Minimizing wirelength in zero and bounded skew clock trees *

Moses Charikar† Jon Kleinberg‡ Ravi Kumar§ Sridhar Rajagopalan‡ Amit Sahai¶
Andrew Tomkins‡

Abstract

An important problem in VLSI design is distributing a clock signal to synchronous elements in a VLSI circuit so that the signal arrives at all elements simultaneously. The signal is distributed by means of a clock routing tree rooted at a global clock source. The difference in length between the longest and shortest root-leaf path is called the *skew* of the tree. The problem is to construct a clock tree with zero skew (to achieve synchronicity) and minimal sum of edge lengths (so that circuit area and clock tree capacitance are minimized).

We give the first constant-factor approximation algorithms for this problem and its variants that arise in the VLSI context. For the zero skew problem in general metric spaces, we give an approximation algorithm with a performance guarantee of $2e$. For the L_1 version on the plane, we give an $(8/\ln 2)$-approximation algorithm.

1 Introduction.

A fundamental problem in VLSI design is *clock routing*, i.e., distributing a clock signal to synchronous elements in a VLSI circuit so that the signal arrives at all elements simultaneously. The signal is distributed by means of a clock routing tree rooted at a global clock source. The difference in length between the longest and shortest root-leaf path is called the *skew* of the tree. To achieve synchronicity, the skew should be zero. This is a significant issue in VLSI design, as non-zero clock skew has been estimated to account for over 10% of overall system cycle time in some high-performance systems [4]. Though it is easy to produce zero skew clock routing trees (see e.g., [5]), naive algorithms may produce trees that are expensive in terms of total *wirelength* (i.e., sum of the edge-lengths in the tree), thereby increasing circuit area and clock tree capacitance. Thus, the ideal clock tree routing algorithm would produce a zero skew clock tree with minimal total wirelength.

This problem, well studied in the VLSI community [15, 8, 10, 9, 24, 18, 25, 6, 16, 21, 7], is precisely the following variant of the classical *Steiner tree problem*:

> Find a Steiner tree, with a distinguished root, so that the lengths of all the root-leaf paths are the same and the sum of the length of edges in the tree is minimized.

*A preliminary version of this paper appeared in *Proc. 10th ACM-SIAM Symposium on Discrete Algorithms*, pp. 177–184, 1999.

†Department of Computer Science, Stanford University, Stanford, CA 94305. Email: moses@theory.stanford.edu. This work was done while the author was visiting IBM Almaden Research Center.

‡Department of Computer Science, Cornell University, Ithaca, NY 14850. Email: kleinber@cs.cornell.edu. This work was done while the author was visiting IBM Almaden Research Center. The author is also supported in part by an Alfred P. Sloan Research Fellowship and by NSF Faculty Early Career Development Award CCR-9701399.

§IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120. Email: {ravi, sridhar, tomkins}@almaden.ibm.com.

¶Laboratory for Computer Science, M.I.T., Cambridge, MA 02139. Email: amits@theory.lcs.mit.edu. This work was done while the author was visiting IBM Almaden Research Center. Also supported in part by a DOD/NDSEG fellowship and DARPA grant DABT63-96-C-0018.

While there are many proposed heuristics for attacking this problem and its variants (see, for instance, the papers cited above), there are no algorithms with non-trivial worst-case performance guarantees known. In this paper we give the first (constant-factor) approximation algorithms for constructing clock trees with zero skew (or a skew of at most a fixed bound), and wirelength as small as possible.

1.1 Clock routing problems.

We focus on the following three versions of the (zero or bounded skew) clock routing problem.

1. **(L_1 clock routing)** A clock signal must be distributed using horizontal and vertical wires on the plane from a source to a set of terminal points. The most common model of delay along a wire is the *linear model*, in which delay corresponds to length. Therefore the distance between points is exactly the L_1 distance. This is the standard formulation of the problem.

2. **(Planar L_1 clock routing)** In general, the embedding of a clock tree may have intersecting wires since the terminals are usually placed first, and then two layers of metal are available for the horizontal and vertical wires of the clock tree. This crossing of wires, however, may necessitate the introduction of many *vias*, or connections between layers, which causes both additional unmodeled delay and attenuation of the clock signal. Therefore one requires a planar-embeddable clock tree [15]. We therefore consider a second version of the routing problem (under the L_1 metric on the plane) with the requirement that the resulting clock tree be a planar embedding.

3. **(General Metric Space clock routing)** The above two versions model the clock routing problem for standard-cell or gate-array design methodologies, which have many small functional modules. In contrast to this, building-block design methodologies use larger functional blocks. These blocks are treated as obstacles and routing must be done in the spaces between blocks. The routing problem is formulated with respect to a graph, called the *channel intersection graph* (CIG) that represents the available routing area. In this model we can think of the terminals V as embedded in a metric space induced by the topology of the CIG. Therefore the third variant of the problem we study is routing a clock tree in an arbitrary metric space.

