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 topology *Extended-DME* (Ex-DME) extends the DME algorithm for exact zero-skew trees via the concept of a *merging region*. (2) For arbitrary topology and arbitrary embedding, *Extended Greedy-DME* (ExG-DME) very closely matches the best known heuristics for the zero-skew case, and for the infinite-skew case (i.e., the Steiner minimal tree problem). (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, and 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 smooth cost-skew tradeoff that enable good engineering solutions.

1 Introduction

Control of signal delay skew has become a dominant objective in clock distribution routing of large global nets. Other objectives also require attention, minimizing wire area or ensuring planar-embeddability (i.e., routability on a single layer). 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 [17, 18, 16], Edahiro [8], and Zhu and Dai [21, 20] use wiresizing to optimize source-sink signal delays in clock distribution; [17] also performs buffer optimization to minimize power dissipation. The works of Zhu and Dai [21, 20] and Pullela et al. [18] propose construction of initial *non-zero skew* clock routing solutions which can be sized to achieve a prescribed skew bound. Our study addresses the underlying *bounded-skew* routing construction. As in [21, 20] we solve the bounded-skew problem under the linear delay model; our methods can also extend to Elmore or other delay approximations. We also note that exact zero skew is never an actual design requirement [13], so engineering tradeoffs also motivate a bounded-skew, rather than exact zero-skew, formulation. We now define the *bounded-skew routing tree* (BST) problem under the linear delay model.

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

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 $\leq B$.*

We also consider the BST variant where no topology G has been prescribed. This is useful in achieving a planar routing, or in fully exploiting the available skew bound B. For this variant, in either the non-planar or the planar case, we typically assume that the source location s_0 is also unspecified.

2 A Review of Three DME Variants

The Deferred-Merge Embedding (DME) algorithm, proposed independently in [5, 3, 1], 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). For linear delay, DME is *optimal*: it returns a tree with minimum cost and minimum source-sink pathlength for any input sink set S and topology G.

We now review DME and its Greedy-DME and Planar-DME variants, following [1, 14]. 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. The segment $ms(v)$ will always be a *Manhattan arc*, i.e., a segment with possibly zero length that has slope $+1$ or -1. 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 $|e_a| + |e_b|$ while preserving zero skew. Given the tree of merging segments, 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 $l(v)$; 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.

The Greedy-DME Algorithm

Note that DME requires an input topology. Several works [1, 3, 6] have thus studied topology constructions that lead to low-cost routing solutions when DME is applied; the most successful is the "Greedy-DME" method of Edahiro [6], which determines the topology of the merging tree in a greedy bottom-up fashion. 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., $ms(u)$ and $ms(v)$ such that $d(ms(u), ms(v))$ is minimum. A new parent merging segment $ms(v)$ is computed for node v from a zero-skew merge of $ms(u)$ and $ms(v)$; K is updated by adding $ms(v)$ and deleting both $ms(u)$ and $ms(v)$. After $n-1$ operations, K consists of the merging segment for the root of the topology.

In [7], $O(n \log n)$ time complexity was achieved by finding several nearest-neighbor pairs at once, i.e., 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. The solution is improved by a post-processing local search that adjusts the resulting topology (cf. "CL + I6" in [7]). Greedy-DME achieves 20% reduction in wiring cost compared with the methods of [3].

The Planar-DME Algorithm

Finally, the Planar-DME algorithm of Kahng and Tsao [14] determines node embeddings and connection topology by top-down partitioning of the routing area and the sink set. Given $S' \subseteq S$ and a convex polygon $P_{S'}$ containing S', Planar-DME 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. Noninterfering wiring implies a planar solution.

3 First Tradeoff: 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 the Merging Region

We will use the following terminology. A *rectilinear* line segment is a horizontal or vertical line segment. An *octilinear polygon* is a convex polygon formed by Manhattan arcs or rectilinear line segments. Such a polygon, along with its interior, defines an *octilinear region*. The *joining region* of two disjoint octilinear 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 (see Figure 1).[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. The skew of point p, denoted $skew(p)$, is defined to be $t_{max}(p) - t_{min}(p)$. If all points of a pointset P have identical max-delay and min-delay, we similarly use the terms $t_{max}(P)$, $t_{min}(P)$ and $skew(P)$. 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 minimum skew.

