Closing the Gap: Near-Optimal Steiner Trees in Polynomial Time *

Jeff Griffith, Gabriel Robins, Jeffrey S. Salowe and Tongtong Zhang

Department of Computer Science, University of Virginia, Charlottesville, VA 22903-2442

Abstract

The minimum rectilinear Steiner tree (MRST) problem arises in global routing and wiring estimation, as well as in many other areas. The MRST problem is known to be NP-hard, and the best performing MRST heuristic to date is the Iterated 1-Steiner (I1S) method recently proposed by Kahng and Robins. In this paper we develop a straightforward, efficient implementation of I1S, achieving a speedup factor of three orders of magnitude over previous implementations. We also give a parallel implementation that achieves near-linear speedup on multiple processors. Several performance-improving enhancements enable us to obtain Steiner trees with average cost within 0.25% of optimal, and our methods produce optimal solutions in up to 90% of the cases for typical nets. We generalize I1S and its variants to three dimensions, as well as to the case where all the pins lie on k parallel planes, which arises in, e.g., multi-layer routing. Motivated by the goal of reducing the running times of our algorithms, we prove that any pointset in the Manhattan plane has a minimum spanning tree (MST) with maximum degree 4, and that in three-dimensional Manhattan space every pointset has an MST with maximum degree of 14 (the best previous upper bounds on the maximum MST degree in two and three dimensional are 6 and 26, respectively); these results are of independent theoretical interest and also settle an open problem in complexity in theory.

1 Introduction

The minimum rectilinear Steiner tree problem is central to VLSI physical-design phases such as global routing and wiring estimation, where we seek low-cost topologies to interconnect the pins of signal nets [30] [37]:

The Minimum Rectilinear Steiner Tree (MRST) problem: Given a set P of n points, find a set S of *Steiner points* such that the minimum rectilinear spanning tree (MST) over $P \cup S$ has minimum cost.

*Professor Gabriel Robins, Professor Jeffrey Salowe, and Ms. Tongtong Zhang are with the Department of Computer Science, University of Virginia, Charlottesville, VA 22903-2442. Mr. Jeff Griffith is with the Department of Computer Science, University of Tennessee, Knoxville, TN 37996-1301. All correspondence and code requests should be addressed to the primary author: Professor Gabriel Robins, robins@cs.virginia.edu, phone: (804) 982-2207, FAX: (804) 982-2214. Professor Robins was partially supported by NSF Young Investigator Award MIP-9457412. Professor Salowe was partially supported by NSF grants MIP-9107717 and CCR-9224789.

The cost of a tree edge is the Manhattan distance between its endpoints, and the cost of a tree is the sum of its edge costs. The terms *points* and *pins* are synonymous and are used interchangeably, depending on the context (a *net* is a set of pins). Figure 1 shows an MST and an MRST for the same four-pin net.

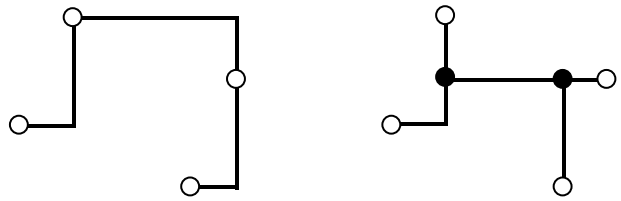

Figure 1: A minimum spanning tree (left) and MRST (right) for a fixed net; hollow dots represent the original pointset P, while solid dots represent the set S of added Steiner points.

Research on the MRST problem has been guided by several fundamental results. First, Hanan [21] has shown that there always exists an MRST with Steiner points chosen from the intersections of all the horizontal and vertical lines passing through all the points in P (see Figure 2), and this result was generalized by Snyder [42] to all higher dimensions. However, a second major result by Garey and Johnson [16] establishes that despite this restriction on the solution space, the MRST problem remains NP-complete; this has given rise to numerous heuristics as surveyed by Hwang, Richards and Winter [27].

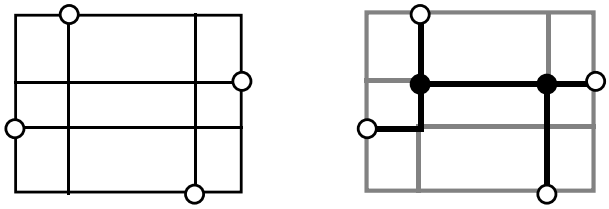

Figure 2: Hanan's theorem: there always exists an MRST with Steiner points chosen from the intersections of all the horizontal and vertical lines passing through all the points.

In solving intractable problems, we often seek provably-good heuristics having bounded worst-case error from optimal. Thus, a third important result is the discovery by Hwang [25] that the rectilinear MST is a fairly good approximation to the MRST, with a worst-case performance ratio of $\frac{cost(MST)}{cost(MRST)} \leq \frac{3}{2}$. This implies that any MST-based strategy that improves upon an initial MST topology will also enjoy a performance ratio of at most $\frac{3}{2}$. This has prompted a large number of Steiner tree heuristics that resemble classic MST construction methods [23] [24] [26] [31] [32], all producing

Steiner trees with average cost 7% to 9% smaller than MST cost [27].

Unfortunately, all MST-based MRST constructions were recently shown by Kahng and Robins [29] to have a worst-case performance ratio of exactly $\frac{3}{2}$. This negative result has motivated research into alternate schemes for MRST approximation, with the best performing among these being the Iterated 1-Steiner (I1S) algorithm [28] [38]. I1S always performs strictly better than $\frac{3}{2}$ times optimal[1], and also performs very well in practice, achieving almost 11% average improvement over MST cost, which is on average less than 0.5% away from optimal [40]. The Iterated 1-Steiner method was generalized to arbitrary weighted graphs by Alexander and Robins [1] [2], and is thus a suitable basis of a practical global router which must handle congestion, obstacles, etc. [15].

The performance success of I1S was achieved at the expense of a high time complexity: although a more efficient variant of I1S can be implemented to run within time $O(n^2 \log n)$ [28], the computational geometric methods employed to achieve this time bound hide large constant factors and are also difficult to code. Thus, actual previous implementations of I1S typically use a more straight-forward approach which requires time $O(n^4 \log n)$.

The first contribution of this paper is a practical implementation of I1S that runs within time $O(n^3)$. Our method is based on a dynamic minimum spanning tree update scheme, and establishes the practicality and viability of the iterated 1-Steiner approach. For 100 points our new implementation is about three orders of magnitude faster than the naive implementation, and the speedup over the naive implementation increases with the number of points. This dramatic improvement in speed enables for the first time the testing of I1S on nets containing several hundred pins.

Since a typical CAD environment consists of a network of workstations, exploiting the available large-grain parallelism provides a natural and effective means of improving the performance of CAD algorithms. With this in mind, a second contribution of our work is a parallel version of I1S that achieves high parallel speedups. Since Steiner tree construction is a computationally expensive part of global routing, our parallel implementation may be viewed as an important first step toward obtaining a "Steiner engine", i.e. an efficient tool for producing near-optimal Steiner trees.

[1] Recently, Berman and Ramaiyer [7] and Foessmeier, Kaufmann and Zelikovsky [6] [13] have extended the fundamental work of Zelikovsky [46] [47] to yield a method similar to I1S (specifically, to the "batched" I1S method described below) with performance ratio bounded by $\frac{11}{8}$; this work settles in the affirmative the longstanding open question of whether there exists a polynomial-time rectilinear Steiner tree heuristic with performance ratio strictly smaller than $\frac{3}{2}$ [25]. At the time of this writing, Berman, Foessmeier, Karpinski, Kaufmann and Zelikovsky [13] further improved the performance bound of their polynomial-time rectilinear Steiner heuristic to $\frac{61}{48} = 1.271$.

Our third contribution entails several performance-improving enhancements to the I1S method. Our methods rely on an approach that deviates from pure greed and instead employs randomness in order to break ties during Steiner point selection. While being asymptotically no slower than the original I1S variants, our methods afford improved average performance. Extensive simulations indicate that for uniformly distributed random nets of up to 8 pins, the average performance of our enhanced I1S algorithm is only 0.25% away from optimal. Moreover, for 8-pin nets, our method produces the optimal Steiner tree in 90% of all instances. We also propose a method of improving performance at the expense of running time, allowing a smooth tradeoff between solution quality and efficiency.

Next, we generalize I1S and its variants to three dimensions, as well as to the intermediate case where all pins lie on k parallel planes. This formulation has several applications, including multi-layer routing [9] [20] [22], and the design of buildings [41]. Empirical testing suggests that this approach is effective for three-dimensional Steiner routing, yielding up to 15% average improvement over MST cost in three-dimensional Manhattan space.

Finally, in order to reduce the running time of the dynamic MST-maintenance component of our algorithms, we prove the following results under the Manhattan metric: (1) every two-dimensional pointset has an MST with maximum degree of at most 4; and (2) every three-dimensional pointset has an MST with maximum degree of at most 14 (the best previously known bounds for two and three dimensions were 6 and 26, respectively). Our results and algorithms on degree-bounded minimum spanning trees are of significant independent theoretical interest [39], and settle several open issues in complexity theory[2].

The rest of the paper is organized as follows. In Section 2 we review the I1S method. Section 3 outlines our more efficient implementation of I1S, and Section 4 describes a variant of I1S with enhanced performance. Section 5 generalizes I1S to three dimensions, and proposes an efficient method for three-dimensional MST-maintenance. Section 6 proves tight bounds for the maximum MST degree in two and three dimensions under the Manhattan metric, and discusses the implications of our results to a problem in complexity theory. Section 7 outlines the parallel implementation, and Section 8 presents extensive simulation results regarding performance, running times, and parallel speedups. We conclude in Section 9 with directions for further research. A preliminary version of this work appeared in [3] and in [4].

[2]Robins and Salowe [39] investigate the maximum MST degree for higher dimensions and other L_p norms, and relate the maximum MST degree to the so-called "Hadwiger" numbers.

2 Review of the Iterated 1-Steiner Method

We begin with a review of the Iterated 1-Steiner method of Kahng and Robins [28]. For two pointsets P and S we define the MST savings of S with respect to P as $\Delta MST(P,S) = cost(MST(P)) - cost(MST(P \cup S))$. We use $H(P)$ to denote the set of Hanan Steiner point candidates (i.e., the intersections of all horizontal and vertical lines passing through points of P). For a pointset P, a *1-Steiner point* $x \in H(P)$ maximizes $\Delta MST(P, \{x\}) > 0$. The I1S method repeatedly finds 1-Steiner points and includes them into S. The cost of the MST over $P \cup S$ will decrease with each added point, and the construction terminates when there is no x with $\Delta MST(P \cup S, \{x\}) > 0$. Although a Steiner tree may contain at most $n-2$ Steiner points [18], I1S may add more than $n-2$ Steiner points; therefore, at each step we eliminate any extraneous Steiner points having degree 2 or less in the MST over $P \cup S$. Figure 3 illustrates a sample execution of I1S, and Figure 4 describes the algorithm formally.

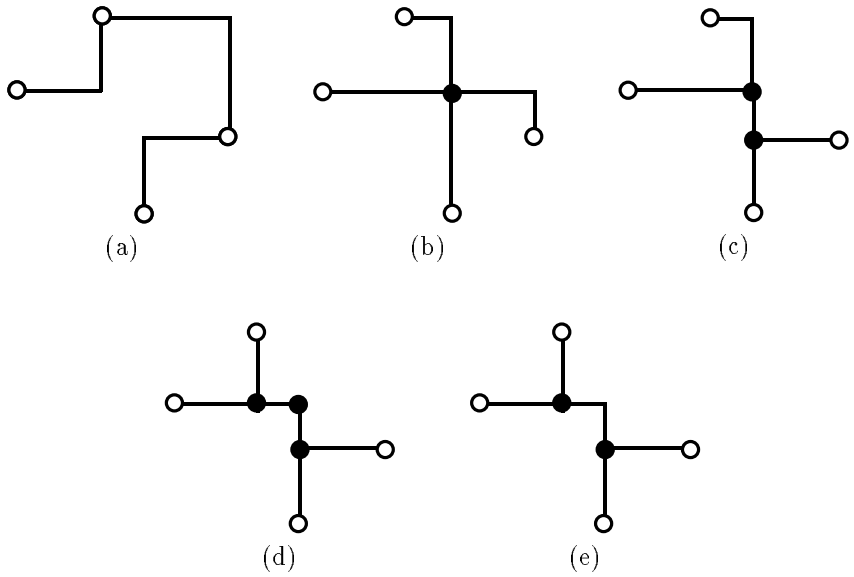

Figure 3: Execution of Iterated 1-Steiner (I1S) on a 4-pin net. Note that in step (d) a degree-2 Steiner point is formed and is thus eliminated from the topology.

