# On the Minimum Density Interconnection Tree Problem*

C. J. Alpert, J. Cong, A. B. Kahng, G. Robins†, and M. Sarrafzadeh‡

CS Dept., University of California at Los Angeles, Los Angeles, CA 90024-1596
† CS Dept., University of Virginia, Charlottesville, VA 22903-2442
‡ EECS Dept., Northwestern University, IL 60208-3118

**Abstract**

We discuss a new *minimum density* objective for spanning and Steiner tree constructions. This formulation is motivated by the minimum-area layout objective, which is best achieved through balancing the usage of horizontal and vertical routing resources. We present two efficient heuristics for constructing low-density spanning trees and prove that their outputs are on average within small constants of optimal with respect to both tree cost and density. Our proof techniques suggest a non-uniform lower bound schema which can afford tighter estimates of solution quality for a given problem instance. Furthermore, the minimum density objective can be transparently combined with a number of previous interconnection objectives (e.g., minimizing tree radius or skew) without affecting solution quality with respect to these previous metrics. Extensive simulation results suggest that applications to VLSI global routing are promising.

**Keywords:** VLSI routing, rectilinear Steiner trees, minimum density, tree radius, stabbing number, computational geometry.

## 1 Introduction

In this paper, we address a new *minimum density* objective for spanning and Steiner tree constructions in the Manhattan plane. Our work is motivated by the area minimization requirement inherent in the global routing phase of VLSI layout (the global routing phase entails construction of spanning or Steiner interconnection trees over prescribed point sets, or *signal nets*; see [20] for a survey). Traditionally, the minimum-area objective has been approximately captured by minimizing the total edgelength in the tree: since wires have a fixed width and must be routed at a fixed separation from each other, the total tree edgelength provides an obvious lower bound on the routing area that must be added to the layout. However, the grid-based structure of integrated circuit routing resources provides additional information for determining the impact of a given interconnection topology on the chip area.

*Partial support for this work was provided by a Department of Defense Graduate Fellowship; by NSF MIP-9110511; by NSF MIP-9110696, NSF Young Investigator Award MIP-9257982, ARO DAAK-70-92-K-0001, and ARO DAAL-03-92-G-0050; and by an IBM Graduate Fellowship. A preliminary version of this work appeared in *Proc. IEEE Intl. Symp. on Circuits and Systems*, Chicago, May 1993.

For the four-terminal signal net shown in Figure 1, the interconnection tree of Figure 1(a) forces at least three wires to cross the dashed line, meaning that the horizontal dimension of the chip must increase by enough to accommodate these three *routing grids*. (We adopt "routing grid" as a generic term that is independent of layout methodology. The term encompasses, e.g., vertical feedthroughs or horizontal routing tracks in a channel [20].) In contrast, the tree of Figure 1(b) forces the horizontal chip dimension to grow by only one routing grid (however, the vertical chip dimension will grow by two grids, as indicated by the horizontal dashed line). In view of manufacturing constraints on the maximum chip dimension, the most effective layouts are generally those which are roughly square, and this suggests balancing the horizontal and vertical routing requirements induced by the interconnection tree. As a result, we formulate the *Minimum Density Interconnection Tree* problem as follows.

INSERT FIGURE 1

A *signal net* $N = \{p_1, p_2, \ldots, p_n\}$ is a set of $n$ points, or *terminals*, in the Manhattan plane. (Note that in discussing related formulations, we will occasionally find it necessary to distinguish a single *source* terminal of the net, with the remaining terminals being *sinks*.) An *interconnection tree* of a net $N$, denoted $T(N)$, is a tree which spans $N$. The *cost* of a tree edge is the Manhattan distance between its endpoints, and the cost of a routing tree is the sum of the costs of its edges. A line *properly* intersects an edge if and only if it intersects the edge at a single point which is not an endpoint of the edge.

**Definition:** The *density* of an interconnection tree is the maximum number of tree edges that can be properly intersected by a horizontal or vertical line in the plane (Figure 2).

**Definition:** For a given net $N$, the *minimum density* of $N$ is the minimum density achievable by an interconnection tree $T(N)$, and a *minimum density interconnection tree* is any $T(N)$ that achieves this minimum density.

INSERT FIGURE 2

**Minimum Density Interconnection Tree (MDIT) Problem:** Given a net $N$, construct a minimum density interconnection tree $T(N)$ which has minimum cost.

Our density criterion recalls the notion of trees with "low stabbing number", which are used in the computational geometry literature to speed up dynamic "ray shooting" queries [1] [8] [13] [14] [24]. However, our work differs from these computational geometry results in several important respects. First, spanning trees with low stabbing number minimize the number of tree edges that can be intersected by a line of *any* orientation, while our MDIT formulation is concerned only with horizontal or vertical intersecting lines. While the low stabbing number formulation is more general, it is also much more difficult to solve: we achieve bounds that are tighter by a logarithmic factor, and our algorithms are considerably simpler as well as more efficient than those achievable through naive adaptation of previous constructions. Second, methods from the computational geometry literature do not bound the tree cost in addition to bounding the tree density; our methods minimize both parameters simultaneously, and are thus much more relevant to VLSI routing applications. Finally, we address the case when Steiner points are allowed, enabling considerably tighter bounds on both density and tree cost.

We also note an attractive feature of the MDIT heuristics presented below, namely, their "compatibility" with existing VLSI routing objectives. The three most prominent interconnection tree criteria in the current literature are all "performance-driven", i.e., they are aimed at improving the maximum speed at which a digital system may be clocked:

1. Minimizing the *total cost* of the interconnection tree corresponds to the well-studied minimum rectilinear Steiner tree problem [4] [19] [21] [25]. The total tree cost contributes a significant portion of the RC delay as well as routing area [6].

2. Minimizing the *maximum source-sink pathlength* in the tree corresponds to the "bounded-radius, bounded-cost" spanning tree formulation which has been treated in [2] [9] [10] [11]. The minimum-radius criterion more accurately captures delay considerations when interconnect resistance dominates driver output resistance, as in multi-chip module substrate or submicron IC interconnects [6] [15].

3. Minimizing the *maximum pathlength skew*, i.e., difference in source-sink pathlengths, captures both the clock skew minimization problem as well as the "min-max" timing constraints which arise in global routing of very high-performance designs. These formulations have been treated in such recent works as [7] [17] [18] [23].

We make note of these existing formulations because our proposed algorithms for minimum-density interconnection trees afford unique multiple optimizations wherein up to three competing objectives may be optimized *simultaneously*. As a result, the area minimization objective of minimum-density routing can be attained without sacrificing performance-driven criteria. In particular, the discussion below describes how tree cost, radius, and density can be simultaneously optimized; we also show how tree cost, skew and density can be simultaneously addressed.

The remainder of this paper is organized as follows. In Section 2, we give two efficient heuristic constructions for minimum density interconnection trees, along with several simple variants. Section 3 establishes a number of performance bounds: we show that our methods have good performance in that on average they produce interconnection trees with both cost and density bounded by constant factors from their optima. Section 4 integrates the minimum density objective with current performance-driven objectives in order to achieve "triple optimizations", and Section 5 concludes with experimental results as well as several directions for future research.

# 2 Heuristics for Minimum Density Interconnection Trees

Without loss of generality, the following discussion will assume that all terminal coordinates lie inside the unit square (the input can always be scaled to meet this condition).

