and parabolic segments.). Note that the underlying DME, Greedy-DME and Planar-DME methods have all been applied under Elmore delay in previous works.

Our Ex-DME algorithm is optimal for any prescribed topology G when (i) the skew bound $B = \infty$, or (ii) $B = 0$ and $FMS(JR(v)) \neq \emptyset$, $\forall v \in G$. For finite skew bounds, Ex-DME is not optimal and thus does not guarantee a monotone tradeoff between tree cost and skew bound. Our experimental results indicate that the tradeoff can be non-monotone especially as the B is increased slightly from zero (i.e., when we make the transition from DME to Ex-DME). We leave open the question of finding an approach which can provide optimal BST solutions for a fixed topology and any skew bound.

We have also described the ExG-DME and ExP-DME tradeoff heuristics, which address the BST variants with unrestricted topology, non-planar and planar routing, respectively. The key feature of ExG-DME is that the topology generation is more dynamic and flexible than in standard DME methods. Also, the asymptotic time complexity can be between $O(n)$ (when $B = 0$) and $O(n \log n)$ ($B = \infty$). (Our current implementation of ExG-DME has $O(n^2 \log n)$ complexity.) ExP-DME achieves a planar routing by constructing planar radius-bounded trees within convex polygons determined by Planar-DME. The skew bound B is satisfied because all root-sink pathlengths are between $\max\{radius(S), B\}$ and $\max\{radius(S) - B, 0\}$. One drawback of this approach is that when B is smaller than the minimum distance between any two sinks, then ExP-DME is identical to Planar-DME. A better approach using the concept of *planar merging regions* is under investigation.

Future work can integrate all of our constructions with parallel research thrusts in delay-constrained routing design, notably the optimal sizing of interconnect widths and the optimal placement and sizing of drivers [18].

References

[1] K. D. Boese and A. B. Kahng, "Zero-Skew Clock Routing Trees With Minimum Wirelength," *Proc. IEEE Intl. Conf. on ASIC*, 1992, pp. 1.1.1 - 1.1.5.

[2] M. Borah and R. M. Owens and M. J. Irwin, "An Edge-Based Heuristic for Rectilinear Steiner Trees", *Technical report* CSE-93-003, Department of Computer Science, Pennsylvania State University (also to appear in *IEEE Trans. on CAD*, Dec. 1994).

[3] T.-H. Chao, Y.-C. Hsu and J.-M. Ho, "Zero Skew Clock Net Routing," in *Proc. ACM/IEEE Design Automation Conf.*, 1992, pp. 518-523.

[4] T.-H. Chao, Y. C. Hsu, J. M. Ho, K. D. Boese and A. B. Kahng, "Zero Skew Clock Routing With Minimum Wirelength", *IEEE Trans. on Circuits and Systems* 39(11), November 1992, pp. 799-814.

[5] M. Edahiro, "Minimum Skew and Minimum Path Length Routing in VLSI Layout Design", *NEC Research and Development* 32(4), October 1991, pp. 569-575.

[6] M. Edahiro, "Minimum Path-Length Equi-Distant Routing", *Proc. IEEE Asia-Pacific Conf. on Circuits and Systems*, December 1992, pp. 41-46.

[7] M. Edahiro, "Clustering-Based Optimization Algorithm in Zero-Skew Routings", *Proc. ACM/IEEE Design Automation Conf.*, June 1993, pp. 612-616.

[8] M. Edahiro, "Delay Minimization for Zero-Skew Routing", *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1993, pp. 563-566.

[9] M. Edahiro, "An Efficient Zero-Skew Routing Algorithm", *Proc. ACM/IEEE Design Automation Conf.*, 1994, pp. 375-380.

[10] E. G. Friedman, "Clock Distribution Design in VLSI Circuits - An Overview", *Proc. IEEE Intl. Symp. on Circuits and Systems*, 1993, pp. 1475-1478.

[11] J.-M. Ho, G. Vijayan and C. K. Wong", "New Algorithms for the Rectilinear Steiner Tree Problem", *IEEE Trans. on CAD*, vol. 9, no. 2, 1990, pp. 185-193.

[12] M. A. B. Jackson, A. Srinivasan and E. S. Kuh, "Clock Routing for High Performance ICs," *Proc. ACM/IEEE Design Automation Conf.*, 1990, pp. 573-579.