A rectilinear line segment l is *well-behaved* if (i) $t_{max}(p)$ and $t_{min}(p)$ are both piecewise linear (with slope $+1$ or -1) 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 it is a well-behaved rectilinear line segment or a well-behaved Manhattan arc. Finally, an octilinear region is well-behaved if its boundary segments are well-behaved. Given a well-behaved octilinear region P, we can compute the max-delay and min-delay of any interior point from the max-delay and min-delay values of P's vertices.

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

For convenience, in the following we denote $JR(mr(a), mr(b))$ as $JR(v)$ and let $L_a = JS(mr(a))$ and $L_b = JS(mr(b))$ for each node $v \in G$ with children a and b.

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

1. Suppose v is a sink s_i. Then $mr(v) = \{s_i\}$.
2. Suppose v is an internal node with children a and b. Then if $FMS(JR(v)) \neq \emptyset$, $mr(v) = FMS(JR(v))$; otherwise, $mr(v) = MSS(JR(v))$.

Similarly, the max-delay and min-delay of a point p in $mr(v)$ are defined recursively as follows:

1. Suppose v is a sink s_i. Then $t_{max}(v) = t_{min}(v) = 0$.
2. Suppose v is an internal node with children a and b. Then
   (a) $t_{max}(p) = \max_{k=a,b}\{t_{max}(p_k) + d(p,p_k)\}$, and $t_{min}(p) = \min_{k=a,b}\{t_{min}(p_k) + d(p,p_k)\}$, where p_k is the point in $mr(k)$ which is closest to p.
   (b) In the case where $FMS(JR(v)) = \emptyset$, $skew(mr(v)) > B$. To meet the skew bound constraint with least increase $\delta = skew(mr(v)) - B$ in merging cost, we set edge lengths as follows. 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 computed as $t_{max}(p) = \max_{k=a,b}\{t_{max}(p_k) + |e_k|\}$ and $t_{min}(p) = \min_{k=a,b}\{t_{min}(p_k) + |e_k|\}$, where p_k is as defined in 2(a).

In case 2(b), we say that edge e_a (e_b) requires *detouring wiring* if $|e_a| > d(mr(a), mr(b))$ ($|e_b| > d(mr(a), mr(b))$). Notice that if $FMS(JR(v)) \neq \emptyset$, then $mr(v)$ is the set of feasible merging points with minimum merging cost. Otherwise, $mr(v)$ may not have this property since some points outside $mr(v)$ may have lower merging cost than points in $JR(v)$.

[1] $JS(P)$ and $JS(Q)$ cannot be a pair of a Manhattan arc and a rectilinear line segment unless both $JS(P)$ and $JS(Q)$ are single points.

3.2 Construction of the Merging Region

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

M1 Compute $JR(v)$ and then compute $FMS(l)$ for each rectilinear boundary segment l of $JR(v)$.

M2 If L_a and 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 (see Figure 1b).

M3 Let F be the set of feasible merging sections computed by M1 and M2. If $F \neq \emptyset$, then construct $mr(v)$ equal to the smallest octilinear region containing F. If $F = \emptyset$, then if $skew(MSS(L_a)) \leq skew(MSS(L_b))$, $mr(v) = MSS(L_a)$; otherwise, $mr(v) = MSS(L_b)$.

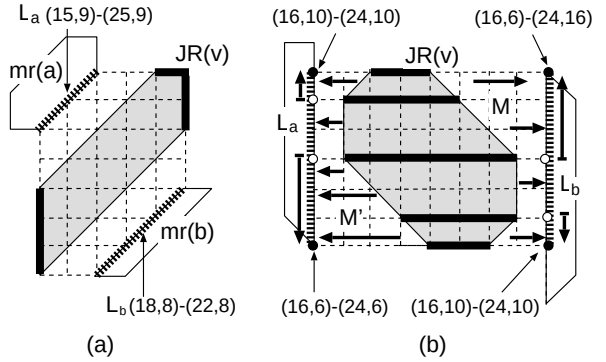


Figure 1: Construction of merging region $mr(v)$, shown as shaded regions, given the merging regions for v's children a and b. The joining region $JR(v)$ is shown as the gridded area. The joining segments L_a and L_b, shown as thick dotted lines, are parallel Manhattan arcs in (a) and parallel vertical line segments in (b). The hollow points in (b) are the skew turning points of L_a and L_b. The first and second coordinate pair associated with points on L_a and L_b represent (max-delay, min-delay) before and after merging, respectively. In (b), $JR(v)$ is divided by $mr(v)$ into two strictly monotone regions M and M'. Arrows indicate the direction of increasing skew in the strictly monotone regions.

