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

Tim Barrera, Jeff Griffith, Gabriel Robins, and Tongtong Zhang

Technical Report No. CS-93-31
June 1, 1993

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

Tim Barrera, Jeff Griffith, Gabriel Robins and Tongtong Zhang

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

Abstract

We propose several efficient enhancements to the Iterated 1-Steiner (I1S) heuristic of Kahng and Robins [17] for the minimum rectilinear Steiner tree problem. For typical nets, our methods obtain average performance of less than 0.25% from optimal, and produce optimal solutions up to 90% of the time. 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. Our algorithms are highly parallelizable, and extend to arbitrary weighted graphs, and thus, our methods are applicable in practical routing regimes. We prove that given a pointset in the Manhattan plane, the minimum spanning tree (MST) degree of any specific point can be made to be 4 or less; similarly, we show that in three dimensions, the MST degree of any specific point can be made 14 or less. Using a perturbative argument, these results have been recently extended to show that for every pointset in the Manhattan plane there exists an MST with maximum degree 4, and for three-dimensional Manhattan space the maximum MST degree is 14 [28]. Aside from having independent theoretical interest, these results help reduce the running times of our algorithms.

1 Introduction

Optimal interconnections are a common theme in a number of areas, including VLSI layout [24], the design of buildings [30], and biology [3], where we typically seek a low-cost topology to interconnect a set of sites. The problem is formulated as follows:

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

In many applications the cost of an edge connecting two points is the Manhattan distance between the points (e.g., in wiring pins on a chip, routing water pipes or air-conditioning ducts through

*Corresponding author is Professor Gabriel Robins, Computer Science Department, Thornton Hall, University of Virginia, Charlottesville, VA 22903-2442, Email: robins@cs.virginia.edu, phone: (804) 982-2207, FAX: (804) 982-2214

a building, etc.) We thus obtain the *Minimum Rectilinear Steiner Tree* (MRST) problem, where edge costs are given by the L_1 metric, and the cost of a tree is the sum of its edge costs. Figure 1 shows an MST and an MRST for a fixed pointset.

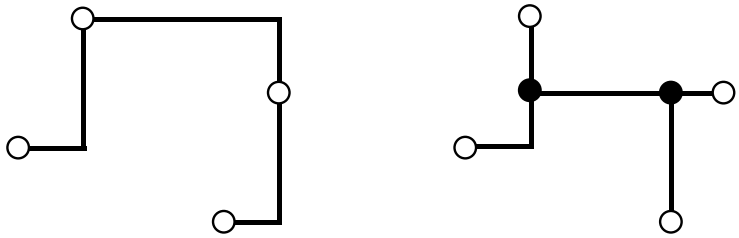

Figure 1: A minimum spanning tree (left) and MRST (right) for a fixed net.

Research on the MRST problem has been guided by several fundamental results. First, Hanan [10] has shown that there always exists an MRST with Steiner points chosen from the intersection of all the horizontal and vertical lines passing through all the points in P (see Figure 2), and this result indeed generalizes to all higher dimensions [31]. However, a second major result establishes that despite this restriction on the solution space, the MRST problem remains NP-complete [5], prompting a large number of heuristics, as surveyed in [16].

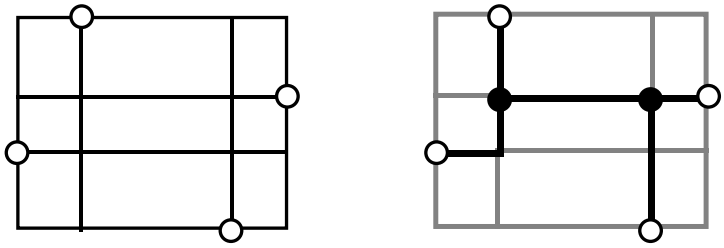

Figure 2: Hanan's theorem: there always exists an MRST with Steiner points chosen from the intersection of all the horizontal and vertical lines passing through all the points.

In solving intractable problems, we often seek provably good heuristics having bounded worst-case error from optimal. Thus, a third important result establishes that the rectilinear MST is a fairly good approximation to the MRST, with a worst-case performance ratio of $\frac{cost(MST)}{cost(MRST)} \leq \frac{3}{2}$ [14]. This implies that any MST-based strategy which improves upon an initial MST topology will also enjoy a performance ratio of at most $\frac{3}{2}$, which has prompted a large number of Steiner tree heuristics that resemble classic MST construction methods [12] [13] [15] [20] [21], all producing

Steiner trees with average cost 7% to 9% smaller than MST cost [26] [32].

Unfortunately, all MST-based MRST constructions were recently shown to have a worst-case performance ratio of exactly $\frac{3}{2}$ [18]. 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 [17]. I1S always performs strictly better than $\frac{3}{2}$ times optimal [27], and achieves almost 11% average improvement over MST cost. Efficient serial and parallel implementations of I1S were given in [1], enabling the benchmarking of the behavior of I1S on nets containing up to several hundred pins.

In this paper we propose several performance-improving enhancements to the I1S method. Our methods rely on a perturbative approach which deviates from pure greed and employs randomness to break ties during Steiner point selection. Our methods are highly parallelizable and are asymptotically no slower than the original I1S variants, yet afford improved average performance; indeed, extensive simulations indicate that for uniformly distributed nets of up to 8 pins, the average performance of our algorithms is only 0.25% away from optimal, refining the observation noted in [29]. 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 performance and efficiency.