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

[14] A. B. Kahng and G. Robins, *On Optimal Interconnections for VLSI*, Kluwer Academic Publishers, 1994.

[15] A. B. Kahng and C.-W. A. Tsao. "Planar-DME: Improved Planar Zero-Skew Clock Routing With Minimum Pathlength Delay," *Proc. ACM/IEEE European Design Automation Conf.*, September 1994.

[16] S. Khuller, B. Raghavachari, and N. Young, "Balancing Minimum Spanning and Shortest Path Trees", *Proc. ACM/SIAM Symp. Discrete Algorithms*, 1993, pp. 243-250

[17] N. Menezes, S. Pullela and L. T. Pillage, "Skew Reduction in Clock Trees Using Wire Width Optimization", *Proc. IEEE Custom Integrated Circuits Conf.*, 1993.

[18] S. Pullela, N. Menezes, J. Omar and L. T. Pillage, "Skew and Delay Optimization for Reliable Buffered Clock Trees", *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1993, pp. 556-562.

[19] S. Pullela, N. Menezes and L. T. Pillage, "Reliable Non-Zero Skew Clock Tree Using Wire Width Optimization", *Proc. ACM/IEEE Design Automation Conf.*, 1993, pp. 165-170.

[20] S. S. Sapetnekar, "RC Interconnect Optimization under the Elmore Delay Model", *Proc. ACM/IEEE Design Automation Conf.*, 1994, pp. 387-391.

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

[22] Q. Zhu and W.M. Dai, "Two-Level Clock Routing Scheme for High Performance and Low Power VLSI Chips and Multichip Modules", manuscript, 1994.

[23] Q. Zhu and W. W.-M. Dai, "Perfect-Balance Planar Clock Routing With Minimal Path-Length", *Proc. IEEE Intl. Conf. on Computer-Aided Design*, Nov. 1992, pp. 473-476.

[24] Q. Zhu and W.M. Dai, "Delay Bounded Minimum Steiner Tree Algorithms for Performance-Driven Routing", *UCSC-CRL-93-46*, Oct. 10, 1993

[25] Q. Zhu, W. M. Dai and J. G. Xi, "Optimal Sizing of High-Speed Clock Networks Based on Distributed RC and Lossy Transmission Line Models", *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1993, pp. 628-633.

Appendix: Proofs of Theorems

Lemma 1: Let $v \in G$ have children a and b with merging regions $mr(a)$ and $mr(b)$. If $mr(a)$ and $mr(b)$ are both well-behaved octilinear polygonal regions, then any rectilinear line segment $l \in JR(v) = JR(mr(a), mr(b))$ is well-behaved. □

Proof: Let $L_a = JS(mr(a))$ and $L_b = JS(mr(b))$.

Case 1: L_a and L_b are parallel Manhattan arcs

(See Figure 10a.) Let $d_1 = d(p, L_a)$ and $d_2 = d(p, L_b)$ for any point $p \in JR(v)$. Recall that all the points on L_a, and on L_b have the same max-delay and min-delay.

Observe that $d_1 + d_2 = d(L_a, L_b) = d$. By definition $t_{max}(p) = \max\{d_1 + t_{max}(L_a), d_2 + t_{max}(L_b)\}$ and $t_{min}(p) = \min\{d_1 + t_{min}(L_a), d_2 + t_{min}(L_b)\}$. In addition, for any rectilinear line segment $l \in JR(v)$, d_1 and $d_2 = d - d_1$ are both linear functions of p's location on l, with slope either $+1$ or -1. Thus, $t_{max}(p)$ and $t_{min}(p)$ are both piecewise linear functions (with slope everywhere being $+1$ or -1) of the position of p on l. Furthermore, $t_{max}(p)$ ($t_{min}(p)$) is a concave (convex) function with at most one turning point. Therefore, the segment l is well-behaved.