1.2 Preliminaries.

We are given a metric space M with distance function d, and a set V of points in M that are designated as *terminals*. As is standard, we define a *Steiner tree* for V to be a tree in M that contains each terminal in V as a node. (The nodes of the tree other than the terminals are referred to as *Steiner points*.) We say that a *clock tree* T for V is a Steiner tree with a distinguished vertex r called the *root*, such that every terminal $v \in V$ occurs as a leaf of T. The tree has an associated length function d_T that assigns a length to every edge in T, subject to the restriction that $d_T(u, v) \geq d(u, v)$ (*i.e.* the tree is allowed to stretch distances). If the metric space is the L_1 plane, for instance, the length of an edge (u, v) in T is at least the L_1 distance between u and v. The *cost* of the tree T is the sum of the lengths of all the edges of T. For $v \in V$, let the length of the path from the root to v be $\ell_v = d_T(r, v)$. The *skew* of T is $\max_{u,v \in V} |\ell_u - \ell_v|$. If T has skew $= 0$, we call it a *zero skew (clock) tree* (ZST) and if T has skew at most s, we call it an *s-skew tree*. (Note that, if necessary, this definition can be modified to allow terminals to be internal nodes of the tree; in the plane, we can instead slightly displace the internal node from the terminal. For general possibly discrete metric spaces, we allow multiple points in the tree to correspond to the same point of the metric space.)

Formally, the *zero* (resp. *bounded*) *skew clock tree problem* is stated as follows.

> Given a set V of terminals in a metric space M, find the minimum cost zero skew tree (resp. tree with skew at most s for a given bound s) for V.

When M is the L_1 plane, we refer to the L_1 variants of these problems. As discussed earlier, intersecting wires in the embedding might cause additional unmodeled delays. This motivates the *planar* variants of the above problem, where the tree T must be planar-embeddable (*i.e.* have no crossing edges).

Existing algorithms for clock routing in the L_1 plane make use of *snaking*, or wiggling an edge in order to lengthen it. Our definition of clock tree incorporates snaking by allowing $d_T(u, v) > d(u, v)$. Without this extension, no zero skew tree may exist. In our model feasibility is no longer a concern — any tree whose leaves are exactly the terminals can be "snaked" to a higher-cost zero skew tree.

The bounded skew clock tree problem is easily seen to be **NP**-complete by setting the skew to infinity so that the problem becomes the classical Steiner tree problem. The same reduction implies that the problem has no approximation scheme in general metrics unless $\mathbf{P} = \mathbf{NP}$. The zero skew problem is also **NP**-complete for general metric spaces. To our best knowledge, the hardness question of the planar zero skew problem is yet unsolved.

We will also refer to these problems as the zero or bounded skew clock routing problems.

1.3 Our results.

For the ZST problem in general metric spaces, we give an approximation algorithm with a performance guarantee of $2e$. We then give a constant-factor approximation algorithm for the bounded skew clock routing problem in general metric spaces. Finally we give an $(8/\ln 2)$-approximation algorithm for the planar ZST problem and a constant factor approximation algorithm for the planar embeddable bounded skew clock routing problem.

1.4 Organization.

Section 2 discusses some related work in clock routing. Section 3 presents a general lower bound for the optimal solution to the problem. This is used to obtain approximation guarantees for our algorithms. Section 4 (resp. Section 5) give the approximation algorithms for the zero (resp. bounded) skew clock routing problems. Section 6 presents an approximation algorithm for the planar ZST problem and Section 7 presents an approximation algorithm for the planar embeddable bounded skew clock routing problems. Section 8 discusses the hardness of the ZST problem.

2 Related work.

Algorithms for clock tree constructions come in two flavors — those that guarantee zero skew and the others that attempt to minimize the skew. Notice however that the aim is typically to minimize total wirelength.

The book by Kahng and Robins [15] contains a detailed account of many of the algorithms for clock tree constructions and several experimental results. The main emphasis on many of the algorithms, however, is to obtain practical solutions (which perform well on standard benchmarks, which in turn may or may not represent the average-case problem instance) rather than obtain solutions which have worst-case performance guarantees. We overview the most relevant algorithms below.

2.1 Minimizing clock skew.

In [21], given a floor plan of modules, a scheme to identify an entry point is presented. The optimal layout of the clock lines from the source to the entry points is determined by an exhaustive search (of course, with some pruning). No theoretical guarantees on the performance of the algorithm are given.

In [8], the authors obtain a clock routing scheme consisting of Manhattan segments with constraints (like blockages) on the routing layers. They obtain a divide-and-conquer algorithm which produces total wirelength of $1.5\sqrt{n}$ for n points distributed randomly on a uniform grid. Contrasting this with the largest possible wirelength of $\sqrt{n}+1$ for a rectilinear Steiner tree for the same distribution, they conclude that, on average, their algorithm is a $3/2$-approximation algorithm when compared to the minimum rectilinear Steiner tree.