We generalize I1S and its variants to three dimensions, as well as to the intermediate case where all the pins lie on k parallel planes. This formulation has several applications, including multi-layer routing [2] [9] [11], and building design [30]. Empirical testing suggests that this approach is very effective for three-dimensional Steiner routing, yielding up to 15% average improvement over MST costs. Our methods extend to arbitrarily weighted graphs, and are thus suitable as the basis of a practical multi-layer global router, where obstruction and congestion considerations affect routing [19].

In order to reduce the running time of our algorithms through a dynamic MST-maintenance scheme [1], we prove the following results under the Manhattan metric: 1) in two dimensions the maximum MST degree of a newly added point can be made to be at most 4; and 2) in three dimensions the maximum MST degree of a newly added point can be made to be at most 14. Using a perturbative argument, these results have been recently extended to show that for every

pointset in the Manhattan plane there exists an MST with maximum degree 4, and for three-dimensional Manhattan space the maximum MST degree is 14 [28] (the best previously known bounds for two and three dimensions were 6 and 26, respectively). Aside from having independent theoretical interest, these results help reduce the running times of our algorithms. The work in [28] also investigates the maximum MST degree for higher dimensions and other L_p norms, and relates the maximum MST degree to the so-called "Hadwiger" numbers.

In Section 2 we review the I1S method. Section 3 describes our enhanced perturbative approach, and Section 4 generalizes I1S to three dimensions. Section 5 presents extensive simulation results, both for two and for three dimensions, and we conclude in Section 6 with some open problems.

2 Overview of the 1-Steiner Method

We begin with a review of the 1-Steiner method [17]. 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 [7], 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. Figure 3 illustrates a sample execution of I1S, and Figure 4 describes the algorithm formally.

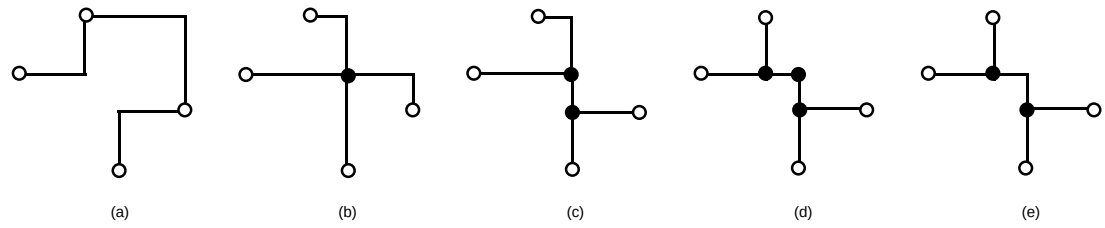

Figure 3: Execution of Iterated 1-Steiner (I1S) on a 4-point example. Note that in step (d) a degree-2 Steiner point is formed and is thus eliminated from the topology (e).

Although a 1-Steiner point may be found in $O(n^2)$ time using complicated techniques from

Algorithm Iterated 1-Steiner (I1S)
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.

computational geometry [6] [17] [25], 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 efficiently adds an entire set of "independent" Steiner points in a single iteration, thereby affording both practicality and reduced time complexity [1] [17]. An example of the output of Batched 1-Steiner (B1S) for a random pointset of size 300 is shown in Figure 5.

3 A Perturbative Approach

At each iteration the basic I1S heuristic uses pure greed to select a 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 which induce equal savings. For example, in Figure 6 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 select one of the 1-Steiner points randomly 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 solution trees), where m is an input parameter.