The Ex-DME algorithm is outlined in Figure 2. Besides the merging region construction rules, there are two main differences between Ex-DME and DME. (1) When merging cost for the subtrees rooted at nodes a and b is greater than the lower bound $d(mr(a), mr(b))$ (i.e., detour wiring is needed), then the edge lengths $|e_a|$ and $|e_b|$ will be determined in the bottom-up phase of Ex-DME. Otherwise, the edge lengths will be determined in the top-down phase. (2) Each node v is embedded at the location in $mr(v)$ that is closest to the location of its parent p, even if $|e_v| > d(mr(v), l(p))$.

Let $v \in G$ have children a and b with merging regions $mr(a)$ and $mr(b)$ which are both well-behaved octilinear regions. We show the following properties of $JR(v)$ in [12].

Lemma 1: Any rectilinear line segment $l \in JR(v)$ is well-behaved. □

Lemma 2: Suppose L_a and L_b are parallel vertical line segments, and l_h is a horizontal line segment in $JR(v)$. Denote $skew\_const(l_h)$ as the portion of l_h with constant skew. Then $skew\_const(l_h) \subseteq FMS(l_h) \subseteq FMS(JR(v))$. □

Procedure Build_Tree_of_Merging_Regions (G,S,B)
Input: Topology G, set of sink locations S, and skew bound B
Output: Tree of merging regions TR

```
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-M3
    if FMS(JR(v)) = ∅ /* detour wiring needed */
      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
    else
      /* |e_a| and |e_b| determined in top-down phase */
```

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

Procedure Find_Exact_Placements(TR)
Input: Tree of merging regions TR
Output: BST $T(S)$ with skew $\leq B$; $|e_v|, \forall v \in G$

```
for each internal node v in G (top-down order)
  if v is the root
    Choose 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 l(p)
```

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

Figure 2: The Ex-DME Algorithm.

By Lemma 1, it is easy to show that any merging region will be a well-behaved octilinear region if L_a and L_b are parallel Manhattan arcs as shown in Figure 1a. Thus, we consider the case when L_a and L_b are parallel rectilinear line segments. As shown in Figure 1b, where $mr(v) = FMS(JR(v)) \neq \emptyset$ and the region $JR(v) - mr(v)$ consists of simple polygonal regions, which we call *strictly monotone regions*. From Lemma 2, we can infer that within each strictly monotone region, the skew and max-delay values increase monotonically from $mr(v)$ toward joining segment, L_a or L_b, while the min-delay value decreases monotonically in the same direction.

Let l and l' be two boundary segments of a strictly monotone region M with l being on $mr(v)$ and l' on L_a or L_b, such that the maximum and minimum y-coordinates of l and l' are the same. From $skew(l) = B$ and the properties of M, we have that l is a vertical line segment if $skew(l')$ is a constant; otherwise, l is a Manhattan arc with slope ± 1 depending on the direction in which the skew increases along l'. Therefore, $mr(v)$ is an octilinear region. Note that the vertices of $mr(v)$ either lie on the rectilinear boundary segments of $JR(v)$ or opposite the skew turning points of L_a and L_b. Thus, $mr(v)$ can be constructed by rules M1-M3. By the property of strictly monotone regions and Lemma 1, the boundary segments of $mr(v)$ are well-behaved.

Finally, if $FMS(JR(v)) = \emptyset$, then $mr(v) = MSS(JR(v))$ is a well-behaved line segment. The construction rules compute feasible merging sections (and minimum skew sections) for at most 8 (2) well-behaved line segments. Hence, $mr(v)$ can be computed in constant time. Therefore, we have

Theorem 1: Each merging region $mr(v)$ for a node $v \in G$ is a well-behaved octilinear region, and can be computed by construction rules M1-M3 in constant time. □

It is obvious that given sink set S, 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 is hence optimal for any prescribed topology. (Experimentally, $FMS(JR(v)) \neq \emptyset$ for most nodes $v \in G$ as long as G is a "good" topology, e.g., the one generated by ExG-DME described below.)