Another algorithm for minimizing skew and wirelength based on matching is given in [6, 16]. They construct a binary tree using geometric matching and show that for cell-based designs, the total wirelength of their routing tree is on average, within a constant factor of the wirelength in an optimal Steiner tree. Their experiments suggest that the skew is near-zero on average.

2.2 Zero clock skew.

An exact zero-skew clock routing algorithm using the Elmore delay model is presented in [22, 23]. The zero skew is obtained by a bottom-up hierarchical approach via a zero-skew merging of the recursive solutions. The main emphasis is on experimental results.

A two-step approach to obtaining zero-skew while simultaneously minimizing wirelength is pioneered in [5]. In this, the authors present the *Deferred Merge Embedding* (DME) algorithm, which embeds any given connection topology to create a zero-skew clock tree. The wirelength is optimal for linear delay. The connection topology is generated by a top-down balanced bipartition (BB) approach. Though the DME algorithm can be shown to produce the optimal tree for a given topology, the BB approach is essentially a heuristic and has no performance guarantees.

3 A lower bound.

We first demonstrate a lower bound on the cost of the s-skew tree in any metric space. Let T be any rooted s-skew tree on the set V of terminals, with radius R'. Since T has skew at most s, the length of every root-leaf path is at least $R' - s$.

We define the *level* of a point $p \in T$ to be its distance from the root (so the root is at level 0). Consider some level $x \in [0, R' - s]$. If there are m points at level x in T then the m spheres of radius $R' - x$ centered at these points must cover all the vertices of V. This observation can be converted into a lower bound as follows. Let $n_V(R)$ be the minimum number of spheres of radius R needed to cover the vertices V. When the set is apparent from context, we suppress the subscript V.

Let Δ be the diameter of the set of vertices V, and Let R^* be the minimum value of x such that $n(x) = 1$ (thus $R^* > \Delta/2$.) Note that $R' \geq R^*$. Then the cost of the minimum cost s-skew tree must be at least:

$$\int_0^{R'-s} n(R'-x)dx = \int_s^{R'} n(R)dR \leq \int_s^{R^*} n(R)dR.$$

This lower bound, and the special case for $s = 0$, will be essential to analyzing our algorithms.

4 An approximation algorithm for general metric spaces.

In this section, we present a $2e$-approximation algorithm for the ZST problem in general arbitrary metric spaces (assuming that snaking is valid). The algorithm is randomized but can be derandomized easily. We place Steiner points on top of vertices from V. For ease of language, when we talk of using a vertex as an internal point in the tree, we mean to place a Steiner point at that vertex and use the Steiner point as the internal point in the tree.

Our algorithm repeatedly partitions the set of vertices to construct the tree. The partitioning proceeds by greedily placing balls of a certain radius $2R$ and grouping all vertices in the same ball together. To obtain

```
Algorithm Connect-Centers:
Initialize:  R := R_0; u_0 := s; U_0 := V; Ū := {U_0}; Ḡ := ∅; T := ∅; i := 0; R_old := Δ/2.
repeat until i = |V|
  S := V; i = 0.
  repeat until S = ∅
    pick g_i arbitrarily from S.
    let G_i be all vertices in S within distance 2R from g_i.
    let S := S \ G_i; i := i + 1.
  for j = 0 to i − 1 do
    let k be such that g_j ∈ U_k.
    add an edge from g_j to u_k of cost exactly 2R_old to T.
  R_old := R; R := R/r; Ū := Ḡ; Ḡ := ∅.
Output T.
```

Figure 1: Algorithm Connect-Centers.

more and more refined partitions, the process repeats with balls of smaller radii. We denote by r the factor by which the radii of balls decrease in each successive refinement of the partitioning process. We will describe our algorithm for any value of r and choose a specific (optimal) value for r at the end.

Algorithm Connect-Centers. Let Δ be the diameter of V. The algorithm first picks an arbitrary vertex s to be the root of the tree, and then chooses an initial partitioning radius $2R_0$ as follows. Let t be chosen uniformly at random from $[0, 1]$, and set $R_0 = (\Delta/2) \cdot \exp(-t \log r)$. The algorithm then proceeds as in Figure 1. At each point in the construction, we take an existing partition of the vertices $\bar{U}$ and refine it to $\bar{G}$. ($\bar{G}$ is not necessarily a strict refinement of $\bar{U}$.) Each set $G_i \in \bar{G}$ has a distinguished member g_i with the property that every $v \in G_i$ has $d(v, g_i) \leq 2R$. Similarly each $U_i \in \bar{U}$ has a member u_i such that every $v \in U_i$ has $d(v, u_i) \leq 2R_{\text{old}} = 2rR$. The tree we construct is denoted by T.

Remark. The algorithm as presented in Figure 1 is only weakly polynomial. But, by constraining R to be $\leq R_0/n^2$, we can obtain a strongly polynomial algorithm at the expense of $O(1/n)$ additive factor in the performance ratio. In fact, we can also modify the algorithm without requiring an additive factor. We omit the details in this version.