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

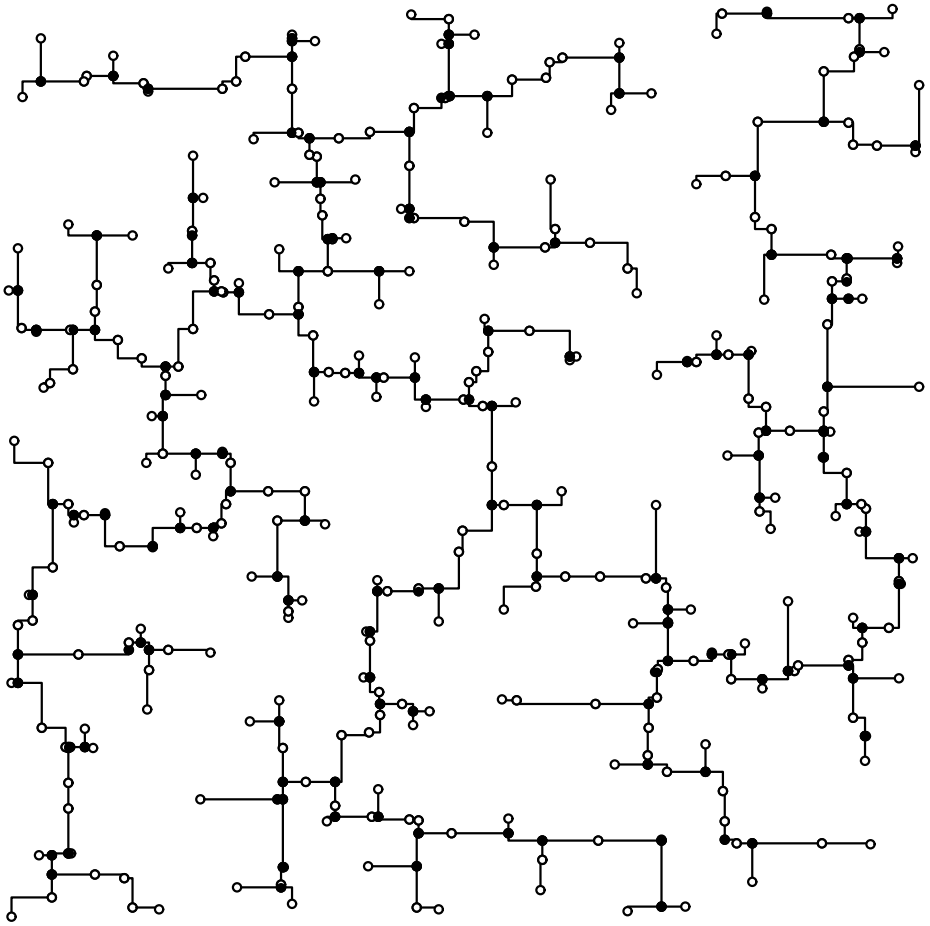

Figure 5: An example of the output of Batched 1-Steiner (B1S) on a random 300-point set (hollow dots). The Steiner points produced by B1S are denoted by dark solid dots.

local minima), we also propose a mechanism that allows small deviations from pure greed: we allow I1S to select a 1-Steiner point 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 slight disimprovement (with respect to the best possible savings), but could possibly create opportunities for greater savings in future iterations.

Finally, we note that performance may improve if instead of looking for 1-Steiner points, we search for *pairs* of Steiner points (having maximum savings with respect to other pairs): for example, such a 2-*Steiner algorithm* will optimally solve the example pointset of Figure 6. Combining these three techniques of (i) non-deterministic tie-breaking, (ii) near-greedy search, and (iii) k-

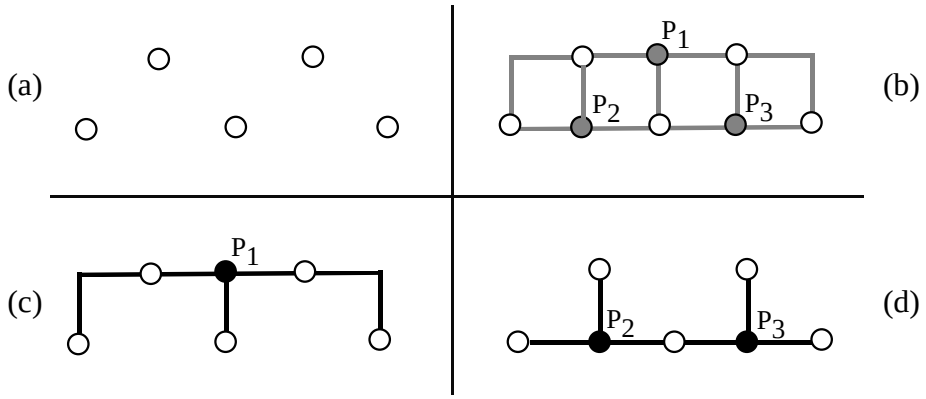

Figure 6: An unfortunate tie-breaking choice for a 1-Steiner point may interfere with the savings of other potential 1-Steiner candidates. If P1 is selected (b), then the MST savings of both P2 and P3 become 0 in the next iteration, yielding a suboptimal tree of cost 7 (c); on the other hand, if P2 is selected first, then P3 may be selected next, yielding an optimal tree of cost 6 (d).

Steiner selection, we obtain a new Perturbated Iterated k-Steiner algorithm (PIkS), as shown in Figure 7.

Algorithm Perturbated Iterated k-Steiner (PIkS)
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** $C = \{X \subseteq H(P) \mid \lvert X\rvert \leq k, \Delta\text{MST}(P \cup S, X) > 0\} \neq \emptyset$ **Do** **Find** $Y \in C$ with maximum $\Delta\text{MST}(P \cup S, Y)$ **Randomly** select $Z \in C$ with $\Delta\text{MST}(P \cup S, Z) > \Delta\text{MST}(P \cup S, Y) - \delta$ $S = S \cup Z$ **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 7: The Perturbated Iterated k-Steiner (PIkS) method.

Note that the original I1S algorithm of Figure 4 is equivalent to our new PIkS algorithm with $k = 1$, $m = 1$, and $\delta = 0$. Our PIkS scheme can also be extended using a "non-interfering" criterion as in [1] [17], to yield a Perturbated Batched k-Steiner algorithm (PBkS), where a maximal number of Steiner points are added during each round. The time complexity of PBkS is $O(m \cdot n^{2(k-1)} \cdot T(n))$, where $T(n)$ is the time complexity of B1S; thus, for fixed m, PB1S runs asymptotically within

the same time as B1S, namely $O(n^3)$. The PBkS template is a method of improving performance at the expense of running time, allowing a smooth tradeoff between performance and efficiency. Although PBkS is guaranteed to always yield optimal solutions for $\leq k-2$ pins, its time complexity increases exponentially with k.