Note 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. Also we know that when $B = \infty$, all merging regions become rectangles. By induction on the maximum depth of any node that violates either of these conditions, we can show that any Steiner tree T which violates the Ex-DME conditions when $B = \infty$ can be transformed into another tree T' such that T' satisfies the Ex-DME conditions and has $cost(T') \leq cost(T)$. Thus, we have:

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. □

However, Ex-DME is not necessarily optimal for any intermediate value of B. A four-sink counterexample is given in [12].

4 Second Tradeoff: Unrestricted Case

We now consider the variant where the topology is not fixed and the embedding is unrestricted. Our *Extended Greedy-DME* (ExG-DME) algorithm matches the best known heuristic for the zero-skew limiting case, and very closely matches the performance of the best known heuristic [2] 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.

Recall that in DME, 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. Indeed, subtrees may be merged at non-root nodes as long as the resulting skew is $\leq B$. This flexibility allows reduced merging cost and is the key merit of the ExG-DME approach.

Consider the example in Figure 3a, where the eight sinks are equally spaced on a horizontal line. 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 in the top example. However, this topology is bad when B is large, even if the costs of the two subtrees are minimum. When the skew bound is large, ideally one should adjust the subtree topology so that the roots of subtrees become closer while the subtree costs remain the same or increase slightly. This is shown in the bottom example. More specifically, our method adjusts the tree topology by changing the position of the root as illustrated in Figure 3b. The root can be repositioned as the parent of nodes u and v, where u and v are the endpoints of any edge in the current tree. When we shift the root of the tree in this way, only a few tree edges will be removed or added so that the basic structure of the subtrees remains the same. In practice, the costs of the two subtrees will have little increase when the topologies are changed this way.

The ExG-DME algorithm follows the Greedy-DME structure, as shown in Figure 4. One key difference between ExG-DME and Greedy-DME is in the construction of the nearest-neighbor graph H. In Greedy-DME, each edge e_{uv} of H represents a possible merging pair of subtrees, T_u and T_v, rooted at u and v. The weight of edge e_{uv}, denoted $w(e_{uv})$, represents the merging cost of T_u and T_v, which can be computed in constant time given merging segments $ms(u)$ and $ms(v)$. In ExG-DME, each edge e_{uv} of H represents a possible way of merging two subtrees T_u (containing u) and T_v (containing v), where u and v are not necessarily the roots of their trees. Therefore, $|H|$ is equal to the number of nodes in the existing subtrees, which is between the number of sinks n and $2n$. Although $w(e_{uv})$ still represents the merging cost of T_u and T_v, the computation becomes somewhat more complicated. If T_u and T_v are the same tree, then $w(e_{uv}) = \infty$ (same trees cannot be merged). Otherwise, we first construct the new subtrees T'_u and T'_v with repositioned roots on edges e_u and e_v, then merge T'_u and T'_v into a new tree T. Then $w(e_{uv}) = cost(T) - cost(T_u) - cost(T_v)$. By maintaining two more variables $mr'(w)$ and $cost(w)$ for each node $w \in H$ we can still compute $w(e_{uv})$ in constant time.[2] Therefore, straightforward computation of all edge weights takes $O(n^2)$ time. 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 [7]. Thus, the time complexity of ExG-DME is $O(n^2 \log n)$. By using the bucket decomposition method of [9], the nearest-neighbor graph can be constructed in linear time, so that the total time complexity becomes $O(n \log n)$.

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, which edges are considered depends heuristically 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.

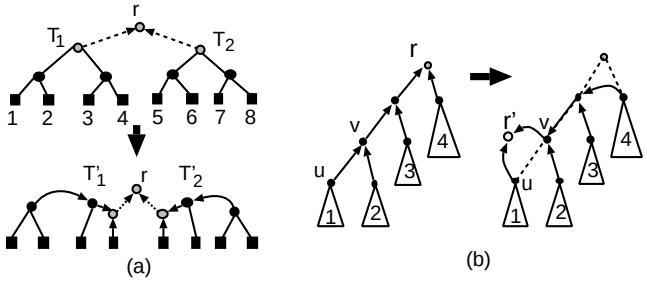


Figure 3: (a) An example showing that given skew bound $B \gg 0$, changing the subtree topology before merging will reduce the merging cost. (b) Repositioning the root in changing the topology.