Case 2: L_a and L_b are parallel rectilinear line segments

(See Figure 10b.) Without loss of generality, we only discuss the case of L_a and L_b being vertical line segments. By the definition of delays for any point $p \in JR(v)$, on any horizontal line segment $l_h \in JR(v)$ the delays of any point on l_h are determined by the delays of the two endpoints of l_h and are well-behaved. Because both L_a and L_b are well-behaved vertical line segments, $t_{max}(p)$ and $t_{min}(p)$ defined over any vertical line segment l_v will be linear functions with at most one turning point (see Figure 10c). So l_v is also well behaved. □

To prove the following lemma and theorem, we need a little more terminology. If l is a well-behaved rectilinear line segment, then the *max-delay increasing section* (*max-delay decreasing section*) of l, denoted $maxd\_incr(l)$ ($maxd\_decr(l)$), is the interval of l such that the max-delay value increases (decreases) toward the upper or right endpoint of l with slope 1. The *min-delay decreasing section* and *max-delay decreasing section* of l, denoted $mind\_incr(l)$ and $mind\_decr(l)$, are defined similarly. The *skew increasing section* and *skew decreasing section* of l, denoted respectively $skew\_incr(l)$ and $skew\_decr(l)$, are also defined similarly except that slopes are $+2$ and -2. The *constant skew section* of l is denoted $skew\_const(l)$ (see Figure 11). It is possible for one or two of the intervals to be empty. For example, if $skew\_incr(l) = l$, then $skew\_decr(l) = skew\_const(l) = \emptyset$. Alternatively, if

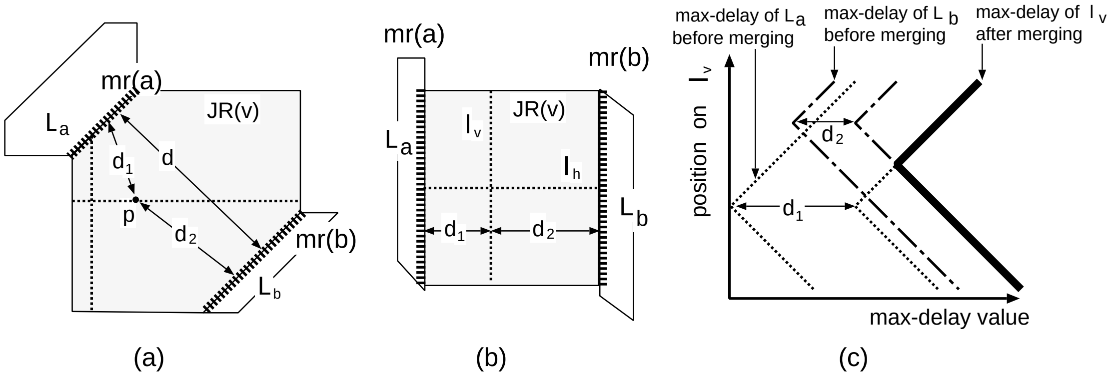


Figure 10: Lemma 1: Any rectilinear line segment $l \in JR(v)$ is well-behaved. In (c), we show that the $t_{max}(p)$ defined over l_v n (b) is a piecewise linear concave function if L_a and L_b are well-behaved.

$skew\_incr(l) \bigcup skew\_decr(l) = l$, then $skew\_const(l)$ is a single point.

From Figure 11, we can have the following fact:

Fact 1: If l is a well-behaved rectilinear line segment, then

- $skew\_incr(l) = maxd\_incr(l) \bigcap mind\_decr(l)$,
- $skew\_decr(l) = maxd\_decr(l) \bigcap mind\_incr(l)$, and
- $skew\_const(l)$ is given by either $maxd\_incr(l) \bigcap mind\_incr(l)$ or $maxd\_decr(l) \bigcap mind\_decr(l)$. □

We can now show the following result. While these are stated for vertical segments, the analogous results hold for the horizontal segments.

Lemma 2: Let $v \in G$ have children a and b with merging regions $mr(a)$ and $mr(b)$, which are both well-behaved octilinear polygonal regions. If $L_a = JS(mr(a))$ and $L_b = JS(mr(b))$ are parallel vertical line segments, then for any horizontal line segment $l_h = \overline{p_1 p_2}$ with $p_1 \in L_a$ and $p_2 \in L_b$, $skew\_const(l_h) \subseteq FMS(l_h) \subseteq FMS(JR(v))$. □

Proof: (See Figure 12.) If $skew\_const(l_h) = \emptyset$, then the lemma holds. Otherwise, we assume that $skew\_const(l_h) = \overline{qq'}$. Let $t'_{max}(p)$, $t'_{min}(p)$, and $skew'(p)$ denote the max-delay, min-delay, and skew value of point $p \in JR(v)$ before $mr(a)$ and $mr(b)$ were merged. Since $mr(a)$ and $mr(b)$ are sets of feasible merging points, $p_1 \in mr(a)$ and $p_2 \in mr(b)$ implying that $skew'(p_1) = t'_{max}(p_1) - t'_{min}(p_1) \leq B$ and $skew'(p_2) = t'_{max}(p_2) - t'_{min}(p_2) \leq B$.