## 2.1 The COMB Construction for Spanning and Steiner Trees

Our first basic algorithm sorts the terminals by increasing $x$-coordinate (ties are broken to favor the larger $y$-coordinate), and then partitions the terminals of net $N$ into $\frac{\sqrt{n}}{\sqrt{2}}$ vertical *strips*, each containing $\sqrt{2n}$ terminals (Figure 3a). (Note that our discussion implicitly assumes use of the floor and ceiling functions as appropriate; this does not affect any of our asymptotic results.) We then connect all the terminals of each strip into a path, in order of decreasing $y$ coordinate (Figure 3b). We complete the routing topology by connecting the terminals with lowest $y$ coordinate in each strip, in order from left to right (Figure 3c). This algorithm is described in Figure 4. The complexity of this algorithm, which we call COMB, is clearly dominated by the partitioning/sorting step (Step 1 of Figure 4), and is therefore $O(n \log n)$.

INSERT FIGURE 3

INSERT FIGURE 4

If the introduction of Steiner points is allowed in constructing the interconnection tree, we can reduce the worst-case density as well as the worst-case cost of our construction via the following method: (i) partition the net $N$ into $\frac{\sqrt{n}}{\sqrt{2}}$ vertical strips, each containing $\sqrt{2n}$ terminals (Figure 5a); (ii) within each strip, connect the terminals in the strip to a central *spine*, i.e., a vertical line which passes through the median terminal of the strip when the terminals are sorted by $x$-coordinate (Figure 5b); then (iii) join all the spines using segments of a single horizontal line (Figure 5c). This variant, which we call COMB_ST, is described in Figure 6 and has complexity $O(n \log n)$, again reflecting the complexity of the partitioning/sorting step.

INSERT FIGURE 5

INSERT FIGURE 6

## 2.2 A Chain-Peeling Method

A different, "chain-peeling" approach to density minimization iteratively computes and superposes chains or antichains (i.e., sets of terminals through which a staircase routing exists). More precisely, a *chain* is a sequence of terminals with coordinates that are monotone nondecreasing in both $x$ and $y$; an *antichain* has coordinates monotone nondecreasing in $x$ and monotone nonincreasing in $y$. According to Dilworth's theorem [12], every partially ordered set of size $n$ must contain either a chain or an antichain of size at least $\sqrt{n}$.

Our chain-peeling method, which we call PEEL (Figure 7), efficiently detects a maximal chain or antichain and then removes it from the net; the process is iterated over the remaining terminals until the net has been covered. Each chain contributes at most 1 to the overall density, and the chains/antichains can be joined together into a tree (Step 7 of Figure 7) without increasing the density further (see Theorem 3.6 below).

The PEEL method is attractive because it escapes such pathological examples as that of Figure 8, where COMB or COMB_ST will yield density an unbounded factor greater than that of PEEL. We show in Section 3.1 that the time complexity of PEEL is $O(n^{\frac{3}{2}} \log\log n)$.

INSERT FIGURE 7

INSERT FIGURE 8

# 3 Performance Bounds

Both the density and the total tree cost of our constructions are on average only small constant factors away from optimal.

## 3.1 Density Bounds

For a net $N$ of $n$ terminals, a lower bound of $\Omega(\sqrt{n})$ can easily be established for the worst-case minimum density of the spanning tree $T(N)$. Moreover, we can show $\Theta(\sqrt{n})$ expected density for the minimum density interconnection tree over $N$.

**Theorem 3.1** *A net $N$ containing the $n$ distinct grid points of the $(\sqrt{n}-1) \times (\sqrt{n}-1)$ grid cannot be spanned by an interconnection tree having density $< \lceil \frac{\sqrt{n}+1}{2} \rceil$.*

**Proof:** In the square array, there are $2(\sqrt{n}-1)$ horizontal and vertical lines between the rows and columns of terminals. For the tree $T(N)$ to be connected, each tree edge must cross at least one horizontal or vertical line. Hence, there are at least $n-1$ line crossings, and the pigeonhole principle implies that at least one of the lines is crossed $\lceil \frac{n-1}{2(\sqrt{n}-1)} \rceil = \lceil \frac{\sqrt{n}+1}{2} \rceil$ times. $\square$

The next theorem requires the following lemma:

**Lemma 3.2** *If $n$ indistinguishable balls are independently placed at random into $n$ indistinguishable boxes, $(1-\frac{1}{e})\cdot n$ boxes are expected to be non-empty.*

**Proof:** The probability of a given ball ending up in a given box $B_i$ is $\frac{1}{n}$, and so the probability of the ball missing box $B_i$ is $1-\frac{1}{n}$. By the independence of the placements, the probability that all $n$ balls miss box $B_i$ is $(1-\frac{1}{n})^n$. Therefore, as $n$ increases, the probability that any given box remains empty is $\lim_{n\to\infty}(1-\frac{1}{n})^n = \frac{1}{e}$. By linearity of expectation, it follows that a constant fraction $\frac{1}{e}$ of the $n$ boxes are expected to remain empty, proving the lemma. $\square$

**Theorem 3.3** *For $n$ terminals chosen randomly from a uniform distribution in the unit square, the minimum density interconnection tree has expected density $\Theta(\sqrt{n})$.*

INSERT FIGURE 9

**Proof:** Partition the unit square into $n$ identical square cells, each of size $\frac{1}{\sqrt{n}}$ by $\frac{1}{\sqrt{n}}$, using $2\sqrt{n}-2$ vertical and horizontal lines (Figure 9a). If we regard cells as "boxes" and terminals as "balls" then by Lemma 3.2 the expected number of cells containing at least one terminal is $(1-\frac{1}{e})\cdot n$. For the spanning tree to be connected, each of these non-empty cells must contain at least one terminal $p_i$ which has an incident tree edge $e_j$ that crosses a boundary of the cell (Figure 9b). By the pigeonhole principle, at least one of the horizontal or vertical lines will intersect $\frac{(1-\frac{1}{e})\cdot n}{(2\sqrt{n}-2)} > (1-\frac{1}{e})\cdot\frac{\sqrt{n}}{2}$ tree edges, yielding the $\Omega(\sqrt{n})$ lower bound on the expected minimum density. Since our algorithms always yield interconnection trees with density $O(\sqrt{n})$ (see the following sequence of results), the expected minimum routing density for a net of $n$ terminals uniformly chosen in the unit square is $\Theta(\sqrt{n})$. $\square$

Interestingly, the proof schema of Theorem 3.3 suggests a computational lower bound for individual instances of the MDIT problem, as follows. Given a net $N$, select integers $i$ and $j$ and partition the unit square into an $i$ by $j$ (not necessarily uniform) rectangular grid such that the greatest number $P$ of the resulting $i\cdot j$ rectangles contain terminals (see Figure 10). By the pigeonhole principle (recall the proof of Theorem 3.3), this induces an immediate lower bound of $\lceil\frac{P-1}{(i-1)+(j-1)}\rceil = \lceil\frac{P-1}{i+j-2}\rceil$ on the minimum routing density of $N$. Various schemes can be used to find a partition which maximizes the quantity $\frac{P-1}{i+j-2}$: for example, one could place $i=\sqrt{n}$ horizontal lines such that at most $\sqrt{n}$ terminals lie between each consecutive pair of horizontal lines, and then place the $j = \sqrt{n}$ vertical lines using a similar criterion. It is open whether there exists a polynomial-time algorithm which computes a rectangular partition that maximizes $\frac{P-1}{i+j-2}$, even for fixed $i$ and $j$ (e.g., $i = j = \sqrt{n}$). Experimental data indicates that this simple computational lower bound can be useful for small net sizes; details are presented in Section 5.

INSERT FIGURE 10

We now establish the density bounds for our heuristics.