Analysis. It is immediate from the description of the algorithm that it will return a ZST, since each vertex is reached after the same number of levels, and the edges in each level are of identical cost. To analyze the cost of the tree produced by this algorithm, we observe:

Lemma 1 *Each time a new partition G is created the number of sets returned in the partition is at most $n_V(R)$.*

Proof. Let $G = \{G_0, \ldots, G_{m-1}\}$. We induct on m. If $m = 1$, there is nothing to prove. Otherwise, consider the $n = n_V(R)$ sets $S_1, S_2, \ldots, S_n$ of radius R that cover all the vertices V. Let S_j be the set that contains g_0. Since G_0 contains all vertices within radius $2R$ from g_0, it must contain all of S_j. Let $V' = V \setminus G_0$. Now, certainly, $n_V(R) \geq 1 + n_{V'}(R)$. But by induction, since the sets $G_1, G_2, \ldots, G_{m-1}$ are the result of a valid execution of the partitioning algorithm on V', it follows that $m - 1 \leq n_{V'}(R)$, and so the claim follows. Note that the claim also follows from the standard analysis for the p-center problem [12, 13, 14]. □

Thus, the total cost of connecting each g_i to some u_j is at most $2R_0 \cdot n(R) \leq 2rR \cdot n(R)$. The expected cost of the tree, therefore, can be seen as bounded by the integral:

$$\int 2rR \cdot n(R) d\mu$$

Here, μ is the probability measure of the algorithm using balls of radius $2R$. Now, recall that once the initial value R_0 for R is chosen, we know that all balls used in the algorithm will have radius $2R_0/r^i$ for some integer i. Note that the probability that R_0 lies in a small range $[x, x + dx]$ is

$$\frac{\ln(x + dx) - \ln(x)}{\ln(r)} = \frac{\ln(1 + dx/x)}{\ln(r)} = \frac{dx}{x \ln(r)}.$$

Thus, the integral above is

$$\int_0^{\Delta/2} \frac{2r}{\ln(r)} \cdot n(R) dR.$$

By our lower bound, the algorithm produces a tree that is at most $2r/\ln(r)$ times the optimal cost. A simple calculation shows that this is minimized when $r = e$, and hence we have

Theorem 2 *The above algorithm achieves an expected approximation ratio of* $2e$.

The basic randomization technique we employed in the algorithm for choosing R_0 has been used previously in [11, 19].

Relative costs of minimum Steiner trees and zero-skew trees. Using the lower bound we developed above, one can show that for n equally spaced points on a line, the optimal zero-skew tree has $\Theta(\log n)$ times the cost of the minimum Steiner tree. We now give a short proof that this is, asymptotically, the largest possible gap. For purposes of simplicity, we do not attempt to optimize the constants in the proof.

Theorem 3 *For a set V of n points in a metric space M, let $St(V)$ denote the cost of the minimum Steiner tree for V, and let $ZST(V)$ denote the minimum cost of a zero-skew tree for V. Then $ZST(V) \leq (4e \ln n + e) \cdot St(V)$.*

Proof. Recall that for a number R, $n(R)$ denotes the minimum number of balls of radius R needed to cover the points in V. We claim that for any R, $n(R) \leq 2St(V)/R$.

To prove the claim, begin with an optimal Steiner tree and convert it into a traveling salesman tour $\{v_{i_1}, \ldots, v_{i_n}\}$ of length at most $2St(V)$. We now construct a set $S \subseteq V$ as follows. We initially include v_{i_1}. Proceeding inductively, suppose S currently consists of $\{v_{i_1}, \ldots, v_{i_p}\}$. Let q denote the minimum index greater than p for which the length of the subtour from v_{i_p} to v_{i_q} is strictly greater than R; we add v_{i_q} to S and continue. At the end of this procedure, we observe that $|S| \leq 2St(V)/R$, since the distance along the traveling salesman tour between consecutive elements of S is at least R. Moreover, an arbitrary element $v_{i_j} \in V$ is within distance R of v_{i_t}, where t is the maximum index less than or equal to j for which $v_{i_t} \in S$; thus if we center a ball of radius R at each element of S, the resulting collection of balls covers V. The claim follows, since $n(R) \leq |S| \leq 2St(V)/R$.

Applying the claim, and using the fact that $St(V) \leq \Delta$, we have

$$\begin{aligned}
ZST(V) &\leq 2e\int_0^{\Delta/2} n(R)dR \\
&= 2e\int_0^{\Delta/2n} n(R)dR + 2e\int_{\Delta/2n}^{\Delta/2} n(R)dR \\
&\leq 2e\left(\frac{\Delta}{2n}\right)n + 4e\int_{\Delta/2n}^{\Delta/2} \frac{St(V)}{R}dR \\
&\leq e\Delta + 4e\cdot St(V)\int_{\Delta/2n}^{\Delta/2}\frac{dR}{R} \\
&= e\Delta + 4e\cdot St(V)\cdot \ln n \\
&\leq (4e\ln n + e)\cdot St(V).
\end{aligned}$$

□

5 Bounded skew clock routing.

We now present a constant factor approximation algorithm for the bounded skew clock routing problem. The algorithm proceeds in two phases. First, we construct a Steiner tree spanning V which we fragment into subtrees. Second, we connect these subtrees using a modification of Algorithm Connect-Centers.