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 a given round are likely to produce positive MST savings in subsequent rounds. Thus, rather than to examine the MST savings of all Hanan candidates in a given round, we may only consider the candidates that produced positive savings in the previous round. This strategy would therefore significantly reduces the time spent during each round, without substantially affecting the performance. We call this streamlined version of B1S the *modified batched 1-Steiner* (MB1S) algorithm.

4 Steiner Routing in Three Dimensions

Three-dimensional packaging is beginning to emerge as a viable VLSI design technology [2] [9] [11]; unfortunately, most existing CAD tools and techniques implicitly address two dimensions only. In contrast, the PIkS 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) version, where the points of P lie on L parallel planes. This formulation corresponds to a multi-layer design containing L layers. The cost of routing between one layer and another (i.e., vias) is likely to be substantially higher than staying on the same single layer, and this may be modeled by varying the distance between the layers. In Section 5 we present extensive benchmark data for several combinations of values for L and n. 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 $L = 1$.

Our three-dimensional PIkS method may be implemented efficiently by using a generalization of Hanan's theorem to higher dimensions [31]. 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 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 $\frac{5}{3}$, 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 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 (see Section 5). Also as expected, the average performance improves as the number of layers increases.

In computing the MST savings of each Steiner candidate, a key observation is that 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 [8]. 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 $R_i(p)$, with the property that only the closest point in each region $R_i(p)$ may be connected to p in the MST(P). This implies that linear time suffices to compute the MST savings of a 1-Steiner candidate [1].

For example, in the Manhattan plane, it is known that to update an MST with a newly added point p, it suffices to search only for the closest point to p in each of the four regions defined by the two lines passing through p at 45 and -45 degrees, respectively [13] (Figure 8(a)). This follows from the fact that each region of such a plane partition has the following *uniqueness property*:

The Uniqueness Property: Given a point p in d-dimensional space, a space 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 distance between the two points u and w. We now prove that the above 4-region partition of the plane has the uniqueness property.

Lemma 4.1 *Given a point $p = (x_0, y_0)$ in the Manhattan plane, each region of the partition of the plane defined by the two lines $y - y_0 = x - x_0$ and $y - y_0 = x_0 - x$ 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)). A boundary between two 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 the names of u and v in this proof). Consider the diamond D in R_1 with one corner at p, with u on the boundary of D (see Figure 8(c)). Let c be the center 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 has the uniqueness property. The other three regions are handled similarly.

□

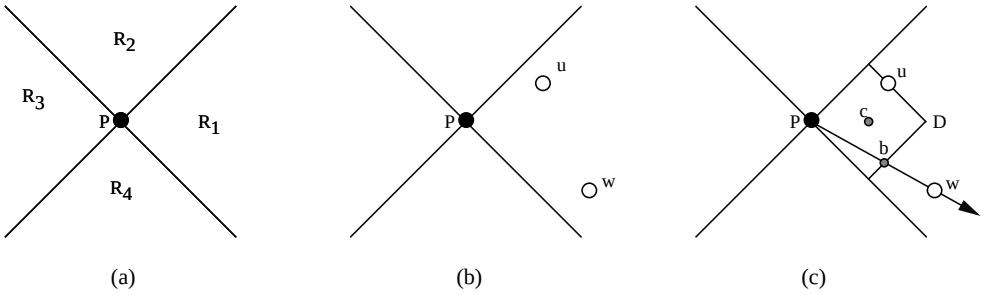

Figure 8: A partition of space into a number of regions with respect to a point p (a) has the *uniqueness property*: if for every two points u and w that lie in the same region (b), either $\text{dist}(w,u) \leq \text{dist}(w,p)$ or else $\text{dist}(u,w) \leq \text{dist}(u,p)$ (c).

A partition of a space into regions is said to have the uniqueness property if each region of the partition has the uniqueness property. Using similar arguments to those of Lemma 4.1, we can show an analogous result for three dimensions, where we partition space into 14 regions corresponding to the faces of a truncated cube (Figure 9(a)), i.e., a solid obtained by chopping off the corners of a cube, yielding 6 square faces and 8 equilateral triangle faces (Figure 9(b)); this solid is known as a "cuboctahedron" [23]. The 14 regions of this partition correspond to the faces of the cuboctahedron, namely 6 pyramids with square cross-section (Figure 9(c)) and 8 pyramids with triangular cross-section (Figure 9(d)). Again, region boundaries may be arbitrarily assigned to either adjacent region. We call this particular partition the *cuboctahedral partition*, and refer to the two types of induced regions as *square pyramids* and *triangular pyramids*. We now show that the cuboctahedral partition has the uniqueness property.