**Theorem 3.4** *Algorithm COMB constructs a spanning tree with density* $\leq \sqrt{2n}$.

**Proof:** Since each strip contains no more than $\sqrt{2n}$ terminals, a vertical line passing through any strip cannot intersect more than $\sqrt{2n}$ tree edges. Since any given horizontal line cannot intersect more than two edges within a strip (one edge from Step 2 and one from Step 3 in Figure 4), the maximum horizontal density is $2 \cdot \frac{\sqrt{n}}{\sqrt{2}} = \sqrt{2n}$. Thus, the density of the COMB output is at most $\sqrt{2n}$. $\square$

**Theorem 3.5** *Algorithm COMB_ST constructs a Steiner tree with density* $\leq \frac{\sqrt{n}}{\sqrt{2}} + 1$.

**Proof:** In the construction of Figure 5, a strip can contain at most $\frac{\sqrt{n}}{\sqrt{2}}$ terminals on each side of its spine, so no vertical line can intersect more than $\frac{\sqrt{n}}{\sqrt{2}}$ of the edges created in Step 2 of Figure 6. No horizontal line can intersect any of the $\frac{\sqrt{n}}{\sqrt{2}}$ vertical spines more than once. Thus, the density of the COMB_ST output is at most $\frac{\sqrt{n}}{\sqrt{2}} + 1$, when we consider the edges added to join the spines together (Step 3). $\square$

INSERT FIGURE 11

A density bound for the chain-peeling algorithm PEEL follows from the following two lemmas, namely, (i) at most $O(\sqrt{n})$ chains or antichains will be "peeled" during the construction, and (ii) these chains/antichains can be connected to form a single component which has density at most the number of chains/antichains.

**Theorem 3.6** *Algorithm PEEL constructs a Steiner tree with density most* $\leq 2 \cdot \sqrt{n}$.

**Proof:** We first show that PEEL computes at most $2 \cdot \sqrt{n}$ chains and/or antichains. Let $a_i$ denote the number of points remaining after we have peeled off $i$ chains and/or antichains. Assume that the algorithm stops when we have peeled off $k$ chains and/or antichains, i.e., $a_k = 0$. We want to show that $k \leq 2 \cdot \sqrt{n}$. According to Dilworth's Theorem [12], the size of the $(i+1)$-th chain/antichain is at least $\sqrt{a_i}$. Thus, $a_{i+1} \leq a_i - \sqrt{a_i}$. Moreover, it is easy to verify that $\sqrt{x - \sqrt{x}} \leq \sqrt{x} - \frac{1}{2}$. Therefore,

$$\sqrt{a_k} \leq \sqrt{a_{k-1} - \sqrt{a_{k-1}}} \leq \sqrt{a_{k-1}} - \frac{1}{2} \leq (\sqrt{a_{k-2}} - \frac{1}{2}) - \frac{1}{2} \leq \ldots \leq \sqrt{a_0} - \frac{k}{2}$$

This implies that $k \leq 2 \cdot (\sqrt{a_0} - \sqrt{a_k}) = 2 \cdot \sqrt{n}$. To complete the proof, we need to show the chains and antichains can be "joined" into a spanning tree without increasing density. This can be accomplished by extending each chain to the top-right corner of the unit square and each anti-chain to the top-left corner; this clearly will not increase total density beyond $k$ (see Figure 11). A simple case analysis shows that the set of chains can then be connected to the set of antichains with no further increase in density, yielding the overall density bound of $2 \cdot \sqrt{n}$. □

Note that when the chains and antichains are joined into a Steiner tree as described in the proof, the tree density will always be *exactly* the total number of chains and antichains since a horizontal line near the top of the square will cut all (extended) chains and antichains. Clearly, lower density constructions might be attainable; however, our experimental results of Section 5 use this simple "joining" construction for Step 7 of the PEEL algorithm.

A result of Hunt and Szymanski [16] shows that the maximum chain or antichain in a pointset can be computed in $O(n \log\log n)$ time. Since PEEL requires at most $O(\sqrt{n})$ iterations, its time complexity is bounded by $O(n^{\frac{3}{2}} \log\log n)$.

### 3.2 Cost Bounds

Probabilistic arguments show that on average, all of our algorithms will produce interconnection trees with low cost.

**Theorem 3.7** *For $n$ terminals distributed arbitrarily in the unit square, algorithm COMB constructs a spanning tree with cost $\leq 2\sqrt{2} \cdot \sqrt{n}$.*

**Proof:** In the COMB construction, the sum of the vertical components of the edges within each strip is bounded by 1 (the height of each strip is one unit). Thus, the sum of the vertical components of all routing tree edges introduced in Step 2 of Figure 4 is bounded by $\frac{\sqrt{n}}{\sqrt{2}}$. Furthermore, the vertical components of edges introduced in Step 3 also sum to at most $\frac{\sqrt{n}}{\sqrt{2}}$. To bound the sum of horizontal components, note that if we pick an arbitrary edge from within each strip, these $\frac{\sqrt{n}}{\sqrt{2}}$ edges have total horizontal span bounded by 1. The horizontal components of all tree edges from Step 2 thus contribute at most $\sqrt{2n}-1$ to the tree cost, and since the edges added in Step 3 have total horizontal span $\leq 1$, we obtain the bound of $2\sqrt{2}\cdot\sqrt{n}$. $\square$

**Theorem 3.8** *For $n$ terminals distributed arbitrarily in the unit square, algorithm COMB_ST constructs a Steiner interconnection tree with cost $\leq \sqrt{2n}+1$.*

**Proof:** In the COMB_ST construction, the vertical spines contribute at most $\frac{\sqrt{n}}{\sqrt{2}}$ to the tree cost. As in the proof of Theorem 3.7, if we pick an arbitrary pair of horizontal edges in each strip, one from either side of the spine, the total cost of these edges is $\leq 1$, so the sum of horizontal edge components is at most $\frac{\sqrt{2n}}{2} = \frac{\sqrt{n}}{\sqrt{2}}$. Finally, the horizontal connector which joins the spines (Step 3 of Figure 6) has cost $\leq 1$, and the desired bound of $\sqrt{2n}+1$ follows. $\square$

**Theorem 3.9** *For $n$ terminals distributed arbitrarily in the unit square, algorithm PEEL constructs a Steiner tree with cost $\leq 4\cdot\sqrt{n}$.*

**Proof:** According to Theorem 3.6, PEEL constructs at most $2\cdot\sqrt{n}$ chains and antichains, which are extended and then joined to yield a Steiner tree over the net $N$. Each extended chain or antichain can have cost at most 2, yielding the desired bound. $\square$

**Theorem 3.10** *For $n$ terminals chosen randomly from a uniform distribution in the unit square, the expected minimum spanning tree cost is $\Theta(\sqrt{n})$.*

**Proof:** While this claim is a consequence of results in the theory of subadditive functionals in the $L_p$ plane [5] [22], we present the following simple proof. Again, we partition the unit square into an array of $n$ square cells, each of size $\frac{1}{\sqrt{n}}$ by $\frac{1}{\sqrt{n}}$. Recall that the expected number of cells that will contain at least one terminal is $(1-\frac{1}{e})\cdot\sqrt{n}$. In any interconnection tree $T(N)$, each terminal will have at least one incident tree edge, and this edge must cross the boundary of the cell. It is easy to show that the expected distance from a terminal to the nearest side of its containing cell is lower-bounded by some constant times the length of the side of the cell (in the Manhattan norm, this constant is $\frac{1}{6}$). We therefore have an $\Omega(\sqrt{n})$ bound on the expected total cost of the interconnection tree. Since our COMB algorithm always yields a spanning tree with cost $O(\sqrt{n})$, the minimum spanning tree cost for a set of $n$ terminals uniformly distributed in the unit square is $\Theta(\sqrt{n})$ on average. $\square$