Although a single 1-Steiner point may be found in $O(n^2)$ time using complicated techniques from computational geometry [17] [28], such methods suffer from large constants in their time complexities, and are notoriously difficult to implement. Thus, a *batched* variant of I1S is usually favored, which

Algorithm Iterated 1-Steiner (I1S) [28]
Input: A set P of n points **Output:** A rectilinear Steiner tree over P

$S = \emptyset$
While $T = \{x \in H(P) \mid \Delta MST(P \cup S, \{x\}) > 0\} \neq \emptyset$ **Do**
  Find $x \in T$ with maximum $\Delta MST(P \cup S, \{x\})$
  $S = S \cup \{x\}$
  Remove from S points with degree ≤ 2 in $MST(P \cup S)$
Output MST$(P \cup S)$

Figure 4: The Iterated 1-Steiner algorithm.

efficiently adds an entire set of "independent" Steiner points in a single *round*, thereby affording both practicality and reduced time complexity [28] [38].

Following Hanan's result, for each candidate Steiner point $x \in H(P)$ the Batched 1-Steiner (B1S) variant computes the induced MST savings $\Delta MST(P, \{x\})$. Next we select a maximal "independent" set of Steiner points, where the criterion for independence is that no candidate Steiner point is allowed to reduce the potential MST cost savings of any other candidate. More formally, a set S of Steiner points is *independent* if $\Delta MST(P, S) \geq \sum_{x \in S} \Delta MST(P, \{x\})$. The weight of set S is $\Delta MST(P, S)$, and our goal is to find an independent set of maximum weight; the Steiner points in that independent set are grouped together during a round of B1S.

Using a reduction from the NP-complete problem of finding a maximum independent set in a graph, it is easy to show that our maximization problem is NP-complete also, even if the independent sets obey the "inheritance property", one of the axioms for matroids (see Cormen et al. [11]). Our independent sets do not necessarily obey the inheritance property or the exchange property for matroids. Nevertheless, we can use a greedy approximation, described in Figure 5, to approximate the weight of a maximum independent set. Note that the greedy algorithm would find a maximum independent set if the subset system was a matroid. This approximation is efficient and seems to work well in practice; it would be interesting to prove nontrivial performance bounds.

Once an approximate maximum independent set S is determined, it is inserted into P, and we iterate this process with P set to $P \cup S$ until we reach a round that fails to induce a Steiner point. The total time required by B1S is $O(n^4 \log n)$ per round (the number of rounds in practice is a small constant independent of net size, i.e., less than 3 on average [28]). The B1S algorithm is summarized in Figure 5.

Algorithm Batched 1-Steiner (B1S) [28]
Input: A set P of n points **Output:** A rectilinear Steiner tree over P

While $T = \{x \in H(P) \mid \Delta MST(P, \{x\}) > 0\} \neq \emptyset$ **Do**
$S = \emptyset$
For $x \in \{T$ in order of non-increasing $\Delta MST\}$ **Do**
If $\Delta MST(P \cup S, \{x\}) \geq \Delta MST(P, \{x\})$ **Then** $S = S \cup \{x\}$
$P = P \cup S$
Remove from P Steiner points with degree ≤ 2 in MST(P)
Output MST(P)

Figure 5: The Batched 1-Steiner (B1S) algorithm.

3 A New Faster Implementation

In speeding up the MST-savings computations, a key observation is that once we have computed an MST over the pointset P, the addition of a single new point x into P can only induce a small constant number of changes between $MST(P)$ and $MST(P \cup \{x\})$. This follows from the observation that each point can have at most 8 neighbors in a rectilinear planar MST, i.e. at most one per octant [24]. Thus, to update an MST with respect to a newly added point x, it suffices to consider only the closest point to x in each of the 8 plane octants with respect to x (below we show that for each point it suffices to examine at most 4 potential candidates for connection in the MST).

Thus we have the following linear-time algorithm for dynamic MST maintenance: connect the new point x to each of its potential neighbors (i.e, the closest point to x in each of the $O(1)$ octants around x), then delete the longest edge on any resulting cycle. This dynamic MST maintenance scheme reduces the time complexity of each round of B1S from $O(n^4 \log n)$ to $O(n^3)$, a substantial savings. An execution example of this method is given in Figure 6, and Figure 7 describes it formally. Note that dynamic MST maintenance can also be achieved in sub-linear time [14], but such methods seem impractical due to their complicated description and large hidden constants. A similar method was also used by Yao [45] to obtain a sub-quadratic MST algorithm in higher dimensions, but no attempt was made to optimize the number of necessary regions, whereas we also optimize the number of regions.

We now show that only 4 regions suffice for dynamic MST maintenance, namely the 4 regions defined by the two lines oriented at 45 and -45 degrees (Figure 8(a)); we call this the *diagonal partition*. We couch the discussion in general terms so that we can later extend our terminology and techniques to the three-dimensional case. We begin by defining the following key property for regions and partitions:

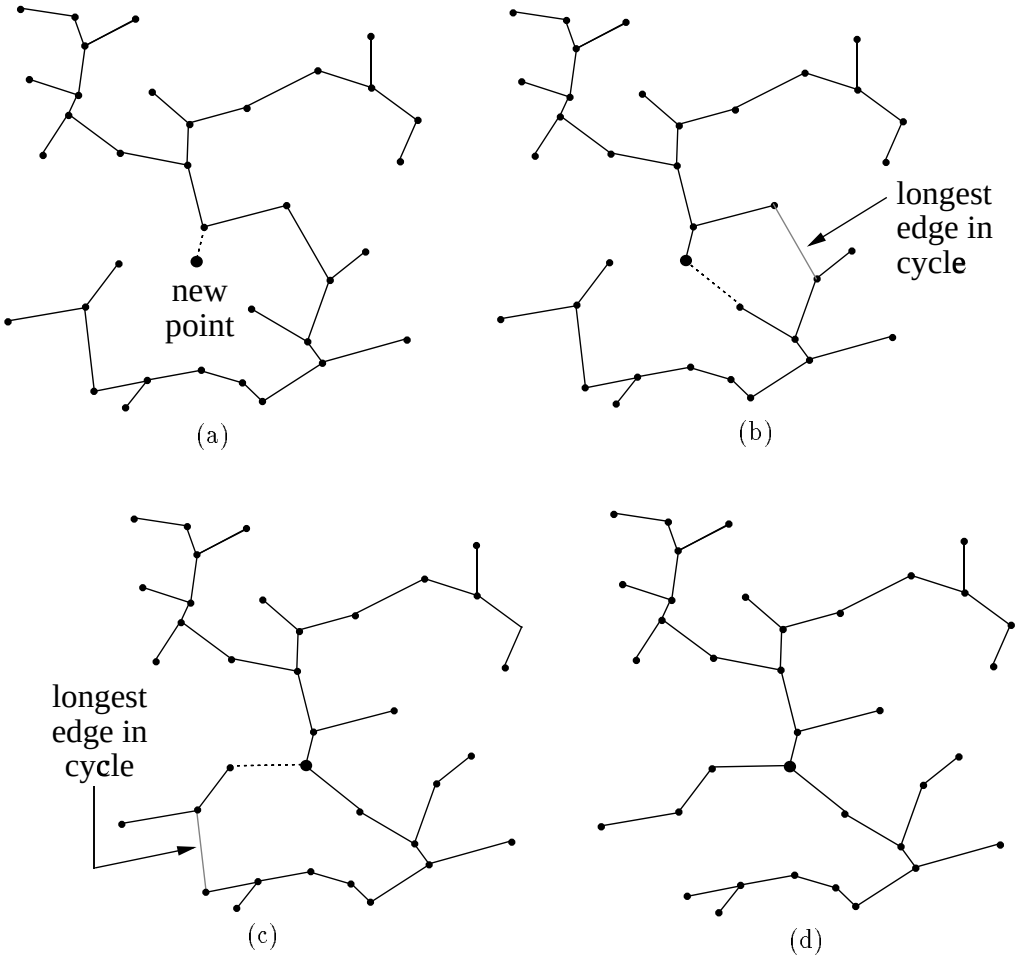

Figure 6: Dynamic MST maintenance: adding a point to an existing MST entails connecting the point to its closest neighbor in each octant, and deleting the longest edge on each resulting cycle (the Euclidean metric has been used for clarity in this example).

The Uniqueness Property: Given a point p, a region R has the *uniqueness property* with respect to p if for every pair of points $u, w \in R$, either $\text{dist}(w, u) \leq \text{dist}(w, p)$ or $\text{dist}(u, w) \leq \text{dist}(u, p)$.

We use $\text{dist}(u, w)$ to denote the Manhattan distance between the two points u and w. A partition of space (i.e., into a finite set of mutually disjoint regions whose union covers the space) is said to have the uniqueness property if each of its regions has the uniqueness property. Clearly, any partition scheme having the uniqueness property can be used in dynamic MST maintenance, since each region having the uniqueness property can contain at most one candidate for connection in the MST. Naturally, simpler partition schemes (i.e., ones containing a smaller number of regions) are preferable to more

Dynamic MST Maintenance (DMSTM)
Input: A set P of n points, $MST(P)$, a new point x **Output:** MST($P \cup \{x\}$)

$T = MST(P)$
For $i = 1$ to #regions **do**
  Find in region $R_i(x)$ the point $p \in P$ closest to x
  Add to T the edge (p, x)
  If T contains a cycle **Then** remove from T the longest edge on the cycle
Output T

Figure 7: Linear-time dynamic MST maintenance.

complicated ones. We now prove that the diagonal partition has the uniqueness property.

Lemma 3.1 *Given a point p in the Manhattan plane, each region of the diagonal partition with respect to p has the uniqueness property.*

Proof: The two diagonal lines through p partition the plane into four disjoint regions R_1 through R_4 (see Figure 8(a)). The boundary points between two neighboring regions may be arbitrarily assigned to either region. Consider one of the 4 regions, say R_1, and let $u, w \in R_1$ (Figure 8(b)). Assume without loss of generality that $\text{dist}(u, p) \leq \text{dist}(w, p)$ (otherwise swap u and w in this proof). Consider the diamond D in R_1 with one corner at p, and with u on the boundary of D (see Figure 8(c)). Let c be the center of D, so that c is equidistant from all points on the boundary of D, and let a ray starting at p and passing through w intersect the boundary of D at the point b. By the triangle inequality, $\text{dist}(w, u) \leq \text{dist}(w, b) + \text{dist}(b, c) + \text{dist}(c, u) = \text{dist}(w, b) + \text{dist}(b, c) + \text{dist}(c, p) = \text{dist}(w, p)$. Thus, w is not closer to p than it is to u and the region R_1 therefore has the uniqueness property. The other three regions are handled similarly. It follows that the diagonal partition has the uniqueness property.

□

A natural question is how to determine an optimal partitioning scheme for a given dimension and metric, i.e., finding a partition scheme that contains the smallest possible number of regions, yet still possesses the uniqueness property. The existence of pointsets in the Manhattan plane where the MST is forced to have degree 4 (i.e., the 5 points corresponding to the center and four corners of a diamond) establishes the optimality of the diagonal partition, in the sense that no partition of the Manhattan plane into less than 4 regions can have the uniqueness property. Section 5 addresses the problem of finding an optimal partition for three-dimensional Manhattan space.