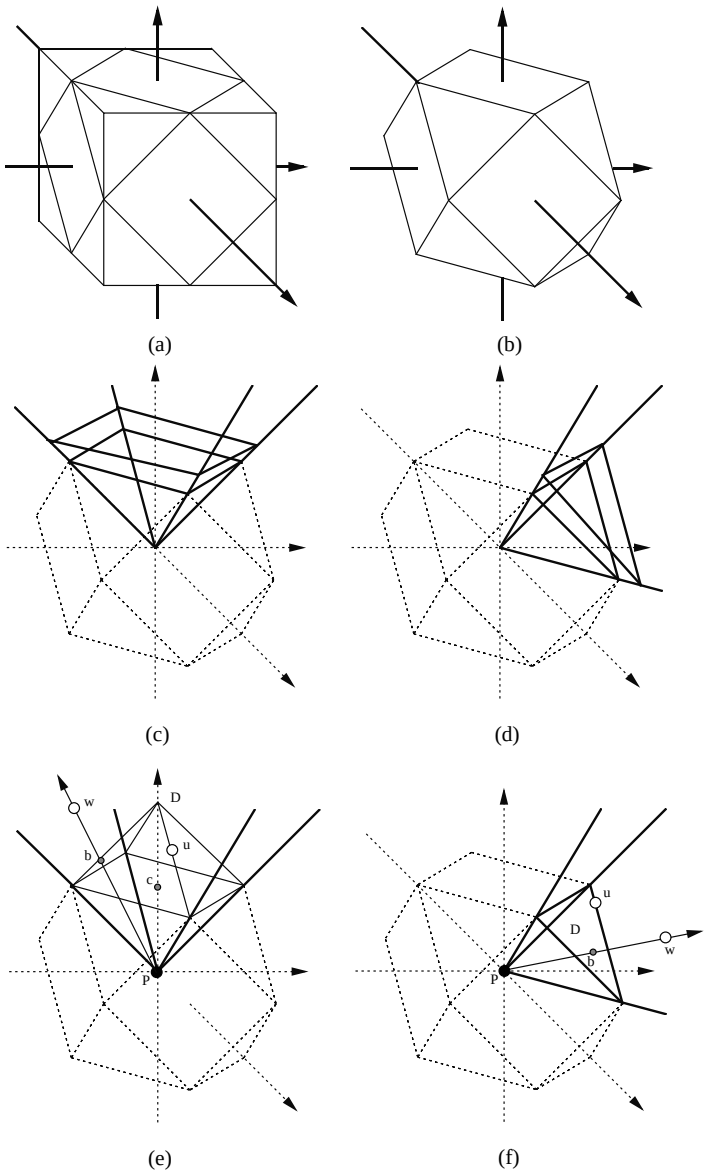

Figure 9: A truncated cube (a-b) induces a three-dimensional cuboctahedral space partition where each region has the uniqueness property. The 14 regions consist of 6 square pyramids (c), and 8 triangular pyramids (d). Using the triangle inequality, the uniqueness property may be shown to hold for each region (e-f).

Theorem 4.2 *Given a point p in three-dimensional space under the Manhattan metric, each of the* 14 *regions in the cuboctahedral partition of space with respect to p has the uniqueness property.*

Proof: This proof is a generalization to three dimensions of the proof of Lemma 4.1. Consider

one of the square pyramids R with respect to p (Figure 9(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 names of u and v in this proof). Consider the locus of points $D \subset R$ that are distance $\text{dist}(u,p)$ from p. (Figure 9(e)); D is actually the boundary of an upper half of an octahedron. Let c be the center of the octahedron determined by D, and 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 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 9(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 names of u and v in this proof). Consider the locus of points D in R that are distance $\text{dist}(u,p)$ from p (Figure 9(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 the region R has the uniquenes property. The other triangular pyramids are handled similarly. □

We thus have the following:

Corollary 4.3 *Given a three-dimensional pointset P and an a additional point x, there exists an MST over $P \cup \{x\}$ where x has degree of at most* 14 *in the MST.*

Proof: The cuboctahedral partition of space as described above 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. Therefore, the degree of any single point in the MST can be forced to be 14 or less. □

It is still an open question whether for three dimensions the cuboctahedral partition has the least number of regions among all possible partitions possessing the uniqueness property; we conjecture that it is. On the other hand, we observe that 13 is a lower bound:

Theorem 4.4 *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 (i.e., all points must connect to the origin) and thus the origin point has degree 13 in the MST. It is still open whether there exists a three-dimensional pointset where the maximum MST degree is forced to be as high as 14 [28]. □

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. A two-dimensional example of this method is given in Figure 10, while Figure 11 gives a formal description of this algorithm. Note that dynamic MST maintenance may also be achieved in sub-linear time [4], but such methods appear impractical due to their complexity and large hidden constants. A similar method was used in [33] to obtain a sub-quadratic MST algorithm in higher dimensions, but no attempt was made to optimize the number of necessary regions.

Note that Theorem 4.2 and Corollary 4.3 do not directly imply that the maximum overall MST degree in three dimensions is 14, since points lying on the boundaries between regions may be the same distance from the origin as they are from each other (e.g., consider the 18 points corresponding to the 6 vertices of an octahedron, and the 12 midpoints of the octahedron edges). It turns out, however, that in all such cases ties for connection during MST construction may be broken appropriately as to keep the overall MST degree low. In order to handle distance ties with respect to points that lie on region boundaries, we define a more restrictive, strict version of the uniqueness property:

The Strict Uniqueness Property: Given a point p in d-dimensional space, 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 $\text{dist}(u,w) < \text{dist}(u,p)$.