From these results, we have:

**Corollary 3.11** *For $n$ terminals chosen randomly from a uniform distribution in the unit square, the algorithms COMB, COMB_ST and PEEL all construct trees which on average have both density and cost bounded by constants times optimal.* □

As noted in Section 1, our notion of density is related to the computational geometry concept of "low stabbing number" which seeks spanning trees having few intersections with lines of *any* orientation [8] [13]. Welzl [24] has proved that there always exists a spanning tree with stabbing number $O(\sqrt{n})$, but his method is both complicated and less efficient (having greater than cubic time complexity). Edelsbrunner et al. [14] have shown that $\Omega(\sqrt{n})$ is a lower bound for the stabbing number of a pointset; our Theorems 3.1 and 3.3 show that this lower bound holds even when only horizontal and vertical stabbing lines are allowed, and moreover establish an *average* case $\Omega(\sqrt{n})$ density lower bound. The authors of [14] also give three spanning tree constructions with low stabbing number, trading off between space, stabbing number, and the use of randomization. These methods obtain bounds on stabbing number ranging from $O(n^{(\frac{1}{2}+\epsilon)})$ to $O(n^{\frac{1}{2}} \cdot polylog)$, and typically run in $O(n^3)$ time and $O(n^2)$ space. By contrast, our algorithms guarantee $O(n^{\frac{1}{2}})$ density, and run in $O(n \log n)$ time and $O(n)$ space. Finally, Agarwal [1] showed that there always exists a *family* of $O(\log n)$ trees such that for an arbitrary given line, *one* of the trees will have a stabbing number of $O(\sqrt{n})$; this family can be computed in time $O(n^{\frac{3}{2}} \cdot polylog)$. However, this is not very useful for VLSI routing, where the goal is to find a *single* tree with good density. In general, since the MDIT formulation restricts the orientation of the stabbing lines, our algorithms are more space- and time-efficient, and have better performance bounds. We are also able to address the case of low-density Steiner trees, while previous work on trees with low stabbing number could address only spanning constructions.

# 4 Triple Optimization

For practical VLSI routing applications, it is often desirable to minimize more than one objective function at once. However, this is difficult: it is unusual for even two competing measures to be treated effectively (e.g., the simultaneous tree radius and tree cost minimization of [11]). In this section, we show that the minimum-density objective is "compatible" with existing performance-driven routing objectives, so that we may simultaneously address up to three separate routing tree measures.

### 4.1 Minimizing Skew, Density, and Total Wirelength

Recall from Section 1 that construction of an interconnection tree with minimum difference among the various source-sink pathlengths captures both minimum-skew clock routing [3] and global routing with min-max timing constraints. The work of [18] gives a general interconnection scheme that achieves extremely small pathlength skews, while keeping the total wirelength on average within a constant factor of optimal, and always bounded by $O(\sqrt{n})$. This clock routing construction of [18], which we refer to as CLOCK, begins with a forest of $n$ isolated terminals, each of which is considered to be a (trivial) tree. An optimal geometric matching on these $n$ points yields $\frac{n}{2}$ segments, each of which defines a tree with two nodes. A tree is rooted at its balance point, i.e., the point that minimizes the pathlength skew to the leaves of its two subtrees. Trees continue to be paired up by geometric matching of their roots, so that at each level of the construction, only half as many points are matched as in the previous level. Thus, after $\lceil \log n \rceil$ matching iterations, a complete tree topology is obtained. Figure 12 formally details the CLOCK algorithm.

INSERT FIGURE 12

In order to construct clock routing trees with low density, we construct a low-density geometric matching via the following variant of algorithm COMB: partition the net into $\frac{\sqrt{n}}{\sqrt{2}}$ strips of $\sqrt{2n}$ terminals each and connect the terminals of each strip from top to bottom as before (Figure 13a). However, instead of connecting the bottom terminals of all strips, connect the terminals in a *serpentine* fashion, i.e., alternate between connecting the bottoms and tops of adjacent pairs of strips as shown in Figure 13b. Arguments similar to those in Section 3 show that this procedure (which we call COMB_SERP) will connect all of the terminals in a single long path topology that has both total cost and overall density simultaneously bounded by $O(\sqrt{n})$ in the worst case.

INSERT FIGURE 13

Taking only every other edge of the tour produced by COMB_SERP will constitute a geometric matching

(Figure 13c) having both total cost and overall density simultaneously bounded by $O(\sqrt{n})$. We may iteratively use such matchings within the CLOCK algorithm of [18] to yield clock routing trees that simultaneously address three competing objectives: pathlength skew, total wirelength, and density. In particular, the latter two quantities are both bounded on average by constants times optimal, which follows from the fact that at each level of the tree construction, only half as many points are being matched as in the previous iteration. For example, the density of the resulting clock tree will be bounded by $O(\sqrt{n}) + O(\sqrt{\frac{n}{2}}) + O(\sqrt{\frac{n}{4}}) + ... = O(\sqrt{n})$. The time complexity of this construction is $O(n \log n)$ since it is dominated by the serpentine matching algorithm.

## 4.2 Minimizing Radius, Density, and Total Wirelength

In [11], a method was proposed to uniformly trade off total routing tree cost with tree radius (i.e., the longest source-sink pathlength in the tree), and simultaneously optimize both parameters to within constants times optimal in the worst case. This "bounded-radius, bounded-cost" (BRBC) construction [11] starts with a low-cost tour of the net terminals (e.g., a depth-first tour of a minimum spanning tree), and then augments this tour by adding shortest paths to the source from certain regularly spaced locations along the this tour. The precise cost/radius tradeoff obtained by BRBC depends on a user-specified parameter $\epsilon \geq 0$. The algorithm returns the shortest-paths tree over the resulting augmented graph. Figure 14 formally details the BRBC construction [11].

INSERT FIGURE 14

We can combine the minimum density objective with with the cost/radius tradeoff of the BRBC algorithm to obtain another "triple optimization". Specifically, we may execute the BRBC algorithm with an initial tour $L$ (Step 3 of Figure 14) that is based on, e.g., the spanning tree constructed by COMB_SERP instead of the minimum spanning tree; recall from Section 4.1 that the COMB_SERP output has total cost and density both bounded by $O(\sqrt{n})$. Aside from this choice of initial traversal, the remainder of the construction proceeds exactly as in [11] and thus has $O(n^2)$ time complexity.

The resulting BRBC spanning tree will have radius bounded by $(1+\epsilon)$ from optimum in the *worst case*, cost bounded by $(1+\frac{2}{\epsilon}) \cdot 2\sqrt{2n}$, and density bounded by $(1+\frac{4}{\epsilon \cdot R}) \cdot \sqrt{2n}$, where $R \leq 2$ is the distance from the source to the farthest sink. This can be seen as follows. The density of the combined COMB_SERP / BRBC construction is bounded by the sum of $\sqrt{2n}$ (the density of the COMB_SERP tree $Q$), plus the number of shortest paths to the source taken during the traversal of $Q$ in the BRBC algorithm (since any shortest path is necessarily monotone, it cannot contribute more than 1 to the density). The latter quantity is determined by noting that the depth-first tour of $Q$ has length equal to twice the COMB tree cost $2\sqrt{2n}$, and that BRBC adds shortest paths to the source at intervals of at least $\epsilon \cdot R$ along the traversal of $Q$. Thus, the density of the overall construction is given by $\sqrt{2n} + \frac{2 \cdot 2\sqrt{2n}}{\epsilon \cdot R} = (1 + \frac{4}{\epsilon \cdot R}) \cdot \sqrt{2n}$. Recalling the $\Omega(\sqrt{n})$ lower bounds for expected cost and density, we see that this construction will on average yield cost, density, and radius within constant factors of their respective optimal values. Indeed, the radius is within a constant factor of optimal in the worst case as well.