5 Third Tradeoff: Planar Routing

Our final heuristic addresses the BST variant where the topology can be arbitrary, but the output tree must be routable on a single layer. Our *Extended Planar-DME* (ExP-DME) algorithm is identical to the Planar-DME algorithm in the limit of $B = 0$, and is

[2]Note that e_{uv} is an edge of nearest-neighbor graph H, while e_u, e_v are edges of topology G. The definitions of $mr'(v)$ and $cost(v)$ for node $v \in H$ are as follows. Let T_v be the merging tree containing node v, and let T'_v be the resulting adjusted tree after the root of T_v is relocated to edge e_v. Then $mr'(v)$ = the root of T'_v and $cost'(v) = cost(T'_v)$. Thus, for any edge e_{uv}, we can compute $w(e_{uv}) = d(mr'(u), mr'(v)) + cost(u) + cost(v) - cost(T_u) - cost(T_v)$ in constant time.

After T_u and T_v are merged into a new tree T, we have to update $mr'(w)$ and $cost(w)$ for each node $w \in T$. By a DFS traversal of the tree, all values $mr'(w)$ and $cost(w)$ can be computed in $(|T|)$ time; the resulting time complexity is dominated by the construction of H.

Algorithm ExG-DME(S, B)
Input: Set of sinks S; skew bound B; parameter k
Output: BST **T(S)** with skew $\leq B$

```
n ← |S| /* starting with n subtrees */
for all sinks v ∈ S
  cost(v) ← 0
  mr'(v) ← mr(v) ← {l(v)}
while (n > 1)
  Construct nearest-neighbor graph H in O(n^2) time
  A ← sorted edges of H in non-decreasing order of
      edge weight in O(n lg n) time
  for i = 1 to min{max{1, n/k}, n − 1} do
    Take edge e_uv with smallest weight from A
    Delete all edges incident to u or v
    Let T_u (T_v) be the subtree containing node u (v)
    if u (v) is not the root of T_u (T_v)
      Reposition the root of T_u (T_v) at e_u (e_v)
      Adjust tree topology accordingly
    T ← merge T_u and T_v in O(|T|) time
    Update mr'(w), cost(w) ∀w ∈ T in O(|T|) time
    n ← n − 1 /* one less subtree */
T(S) ← Find_Exact_Placements(T)
```

Figure 4: The ExG-DME Algorithm.

identical to the standard SMT heuristic of edge-overlapping from a minimum spanning tree in the limit $B = \infty$. 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 radius less than B, then construct a planar radius-bounded tree over each cluster. A similar approach is used by Zhu and Dai [21], but we expect substantial cost savings (e.g., using Planar-DME instead of the method in [19] represents over 20% cost reduction).

To achieve the planar radius-bounded tree construction over the sinks in any given cluster, we modify the KRY method of Khuller et al. [15], so that it does not cross the clock tree edge that leads into the given sink cluster.[3] 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 preserves the spanning tree radius within the eventual Steiner tree output. Our greedy edge-overlapping 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.[4]

The final step in ExP-DME removes edge crossings, which further reduces the tree cost and still preserves the spanning tree radius (see Figure 5). 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 lowest common ancestor of v_1 and u_1 to v_1 is shorter than that to u_1. Replacing $\overline{v_1v_2}$ and $\overline{u_1u_2}$ by $\overline{v_1w}$, $\overline{wv_2}$ and $\overline{u_1w}$ removes the edge crossing without increasing the tree radius. Furthermore, the tree cost is reduced by $|\overline{wu_2}|$. The number of operations is bounded by the number of edge crossings.