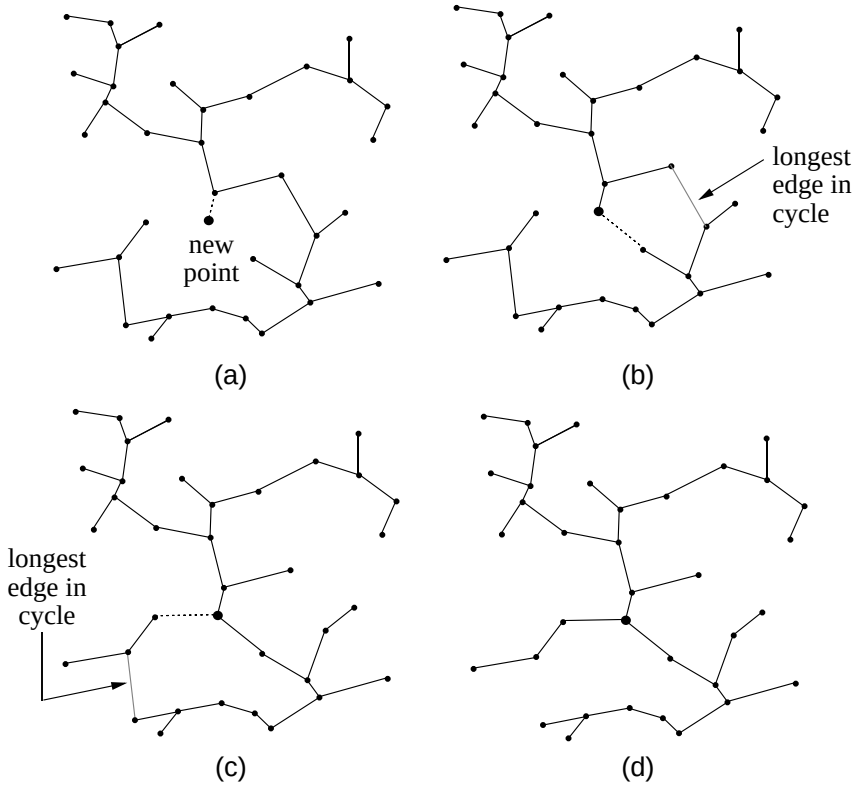

Figure 10: Dynamic MST maintenance: adding a point to an existing MST entails connecting the point to its closest neighbor in each region, and deleting the longest edge on each resulting cycle (the Euclidean metric has been used for clarity in this illustration).

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 11: Linear-time dynamic MST maintenance.

Note that is 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. Using a perturbative argument, Robins and

Salowe [28] prove that by breaking ties judiciously, lower-dimensional regions do not increase the maximum MST degree:

Theorem 4.5 *Given a partition of d-dimensional Manhattan 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 being lower-dimensional, and are not required to have the uniqueness property at all), then the maximum MST degree in this space is r' or less.*

Proof Sketch: Given a pointset P, perturb the coordinates of the points by a tiny amount so that the lower-dimensional regions become empty of points. This is always possible to do, and yields a new perturbed pointset P'. Clearly, interpoint distances in P' are not much different from the corresponding interpoint distances in P, and therefore the cost of the MSTs over P' and P can differ by only a tiny amount which we can make arbitrarily small. This implies that the MSTs over P and over P' have the *same* topology. But the MST over P' has maximum degree r', since only the r' regions are nonempty with respect to the pointset P'. Finally, use the topology of the MST for P' to connect the points of P; this would correspond to an MST over P. For more details, the reader is referred to [28]. □

Corollary 4.6 *Every pointset in the Manhattan plane has an MST with maximum degree* 4.

Corollary 4.7 *Every pointset in three-dimensional Manhattan space has an MST with maximum degree* 14.

The bound of 4 of the maximum MST degree in the Manhattan plane is tight, since there are examples which achieve that bound (e.g., the center and vertices of a diamond). While Theorem 4.4 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 any examples in three-dimensional Manhattan space which force an MST degree of 14. We conjecture that none exist.

5 Experimental Results

We have implemented the PIkS and the PBkS algorithms, as well as several variants, using C in the SUN IPC workstation environment. Our code is available from the authors upon request. The following algorithm variants were tested:

- **B1S** - The batched variant of iterated 1-Steiner [17];
- **MB1S** - The B1S variant modified for faster execution;
- **PB1S** - The perturbated version of B1S;
- **I2S** - The iterated 2-Steiner algorithm;
- **PI2S** - The perturbated 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(PB1S,I2S,PI2S)** - The metaheuristic over heuristics PB1S, I2S, and PI2S, i.e., the best solution found by any of these heuristics; and
- **META(B1S,MB1S,PB1S,I2S,PI2S)** - The metaheuristic over heuristics B1S, MB1S, PB1S, I2S, and PI2S, i.e., the best solution found by any of these heuristics.

For comparison, we have also tested:

- **UKy** - The Steiner heuristic of Lewis et al. [22]; and
- **OPT** - The optimal Steiner tree algorithm of Salowe and Warme [29].