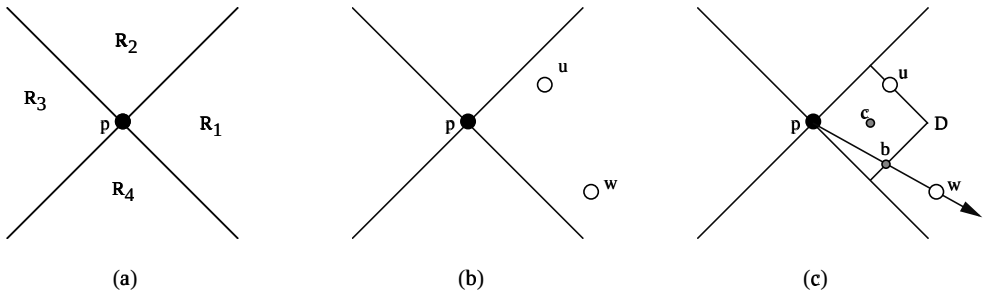

Figure 8: The *diagonal partition* of the plane into 4 regions with respect to a point p (a) has the *uniqueness property*: for every two points u and w that lie in the same region (b), either $\text{dist}(w,u) \leq \text{dist}(w,p)$ or else $\text{dist}(u,w) \leq \text{dist}(u,p)$ (c).

We have shown above that in the Manhattan plane, the degree of any particular single MST node can be made to be 4 or less. But note that this does not imply that the degrees of *all* nodes can be made 4 or less *simultaneously*, since decreasing the degree of one node can increase the degree of neighboring nodes. Thus, it is not immediately obvious that in the Manhattan plane there always exists an MST with maximum degree 4; this requires additional proof as detailed in Section 6 below.

4 A Variant With Improved Performance

At each iteration, the basic I1S heuristic uses pure greed to select a 1-Steiner point, and this may unfortunately preclude additional savings in subsequent iterations. A similar phenomenon may occur due to tie-breaking among 1-Steiner candidates that induce equal savings. For example, in Figure 9 we observe that an unfortunate choice for a 1-Steiner point can interfere with the savings of future potential 1-Steiner candidates, resulting in a suboptimal solution.

Empirical tests indicate that ties in MST savings for the various 1-Steiner point candidates occur very often. Therefore, in order to avoid breaking ties in ways that would preclude possible future savings, we propose the following scheme: when an MST savings tie occurs among a number of 1-Steiner candidates, rather than using a deterministic tie-breaking rule, we instead randomly select one of the 1-Steiner candidates and proceed with the execution. We then run this randomized variant of I1S m times on the same input, and select the best solution (i.e., the least costly of the m trees produced), where m is an input parameter.

In order to further avoid the pitfalls of a purely greedy strategy (i.e., getting trapped in local

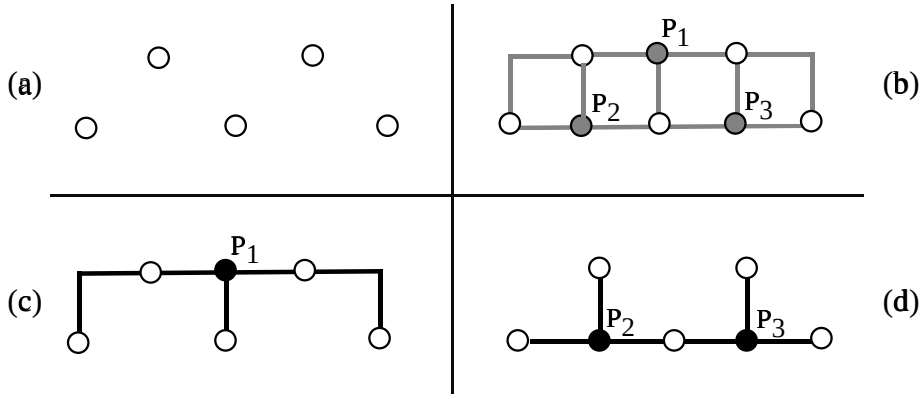

Figure 9: An unlucky tie-breaking choice for a 1-Steiner point may interfere with the savings of other potential 1-Steiner candidates. If P_1 is selected in the first iteration (b), then the MST savings of both P_2 and P_3 vanish during the second iteration, yielding a suboptimal tree of cost 7 (c); on the other hand, if P_2 is selected in the first iteration, then P_3 may be selected in the second iteration, yielding an optimal tree of cost 6 (d).

minima), we also propose a mechanism that allows I1S to select a 1-Steiner candidate if its MST savings is within δ units from that of the best candidate, where δ is again an input parameter. This strategy would enable the acceptance of a slightly suboptimal choice (with respect to the best immediate possible savings), with the possibility of realizing greater savings in future iterations.

Finally, we note that performance may be further improved if instead of looking for individual 1-Steiner points, we search for *pairs* of Steiner candidates that offer maximum savings with respect to other candidates or pairs of candidates. For example, such an *Iterated 2-Steiner algorithm* (I2S) will optimally solve the example pointset of Figure 9. Combining these three techniques of (1) non-deterministic tie-breaking, (2) near-greedy search, and (3) k-Steiner selection, we obtain a new *Enhanced Iterated k-Steiner* (EIkS) algorithm, as shown in Figure 10.

Note that the original I1S algorithm of Kahng and Robins [28] (see Figure 4) is equivalent to our new EIkS algorithm with $k = 1$, $m = 1$, and $\delta = 0$. Our EIkS scheme can also be extended using a "non-interfering" criterion as in [28], to yield an *enhanced batched k-Steiner* (EBkS) algorithm, where a maximal number of Steiner points are added during each round. The time complexity of EBkS is $O(m \cdot n^{2(k-1)} \cdot T(n))$, where $T(n)$ is the time complexity of B1S. For fixed m, EB1S runs asymptotically within the same time as B1S, namely $O(n^3)$ per round. The EBkS method improves the quality of the solutions at the expense of running time, allowing a smooth tradeoff between performance and efficiency. Although EBkS is guaranteed to always yield optimal solutions for $\leq k + 2$ pins, its time

Algorithm Enhanced Iterated k-Steiner (EIkS)
Input: A set P of n points, parameters $\delta \geq 0$, $k \geq 1$, and $m \geq 1$ **Output:** A rectilinear Steiner tree over P

T =MST(P)
Do m times
  $S = \emptyset$
  While $C = \{X \subseteq H(P) \mid \|X\| \leq k, \Delta\text{MST}(P \cup S, X) > 0\} \neq \emptyset$ **Do**
  Find $Y \in C$ with maximum $\Delta\text{MST}(P \cup S, Y)$
  Randomly select $Z \in C$ with $\Delta\text{MST}(P \cup S, Z) \geq \Delta\text{MST}(P \cup S, Y) - \delta$
  $S = S \cup Z$
  Remove from S points with degree ≤ 2 in MST($P \cup S$)
  If cost(MST($P \cup S$)) $<$ cost(T) **Then** T =MST($P \cup S$)
Output T

Figure 10: The Enhanced Iterated k-Steiner (EIkS) method.

complexity increases exponentially with k; thus in order to remain within polynomial time, k must be fixed.

Finally, it was observed empirically that only a small fraction of the Hanan candidates have positive MST savings in a given round of B1S; moreover, only candidates with positive MST savings in an earlier round are likely to produce positive MST savings in subsequent rounds. Therefore, rather than examine the MST savings of all Hanan candidates in a given round, we may consider only the candidates that produced positive savings in the previous round. The empirical simulations described in Section 8 indicate that this strategy significantly reduces the time spent during a round, without degrading solution quality. We call this streamlined version of B1S the *modified batched 1-Steiner* (MB1S) algorithm.

5 Steiner Routing in Three Dimensions

Three-dimensional packaging is emerging as a viable VLSI design technology [9] [20] [22]; however, most existing CAD routing tools and techniques still implicitly address two dimensions only. In contrast, the EIkS method readily generalizes to arbitrary dimensions. We distinguish between the general three-dimensional version of the Steiner problem, and the less general (but more realistic) multi-layer formulation, where the points of P lie on L parallel planes. Note that the unrestricted three-dimensional version of the Steiner problem occurs in the limit when $L = \infty$, and the standard two-dimensional formulation is the case $L = 1$. The cost of routing between one layer and another (i.e., using vias) is

likely to be substantially higher than staying on the same single layer (i.e., in terms of manufacturing expense, signal propagation delay, etc.), and this may be modeled by varying the distance between the layers. In Section 8 we present simulation data for several combinations of values for L and n.

Our three-dimensional EIkS method may be implemented efficiently by using Snyder's [42] generalization of Hanan's theorem to higher dimensions. In particular, there always exists an optimal Steiner tree whose Steiner points are chosen from the $O(n^3)$ intersections of all orthogonal planes (i.e., planes parallel to the coordinate axis) passing through all points in P. The three-dimensional analog of Hwang's result suggests that the maximum MST/MRST ratio for three dimensions is at most $\frac{5}{3}$ (this is a consequence of a more general conjecture for higher dimensions [19]), although there is currently no known proof of this. An example consisting of six points located in the middles of the faces of a rectilinear cube establishes that $\frac{5}{3}$ is a lower bound for the worst MST/MRST performance ratio in three dimensions. Thus, we expect the average performance of our heuristics, expressed as percent improvement over MST, to be higher in three dimensions than it is in two dimensions; this is indeed confirmed by our experimental results in Section 8. Also as expected, the average performance improves as the number of layers L increases.

As noted above, once we have computed an MST over a pointset P, the addition of a single new point p into P can only induce a small constant number of topological changes between $MST(P)$ and $MST(P \cup \{p\})$. This follows from the fact that in a fixed dimension, each point can have at most $O(1)$ neighbors in a rectilinear MST, as noted above in Section 3. Thus, MST savings in three dimensions may be efficiently calculated by partitioning the space with respect to the new point p into $O(1)$ mutually disjoint regions $R_i(p)$ having the uniqueness property, namely that only the closest point to p in each region $R_i(p)$ may be connected to p in the MST(P). This implies that in three-dimensional Manhattan space, linear time suffices to compute the MST savings of each 1-Steiner candidate.

Using similar arguments to those of Lemma 3.1, we can partition three-dimensional Manhattan space into 14 regions, with the uniqueness property holding for each region. Such a partition corresponds to the faces of a truncated cube (Figure 11(a)), i.e., a solid obtained by chopping off the corners of a cube, yielding 6 square faces and 8 equilateral triangle faces (Figure 11(b)); this solid is known as a "cuboctahedron" [34]. The 14 solid regions of this partition are induced by the 14 faces of the cuboctahedron, namely the 6 pyramids with square cross-section (Figure 11(c)) and the 8 pyramids with triangular cross-section (Figure 11(d)). Again, points located on region boundaries may be arbitrarily

assigned to any of the adjacent regions. We call this particular partition of space the *cuboctahedral partition*, and refer to the two types of induced regions as *square pyramids* and *triangular pyramids*, respectively. We now show that the cuboctahedral partition has the uniqueness property.