From above $skew\_const(l_h)$ is either $maxd\_incr(l_h) \bigcap mind\_incr(l_h)$ or $maxd\_decr(l_h) \bigcap mind\_decr(l_h)$. Suppose that $skew\_const(l_h) = maxd\_incr(l_h) \bigcap mind\_incr(l_h)$, and let $d_1 = d(p_1, q)$ and $d_2 = d(p_2, q)$. As illustrated in Figure 11a, we then have $t_{max}(q) = \max_{k=1,2}\{t'_{max}(p_k) + d_k\} = t'_{max}(p_1) + d_1$ and $t_{min}(q) =$

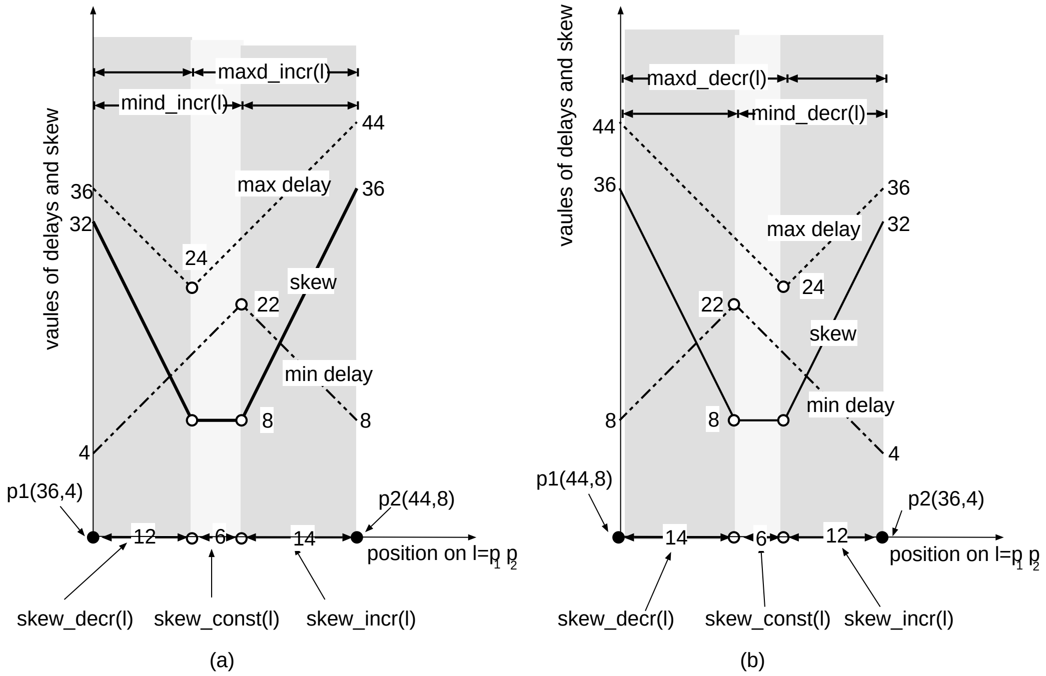


Figure 11: A well-behaved line segment $l = \overline{p_1 p_2}$ can be divided into three intervals, $skew\_incr(l)$, $skew\_const(l)$, and $skew\_decr(l)$, one or two of which might be empty. The coordinates associated with the endpoints of l represent (max-delay, min-delay). $|\overline{p_1 p_2}| = 32$. Interval $skew\_const(l)$, between the skew turing points (hollow dots) of l, is equal to $maxd\_incr(l) \bigcap mind\_incr(l)$ in (a) and equal to $maxd\_decr(l) \bigcap mind\_decr(l)$ in (b).