We first construct a Steiner tree T' spanning V. To do this, we use the currently best known approximation algorithm for Steiner trees in general metric spaces due to Prömel and Steger [20]. In case the points are in the plane, we can use a PTAS for Steiner trees [1, 2], with an approximation ratio of $1+\epsilon$ for any $\epsilon > 0$. Let $W \subseteq V$ be a maximal subset of terminals such that the distance between any two of them in T' is at least s. W can be chosen by a greedy algorithm.

Lemma 4

$$|W| \leq 2\text{cost}(T')/s.$$

Proof. For $v \in W$, let B_v be a ball of radius $s/2$ about v, distances being computed in the metric induced by the tree T'. Then, for $u, v \in W$, $B_u \cap B_v = \emptyset$. Now, the Steiner tree T' has a path P_v of length $s/2$ within each ball B_v. (Here, P_v could include a fractional part of an edge.) The sum of the lengths of the paths P_v is at most cost(T'). Hence, the number of paths (and therefore, the number of vertices in W) is at most $2\text{cost}(T')/s$. □

For each $v \in W$, we construct a tree T_v rooted at v, such that the distance from v to every vertex in T_v is at most s. To do this, we order the vertices in W arbitrarily, say $W = \{v_1, \ldots, v_k\}$. Now, we assign every vertex in V to the closest vertex in W, breaking ties in favor of vertices with smaller indices. Here distances are computed in T'. Note that every vertex in V is within a distance of at most s from some vertex in W (by the maximality of W.) For $v \in V$, let $c(v)$ denote the vertex in W that it is assigned to; let P_v denote the path in T' from v to $c(v)$. The length of P_v is at most s.

Lemma 5 *For $v_1 \neq v_2$, if $c(v_1) \neq c(v_2)$, the paths P_{v_1} and P_{v_2} are edge disjoint.*

Proof. Suppose P_{v_1} and P_{v_2} share an edge. An easy case analysis shows that this contradicts the choice of either $c(v_1)$ or $c(v_2)$. □

For vertex $v \in W$, let $S(v)$ denote the set of vertices assigned to it. Let T_v be the subtree of T' that spans $S(v)$; in other words, $T_v = \cup_{u \in S(v)} P_u$. Then Lemma 5 implies that the subtrees T_v are disjoint. Clearly, we also have:

Lemma 6

$$\sum_{v \in W} \text{cost}(T_v) \leq \text{cost}(T').$$

Also, the distance from v to every vertex in T_v is at most s.

Now we describe how to modify Algorithm Connect-Centers using the subtrees T_v constructed above to produce the final tree with skew at most s. We execute Algorithm Connect-Centers, but stop the process of construction of the tree at the last step when $R < s$ for the first time (i.e., we stop *before* a value for R smaller than s is used to create a partition). Let R_f be the final value of R (so $R_f < s$). At this time, $\bar{U}$ is a partition of V such that every vertex in U_i is at a distance at most $2rR_f$ from u_i. Let T be the partial tree constructed by the algorithm so far. We will connect each of the subtrees T_v to the tree T in the following way: For $v \in W$, let i_v be such that $v \in U_{i_v}$. Connect T_v to the tree T by adding an edge of weight $2rR_f$ from v to u_{i_v}. It is easy to see that the tree so constructed has skew at most s.

Now, we shall analyze the cost of the tree we obtain. Let C_1 be the cost of the tree that the algorithm constructs until $R < s$ for the first time. Let C_2 be the total cost of all the edges from vertices $v \in W$ to u_{i_v}. Let C_3 be the total cost of the trees T_v for $v \in W$.

Then, by the previous analysis,

$$\mathbf{E}[C_1] \leq \frac{2r}{\ln r} \int_s^{\Delta/2} n(R) dR.$$

Also,

$$C_3 \leq \text{cost}(T')$$

Now,

$$\begin{aligned} \mathbf{E}[2rR_f] &= 2\int_s^{rs} \frac{1}{\ln r} dx \\ &= \frac{2(r-1)s}{\ln r}. \end{aligned}$$

Hence,

$$\begin{aligned} \mathbf{E}[C_2] &= |W| \cdot \mathbf{E}[2rR_f] \\ &= |W| \frac{2(r-1)s}{\ln r} \\ &\leq \frac{4(r-1)}{\ln r} \text{cost}(T'). \end{aligned}$$