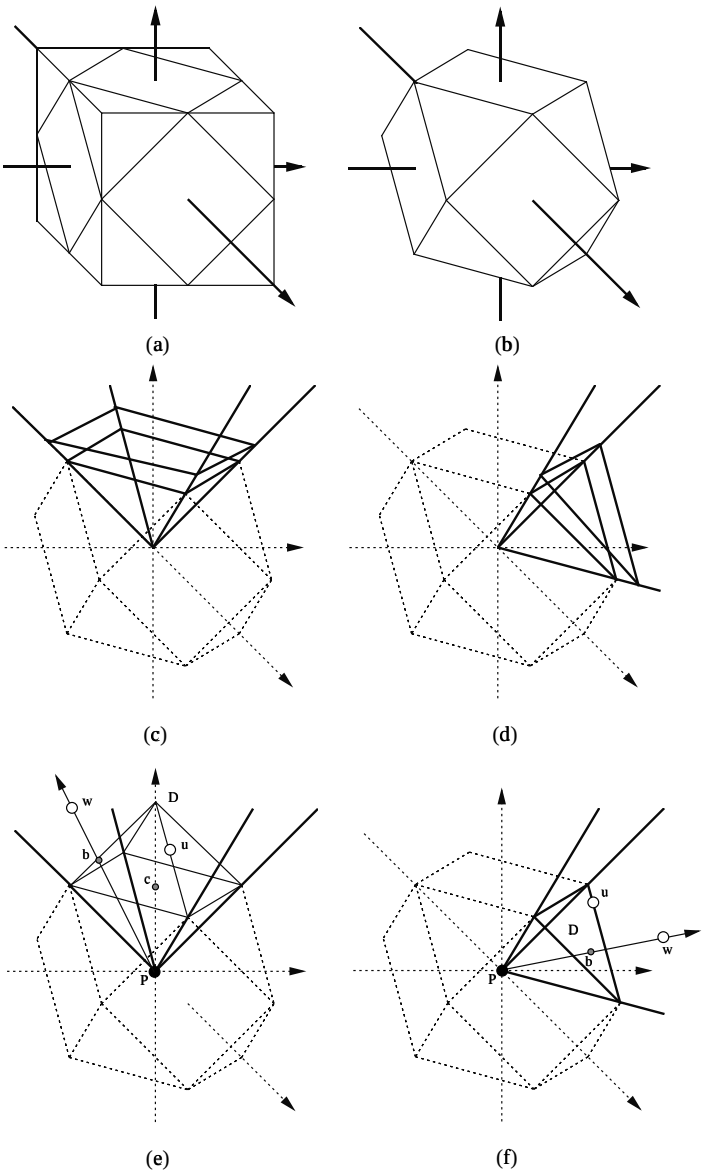

Figure 11: A truncated cube (a-b) induces a three-dimensional cuboctahedral space partition where each region has the uniqueness property. The 14 induced regions consist of 6 square pyramids (c), and 8 triangular pyramids (d). Using the triangle inequality, the uniqueness property may be shown to hold for each region (e-f).

Theorem 5.1 *Given a point p in three-dimensional space under the Manhattan metric, each of the* 14

regions of the cuboctahedral partition of space with respect to p has the uniqueness property.

Proof: To show the uniqueness property for the square pyramids, consider one of the square pyramids R with respect to p (Figure 11(c)), and let $u, w \in R$. Assume without loss of generality that $\text{dist}(u,p) \leq \text{dist}(w,p)$ (otherwise swap the roles of u and w in this proof). Consider the locus of points $D \subset R$ that are distance $\text{dist}(u,p)$ from p. (Figure 11(e)); D is the upper half of the boundary of an octahedron. Let c be the center of the octahedron determined by D, so that c is equidistant from all points of D. Let b be the intersection of the surface of D with a ray starting from p and passing through w. By the triangle inequality, $\text{dist}(w,u) \leq \text{dist}(w,b) + \text{dist}(b,c) + \text{dist}(c,u) = \text{dist}(w,b) + \text{dist}(b,c) + \text{dist}(c,p) = \text{dist}(w,p)$. Thus, w is not closer to p than it is to u and therefore the region R has the uniqueness property. The other square pyramids are handled similarly.

To show the uniqueness property for the triangular pyramids, consider one of the triangular pyramids R with respect to p (Figure 11(d)), and let $u, w \in R$. Assume without loss of generality that $\text{dist}(u,p) \leq \text{dist}(w,p)$ (otherwise swap the roles of u and w). Consider the locus of points D in R that are distance $\text{dist}(u,p)$ from p (Figure 11(f)). Let b be the intersection of D with a ray starting from p and passing through w. By the triangle inequality, $\text{dist}(w,u) \leq \text{dist}(w,b) + \text{dist}(b,u) \leq \text{dist}(w,b) + \text{dist}(b,p) = \text{dist}(w,p)$. Thus, w is not closer to p than it is to u and therefore the region R has the uniquenes property. The other triangular pyramids are handled similarly. □

We thus have the following:

Corollary 5.2 *Given a pointset P in three-dimensional Manhattan space and an additional new point p, there exists an MST over $P \cup \{p\}$ where p has degree of at most* 14 *in the MST.*

Proof: The cuboctahedral partition of space with respect to p yields 14 regions, each possessing the uniqueness property. This implies that for any two points inside a region, one is closer to the other rather than to the origin; thus only one point inside each region is a viable candidate for an MST neighbor of p. Therefore, the degree of p (or any other single point) in the MST can be made 14 or less. □

It is still an open question whether for three dimensions the cuboctahedral partition is optimal (i.e., whether the cuboctahedral partition has a minimum number of regions among all partitions having the uniqueness property); we conjecture that it is. On the other hand, we can show that 13 is a lower

bound on the maximum MST degree in three-dimensional Manhattan Space:

Theorem 5.3 *There are three-dimensional pointsets for which the maximum degree of* any *MST is at least* 13.

Proof: Consider the pointset $P = \{(0,0,0), (\pm 100, 0, 0), (0, \pm 100, 0), (0, 0, \pm 100), (47, -4, 49), (-6, -49, 45), (-49, 8, 43), (-4, 47, -49), (-49, -6, -45), (8, -49, -43), (49, 49, 2)\}$.

The distance between every point and the origin is exactly 100 units, but the distance between any two non-origin points is strictly greater than 100 units. Therefore, the MST over P is unique with a star topology (i.e., all 13 points must connect to the origin in the MST) and thus the origin point has degree 13 in the MST. □

A remaining open question is whether there exists a three-dimensional pointset where the maximum MST degree is forced to be as high as 14. Given that each point can connect to at most 14 neighbors in the MST, we obtain the following linear-time algorithm for dynamic MST maintenance (DMSTM) in three dimensions: connect the new point in turn to each of its ≤ 14 potential neighbors, then delete the longest edge on each resulting cycle. This is a three-dimensional analog of the two-dimensional method given in Figures 6 and 7.

6 On The Maximum MST Degree

As noted above, Lemma 3.1 does not imply that in the Manhattan plane the maximum MST degree is 4, nor does Corollary 5.2 imply that the maximum MST degree in three dimensions is 14, since reducing the MST degree of any one point may increase the MST degree of other points. It turns out, however, that ties for connection during MST construction may always be broken appropriately so as to keep the overall MST degree low. We begin by defining a strict version of the uniqueness property:

The Strict Uniqueness Property: Given a point p, a space region R has the *strict uniqueness property* with respect to p if for every pair of points $u, w \in R$, either $\text{dist}(w, u) < \text{dist}(w, p)$ or $\text{dist}(u, w) < \text{dist}(u, p)$.

Note that if a region has the strict uniqueness property it also has the (non-strict) uniqueness property. Clearly each d-dimensional region satisfying the strict uniqueness property may contribute at most 1 to the maximum MST degree. We now prove that by breaking ties judiciously, the maximum MST

degree can be made no larger than the number of d-dimensional regions in a partition having the strict uniqueness property:

Theorem 6.1 *Given a partition of d-dimensional space into r regions, and given that only $r' \leq r$ of these regions are d-dimensional and have the strict uniqueness property (the rest of the $r - r'$ regions being lower-dimensional, and are not required to have the uniqueness property), then the maximum MST degree in this space is r' or less.*

Proof: Given a pointset P, perturb the coordinates of each point by a tiny amount so that the lower-dimensional regions with respect to each point do not contain any other points. This is always possible to do, and yields a new perturbed pointset P'. Clearly, interpoint distances in P' only differ by a tiny amount from the corresponding interpoint distances in P, and therefore the cost of the MSTs over P' and P can differ by only a similarly tiny amount which can be made arbitrarily small. But the MST over P' has maximum degree r', since only the r' d-dimensional regions of the partition are nonempty with respect to the points of P'. We now use the topology of the MST for P' to connect the corresponding points of P; this would induce an MST over P having maximum degree r'. □

Applications of this technique to 2 and 3 dimensions are immediate:

Corollary 6.2 *Every pointset in the Manhattan plane has an MST with maximum degree* 4.

Proof: Modify the diagonal partition into a *strict* diagonal partition having a total of 8 regions: 4 two-dimensional open wedges (i.e., not containing any of their own boundary points), and 4 one-dimensional rays (i.e., the boundaries between the wedges). By arguments similar to those of Lemma 3.1, each of the open wedges possesses the strict uniqueness property, and thus by Theorem 6.1 points lying on the boundaries between wedges can be perturbed into the interiors of the wedges themselves. Thus the maximum MST degree given such a strict diagonal partitioning scheme is 4. □

The bound of 4 on the maximum MST degree in the Manhattan plane is tight, since there are examples which achieve this bound (e.g., the center and vertices of a diamond). Note that the best previously known upper bound for the maximum MST degree in the Manhattan plane was 6, as was stated without proof in [24].

Corollary 6.3 *Every pointset in three-dimensional Manhattan space has an MST with maximum degree* 14.

Proof: Modify the cuboctahedral partition into a *strict* cuboctahedral partition having a total of 38 regions: 14 three-dimensional open pyramids (i.e., 8 triangular pyramids and 6 square pyramids, each not containing any of their own boundary points), and 24 two-dimensional regions (i.e., corresponding to all the boundaries between the pyramids). By arguments identical to those of Theorem 5.1, each of the open pyramids possesses the strict uniqueness property, and thus by Theorem 6.1, points lying on the boundaries between the 14 pyramids can be perturbed into the interiors of the pyramids themselves. Thus, the maximum MST degree given such a strict cuboctahedral partition scheme is equal to the number of three-dimensional regions of the cuboctahedral partition, i.e. 14. □

While Theorem 5.3 gives an example that illustrates that the maximum MST degree in three-dimensional Manhattan space can be as large as 13, it is still open whether there exist examples in three-dimensional Manhattan space that force an MST degree of 14. Note that the best previously known bound for the maximum MST degree in three-dimensional Manhattan space was 26, as implied by a result from the theory of sphere packing [12] [44].

Our results regarding MST bounds also settle some open questions in complexity theory, since it is known that the problem of finding a degree-bounded MST is NP-complete, even when the degree bound is fixed at 2 (yielding the Traveling Salesman Problem), or at 3 as shown by Papadimitriou and Vazirani [36]. Corollary 6.2 implies that the degree-bounded MST problem in the Manhattan plane is solvable in polynomial time when the degree bound is fixed at 4 or more, since we have shown how to efficiently find an MST that meets these maximum degree constraints; this was previously an open problem. Similarly, Corollary 6.3 implies that the degree-bounded MST problem in three-dimensional Manhattan space is solvable in polynomial time when the degree bound is fixed at 14 or more (since we have shown how to efficiently find an ordinary MST that meets such maximum degree constraints). Monma and Suri [35] used a similar perturbation argument to prove that for any poinset in the Euclidean plane, there is an MST with maximum degree of 5.

7 A Parallel Implementation

Since a typical CAD environment consists of a network of workstations, exploiting the available large-grain parallelism provides a natural and effective means of improving the performance of CAD algorithms. We therefore propose a parallel implementation of the Iterated 1-Steiner method that achieves high parallel speedups. The I1S algorithm is highly parallelizable because each one of p processors can

compute independently and in parallel the MST savings of $O(\frac{n^2}{p})$ of the Steiner candidates. In our parallel implementation, all processors send their best candidate to a master processor, which selects the best of these candidates for inclusion into the pointset. This procedure is iterated until no improving candidates can be found (the B1S algorithm parallelizes similarly).

We used the Parallel Virtual Machine (PVM) system [5] [43] to control remote processors. PVM is a widely-available software package that allows a heterogeneous network of parallel and serial computers to be used as a single computational resource. The PVM system consists of two parts, a daemon process and a user library, providing mechanisms for initiating processes on other machines and for controlling synchronization and communication among processes. Using PVM as a framework for parallelism alleviates the need to hand-code the synchronization and communication protocols from scratch; for the specific details on how PVM manages the underlying system resources see [5] [43].