$\min_{k=1,2}\{t'_{min}(p_k) + d_k\} = t'_{min}(p_1) + d_1$, Hence, $skew(q) = t_{max}(q) - t_{min}(q) = t'_{max}(q) + d_1 - t'_{min}(q) - d_1 = skew'(p_1) \leq B$. Similarly, if $skew\_const(l_h) = maxd\_decr(l_h) \bigcap mind\_decr(l_h)$, we have the situation illustrated in Figure 11b. In this case, $t_{max}(q) = t'_{max}(p_2) + d_2$ and $t_{min}(q) = t'_{min}(p_2) + d_2$, and again, $skew(q) = skew'(p_2) \leq B$. Therefore, $skew\_const(l_h) \subseteq FMS(l_h) \subseteq FMS(JR(v))$. $\square$

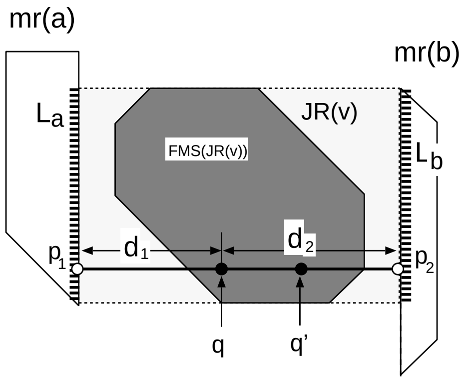


Figure 12: Lemma 2: $skew\_const(l_h) = \overline{qq'} \subseteq FMS(l_h) \subseteq FMS(JR(v))$

Theorem 1: Each merging region $mr(v)$ for a node $v \in G$ is a well-behaved octilinear polygonal region, and can be computed by construction rules M1-M5 in constant time. $\square$

Proof: Initially, each merging region is only a single point with zero max-delay and min-delay, and the theorem holds.

Assume v is an internal node with children a and b whose merging regions $mr(a)$ and $mr(b)$ are both well-behaved octilinear polygonal regions. Let $L_a = JS(mr(a))$ and $L_b = JS(mr(b))$. We will prove that the merging region $mr(v)$ is also a well-behaved polygonal region for each of the cases as we construct the merging region.

Case 1: L_a and L_b are parallel Manhattan arcs

All points on a Manhattan arc $l \in JR(v)$ that is parallel to the joining segments will have the same max-delay, min-delay and skew values. Any such Manhattan arc that contains a feasible merging point will therefore belong to $FMS(JR(v))$. Let F be the set of feasible merging sections of rectilinear boundary segments of $JR(v)$. If $F \neq \emptyset$, $mr(v)$ is the smallest octilinear polygonal region containing F. By Lemma 1, the rectilinear boundary segments of $mr(v)$ are well-behaved. Furthermore, a boundary segment l that is a Manhattan arc has identical max-delay and min-delay values at all the points. Thus, $mr(v)$ is well-behaved.

On the other hand, if $F = \emptyset$ then $MSS(JR(v))$ must be either L_a or L_b, both of which are well-behaved; Thus, $mr(v) = MSS(JR(v))$ is well-behaved.

Case 2: L_a and L_b are parallel rectilinear line segments

Without loss of generality, we assume parallel vertical line segments with L_a to the left of L_b.

By definition, $FMS(JR(v))$ is the union of the feasible merging sections of all horizontal line segments in the joining region. We say that the *infeasible merging region* of $JR(v)$ is given by $IMS(JR(v)) = JR(v) - FMS(JR(v))$. This infeasible merging region $IMS(JR(v))$ consists of several simple polygonal regions, called *strictly monotone regions*, each of which is bounded by the boundary segments of $mr(v)$ and $JR(v)$ (see Figure 13). Note the following:

Fact 2: By Lemma 2, within a strictly monotone region between $FMS(JR(v))$ and L_a (L_b), the skew value is monotonically increasing in the direction from right to left (from left to right). □

Fact 3: By Fact 1, for any horizontal line segment $l_h \in JR(v)$, $skew\_incr(l) \subseteq maxd\_incr(l)$, $skew\_incr(l) \subseteq mind\_decr(l)$, $skew\_decr(l) \subseteq maxd\_decr(l)$, and $skew\_decr(l) \subseteq mind\_incr(l)$. □

Fact 4: By Facts 2 and 3, in a strictly monotone region the max-delay value within M will increase in the same direction in which the skew value increases, and the min-delay value will increase in the opposite direction. □

Fact 5: For any point p on the boundary between $FMS(JR(v))$ and the strictly monotone regions, $skew(p) = B$. □

Consider without loss generality the example in Figure 13. When $FMS(JR(v)) \neq \emptyset$. Let l be a boundary segment between $FMS(JR(v))$ and a strictly monotone region M. Let l' be another boundary segment of M such that the maximum and minimum y-coordinates of l and l' are the same. From the above facts:

- If $l' \subseteq skew\_incr(L_a)$ ($skew\_incr(L_b)$), then l is a Manhattan arc with slope +1 (−1).
- If $l' \subseteq skew\_decr(L_a)$ ($skew\_incr(L_b)$), then l is a Manhattan arc with slope −1 (+1).
- By Fact 4, if l is a Manhattan arc, then all points on l have the same max-delay and min-delay values, i.e., l is well-behaved.
- If $l' = skew\_const(L_a)$ or $skew\_const(L_b)$, then l is a vertical line segment. By Lemma 1, l is well-behaved. Furthermore, the endpoints of l are opposite the skew turning points of L_a or (L_b).

Hence, $mr(v) = FMS(JR(v)) \neq \emptyset$ is a well-behaved octilinear polygonal region. The vertices of $FMS(JR(v))$ all lie on the feasible merging sections of the boundary segments of $JR(v)$ or opposite the skew turning points of L_a and L_b.

If $FMS(JR(v)) = \emptyset$, then all of $JR(v))$ will be a strictly monotone region where the skew value will strictly monotone increase either from right to left or from left to right. If the skew value increases from right to left then $mr(v) = MSS(JR(v)) = MSS(L_a)$; otherwise, $mr(v) = MSS(L_b)$. In either event, $mr(v)$ is a well-behaved line segment.

So we conclude that $mr(v)$ is a well-behaved octilinear polygonal region. By the discussion in Section 3.2, $mr(v)$ can be obtained using construction rules M1-M5 in constant time. □

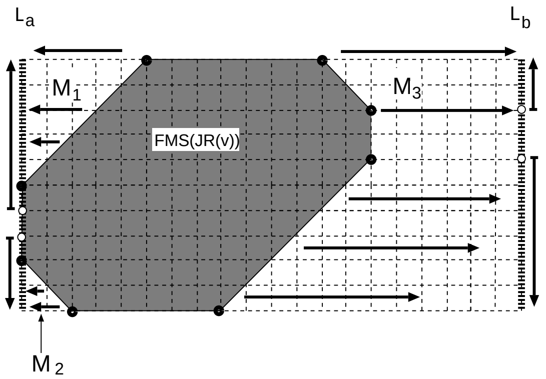


Figure 13: An example where merging region $mr(v) = FMS(JR(v)) \neq \emptyset$. Region $JR(v)$ is divided by $mr(v)$ into three strictly monotone regions M_1, M_2, and M_3. Regions M_1 and M_2 are horizontally skew decreasing. and region M_3 is horizontally skew increasing. The hollow points are the skew turning points of L_a and L_b. The arrows indicate the direction in which skew increases.

Theorem 2: When $B = \infty$, for any sink set S and topology G, Ex-DME returns a Steiner tree over S with minimum cost for topology G. □

Proof: Our proof is similar in structure to the proof of Theorem 1 [4]. Notice that Ex-DME places only two conditions on the placement of a node $v \in G$: (i) $l(v) \in mr(v)$, and (ii) $d(l(p), l(v)) = d(l(p), mr(v))$, where node p is the parent of node v. Suppose T is a Steiner tree over S with minimum cost for topology G, but contains a node placement violating one of the two conditions. We define T_v as the subtree in T which is rooted at node v. Let v be a node with greatest depth in T that violates either condition. Also let a and b be the children of node v. Because v has maximum depth, subtrees T_a and T_b do not violate Ex-DME constraints. We will prove below that there will be another tree T' such that $cost(T') \leq cost(T)$ and neither condition is violated by subtree $T'_v \in T'$.

Assume that node v violates condition (i), i.e., $l(v) \notin mr(v)$. We construct a new tree T' by first copying tree T to T' and then changing subtree $T'_v \subseteq T'$ as follows. First, move node v to location v' which is the point in $mr(v)$ that is closest to the original location of v, $l(v)$. Then, move v's children a and b to locations $a' \in mr(a)$ and $b' \in mr(b)$ which are closest to v'. Finally, use Ex-DME to construct the new subtrees T'_a and T'_b which are rooted at new locations a' and b', respectively. From Figure 14, it is fairly clear that $|\overline{a'v'}| + |\overline{b'v'}| + |\overline{v'v}| \leq |\overline{av}| + |\overline{bv}|$. In general, this claim is true under Manhattan distance metrics because all merging regions are rectangles when $B = \infty$. Therefore, $|\overline{a'v'}| + |\overline{b'v'}| + |\overline{v'p}| \leq |\overline{a'v'}| + |\overline{b'v'}| + |\overline{v'v}| + |\overline{vp}| \leq |\overline{av}| + |\overline{bv}| + |\overline{vp}|$. Also, $cost(T'_a) = cost(T_a)$ and $cost(T'_b) = cost(T_b)$ since T'_a and T'_b are produced by Ex-DME and thus do not violate the Ex-DME conditions. Therefore, $cost(T') \leq cost(T)$.