# 5 Results and Conclusions

We have implemented the COMB_SERP variant of the COMB algorithm (see Section 4.1), the COMB_ST and the PEEL algorithms using ANSI C for both the Macintosh and Sun Sparc environments. The code is available from the first-listed author upon request. Results are presented in Tables 1 and 2. For each pointset cardinality, each algorithm was executed on 100 pointsets randomly chosen from a uniform distribution in the unit square. Table 1 reports the minimum, average, and maximum densities of the resulting interconnection trees. Note that for algorithm PEEL, we simply report the number of chains and antichains computed by the algorithm; this gives the spanning tree density when we use the simple joining method described in the proof of Theorem 3.6. The tree cost of PEEL will be somewhat higher than shown in Table 2 since we report the sum of chain/antichain costs, but not the extra edgelength needed to join the chains together.

The data indicates that the average density of the tree produced by the COMB_SERP algorithm is on par with the density of the simple minimum spanning tree. However, the density of the minimum spanning tree has markedly higher variance, and in the worst case can be as large as $\Omega(n)$. Thus, the COMB or COMB_SERP constructions may have practical utility due to their predictable performance. The average density of the trees produced by the COMB_ST algorithm is considerably better than the average density of the corresponding minimum spanning trees: for example, with signal nets of size 10, COMB_ST yields trees with average density = 3.00, in contrast to average minimum spanning tree density = 3.82. For $n = 10$, this 21% decrease in average density is achieved with a corresponding 21% increase in the tree cost over the MST cost, shown in Table 2. Note that there is essentially no variance in the density of the COMB_ST output.

As discussed in Section 3, for a given net $N$, any partition of the unit square into an $i$ by $j$ rectangular grid, such that $P$ of the resulting $i \cdot j$ rectangles contain terminals of $N$ (Figure 10), induces a lower bound $\lceil \frac{P-1}{i+j-2} \rceil$ on the minimum routing density of $N$. Recall that a simple version of this lower bound schema places $i = \sqrt{n}$ horizontal lines so as to leave at most $\sqrt{n}$ terminals between consecutive lines, and then places $j = \sqrt{n}$ vertical lines using the same criterion. A comparison of the COMB_ST density versus the results of this computational lower bound are given in the rightmost three columns of Table 1 (note that any fractional computational lower bound values are rounded up to the nearest integer, since density takes on only integer values). The lower bound can be used to assessing algorithm quality on an instance-wise basis; in particular, we see that the performance ratio for COMB_ST is quite good despite the simplicity of the lower bound.

| net | MST | | | COMB_SERP | | | PEEL | | | COMB_ST | | | COMB_ST / LB | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| size | min | ave | max | min | ave | max | min | ave | max | min | ave | max | min | ave | max |
| 3 | 1 | 1.69 | 2 | 1 | 1.69 | 2 | 1 | 1.69 | 2 | 1 | 1.00 | 1 | 1.00 | 1.00 | 1.00 |
| 5 | 2 | 2.57 | 4 | 2 | 2.70 | 3 | 2 | 2.00 | 2 | 2 | 2.00 | 2 | 1.00 | 1.36 | 2.00 |
| 7 | 2 | 2.97 | 5 | 2 | 3.64 | 4 | 2 | 2.66 | 3 | 3 | 3.00 | 3 | 1.00 | 1.97 | 3.00 |
| 10 | 2 | 3.82 | 6 | 3 | 3.54 | 5 | 2 | 3.08 | 4 | 3 | 3.00 | 3 | 1.50 | 1.54 | 3.00 |
| 15 | 3 | 4.35 | 6 | 3 | 4.29 | 5 | 3 | 3.93 | 5 | 3 | 3.00 | 3 | 1.50 | 1.50 | 1.50 |
| 20 | 4 | 4.98 | 8 | 4 | 4.80 | 5 | 4 | 4.76 | 6 | 4 | 4.00 | 4 | 1.33 | 1.85 | 2.00 |
| 30 | 4 | 5.99 | 8 | 5 | 5.89 | 7 | 5 | 5.88 | 7 | 5 | 5.00 | 5 | 1.67 | 1.85 | 2.50 |
| 50 | 5 | 7.11 | 10 | 6 | 7.36 | 9 | 7 | 7.85 | 9 | 6 | 6.00 | 6 | 1.50 | 1.97 | 2.00 |
| 100 | 7 | 9.48 | 12 | 9 | 10.95 | 13 | 10 | 11.48 | 13 | 8 | 8.00 | 8 | 2.00 | 2.00 | 2.00 |
| 300 | 12 | 14.59 | 17 | 15 | 17.55 | 21 | 19 | 20.69 | 22 | 13 | 13.00 | 13 | 1.86 | 2.08 | 2.17 |

Table 1: Tree density statistics for minimum spanning tree and for the three heuristic constructions. Averages are taken over 100 instances for each net size. The rightmost columns give the ratio of COMB_ST density to the instance-wise computational lower bound of Section 3.1; for small net sizes in particular, the closeness of the COMB_ST result to this simple lower bound is encouraging.

In conclusion, we have proposed a new spanning and Steiner tree formulation based on a minimum density criterion. We have also presented several efficient heuristics for constructing low-density routing trees. The average performance of all our algorithms has been shown to be within constant factors of optimal in terms of both tree cost and density. Our techniques can be used to unify the new density criterion with previous "performance-driven" interconnection objectives in order to achieve simultaneous optimization of up to three separate and competing interconnection tree measures. Extensive simulations indicate that our constructions are effective in practice, and hold promise for balanced-resource routing applications in VLSI layout.

| net | MST | | | COMB_SERP | | |
|---|---|---|---|---|---|---|
| size | min | ave | max | min | ave | max |
| 3 | 417 | 1103.66 | 2227 | 466 | 1210.13 | 2561 |
| 5 | 804 | 1658.39 | 2554 | 1010 | 2154.82 | 4233 |
| 7 | 1322 | 2039.34 | 2983 | 1520 | 2851.53 | 4427 |
| 10 | 1781 | 2662.36 | 3462 | 2287 | 3682.77 | 4766 |
| 15 | 2296 | 3224.41 | 4045 | 2663 | 4692.03 | 6465 |
| 20 | 2766 | 3789.89 | 4558 | 3819 | 5265.67 | 6567 |
| 30 | 4107 | 4651.00 | 5403 | 5524 | 6841.33 | 8529 |
| 50 | 5190 | 5945.47 | 6668 | 7542 | 8708.12 | 10177 |
| 100 | 7481 | 8384.32 | 8887 | 11630 | 12519.36 | 13487 |
| 300 | 13850 | 14318.99 | 14865 | 20357 | 21118.83 | 21839 |
| net | PEEL | | | COMB_ST | | |
| size | min | ave | max | min | ave | max |
| 3 | 86 | 736.72 | 1577 | 366 | 1164.96 | 1882 |
| 5 | 758 | 1495.57 | 2481 | 1063 | 2260.09 | 3229 |
| 7 | 770 | 1852.91 | 3209 | 1992 | 3009.31 | 4141 |
| 10 | 1595 | 2776.06 | 4080 | 2307 | 3224.01 | 3974 |
| 15 | 2562 | 3721.27 | 5071 | 3143 | 4216.83 | 4941 |
| 20 | 2871 | 4720.69 | 6350 | 3692 | 4823.63 | 5649 |
| 30 | 4873 | 6318.27 | 8085 | 5594 | 6570.46 | 7740 |
| 50 | 7447 | 9298.73 | 11629 | 7070 | 8029.99 | 8945 |
| 100 | 12552 | 14717.75 | 16579 | 10390 | 11083.93 | 11911 |
| 300 | 26123 | 28960.44 | 31549 | 17963 | 18681.10 | 19614 |

Table 2: Tree cost statistics.