Each of the processors in our parallel implementation is a SUN workstation, communicating over an Ethernet. The master processor sends to the p available processors equal-sized subsets of the Hanan candidate set $H(P)$; each slave process then determines from among the candidates that it received the one with the best ΔMST, and returns the winner to the master process. Out of all the p candidates received, the master process then determines the candidate with the highest MST savings and includes it into the pointset. This parallel version of the Iterated 1-Steiner algorithm is shown in Figure 12.

To further increase the parallel execution speed, we implemented a load-balancing scheme in order to mitigate any significant variations that may exist between the speeds of different processors (due to a heavy CPU load, the underlying architecture type, swapping behavior, etc.) To this end, the wall-clock response time of each processor is tracked. If any individual processor is determined to be considerably slower than the rest, it is henceforth given smaller tasks to perform (i.e., it is sent less Hanan candidates, in an amount proportional to its observed speed). If a processor does not complete a task within reasonable time, it is sent an abort message, and its computation task is reassigned altogether to the fastest idle processor available. This prevents individual slow (or crashed) processors from seriously impeding the speed of the overall computation.

Algorithm Parallel Iterated 1-Steiner (PI1S)

Input: A set P of n points (assuming p processors available)
Output: A rectilinear Steiner tree over P

Master Process:

$S = \emptyset$
Do Forever
  For $i = 1$ to p **Do**
    Send the i^{th} processor $P \cup S$ and $\frac{|H(P)|}{p}$ different Hanan candidates from $H(P)$
  For $i = 1$ to p **Do**
    Receive from the i^{th} processor the candidate h_i with the largest ΔMST
  Let x be the h_i with the largest $\Delta MST(P \cup S, \{h_i\})$ for all $1 \leq i \leq p$
  If $\Delta MST(P \cup S, \{x\}) > 0$ **Then**
    $S = S \cup \{x\}$
    Remove from S points with degree ≤ 2 in $MST(P \cup S)$
  Else Output MST$(P \cup S)$ and **Stop**

Slave Process:

Receive from the master process $P \cup S$ and a set of Hanan candidates $h_1, \ldots, h_k$
Return the candidate h_i with the largest $\Delta MST(P \cup S, \{h_i\})$

Figure 12: The Parallel Iterated 1-Steiner algorithm. The master process sends each one of p slave processors $\frac{1}{p}$ of the Hanan candidates; a slave process then determines the candidate with the best ΔMST and returns it to the master process. Out of all the p candidates received, the master process then uses the candidate with the highest MST savings for inclusion into the pointset. To further increase the parallel execution speed, slower processes are sent smaller sets of candidates, in proportion to their observed speed (for clarity, this simple load-balancing scheme is not shown in the above template).

8 Experimental Results

We have implemented both serial and parallel enhanced versions of the B1S algorithm, using C in the SUN workstation environment. Our code is available upon request. The serial B1S heuristic has been benchmarked on up to 10000 instances of each net size. The instances were generated randomly by independently choosing the coordinates of each point from a uniform distribution in a 10000×10000 grid; such instances are statistically similar to the pin locations of actual VLSI nets and are the standard testbed for Steiner tree heuristics [27]. As is the convention in the Steiner approximation literature [27], we evaluate the performance of our method by comparing the cost of our solutions to the MST cost over the same inputs. Performance results are summarized in Table 1 and are illustrated in Figures 14(a) through 14(c). B1S yields Steiner trees with cost averaging almost 11% less than the MST cost.[3]

We timed the execution of the serial and parallel versions of B1S, using both the naive $O(n^4 \log n)$

[3]Recently, other Steiner heuristics with performance approaching that of I1S were proposed by Borah, Owens, and Irwin [8], Chao and Hsu [10] and by Lewis, Pong and VanCleave [33].

implementation [28] and our new $O(n^3)$ implementation, which incorporates the efficient, dynamic MST maintenance as described in Section 3. We observe that the number of rounds required by B1S is always very small, and may therefore be considered a constant; for example, when $n = 300$, the average number of rounds is only about 2.5, and we never observed B1S to require more than 5 rounds.

The parallel implementation uses 9 SUN 4/40 (IPC) workstations, with a SUN 4/75 (Sparc2) as the master processor. The improvements in running time are dramatic: for $n = 100$ our fast serial implementation is 247 times faster than the naive serial implementation (as measured by CPU time), and our parallel implementation running on 10 processors is 1163 times faster than the naive implementation (as measured by elapsed wall-clock time). It should be noted that for the serial implementation, the CPU time and the elapsed wall-clock time are essentially identical, since we report averages over hundreds of cases, and all of our algorithms are CPU-bound (i.e., swapping and I/O time is negligible). Detailed execution timings for various cardinalities are given in Table 2, and are illustrated in Figure 14(d). Note that most of the observed speedup over the naive implementation stems from the fast serial implementation using the dynamic MST update scheme (i.e., as described in Section 3). The parallel implementation only adds a speedup factor somewhat less than the number of 'processors/workstations available (10 SUNs in our case). The serial speedup grows as a function of the net size, since our new serial implementation is asymptotically faster than the naive serial implementation.

As noted by Ganley and Cohoon [15], most nets in actual VLSI designs have a small number of pins (i.e., less than 10). It is therefore of particular interest to observe the behavior of our algorithms on small nets. Even for small pointsets, our new implementations are considerably faster than the previous, naive ones: for example, for $n = 5$ the new serial B1S is on average twice as fast as the naive implementation, while for $n = 10$ it is 7 times as fast. Our parallel speedup increases with problem size, reaching about 7.2 for $n = 250$ running on 10 processors. This enables us to examine the asymptotic behavior of B1S for much larger pointsets than was previously possible: the output of Batched 1-Steiner (B1S) for a random pointset of size 300 is shown in Figure 13.

We have also implemented the EIkS and the EBkS algorithms, and the following algorithms were tested side-by-side on the same inputs:

- **B1S** - The Batched 1-Steiner method, with the code further streamlined for speed;

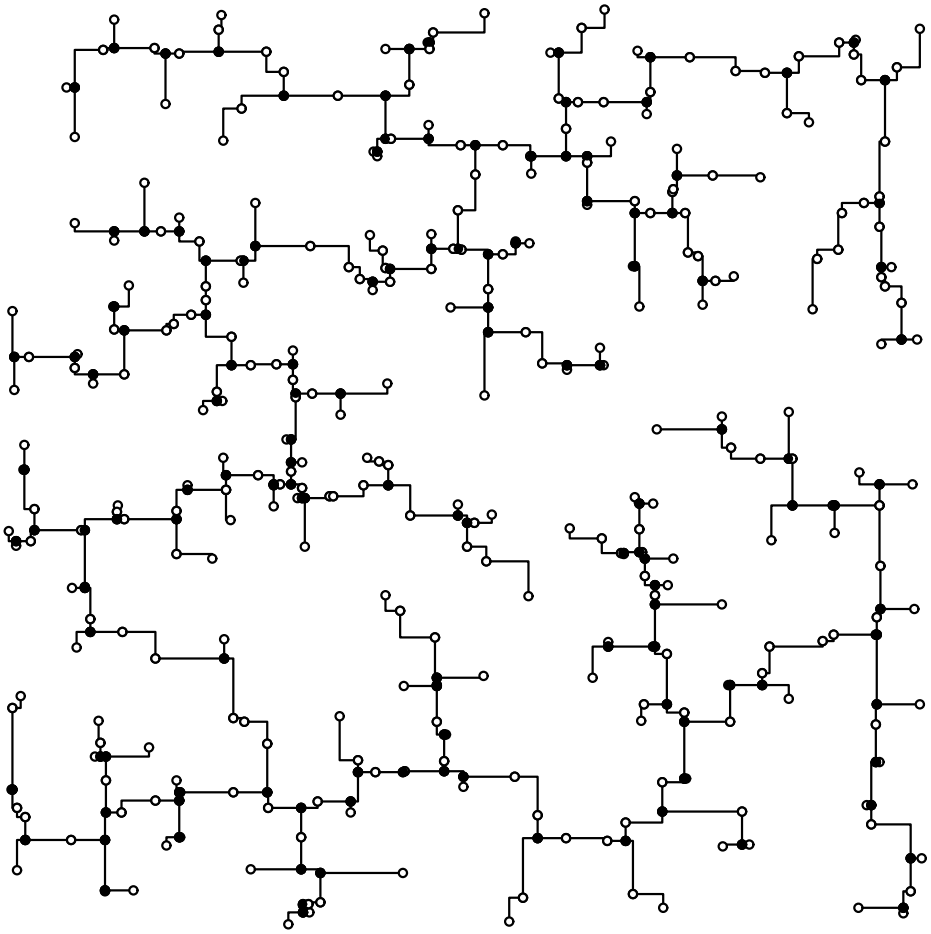

Figure 13: An example of the output of Batched 1-Steiner (B1S) on a random 300-point set (hollow dots). The Steiner points produced by B1S are denoted by dark solid dots.

- **MB1S** - The "modified" B1S variant that uses only winning candidates from previous rounds;
- **EB1S** - The enhanced version of B1S;
- **I2S** - The Iterated 2-Steiner algorithm;
- **EI2S** - The enhanced version of I2S;
- **META(B1S,MB1S)** - The *metaheuristic* over heuristics B1S and MB1S, i.e., the best solution found by either of these heuristics;
- **META(EB1S,I2S,EI2S)** - The metaheuristic over EB1S, I2S, and EI2S;
- **META(B1S,MB1S,EB1S,I2S,EI2S)** - The metaheuristic over B1S, MB1S, EB1S, I2S, and

EI2S;

- **OPT** - The optimal Steiner tree algorithm of Salowe and Warme [40].

Recall from the discussion at the end of Section 4 that MB1S is a more efficient version of B1S, since it only examines a fraction of the Hanan candidates in a typical round (i.e., only the ones with positive MST savings in the previous round). OPT is the fastest known *optimal* rectilinear Steiner tree algorithm [40]. All variants have been benchmarked on up to 10000 random instances of each net size. Figure 15(a) shows the performance comparison of MB1S, EI2S, and OPT, while Table 3 gives more detailed performance data. We observe that the average performance of our methods is extremely close to optimal: for $n = 8$, EI2S is on average only about 0.11% away from optimal, and solutions are optimal in about 90% of the cases. Even for $n = 30$, MB1S is only about 0.30% away from optimal, and yields optimal solutions in about one quarter of all cases. Table 4 tracks the percentage of cases where the various heuristics find the optimal solution, and this data is also depicted pictorially in Figure 15(b).

Table 5 gives the average CPU times, in seconds, for each heuristic and net size. Our most time-efficient algorithm is MB1S, requiring an average of 0.009 CPU seconds per 8-pin net, and an average of 0.375 seconds per 30-pin net. Using EIkS (or EBkS) with values of k greater than 2 improves the performance, but slows down the algorithm; it is easy to see that for arbitrary k, EIkS (EBkS) always yields optimal solutions for $\leq k+2$ pins, but has time complexity greater by a factor of $n^{2(k-1)}$ than that of EB1S. This allows a smooth tradeoff between performance and efficiency. However, the performance of the EBkS algorithm with $k = 2$ is already so close to optimal, that in most applications increasing k further is not likely to justify the incurred time penalty.

In three dimensions, we observed that the average performance of EB1S approaches 15% improvement over MST cost, and the performance increases with the number of planes L. It is not surprising that the average savings over MST cost in three dimensions is higher than it is in two dimensions, since the worst-case performance ratio in three dimensions is higher also (i.e., $\frac{5}{3}$ for three dimensions vs. $\frac{3}{2}$ for two dimensions). Figure 15(c) shows the performance of our method in three dimensions for various values of the number of parallel planes L, including the unrestricted three-dimensional case, corresponding to the limit when L approaches ∞. Table 7 gives statistics for three dimensions on the number of Steiner points induced by B1S (see Figure 15(d)), as well as on the number of rounds that occur in B1S before termination. As is the case in two dimensions, the number of rounds for B1S in

three dimensions is on average very small. Table 6 gives more detailed performance data. In all cases, the L parallel planes were uniformly spaced in the unit cube (i.e., they were separated by $\frac{G}{L}$ units apart, where $G = 10000$ is the gridsize). Unfortunately, the OPT algorithm of Salowe and Warme [40] does not easily generalize to higher dimensions; thus, we were not able to compare our three-dimensional version of EB1S to optimal.