Now, we can assume that node v does not violates condition (i). If node v violates condition (ii), then we simply move node v to new location $v' \in mr(v)$ which is closest to the location of v's parent p, and use Ex-DME to construct subtree $T'_v \in T'$ such that T'_v is rooted at location v'. Since $cost(T_v) = cost(T'_v)$ and $|\overline{v'p}| \leq |\overline{vp}|$, so we have $cost(T') \leq cost(T)$.

Thus, if either Ex-DME condition is violated by any tree T, we can construct a tree T' such that $cost(T') \leq cost(T)$ and no Ex-DME constraints are violated. We conclude that Ex-DME can construct a Steiner minimal tree over S with topology G. $\square$

Theorem 3: The routing tree $T(S)$ constructed by ExP-DME is skew-bounded and planar. $\square$

Proof: The skew bound B is satisfied because all root-sink pathlengths are between $\max\{radius(S), B\}$ and $\max\{radius(S) - B, 0\}$.

Let $S' \subseteq S$ be a sink set determined by Planar-DME for the construction of a radius-bounded tree $T(S')$. Planar-DME guarantees that sinks in S' are contained in a polygon $P_{S'}$, and all existing edges are outside $P_{S'}$ except for a single edge l that connects the root of $T(S')$ to the portion of the tree outside $P_{S'}$.

In the following, we use $E(T)$ to denote the set of edges in a tree T. Let $F = \{e \mid e \in T(S),\ e \notin P_{S'}\} + \{l\}$.

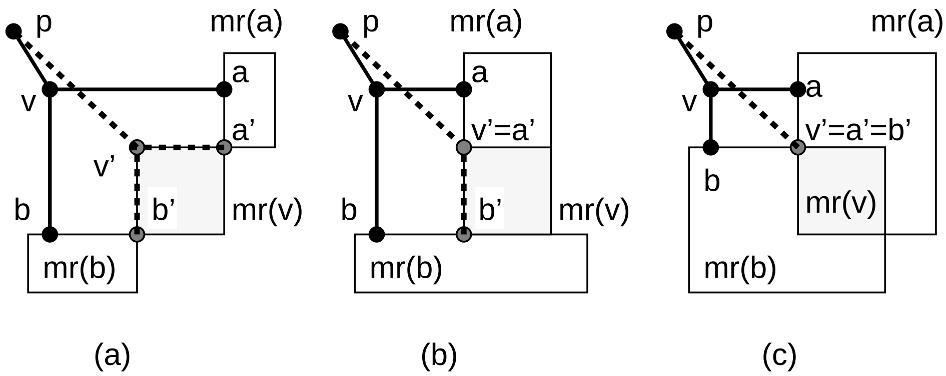


Figure 14: Examples showing that any Steiner tree T which violates condition (i), i.e., $l(v) \notin mr(v)$ can be transformed into another tree T' such that T' satisfies condition (i), and $cost(T') \leq cost(T)$. For convenience, we simply use p, v, a, and v to denote both the nodes and their original locations in tree T. Note that when the skew bound $B = \infty$, all merging regions are rectangles (shown as the shaded regions).

Since sink set S' is contained inside $P_{S'}$, we can construct an MST T_M and an SPT T_S such that edges in $E(T_M)$ and $E(T_S)$ will not cross the edges in F. Let T_K denote the KRY tree constructed using T_M and T_S as inputs. Since $E(T_K) \subseteq E(T_M) \bigcup E(T_S)$, edges in T_K will not cross edges in F. The edge-overlapping heuristic is applied when the newly generated tree edges will not cross edges in F. Thus only edges in T_K will cross each other. Since the crossing edges are removed without increasing the tree radius (recall Figure 7), then $T(S')$ will be skew-bounded and planar. Therefore, $T(S)$ is planar. $\square$