It is still an open question whether there exists a polynomial-time algorithm that constructs a routing tree with both cost and density bounded by constants times optimal in the *worst* case. It is also unknown whether the MDIT problem is NP-complete. Recall that the chain-peeling method, PEEL, holds some promise in the sense that there exist examples where it outperforms COMB and COMB_ST by a factor of $\Theta(\sqrt{n})$ (Figure 8); we conjecture that PEEL can be shown to yield worst-case density that is within a small constant factor of optimal. Indeed, we offer two closely related conjectures: (i) that the minimum density of a spanning tree over net $N$ is at least the minimum of the number of chains or the number of antichains needed to cover $N$; and (ii) the PEEL algorithm will use at most two times the minimum possible number of chains/antichains that cover $N$.

# References


[1] P. K. Agarwal, *Intersection and Decomposition Algorithms for Planar Arrangements*, Cambridge University Press, Cambridge, England, 1991.

[2] C. J. Alpert, T. C. Hu, J. H. Huang, and A. B. Kahng, *A Direct Combination of the Prim and Dijkstra Constructions for Improved Performance-Driven Global Routing*, in to appear in Proc. IEEE Intl. Symp. on Circuits and Systems, Chicago, IL, May 1993.

[3] H. Bakoglu, J. T. Walker, and J. D. Meindl, *A Symmetric Clock-Distribution Tree and Optimized High-Speed Interconnections for Reduced Clock Skew in ULSI and WSI Circuits*, in Proc. IEEE Intl. Conf. on Computer Design, Port Chester, NY, October 1986, pp. 118–122.

[4] T. Barrera, J. Griffith, S. A. McKee, G. Robins, and T. Zhang, *Toward a Steiner Engine: Enhanced Serial and Parallel Implementations of the Iterated 1-Steiner Algorithm*, in to appear in the Great Lakes Symposium on VLSI, Kalamazoo, MI, March 1993.

[5] J. Beardwood, H. J. Halton, and J. M. Hammersley, *The Shortest Path Through Many Points*, Proc. Cambridge Philos. Soc., 55 (1959), pp. 299–327.

[6] K. D. Boese, J. Cong, A. B. Kahng, K. S. Leung, and D. Zhou, *On High-Speed VLSI Interconnects: Analysis and Design*, Proc. Asia-Pacific Conf. on Circuits and Systems, (1992), pp. 35–40.

[7] K. D. Boese and A. B. Kahng, *Zero-Skew Clock Routing Trees with Minimum Wirelength*, in Proc. IEEE Intl. ASIC Conf., Rochester, NY, September 1992, pp. 17–21.

[8] B. Chazelle, *Tight Bounds on the Stabbing Number of Spanning Trees in Euclidean Space*, Tech. Rep. CS-TR-155-88, Department of Computer Science, Princeton University, 1988.

[9] J. Cohoon and J. Randall, *Critical Net Routing*, in Proc. IEEE Intl. Conf. on Computer Design, Cambridge, MA, October 1991, pp. 174–177.

[10] J. Cong, A. B. Kahng, G. Robins, M. Sarrafzadeh, and C. K. Wong, *Performance-Driven Global Routing for Cell Based IC's*, in Proc. IEEE Intl. Conf. on Computer Design, Cambridge, MA, October 1991, pp. 170–173.

[11] ——, *Provably Good Performance-Driven Global Routing*, IEEE Trans. on Computer-Aided Design, 11 (1992), pp. 739–752.

[12] R. P. Dilworth, *A Decomposition Theorem for Partially Ordered Sets*, Ann. Math, 51 (1950), pp. 161–166.

[13] H. Edelsbrunner, *Algorithms in Combinatorial Geometry*, Springer-Verlag, Berlin, 1987.

[14] H. Edelsbrunner, L. Guibas, J. Hershberger, R. Seidel, M. Sharir, J. Snoeyink, and E. Welzl, *Implicitly Representing Arrangements of Lines or Segments*, in Proc. ACM Symp. on Computational Geometry, Urbana-Champaign, IL, June 1988, pp. 56–69.

[15] C. Hilbert and C. Rathmell, *Multichip Modeuls: System Advantages, Major Constructions, and Materials Technologies*, IEEE Press, 1991.

[16] J. Hunt and S. Szymanski, *A Fast Algorithm for Computing Longest Common Subsequence*, Comm. of ACM, 20 (1977), pp. 350–353.

[17] M. A. B. Jackson, A. Srinivasan, and E. S. Kuh, *Clock Routing for High-Performance IC's*, in Proc. ACM/IEEE Design Automation Conf., 1990, pp. 573–579.

[18] A. B. Kahng, J. Cong, and G. Robins, *High-Performance Clock Routing Based on Recursive Geometric Matching*, in Proc. ACM/IEEE Design Automation Conf., June 1991, pp. 322–327.

[19] A. B. Kahng and G. Robins, *A New Class of Iterative Steiner Tree Heuristics With Good Performance*, IEEE Trans. on Computer-Aided Design, 11 (1992), pp. 893–902.

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

[21] D. Richards, *Fast Heuristic Algorithms for Rectilinear Steiner Trees*, Algorithmica, 4 (1989), pp. 191–207.

[22] J. M. Steele, *Growth Rates of Euclidean Minimal Spanning Trees With Power Weighted Edges*, Annals of Probability, 16 (1988), pp. 1767–1787.

[23] R. S. Tsay, *Exact Zero Skew*, in Proc. IEEE Intl. Conf. on Computer-Aided Design, Santa Clara, CA, November 1991, pp. 336–339.

[24] E. Welzl, *Partition Trees for Triangle Counting and Other Range Searching Problems*, in Proc. ACM Symp. on Computational Geometry, Urbana-Champaign, IL, June 1988, pp. 23–33.

[25] P. Winter, *Steiner Problem in Networks: A Survey*, Networks, 17 (1987), pp. 129–167.

Charles J. Alpert was born in January 1969 in Bethesda, MD. He earned his B. S. in Math and Computational Sciences and his B. A. in History from Stanford University in 1991. He received his M.S. degree in Computer Science from UCLA in 1993 and is currently pursuing his Ph.D. there. His research interests include combinatorial algorithms, computational geometry, VLSI circuit partitioning, and classification. He received the National Defence Science and Engineering Graduate Fellowship in 1991 and is a member of IEEE.

Jingsheng (Jason) Cong received the B. S. degree in computer science from Peking University in 1985, and the M. S. and Ph.D. degrees in computer science from the University of Illinois at Urbana-Champaign in 1987 and 1990, respectively. Currently, he is an Associate Professor in the Computer Science Department at UCLA. His research interests include computer-aided design of VLSI circuits, fault-tolerant design of VLSI systems, and design and analysis of efficient combinatorial and geometric algorithms. He is a member of ACM.