9 Conclusion

We have proposed enhanced serial and parallel implementations of the Batched 1-Steiner heuristic (B1S), achieving speeds of up to three orders of magnitude faster than previous implementations. Moreover, the speedup increases with the number of points. This has enabled the testing of B1S on several hundred points for the first time, and we observed that for such large pointsets B1S consistently improves 11% over MST cost.

Next, we enhanced B1S by using a near-greedy approach with random tie-breaking. Our method enjoy the same asymptotic time complexity as B1S, yet offer improved average performance. We also allow performance increase at the expense of running time, creating a smooth tradeoff between solution quality and computational efficiency. Extensive simulations indicate that for typical nets, the average performance of our methods is less than 0.25% away from optimal, and our solutions are actually optimal for up to 90% of uniformly distributed nets of typical sizes.

We generalized B1S and its variants to three dimensions, as well as to the case where all the pins lie on L parallel planes, which arises in, e.g., three-dimensional VLSI and multi-layer routing. Our methods are highly parallelizable and generalize to arbitrary weighted graphs; thus, they are suitable to support a multi-layer global router, where obstruction and congestion considerations affect routing. Since Steiner tree construction is a computationally expensive component of global routing, our techniques suggest the feasibility of a "Steiner engine" for efficiently computing near-optimal Steiner trees.

We reduced the running time of our algorithms through a dynamic MST-maintenance scheme, and we proved that under the Manhattan metric: (1) in two dimensions we can always find an MST with maximum degree 4; and (2) in three dimensions we can always find an MST with maximum degree 14. The best previously known bounds for two and three dimensions were 6 and 26, respectively. These results were used to decrease the running times of our algorithms, and moreover they have independent

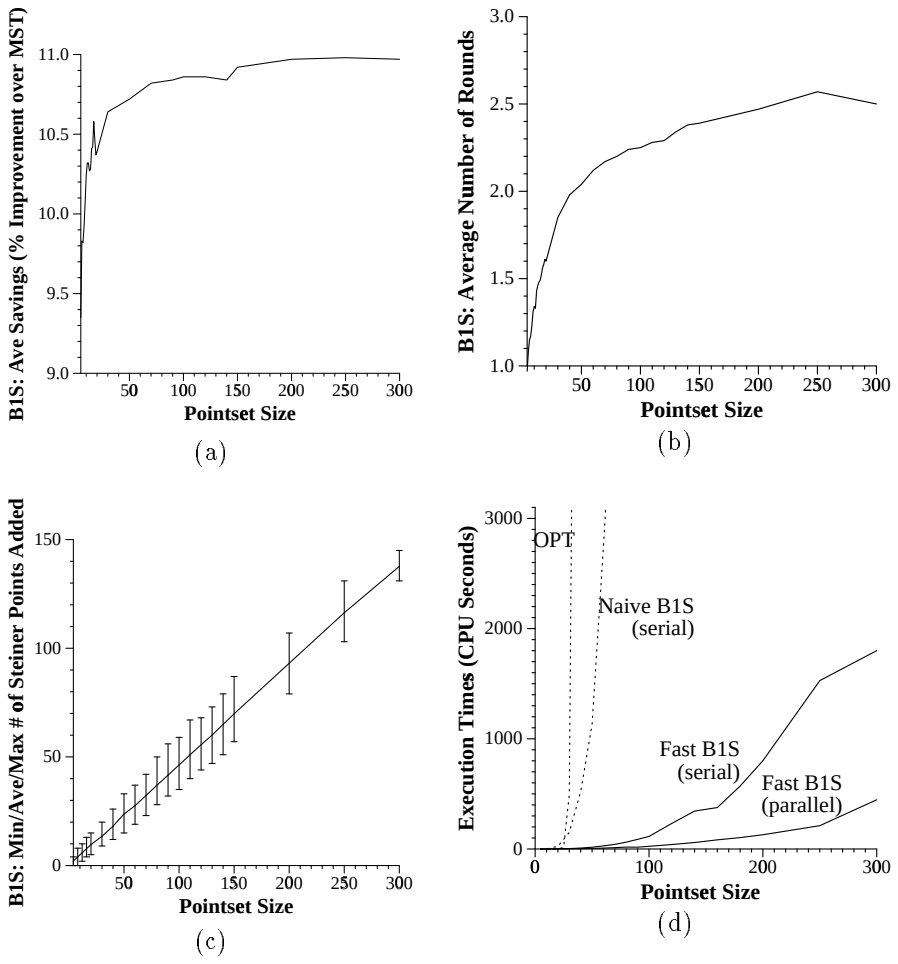

Figure 14: (a) Average performance of B1S, shown as percent improvement over MST cost; (b) Average number of rounds for B1S; (c) Average number of Steiner points induced by B1S (vertical bars indicate the range of the minimum and maximum number of Steiner points added); (d) Average execution times (in elapsed wall-clock seconds) for the serial and parallel B1S, using both the naive MST implementation and our new incremental MST maintenance scheme, over a wide range of net sizes; note that for the serial versions, the elapsed wall-clock times are essentially the same as the CPU time, since the benchmarks represent average running times over hundreds of cases (the algorithms are CPU-bound, so that swapping and I/O have negligible effect on the running time).

theoretical significance; for example, our bounds on the maximum MST degree were used to settle an open problem in complexity theory [39].

Remaining open research questions include finding an MRST heuristic with performance consistently higher than EB1S (or MB1S), but with a significantly better running time (note that it would

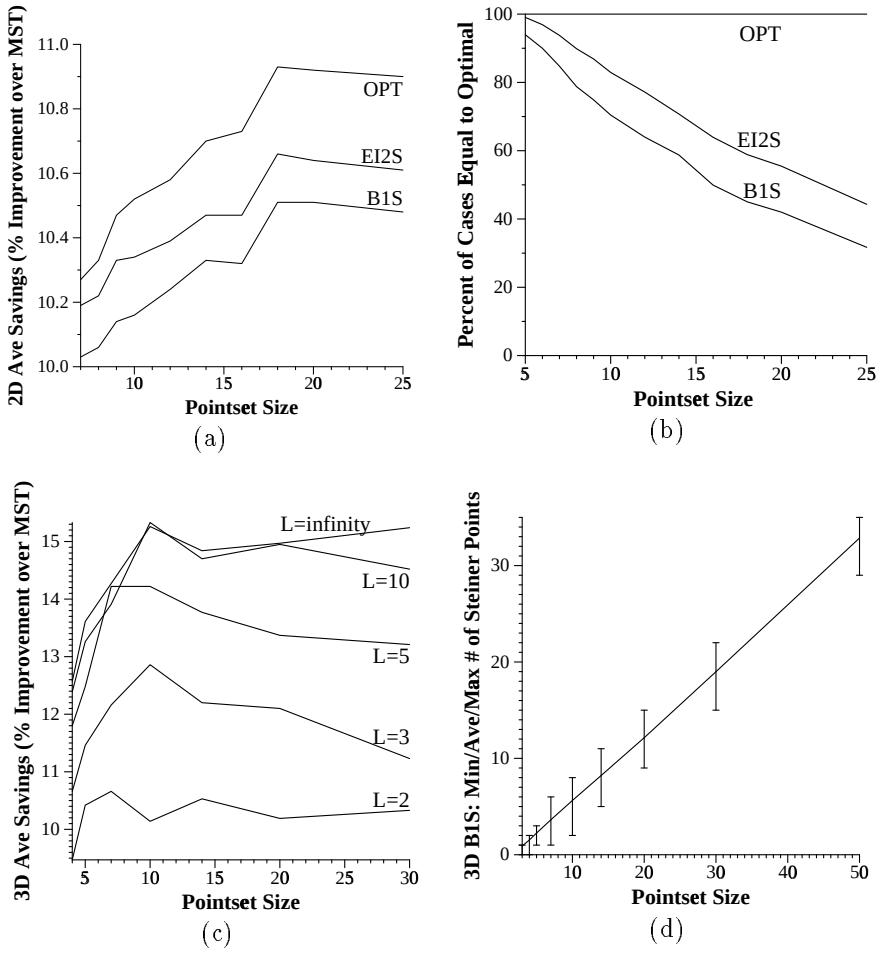

Figure 15: (a) Average performance in two dimensions of EI2S, B1S, and OPT; note that EI2S is only 0.25% (or less) away from optimal; (b) Percentage of all cases when the heuristics find the optimal solution (note that EI2S yields optimal solutions a large percentage of the time); (c) Average performance of EB1S in three dimensions for various values of L = number of parallel planes; (d) Average number of Steiner points added by B1S in three dimensions for $L = \infty$.

not suffice to just find a heuristic with better performance but which runs more slowly, since increasing the parameter k in EBkS will achieve exactly that). On the theoretical front, it would be interesting to determine whether the maximum MST degree in three dimensional Manhattan space is 13 or 14 and extend such results to higher dimensions and to different L_p norms.

10 Acknowledgments

We would like to thank Mr. Tim Barrera for his help with the coding, and the referees for their helpful comments.

References

[1] M. J. Alexander and G. Robins, *An Architecture-Independent Unified Approach to FPGA Routing*, Tech. Rep. CS-93-51, Department of Computer Science, University of Virginia, October 1993.

[2] M. J. Alexander and G. Robins, *A New Approach to FPGA Routing Based on Multi-Weighted Graphs*, in Proc. ACM/SIGDA International Workshop on Field-Programmable Gate Arrays, Berkeley, CA, February 1994.

[3] 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 Proc. Great Lakes Symp. VLSI, Kalamazoo, MI, March 1993, pp. 90–94.

[4] T. Barrera, J. Griffith, G. Robins, and T. Zhang, *Narrowing the Gap: Near-Optimal Steiner Trees in Polynomial Time*, in Proc. IEEE Intl. ASIC Conf., Rochester, NY, September 1993, pp. 87–90.

[5] A. Beguelin, J. J. Dongarra, G. A. Geist, R. Manchek, and V. S. Sunderam, *A User's Guide to PVM: Parallel Virtual Machine*, Tech. Rep. ORNL/TR-11826, Oak Ridge National Laboratory, 1991.

[6] P. Berman, U. Foessmeier, M. Karpinski, M. Kaufmann, and A. Z. Zelikovsky, *Approaching the 5/4 - Approximation for Rectilinear Steiner Trees*, Tech. Rep. WSI-94-06, Wilhelm Schickard-Institut fur Informatik, 1994.

[7] P. Berman and V. Ramaiyer, *Improved Approximations for the Steiner Tree Problem*, in Proc. ACM/SIAM Symp. Discrete Algorithms, San Francisco, CA, January 1992, pp. 325–334.

[8] M. Borah, R. M. Owens, and M. J. Irwin, *An Edge-Based Heuristic for Rectilinear Steiner Trees*, Tech. Rep. CS-93-003, Department of Computer Science, Pennsylvania State University, 1993.

[9] D. Braun, J. Burns, S. Devadas, H. K. Ma, K. M. F. Romeo, and A. Sangiovanni-Vincentelli, *Chameleon: A New Multi-Layer Channel Router*, in Proc. ACM/IEEE Design Automation Conf., 1986, pp. 495–502.

[10] T. H. Chao and Y. C. Hsu, *Rectilinear Steiner Tree Construction by Local and Global Refinement*, IEEE Trans. Computer-Aided Design, (1994), pp. 303–309.

[11] T. H. Cormen, C. E. Leiserson, and R. Rivest, *Introduction to Algorithms*, MIT Press, 1990.

[12] J. T. Croft, K. J. Falconer, and R. K. Guy, *Unsolved Problems in Geometry*, Springer-Verlag, New York, 1991.