When $B = 0$, the time complexity of ExP-DME is $O(n \log n)$ since ExP-DME is identical to Planar-DME. When the skew bound increases, the running time of ExP-DME becomes dominated by KRY and edge-overlapping heuristics. Finally, in [12] we show that

Theorem 3: The routing tree $T(S)$ constructed by ExP-DME is planar and satisfies the skew bound B. □

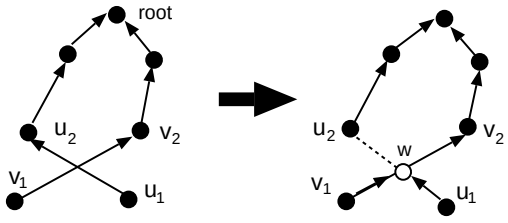


Figure 5: Removal of crossing edges.

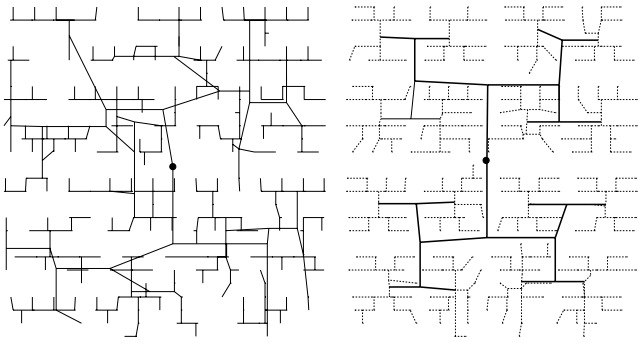
6 Experimental Results

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 and r1-r5. Table 1 shows the total wirelengths of BSTs constructed by ExG-DME/ExP-DME for different values of the skew bound. In the last row of the table gives the results obtained by the Steiner heuristic in [2], which has $O(n \log n)$ time complexity and is competitive with Iterated 1-Steiner [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, 14] in the literature. (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 average only 0.21% higher cost than [2]; on the other hand, ExP-DME is essentially similar to the "MST + edge overlapping" heuristic [11] for the Steiner minimal tree problem.

When $0 < B < \infty$, both ExG-DME and ExP-DME obtain smooth skew-cost tradeoffs. The effectiveness of these methods is due to their novel topology generation strategies, which allow very natural transitions from zero-skew tree to Steiner minimal tree.

Experimentally, for any skew bound ExG-DME takes less than 120 minutes and ExP-DME takes less than 43 minutes to solve all benchmarks on a Sparc-10 workstation. Figure 6 shows the routing solutions constructed by ExG-DME and ExP-DME for the prim1 benchmark with skew bound equal to $100\mu m$.



(a) cost=109.6, non-planar (b) cost=111.2, planar

Figure 6: 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 dotted lines; the tree edges output by Planar-DME are shown by think solid lines.

[3] KRY is a best possible "shallow-light" construction, where "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).

[4] Ho et al. [11] provided an optimal edge-overlapping construction, but it cannot always be applied to the KRY spanning trees because high-degree nodes may occur. The output of our edge-overlapping heuristic is nearly identical to that of the optimal edge-overlapping algorithm of Ho et al. (called S-RST in [11]), with about 0.2% average cost difference.

Skew Bound	Total wirelengths of ExG-DME/ExP-DME						
	prim1	prim2	r1	r2	r3	r4	r5
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	3,283.8/3,940.0	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 obtained by ExG-DME and ExP-DME for different skew bounds. The unit is $100\mu m$. When $B = \infty$, the results are competitive with those obtained by [2], shown in the last row.

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.[5] 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 experimental results indicate that the Ex-DME tradeoff can be non-monotone, especially as B is increased slightly from zero (i.e., when we make the transition from DME to Ex-DME). We leave open the questions of improving the approach, and finding optimal BST solutions for a fixed topology and any skew bound. Similarly, a drawback of ExP-DME 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.

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", *IEEE Trans. on CAD* 13(12), Dec. 1994, pp. 1563-1568.

[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] J. Cong and C-K Koh, "Minimum-Cost Bounded-Skew Clock Routing", to appear in *Proc. Int'l Symposium on Circuits and Systems*, May 1995.

[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] D. J. H. Huang, A. B. Kahng and C.-W. A. Tsao, "On the Bounded-Skew Clock and Steiner Routing Problems", UCLA CS Dept. *technical report* TR-940026x, 1994.

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

[14] 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.

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

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

[17] 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.

[18] 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.

[19] 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.

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

[21] Q. Zhu and W.M. Dai, manuscript, 1994.

[5] Note: Recently an independent study of the same problem has been given in [4]