Andrew B. Kahng (b. Oct. 1963, San Diego, CA) holds the A.B. degree in applied mathematics and physics from Harvard College, and the M.S. and Ph.D. degrees in computer science from the University of California at San Diego. Since July 1989, he has been an assistant professor in the Computer Science Department at UCLA, where he has received NSF Research Initiation and Young Investigator awards. His research interests include: VLSI layout and logic synthesis, pattern recognition and image processing, combinatorial and parallel algorithms, computational geometry, and the theory of global optimization. He is a member of ACM, IEEE and SIAM.

Gabriel Robins received the Ph.D. degree from the UCLA Computer Science Department, where he won the Distinguished Teaching Award and held an IBM Graduate Fellowship. Currently, Dr. Robins is Assistant Professor of Computer Science in the University of Virginia at Charlottesville. His primary areas of research are VLSI CAD and geometric algorithms, with recent work focusing on performance-driven routing, Steiner tree heuristics, motion planning, pattern recognition algorithms, and computational biology. Dr. Robins is the author of a Distinguished Paper at the 1990 IEEE International Conference on Computer-Aided Design. He is a member of ACM, IEEE, MAA and SIAM.

Majid Sarrafzadeh received the B.S., M.S., and Ph.D. degrees in 1982, 1984, and 1986, respectively, all from the University of Illinois at Urbana-Champaign in electrical and computer engineering. He is currently Assistant Professor of Electrical Engineering and Computer Science at Northwestern University. His research interests lie in the area of design and analysis of algorithms and computational complexity, with emphasis in VLSI. Dr. Sarrafzadeh received a NSF Engineering Inititiation Award in 1987, and a Design Automation

Award in 1988.

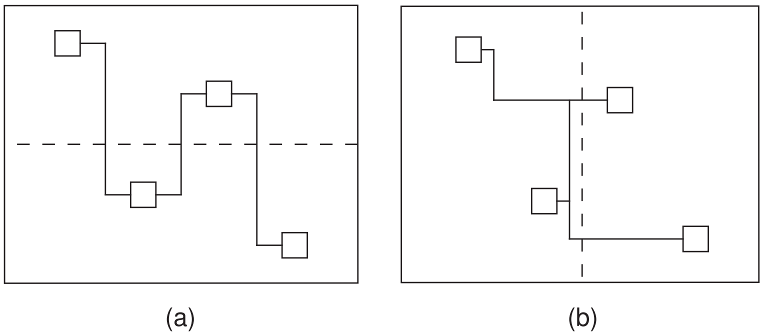



Figure 1: A four-terminal signal net for which the tree on the left increases the required layout dimension by three routing grids, while the tree on the right requires only two routing grids.

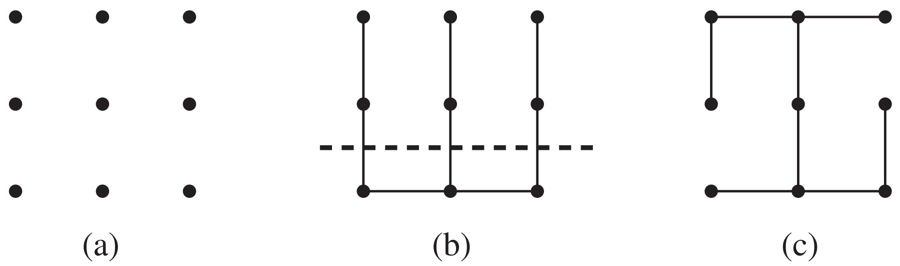
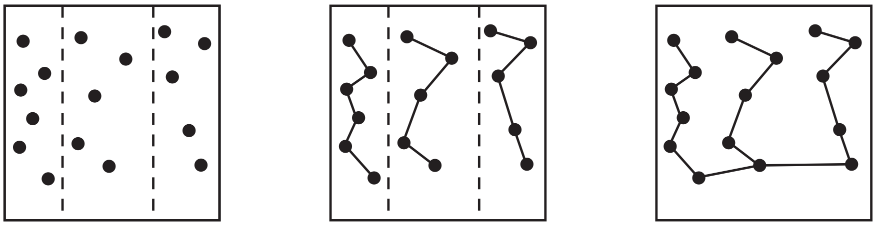



Figure 2: (a) Example of a signal net, along with (b) an interconnection tree with density = 3, and (c) a minimum density tree with density = 2.

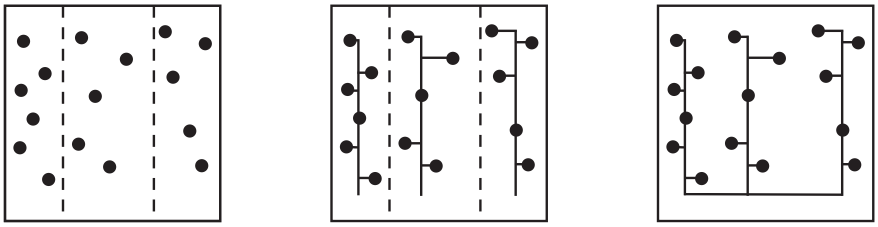


(a) (b) (c)

Figure 3: Execution of the COMB spanning tree construction on a net of size $n = 16$.

| **Algorithm: COMB** |
|---|
| **Input:** a net $N$, containing $\lvert N \rvert = n$ terminals<br>**Output:** a low-density low-cost interconnection tree spanning $N$ |
| **Partition** $N$ into $\frac{\sqrt{n}}{\sqrt{2}}$ vertical strips each containing $\sqrt{2n}$ terminals<br>**Connect** in monotone $y$-order the terminals within each strip<br>**Connect** in monotone $x$-order the bottom terminals of all strips<br>**Output** resulting spanning tree |

Figure 4: Algorithm COMB: heuristic minimum-density spanning tree construction.



(a) (b) (c)

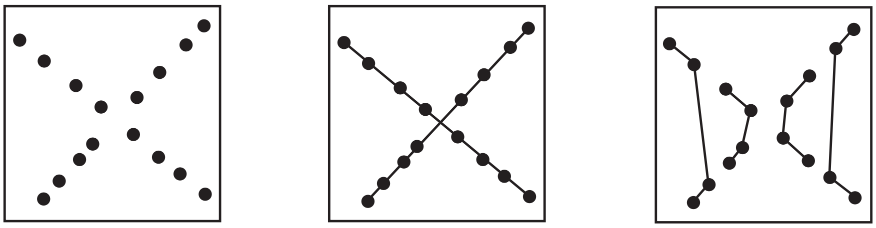
Figure 5: Execution of the COMB_ST Steiner tree construction on a net of size $n = 16$. Note that density $= 3$ is achieved by the construction, while the COMB construction yielded density $= 5$ for the same instance.

| **Algorithm: COMB_ST** |
|---|
| **Input:** a net $N$, containing $\lvert N \rvert = n$ terminals |
| **Output:** a low-density low-cost Steiner tree connecting $N$ |
| 1: **Partition** $N$ into $\frac{\sqrt{n}}{\sqrt{2}}$ vertical strips each containing $\sqrt{2n}$ terminals |
| 2: **Connect** the terminals within each strip to a central spine |
| 3: **Connect** the bottoms of all spines |
| 4: **Output** resulting Steiner tree |

Figure 6: Algorithm COMB_ST: heuristic minimum-density Steiner tree construction.

| **Algorithm: PEEL** |
|---|
| **Input:** a net $N$, containing $\lvert N \rvert = n$ terminals |
| **Output:** a low-density low-cost tree spanning $N$ |
| 1: $S = N$ |
| 2: $T = \emptyset$ |
| 3: **While** $S \neq \emptyset$ **Do** |
| 4: $\quad C =$ maximum chain or antichain of $S$ |
| 5: $\quad T = T \cup C$ |
| 6: $\quad S = S - C$ |
| 7: **Join** all chains/antichains in $T$ and output resulting tree |

Figure 7: Algorithm PEEL produces a low-density tree by iteratively computing maximum chains or antichains, then joining them into an interconnection tree.