Recall from the end of Section 3 that MB1S is a more efficient version of B1S, since it only examines a fraction of the Hanan candidates in each round (i.e., only the ones with positive MST savings in the previous round). UKy has average performance on par with the best existing methods [22], while OPT is the fastest known *optimal* rectilinear Steiner tree algorithm [29]. All algorithms have been extensively benchmarked using various net cardinalities. Up to 10000 instances of each

cardinality were solved using each algorithm. The instances were generated randomly by 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 [16] [17].

Figure 12 shows the performance comparison of MB1S, PI2S, UKy, and OPT, while Table 1 in the Appendix gives more detailed performance data. We observe that the average performance of our method is extremely close to optimal: for $n = 8$, PI2S is on average only about 0.25% away from optimal. Moreover, the solutions found by PI2S are optimal about 90% of the time for 8-pin nets. Table 2 in the Appendix tracks the percentage of cases where the various heuristics find the optimal solution, and this information is also depicted pictorially in Figure 13.

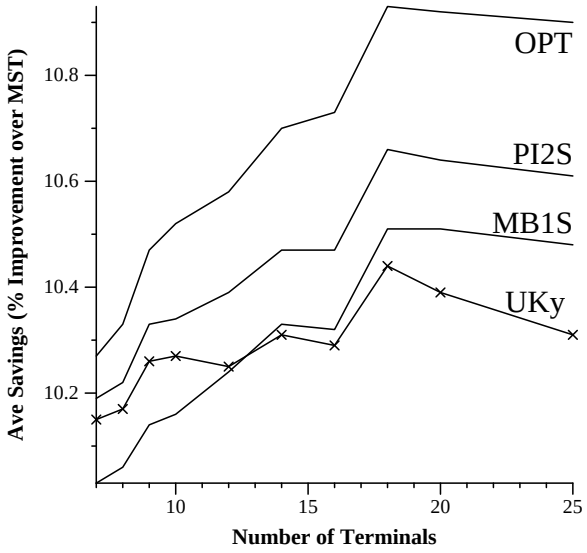

Figure 12: Average performance in two dimensions of several of the heuristics. Note that PI2S is only 0.25% (or less) away from optimal.

The average running times of the algorithms for various cardinalities are compared in Figures 14 and 15. Table 3 in the Appendix gives the average CPU times, in seconds, for each heuristic and cardinality. 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. Note that in the net cardinality ranges depicted in Figures 14 and 15, the polynomial-time UKy heuristic requires more time than the exponential-time OPT heuristic! naturally, OPT eventually runs slower than

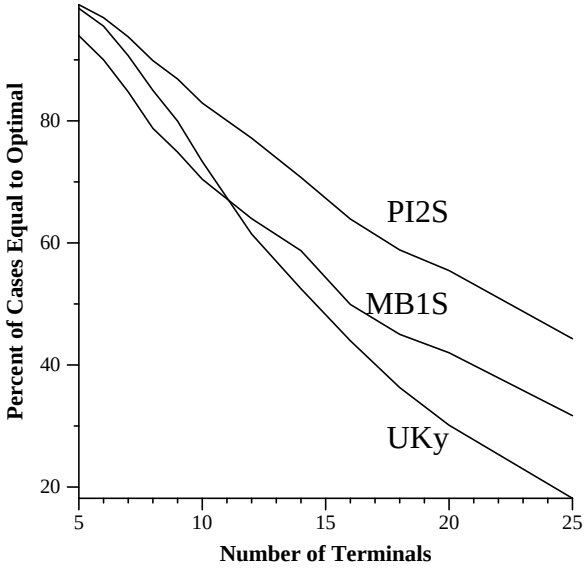

Figure 13: The percentage of all cases when the heuristics find the optimal solution. Note that PI2S yields optimal solutions a large percentage of the time.

UKy, which happens at around 27 points. Using PIkS (or PBkS) with values of k greater than 2 improves the performance, but slows down the algorithm; it is easy to see that for arbitrary k, PIkS (PBkS) 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 PI1S (PB1S). This allows a smooth tradeoff between performance and efficiency. However, the performance of the PBkS algorithm with $k = 2$ is already so close to optimal, that in most applications increasing k further may not justify the incurred time penalty.

In three dimensions, we observed that the average performance of PB1S 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., $\frac{5}{3}$ vs. $\frac{3}{2}$). Figure 16 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 4 in the Appendix gives more detailed performance data, for various values of L. In all cases, the L parallel planes were uniformly spaced in the unit cube (i.e., there were separated by $\frac{G}{L}$ units apart, where $G = 10000$ is the gridsize. Unfortunately, the OPT algorithm of Salowe and Warme [29] does not easily generalize to higher dimensions; thus,

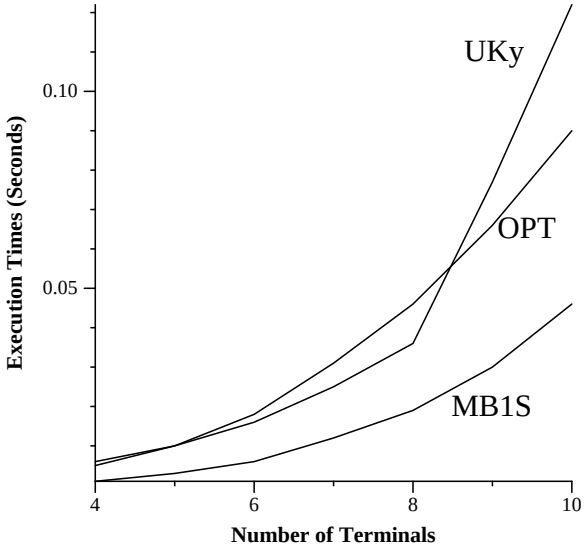

Figure 14: Average execution times, in CPU seconds, for 4 through 10 points.