Let OPT_{ST} be the cost of the optimal Steiner tree on the set of terminals. Since the Steiner tree T' is constructed using the algorithm of Prömel and Steger [20], this guarantees that

$$\text{cost}(T') \leq \frac{5}{3} \mathsf{OPT}_{ST}.$$

Let OPT be the cost of the optimal clock tree with skew at most s. Then, we have two lower bounds for OPT. First, the lower bound given in Section 3 can easily be seen to generalize to the case of bounded skew

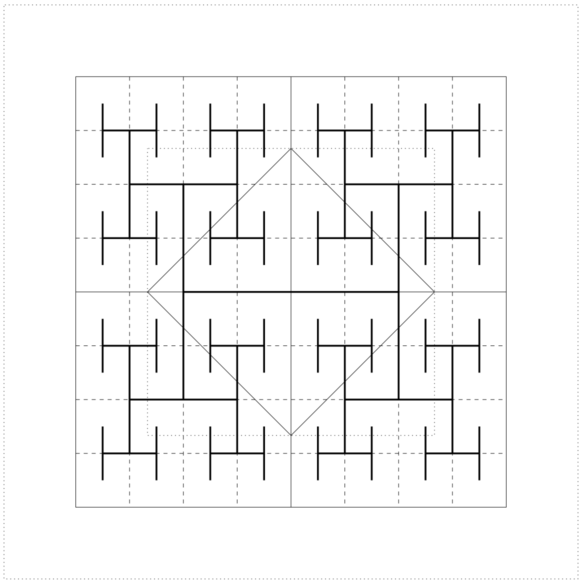

Figure 2: The first few levels of an H-tree.

as follows:

$$\begin{aligned} \mathsf{OPT} &\geq \int_s^{\Delta/2} n(R)dR. \\ \mathsf{OPT} &\geq \mathsf{OPT}_{ST}. \end{aligned}$$

Now, we can bound the expected cost of the tree we obtain in terms of OPT as follows:

$$\begin{aligned} \mathbf{E}[C_1 + C_2 + C_3] &\leq \left(\frac{2r}{\ln r} + \frac{5}{3}\left(1 + \frac{4(r-1)}{\ln r}\right)\right)\mathsf{OPT} \\ &\leq \left(\frac{5}{3} + \frac{26r - 20}{3\ln r}\right)\mathsf{OPT}. \end{aligned}$$

The approximation ratio is optimized by choosing $r \approx 1.753$, which gives an approximation ratio of ≤ 16.86.

6 Planar zero skew clock routing.

We now present a constant factor approximation algorithm for the planar ZST problem.

Let R^* be the smallest radius of an L_1 ball that encloses all the points. We first find a center point r such that every terminal is within an L_1 distance of R^* from r. We now construct a square S of side $2R_0$ centered at r. The value R_0 is chosen by selecting t uniformly and at random from $[0, 1]$ and setting $R_0 = R^* \cdot 2^t$. The square S is then subdivided into four equal sized squares S_1, S_2, S_3, S_4. The squares S_i are called the *children* of S and S is called the *parent* of each S_i. The center of S is connected to the centers of each S_i by an H-shaped structure. We proceed recursively in each S_i, dividing each into 4 equal squares and so on, so long as there is at least one point in the square. This produces a tree that we refer to as an *H-tree* (see Figure 2.)

This tree spans all the terminals. In fact, we only construct the subtree of the H-tree that spans all the terminals. To do this, we ensure that the tree construction proceeds only inside squares that contain at least one terminal. At any point in the execution of the algorithm, consider a square S produced by the algorithm and subdivided into S_1, S_2, S_3, S_4. Then the center of S is connected to the center of S_i and the tree construction proceeds recursively in S_i only if S_i contains a terminal. Also, we stop the recursive subdivision when the squares that the algorithm constructs have side lengths smaller than R^*/n^2. At this stage, we connect the centers of all squares to the terminals inside them by edges of length R^*/n^2.

In order to analyze the cost of the tree returned by the algorithm, we associate, with each square S constructed by the algorithm, the cost of the connection from the center of S to the center of the parent of S. Thus, the charge to a square of side $2x$ is $2x$. Note that when the algorithm terminates, the cost of connecting the n terminals to the centers of their corresponding squares is $n \cdot R^*/n^2 = R^*/n$. Since the cost of the optimal ZST is at least $2R^*$, this is at most $1/n$ times the optimal cost. We ignore this cost in our calculations, and in fact, the algorithm can be modified so that this cost is not incurred. We omit the details in this version.

Let $n(x)$ be the minimum number of L_1 balls of radius x required to cover the terminals. Let $n'(y)$ be the number of squares of radius y produced by the algorithm.