[13] U. Foessmeier, M. Kaufmann, and A. Z. Zelikovsky, *Fast Approximation Algorithms for the Rectilinear Steiner Tree Problem*, Tech. Rep. WSI-93-14, Wilhelm Schickard-Institut fur Informatik, 1993.

[14] G. N. Fredrickson, *Data Structures for On-Line Updating of Minimum Spanning Trees*, SIAM J. Comput., 14 (1985), pp. 781–798.

[15] J. L. Ganley and J. P. Cohoon, *Routing a Multi-Terminal Critical Net: Steiner Tree Construction in the Presence of Obstacles*, in Proc. IEEE Intl. Symp. Circuits and Systems, London, England, May 1994.

[16] M. Garey and D. S. Johnson, *The Rectilinear Steiner Problem is NP-Complete*, SIAM J. Applied Math., 32 (1977), pp. 826–834.

[17] G. Georgakopoulos and C. H. Papadimitriou, *The 1-Steiner Tree Problem*, J. Algorithms, 8 (1987), pp. 122–130.

[18] E. N. Gilbert and H. O. Pollak, *Steiner Minimal Trees*, SIAM J. Applied Math., 16 (1968), pp. 1–29.

[19] R. L. Graham and F. K. Hwang, *A Remark on Steiner Minimal Trees*, Bull. Inst. Math. Acad. Sinica, 4 (1976), pp. 177–182.

[20] A. Hanafusa, Y. Yamashita, and M. Yasuda, *Three-Dimensional Routing for Multilayer Ceramic Printed Circuit Boards*, in Proc. IEEE Intl. Conf. Computer-Aided Design, Santa Clara, CA, November 1990, pp. 386–389.

[21] M. Hanan, *On Steiner's Problem With Rectilinear Distance*, SIAM J. Applied Math., 14 (1966), pp. 255–265.

[22] A. C. Harter, *Three-Dimensional Integrated Circuit Layout*, Cambridge University Press, New York, 1991.

[23] N. Hasan, G. Vijayan, and C. K. Wong, *A Neighborhood Improvement Algorithm for Rectilinear Steiner Trees*, in Proc. IEEE Intl. Symp. Circuits and Systems, New Orleans, LA, 1990.

[24] J.-M. Ho, G. Vijayan, and C. K. Wong, *New Algorithms for the Rectilinear Steiner Tree Problem*, IEEE Trans. Computer-Aided Design, 9 (1990), pp. 185–193.

[25] F. K. Hwang, *On Steiner Minimal Trees with Rectilinear Distance*, SIAM J. Applied Math., 30 (1976), pp. 104–114.

[26] F. K. Hwang, *An O(n log n) Algorithm for Rectilinear Minimal Spanning Trees*, J. ACM, 26 (1979), pp. 177–182.

[27] F. K. Hwang, D. S. Richards, and P. Winter, *The Steiner Tree Problem*, North-Holland, 1992.

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

[29] A. B. Kahng and G. Robins, *On Performance Bounds for a Class of Rectilinear Steiner Tree Heuristics in Arbitrary Dimension*, IEEE Trans. Computer-Aided Design, 11 (1992), pp. 1462–1465.

[30] A. B. Kahng and G. Robins, *On Optimal Interconnections for VLSI Layout*, Kluwer Academic Publishers (to appear), Boston, MA, 1994.

[31] J. H. Lee, N. K. Bose, and F. K. Hwang, *Use of Steiner's Problem in Sub-Optimal Routing in Rectilinear Metric*, IEEE Trans. Circuits and Systems, 23 (1976), pp. 470–476.

[32] K. W. Lee and C. Sechen, *A New Global Router for Row-Based Layout*, in Proc. IEEE Intl. Conf. Computer-Aided Design, Santa Clara, CA, November 1990, pp. 180–183.

[33] F. D. Lewis, W. C. Pong, and N. VanCleave, *Local Improvement in Steiner Trees*, in Proc. Great Lakes Symp. VLSI, Kalamazoo, MI, March 1993, pp. 105–106.

[34] A. L. Loeb, *Space Structures: Their Harmony and Counterpoint*, Birkhauser, New York, 1991.

[35] C. Monma and S. Suri, *Transitions in geometric minimum spanning trees*, Discrete & Computational Geometry, 8 (1992), pp. 265–293.

[36] C. H. Papadimitriou and U. V. Vazirani, *On Two Geometric Problems Relating to the Traveling Salesman Problem*, J. Algorithms, 5 (1984), pp. 231–246.

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

[38] G. Robins, *On Optimal Interconnections*, Ph.D. Dissertation, CSD-TR-920024, Department of Computer Science, UCLA, 1992.

[39] G. Robins and J. S. Salowe, *On the Maximum Degree of Minimum Spanning Trees*, in Proc. ACM Symp. Computational Geometry, Stony Brook, NY, June 1994.

[40] J. S. Salowe and D. M. Warme, *An Exact Rectilinear Steiner Tree Algorithm*, in Proc. IEEE Intl. Conf. Computer Design, Cambridge, MA, October 1993, pp. 472–475.

[41] J. M. Smith and J. S. Liebman, *Steiner Trees, Steiner Circuits and the Interference Problem in Building Design*, Engineering Optimization, 4 (1979), pp. 15–36.

[42] T. L. Snyder, *On the Exact Location of Steiner Points in General Dimension*, SIAM J. Comput., 21 (1992), pp. 163–180.

[43] V. S. Sunderam, *PVM: A Framework for Parallel Distributed Computing*, Concurrency: Practice and Experience, 2 (1990), pp. 315–339.

[44] G. F. Tóth, *New Results in the Theory of Packing and Covering*, Convexity and its Applications, 1983.

[45] A. C. C. Yao, *On Constructing Minimum Spanning Trees in k-Dimensional Spaces and Related Problems*, SIAM J. Comput., 11 (1982), pp. 721–736.

[46] A. Z. Zelikovsky, *An 11/6 Approximation Algorithm for the Network Steiner Problem*, Algorithmica, 9 (1993), pp. 463–470.

[47] A. Z. Zelikovsky, *A Faster Approximation Algorithm for the Steiner Tree Problem in Graphs*, Information Processing Letters, 46 (1993), pp. 79–83.

	Batched 1-Steiner (B1S) Performance Results												
	Performance			Number of SPs			Number of rounds			Ave #SPs per round			
n	Min	Ave	Max	Min	Ave	Max	Min	Ave	Max	1	2	3	4
4	0.00	8.83	27.19	0	2.10	4	0	0.97	2	1.14	0.37	0.00	0.00
5	0.00	9.35	23.59	0	2.59	5	0	1.08	3	1.51	0.46	0.06	0.00
6	0.00	9.83	25.99	0	3.09	6	0	1.15	3	1.91	1.42	0.03	0.00
7	0.41	9.82	23.44	2	3.52	7	1	1.17	3	2.32	0.59	0.02	0.00
8	0.00	9.92	20.93	0	3.99	8	0	1.23	3	2.73	0.96	0.17	0.00
9	1.04	10.07	20.62	2	4.48	7	1	1.31	3	3.12	1.16	0.19	0.00
10	1.26	10.24	20.03	3	4.95	8	1	1.34	4	3.56	1.11	0.10	0.01
11	2.01	10.32	20.81	3	5.42	10	1	1.33	3	4.01	1.01	0.03	0.00
12	1.85	10.32	18.83	2	5.88	10	1	1.43	4	4.31	1.28	0.09	0.00
13	2.06	10.27	19.74	3	6.39	11	1	1.46	4	4.80	1.23	0.09	0.04
14	3.01	10.28	19.56	3	6.82	11	1	1.48	4	5.18	1.25	0.14	0.07
15	2.78	10.41	18.10	4	7.27	12	1	1.49	4	5.59	1.33	0.12	0.00
16	2.47	10.42	18.45	4	7.76	13	1	1.52	4	6.01	1.33	0.13	0.00
17	3.00	10.58	17.94	5	8.33	14	1	1.56	3	6.50	1.38	0.27	0.00
18	3.62	10.46	18.19	5	8.70	14	1	1.58	4	6.85	1.52	0.26	0.01
19	3.71	10.37	18.05	5	9.14	14	1	1.61	5	7.21	1.58	0.10	0.01
20	4.35	10.39	17.90	5	9.63	15	1	1.60	4	7.68	1.58	1.00	0.00
30	6.49	10.64	14.73	9	13.47	20	1	1.85	4	11.61	1.75	0.21	0.01
40	6.28	10.68	14.55	12	18.10	26	1	1.98	5	15.95	2.47	0.23	0.00
50	7.25	10.72	15.28	15	23.70	33	1	2.04	4	19.97	2.78	0.56	0.00
60	8.67	10.77	14.29	19	27.53	37	1	2.12	5	23.88	3.81	0.58	0.00
70	7.80	10.82	13.55	23	32.26	42	1	2.17	4	28.02	3.52	0.62	0.00
80	7.34	10.83	12.88	28	36.99	50	1	2.20	4	32.07	4.66	0.64	0.01
90	7.99	10.84	13.52	32	41.61	56	1	2.24	4	36.18	5.14	1.17	0.04
100	8.78	10.86	13.60	35	46.30	59	1	2.25	4	40.21	5.55	1.31	0.01
110	8.04	10.86	12.86	40	50.98	67	1	2.28	4	44.96	5.77	0.53	0.08
120	7.83	10.86	12.62	44	55.59	68	1	2.29	5	48.64	7.15	1.02	0.00
130	8.25	10.85	12.63	47	60.03	73	1	2.34	4	52.87	7.22	0.82	0.00
140	8.45	10.84	12.57	51	64.97	79	1	2.38	5	56.94	8.12	1.07	0.22
150	9.07	10.92	12.63	57	69.85	87	1	2.39	4	59.92	8.28	1.13	0.00
200	9.39	10.97	12.32	79	93.10	107	2	2.47	4	80.04	11.68	1.48	0.50
250	10.26	10.98	11.68	103	116.32	131	2	2.57	4	100.00	14.53	0.82	0.00
300	9.76	10.97	12.18	131	137.67	145	2	2.50	4	121.85	16.85	1.33	0.00

Table 1: Batched 1-Steiner (B1S) statistics: the performance figures denote percent improvement over MST cost. Also given are statistics regarding the number of Steiner points produced, the number of rounds, and the number of Steiner points produced in each round.

	Batched 1-Steiner (B1S) Average Execution Speeds (CPU Seconds)								
	Serial			Parallel			Speedup		overall
	old	new	ratio	old	new	ratio	old	new	gain
n	A	B	A/B	C	D	C/D	A/C	B/D	A/D
4	0.01	0.01	1.00	0.16	0.15	1.07	0.06	0.07	0.07
5	0.04	0.02	2.00	0.20	0.19	1.05	0.20	0.11	0.27
6	0.10	0.04	2.50	0.21	0.20	1.05	0.48	0.20	0.53
7	0.20	0.06	3.33	0.24	0.22	1.09	0.83	0.27	0.91
8	0.37	0.09	4.11	0.27	0.24	1.13	1.37	0.38	1.54
9	0.63	0.12	5.25	0.39	0.27	1.44	1.62	0.44	2.33
10	1.02	0.16	6.38	0.51	0.29	1.76	2.00	0.55	3.52
12	2.28	0.26	8.77	1.06	0.35	3.03	2.15	0.74	6.51
14	4.69	0.44	10.66	1.57	0.41	3.83	2.99	1.07	11.44
16	8.40	0.61	13.77	2.07	0.47	4.40	4.06	1.30	17.87
18	14.67	0.81	18.11	4.14	0.57	7.26	3.54	1.42	25.73
20	24.07	1.09	22.08	5.57	0.61	9.13	4.32	1.79	39.46
30	151	3.80	39.74	40.2	1.79	22.46	3.75	2.12	84.36
40	522	8.59	60.77	126	3.23	39.01	4.14	2.66	161.61
50	1130	16.1	70.19	200	5.03	39.76	5.65	3.20	224.65
60	2745	28.1	97.69	753	8.03	93.77	3.65	3.50	341.84
70	5520	41.9	131.74	1002	12.1	82.81	5.51	3.46	456.20
80	10350	62.2	166.40	2084	17.1	121.87	4.97	3.64	605.26
90	15450	87.5	176.57	2582	16.5	156.48	4.97	5.30	936.36
100	28140	114	246.84	4748	24.2	196.20	5.93	4.71	1162.81
120		232			40.5			5.73	
140		344			58.6			5.87	
160		376			79.67			4.72	
180		571			103			5.54	
200		801			129			6.21	
250		1528			212			7.21	
300		1800			447			4.03	

Table 2: Execution times for Batched 1-Steiner (B1S): the serial execution times are given in CPU seconds, while the parallel execution times are elapsed wall-clock times. Both the serial and parallel versions were tested with the old naive implementation, as well as the new faster implementation. The overall gain is the ratio of the old serial time to the new parallel time. The parallel implementation uses 9 SUN 4/40 (IPC) workstations, with a SUN 4/75 (Sparc2) as the master processor.