(a) (b) (c)

Figure 8: Illustration (a) of class of examples on which PEEL (b) performs an unbounded factor better than either COMB (c) or COMB_ST. The connecting edges between the strips (Step 3 of COMB) are not shown in (c). For points in an "X" configuration, PEEL will always yield a constant density = 2, while COMB or COMB_ST density will grow as the square root of $n$.

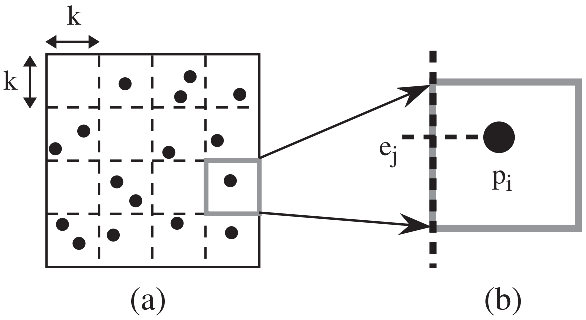



Figure 9: Expected minimum density of a net: (a) the unit square is partitioned into $n$ congruent cells; (b) each non-empty cell contains some terminal $p_i$ which contributes at least one edge $e_j$ that crosses a cell boundary.

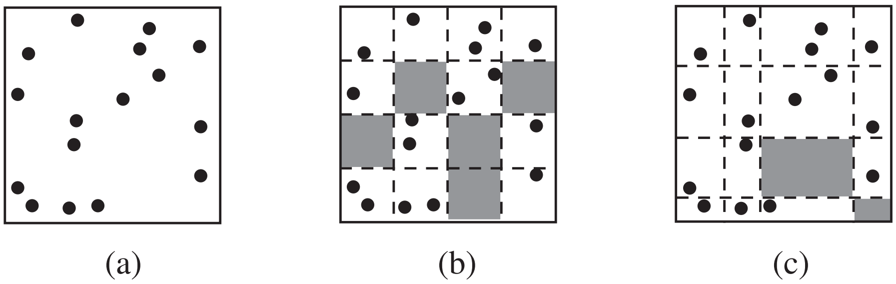



Figure 10: Computing a non-uniform lower bound on the density. For the net in (a), a uniform partition of the unit square into 16 squares of size $\frac{1}{4} \times \frac{1}{4}$ each, shown in (b), yields 11 non-empty cells which imply a density lower bound of $\lceil \frac{11-1}{(4-1)+(4-1)} \rceil = 2$. On the other hand, the non-uniform partition shown in (c) yields 14 non-empty cells, which imply an improved density lower bound of $\lceil \frac{14-1}{(4-1)+(4-1)} \rceil = 3$.

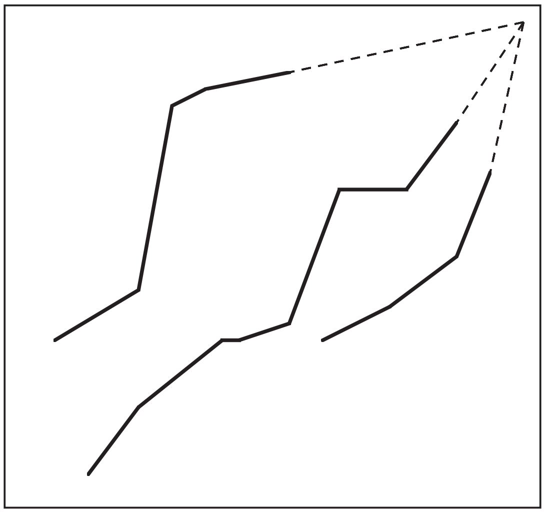


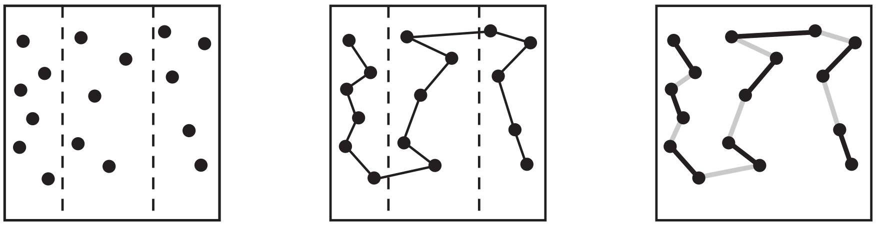
Figure 11: Combining chains into a low-density tree.

**Algorithm: CLOCK**

**Input:** a net $N$, containing $|N| = n$ terminals
**Output:** A low pathlength skew tree topology $T$

$T = \emptyset$
$P = N$
**While** $|P| > 1$ **Do**
  $M =$ the edges of the optimal geometric matching over $P$
  $P' = \emptyset$
  **For** $(p_1, p_2) \in M$ **Do**
    $T_1 =$ the subtree of $T$ rooted at $p_1$
    $T_2 =$ the subtree of $T$ rooted at $p_2$
    $p =$ a point lying *between* $p_1$ and $p_2$ on the line containing $p_1$ and $p_2$, such that $p$ minimizes skew of the tree $T_1 \cup T_2 \cup \{(p, p_1), (p, p_2)\}$ rooted at $p$
    $P' = P' \cup \{p\}$
    $T = T \cup \{(p, p_1), (p, p_2)\}$
  $P = P'$ plus a possible unmatched node if $|P|$ is odd
clock routing tree $= T$
root of $T =$ single remaining point in $P$

Figure 12: Matching-based, minimum-skew clock tree construction of [18].



(a) (b) (c)

Figure 13: (a) Partitioning a net into strips/chains; (b) a serpentine tour with low density and low average cost and low density; and (c) an embedded geometric matching which also has low density and low average cost.

| **Algorithm: BRBC** |
|---|
| **Input:** a net $N$, containing $\|N\| = n$ terminals, source $s \in N$, and a real parameter $\epsilon \geq 0$ <br> **Output:** a tree $T$ with radius $\leq (1+\epsilon) \cdot R$ and cost $\leq (1 + \frac{2}{\epsilon}) \cdot cost(MST)$. |
| $R$ = radius of $N$ <br> $Q = MST(N)$ = minimum spanning tree over $N$ <br> $L$ = depth-first tour of $Q$ <br> $S = 0$ <br> **For** $i = 1$ to $\|L\| - 1$ **Do** <br>     $S = S + dist(L_i, L_{i+1})$ <br>     **If** $S \geq \epsilon \cdot R$ **Then** <br>         $Q = Q \cup \{(s, L_{i+1})\}$ <br>         $S = 0$ <br> $T$ = shortest path tree of $Q$ |

Figure 14: BRBC algorithm of [11], which computes a bounded-radius spanning tree $T$ for a net $N$, with source $s \in N$ and radius $R$, using parameter $\epsilon$ (we use $dist(L_i, L_{i+1})$ to denote the edge cost between two consecutive points in the depth-first tour $L$). $T$ will have radius at most $(1+\epsilon) \cdot R$, and cost at most $(1 + \frac{2}{\epsilon}) \cdot cost(MST(N))$.