Lemma 7

$$n'(2x) \leq 4n(x).$$

Proof. Consider a grid with the center point r as the origin, produced by equispaced horizontal and vertical lines such that the distance between consecutive lines is $2x$. The squares of side $2x$ produced by the algorithm are precisely the squares in this grid that contain at least one terminal. Consider the optimal partitioning of the terminals into $n(x)$ L_1 balls of radius x. Each L_1 ball in the partition intersects at most 4 squares in the grid. Thus there can be at most $4n(x)$ squares in the grid that contain at least one terminal. □

Since the algorithm constructs squares of side lengths in the range $[0, 2R^*]$, the expected cost of the tree is bounded by

$$\int_0^{2R^*} n'(y) \cdot y \cdot d\mu.$$

Here, $d\mu$ is the probability that the algorithm constructs squares with side length in the range $[y, y + dy]$. Hence $d\mu = dy/(y \ln 2)$. The expected cost is thus bounded by

$$\begin{aligned} \int_0^{2R^*} \frac{n'(y)}{\ln 2} dy &= \int_0^{R^*} \frac{n'(2x)}{\ln 2} 2dx \\ &\leq \int_0^{R^*} \frac{8}{\ln 2} n(x) dx. \end{aligned}$$

Hence, the expected cost of the tree produced by the algorithm is at most $8/\ln 2 \approx 11.54$ times the optimal cost.

Theorem 8 *The above algorithm achieves an expected approximation ratio of* $8/\ln 2$.

The algorithm can be derandomized easily by choosing a set of $O(n^2)$ values of R_0 in the range $[R^*, 2R^*]$, running the algorithm for each of them and returning the best tree produced.

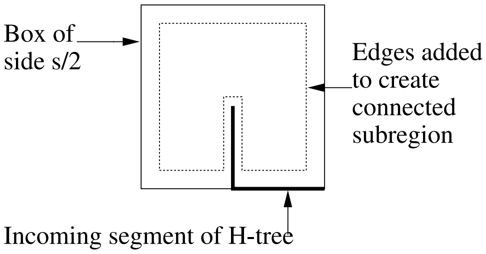

Figure 3: New edges added to each box.

7 Planar embeddable bounded skew clock routing

We now give a constant factor approximation algorithm for creating planar-embedded s-skew trees. We apply the lower bound of section 3, namely, $\mathsf{OPT} \geq \int_s^{R^*} n(R)dR$.

Our strategy will be similar to the bounded skew case for general metrics. We construct the zero skew tree as in the previous section but stop when the sides of the squares become smaller than $s/2$. We will then connect the points in each square to the center using a tree whose cost is comparable to the MST for the point set and has radius at most s. We will separately bound the cost of both the truncated zero skew tree and the trees within each square to within a constant factor of OPT.

We present a deterministic version of the algorithm here. Let R_0 be the unique value in $[R^*, 2R^*)$ of the form $2^t s$, where t is integral. Let $R_i = 2^{t-i}s$. Enclose the point set in a box of side $2R_0$. We iteratively divide the square as before into four squares, but we stop after $t+1$ iterations, when the side of the resulting square has size $s/2$. We then build a zero-skew H-tree terminating at the centers of every populated square of size $s/2$. We will now connect the points within each square to the center.

We first construct an MST connecting all the points in the point set. We divide this MST into pieces using the ZST built above. Recall that the ZST include a single edge into the center of each square. We cut each edge in the MST at points where it intersects existing edges in the ZST, or boundaries of the squares of side $s/2$. We augment the MST edges within each square to produce a connected planar graph, by adding the new edges shown in Figure 3. This results in a connected graph within each square of side $s/2$, from which we take any spanning subtree.

We apply the following result (see [3, 17]).

Lemma 9 *Given any $\epsilon > 0$ and point set in the plane with radius r, and spanning tree T with cost c rooted at p, there exists a polynomial time algorithm to find a spanning tree T' with radius $r' \leq (1+\epsilon)r$ and cost $c' \leq (1 + 1/\epsilon)c$.*

We run this algorithm for $\epsilon = 1$ on each square of side $s/2$, and attach the resulting spanning tree to the ZST at the center of the square.

Now, notice that the cost of the resulting structure has two components, each of which we bound separately. First, the edges of the ZST and the additional edges of Figure 3 are bounded by five times the cost of the ZST. We can bound the cost of the ZST using techniques similar to the previous section, with the caveat that rather than bounding $n'(x)$ in terms of $n(x/2)$, we instead bound it in terms of $n(4x)$. The details are omitted.

8 Hardness of zero skew clock routing.

Theorem 10 *The zero skew clock routing problem for general metric spaces is* **NP**-*hard.*

Proof. The reduction is from set cover. Let $[n] = \{1, 2, \ldots, n\}$. A set cover instance consists of an integer k, and m sets $S_1, \ldots, S_m$ such that each S_i is a subset of $[n]$. We are required to determine if there exist k sets $S_{i_1}, \ldots, S_{i_k}$ such that $[n] \subseteq \bigcup_{j=1}^{k} S_{i_j}$. Given an instance I of set cover, we define an instance of the zero skew clock routing problem as follows. We first construct a weighted graph G from the instance I: G has a vertex s, vertices $x_1, \ldots, x_m$ (one corresponding to each set), and vertices $y_1, \ldots, y_n$ (one corresponding to each element of $[n]$) and an edge from x_i to y_j iff $j \in S_i$; such an edge has length 1. Also, every x_i is connected to s by an edge of length $1/n$. Consider the zero skew clock routing problem for the set of terminals $\{y_1, \ldots, y_n\}$ in the metric space induced by distances in G. If k' is the minimum number of sets in the instance I required to cover $[n]$, it is easy to show that the optimal solution to the zero skew clock routing problem is $n + k'/n$. □

9 Open questions.

The complexity of the planar ZST problem is still open. We do not know if the problem is **NP**-hard.