Average Performance Results in Two Dimensions (% Improvement over MST)										
n	# nets	(1) B1S	(2) MB1S	(3) EB1S	(4) I2S	(5) EI2S	Meta (1-2)	Meta (3-5)	Meta (1-5)	OPT
4	10000	8.54	8.54	8.54	8.54	8.54	8.54	8.54	8.54	8.54
5	10000	9.34	9.34	9.39	9.44	9.45	9.37	9.46	9.46	9.47
6	10000	9.79	9.78	9.85	9.90	9.92	9.82	9.93	9.93	9.97
7	5000	10.04	10.03	10.12	10.15	10.19	10.08	10.21	10.21	10.27
8	5000	10.06	10.06	10.15	10.17	10.22	10.11	10.24	10.24	10.33
9	5000	10.16	10.14	10.25	10.27	10.33	10.20	10.34	10.34	10.47
10	5000	10.19	10.16	10.27	10.28	10.34	10.22	10.36	10.36	10.52
12	5000	10.24	10.24	10.34	10.33	10.39	10.29	10.41	10.41	10.58
14	5000	10.33	10.33	10.42	10.40	10.47	10.37	10.49	10.49	10.70
16	5000	10.33	10.32	10.42	10.40	10.47	10.37	10.49	10.49	10.73
18	4000	10.51	10.51	10.61	10.58	10.66	10.56	10.67	10.67	10.93
20	3000	10.51	10.51	10.61	10.56	10.64	10.55	10.66	10.66	10.92
25	2000	10.47	10.48	10.57	10.53	10.61	10.52	10.62	10.62	10.90
30	100	10.45	10.59	10.76	10.55	10.62	10.51	10.63	10.63	10.89
50	500	10.89	10.89	10.99	10.93	11.03	10.93	11.04	11.05	
70	100	10.73	10.77	10.86	10.76	10.88	10.80	10.91	10.91	

Table 3: Performance statistics: the figures denote average percent improvement over MST cost.

Percent of the Cases When Solution Was Optimal										
n	# nets	(1) B1S	(2) MB1S	(3) EB1S	(4) I2S	(5) EI2S	Meta (1-2)	Meta (3-5)	Meta (1-5)	OPT
4	10000	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
5	10000	94.29	93.96	96.53	98.42	99.04	95.30	99.26	99.27	100.00
6	10000	90.34	90.00	93.68	94.92	96.91	91.98	97.46	97.46	100.00
7	5000	85.20	84.75	89.96	90.72	93.77	87.45	94.75	94.75	100.00
8	5000	79.62	78.77	85.26	84.73	89.87	82.13	90.94	90.95	100.00
9	5000	75.80	74.88	82.23	81.70	86.86	78.64	87.92	87.92	100.00
10	5000	72.03	70.44	79.12	77.01	82.92	75.09	84.49	84.49	100.00
12	5000	65.15	63.98	73.01	70.41	77.17	68.49	78.72	78.72	100.00
14	5000	57.86	58.73	66.69	62.84	70.73	62.31	72.33	72.33	100.00
16	5000	50.36	49.92	60.17	54.84	63.92	54.99	65.73	65.73	100.00
18	4000	44.90	45.02	55.65	49.33	58.85	49.45	61.02	61.02	100.00
20	3000	42.87	42.00	53.20	45.70	55.47	47.00	58.23	58.23	100.00
25	2000	31.25	31.70	41.45	34.15	44.30	35.60	45.90	45.90	100.00
30	100	27.00	26.00	41.00	32.00	45.00	31.00	47.00	47.00	100.00

Table 4: Optimality Percentages: figures denote the percent of the cases where the various heuristics found the optimal solution.

Average CPU Time Per Net (CPU Seconds)									
n	(1) B1S	(2) MB1S	(3) EB1S	(4) I2S	(5) EI2S	Meta (1-2)	Meta (3-5)	Meta (1-5)	OPT
4	0.002	0.001	0.054	0.001	0.072	0.003	0.127	0.130	0.006
5	0.004	0.002	0.107	0.003	0.068	0.006	0.178	0.184	0.010
6	0.006	0.004	0.184	0.006	0.128	0.010	0.318	0.328	0.018
7	0.009	0.006	0.287	0.012	0.231	0.015	0.530	0.545	0.031
8	0.013	0.009	0.432	0.019	0.384	0.022	0.835	0.857	0.046
9	0.019	0.012	0.571	0.030	0.634	0.031	1.235	1.266	0.066
10	0.025	0.016	0.806	0.046	0.951	0.041	1.803	1.844	0.090
12	0.043	0.026	1.324	0.096	1.967	0.069	3.387	3.456	0.158
14	0.066	0.041	2.127	0.180	3.623	0.107	5.930	6.037	0.268
16	0.096	0.059	3.288	0.317	6.372	0.155	9.977	10.13	0.405
18	0.134	0.081	4.216	0.540	9.700	0.215	14.46	14.68	0.774
20	0.181	0.115	6.440	0.833	15.87	0.296	23.14	23.44	1.618
25	0.341	0.220	11.94	2.050	48.09	0.561	62.08	62.64	13.88
30	0.569	0.375	18.93	4.184	86.10	0.944	109.2	110.1	495.8
50	2.732	1.694	79.78	35.91	764.6	4.426	880.3	884.7	
70	6.664	4.236	255.4	150.3	5668	10.90	6074	6085	

Table 5: Average execution times, in CPU seconds, for each of the heuristics.

Average Performance of EB1S in Three Dimensions (% Improvement over MST)									
n	L=2	L=3	L=4	L=5	L=7	L=10	L=14	L=20	L=∞
3	8.01	9.51	9.72	10.11	10.24	10.65	10.26	10.63	10.66
4	9.47	10.66	11.14	11.80	12.33	12.39	12.17	12.01	12.57
5	10.42	11.46	12.46	12.48	13.12	13.26	13.48	13.45	13.61
7	10.66	12.16	13.07	13.24	13.95	13.91	14.19	14.20	14.23
10	10.14	12.86	13.64	14.22	13.54	15.33	14.26	14.34	15.26
14	10.53	12.20	13.25	13.77	14.38	14.70	14.27	14.77	14.84
20	10.19	12.10	13.48	13.37	14.25	14.95	14.74	14.84	14.97
30	10.33	11.23	11.92	13.21	13.82	14.52	15.02	15.04	15.24
50	10.28	10.20	11.31	13.01	13.49	14.49	15.14	15.08	15.16

Table 6: Average percent improvement of EB1S over MST cost in three dimensions for L planes equally spaced in the unit cube. The unrestricted three-dimensional case occurs when $L = \infty$ (last column).

#SPs and #rounds for B1S in 3D						
	#SPs			#rounds		
n	min	ave	max	min	ave	max
3	0	0.90	1	1	1.90	2
4	0	1.53	2	1	2.18	3
5	1	2.25	3	2	2.33	4
7	1	3.63	6	2	2.60	5
10	2	5.63	8	2	2.92	7
14	5	8.22	11	2	3.18	6
20	9	12.14	15	2	3.26	5
30	15	18.99	22	3	3.53	6
50	29	32.86	35	3	4.14	6

Table 7: Statistics regarding the number of Steiner points induced by B1S in the unrestricted three-dimensional case (i.e., $L = \infty$). Also given are the number of rounds executed by B1S. Shown are the minimum, average, and maximum values.

Jeff Griffith received his Bachelor's degree in Mathematics in 1992 from the University of Virginia in Charlottesville. He is presently working on his Ph.D. in computer science at the University of Tennessee in Knoxville. His areas of interest include physical design for VLSI circuits, as well as scientific computing.

Gabriel Robins received the Ph.D. degree in 1992 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 at the University of Virginia in Charlottesville, where he recently won the NSF Young Investigator Award. His primary areas of research are VLSI CAD and geometric algorithms, with recent work focusing on performance-driven routing, Steiner tree heuristics, computational geometry, motion planning, pattern recognition, and computational biology. Dr. Robins is the author of a Distinguished Paper at the 1990 IEEE International Conference on Computer-Aided Design, and his Ph.D. dissertation was nominated for the ACM Distinguished Dissertation Award. He is a member of ACM, IEEE, MAA, and SIAM.

Jeffrey S. Salowe received the B.A. Degree in mathematics from the University of Virginia, Charlottesville, VA, in 1983, and the M.S. and Ph.D. degrees in computer science from Rutgers University, New Brunswick, NJ, in 1985 and 1987, respectively. Currently, he is an Assistant Professor of Computer Science at the University of Virginia, Charlottesville, VA. His research interests include the analysis of algorithms, computational geometry, data structures, and graph algorithms.

Tongtong Zhang studied at the University of Beijing, China for three years and received her Bachelor's degree in 1992 from Bridgewater College, Virginia. She is currently working on her Ph.D. in computer science at the University of Virginia in Charlottesville. Her areas of interest include physical design for VLSI circuits and computational biology.

Footnotes

Professor Gabriel Robins, Professor Jefferey Salowe, and Ms. Tongtong Zhang are with the Department of Computer Science, University of Virginia, Charlottesville, VA 22903-2442.

Mr. Jeff Griffith is with the Department of Computer Science, University of Tennessee, Knoxville, TN 37996-1301.

Professor Gabriel Robins, Professor Jeffrey Salowe, and Ms. Tongtong Zhang are with the Department of Computer Science, University of Virginia, Charlottesville, VA 22903-2442. Mr. Jeff Griffith is with the Department of Computer Science, University of Tennessee, Knoxville, TN 37996-1301. All correspondence and code requests should be addressed to the primary author: Professor Gabriel Robins, robins@cs.virginia.edu, phone: (804) 982-2207, FAX: (804) 982-2214. Professor Robins was partially supported by NSF Young Investigator Award MIP-9457412. Professor Salowe was partially supported by NSF grants MIP-9107717 and CCR-9224789.

Footnote 1: Recently, Berman and Ramaiyer [7] and Foessmeier, Kaufmann and Zelikovsky [6] [13] have extended the fundamental work of Zelikovsky [46] [47] to yield a method similar to I1S (specifically, to the "batched" I1S method described below) with performance ratio bounded by $\frac{11}{8}$; this work settles in the affirmative the longstanding open question of whether there exists a polynomial-time rectilinear Steiner tree heuristic with performance ratio strictly smaller than $\frac{3}{2}$ [25]. At the time of this writing, Berman, Foessmeier, Karpinski, Kaufmann and Zelikovsky [13] further improved the performance bound of their polynomial-time rectilinear Steiner heuristic to $\frac{61}{48} = 1.271$.

Footnote 2: Robins and Salowe [39] investigate the maximum MST degree for higher dimensions and other L_p norms, and relate the maximum MST degree to the so-called "Hadwiger" numbers.

Footnote 3: Recently, other Steiner heuristics with performance approaching that of I1S were proposed by Borah, Owens, and Irwin [8], Chao and Hsu [10] and by Lewis, Pong and VanCleave [33].