Since our algorithm can be thought of yielding a clock tree topology, it will be interesting to see how it performs in practice, especially when combined with the Deferred Merge Embedding technique.

References

[1] S. Arora. Polynomial time approximation schemes for Euclidean TSP and other geometric problems. *Proc. 37th Symposium on Foundations of Computer Science*, pages 2–11, 1996.

[2] S. Arora. Nearly linear time approximation schemes for Euclidean TSP and other geometric problems. *Proc. 38th Symposium on Foundations of Computer Science*, pages 554–563, 1997.

[3] B. Awerbuch, A. Baratz, and D. Peleg. Cost-sensitive analysis of communication protocols. *Proc. 9th Symposium on Principles of Distributed Computing*, pages 177–187, 1990.

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

[5] 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 — II: Analog and Digital Signal Processing*, 39(11):799–814, 1992.

[6] J. Cong, A. Kahng, and G. Robins. Matching-based methods for high-performance clock routing. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 12(8):1157–1169, 1993.

[7] M. Edahiro. An efficient zero-skew routing algorithm. *Proc. 31st Design Automation Conference*, pages 375–380, 1994.

[8] M. Jackson, A. Srinivasan, and E. Kuh. Clock routing for high-performance ICs. *Proc. 27th Design Automation Conference*, pages 573–579, 1990.

[9] J. Cong, A. Kahng, C.-K. Koh, and C.-W. Tsao. Bounded-skew clock and Steiner routing under Elmore delay. *TR-950030*, University of California, Los Angeles, Computer Science Department, 1995.

[10] J. Cong and C.-K. Koh. Minimum-cost bounded-skew clock routing. *TR-950003*, University of California, Los Angeles, Computer Science Department, 1995.

[11] M. Goemans and J. Kleinberg. An improved approximation ratio for the minimum latency problem. *Proc. 7th Symposium on Discrete Algorithms*, pages 152–157, 1996.

[12] T.E. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science,* 38:293–306, 1985.

[13] D. Hochbaum. Various notions of approximations: good, better, best, and more. In: D. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*. PWS Publishing Company, 1996.

[14] D.S. Hochbaum and D.B. Shmoys. A best possible heuristic for the k-center problem. *Mathematics of Operations Research,* 10:180–184, 1985.

[15] A. Kahng and G. Robins. *On Optimal Interconnections for VLSI.* Kluwer Academic Publishers, 1995.

[16] A. Kahng, J. Cong, and G. Robins. High-performance clock routing based on recursive geometric matching. *Proc. 28th Design Automation Conference*, pages 322–327, 1991.

[17] S. Khuller, B. Raghavachari, and N. Young. Balancing minimum spanning trees and shortest-path trees. *Algorithmica*, 14(4):305-321, 1995.

[18] Y.-M. Li and M. Jabri. A zero-skew clock routing scheme for VLSI circuits. *Proc. International Conference on Computer-Aided Design*, pages 458–463, 1992.

[19] R. Motwani, S. Phillips and E. Torng. Non-clairvoyant scheduling, *Proc. 4th Symposium on Discrete Algorithms*, pages 422–431, 1993. See also: *Theoretical Computer Science,* 130:17–47, 1994.

[20] H. Prömel and A. Steger. RNC-approximation algorithms for the Steiner problem. *Proc. Symposium on Theoretical Aspects of Computer Science*, volume 1200 of *Lecture Notes in Computer Science*, pages 559–570. Springer-Verlag, 1997.

[21] Ramanathan, Dupont, and Shin. Clock distribution in general VLSI circuits. *IEEE Trans. on Circuits and Systems*, 41, 1994.

[22] R.-S. Tsay. Exact zero skew. *Proc. International Conference on Computer-Aided Design*, pages 336–339, 1991.

[23] R.-S. Tsay. An exact zero-skew clock routing algorithm. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 12(2):242–249, 1993.

[24] J. G. Xi and W. W.-M. Dai. Jitter-tolerant clock routing in two-phase synchronous systems. *Proc. International Conference on Computer-Aided Design*, pages 316–321, 1996.

[25] Q. Zhu and W. W.-M. Dai. Perfect-balance planar clock routing with minimal path-length. *Proc. International Conference on Computer-Aided Design*, pages 473–477, 1992.