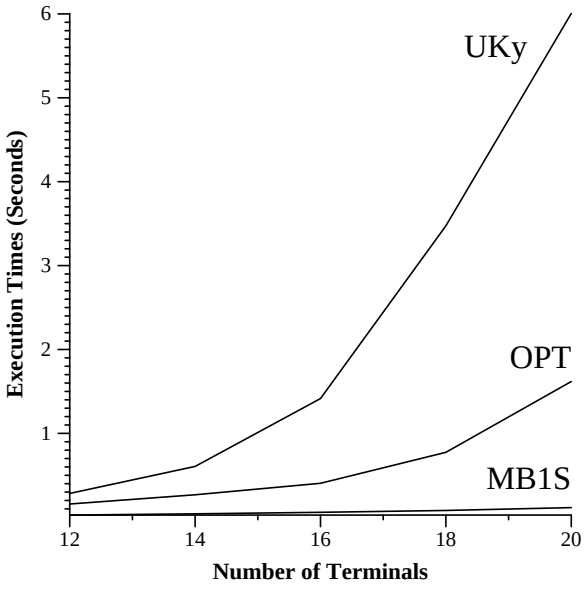

Figure 15: Average execution times, in CPU seconds, for more than 10 points.

we did not compare our three-dimensional version of PB1S to optimal.

Table 5 gives statistics for three dimensions on the number of Steiner points induced by B1S, as well as on the number of rounds that occur in B1S before termination. As is the case in two dimensions [1], the number of rounds for B1S in three dimensions is on average very small.

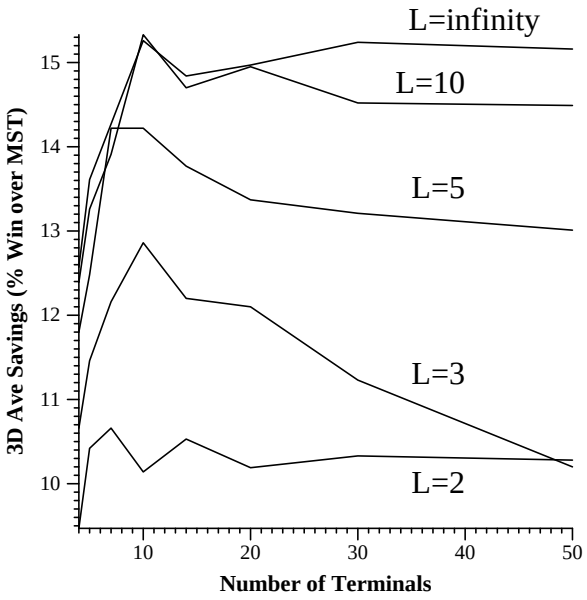

Figure 16: Average performance of PB1S in three dimensions for various values of L = number of parallel planes.

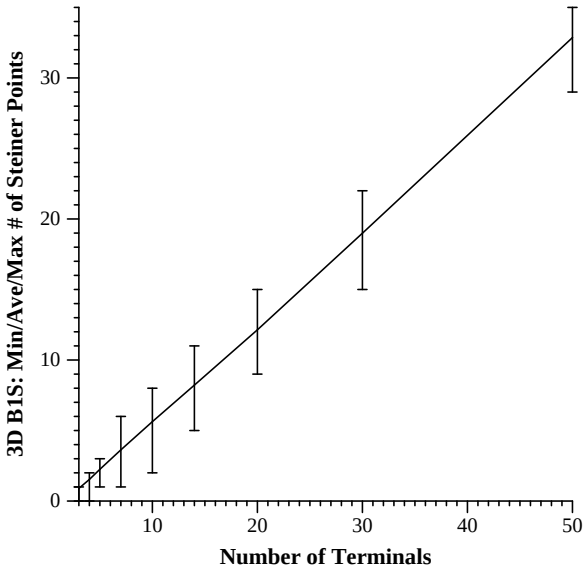

Figure 17: Minimum, average, and maximum number of Steiner points added by MB1S in three dimensions

6 Conclusion

We proposed several enhancements to the Iterated (Batched) 1-Steiner heuristic [17], based on a perturbative approach. Our methods enjoy the same asymptotic time complexity as Iterated

(Batched) 1-Steiner, yet offer improved average performance. We also proposed a method of increasing performance at the expense of running time, allowing a smooth tradeoff between performance and efficiency. Extensive simulations indicate that for typical nets, the average performance of our new algorithms is only a small fraction of a percent away from optimal, and our solutions are actually optimal for most random uniformly distributed instances. We then generalized I1S 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 [11] and multi-layer routing [2] [9]. The perturbative 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 [19].

We reduced the running time of our algorithms through a dynamic MST-maintenance scheme [1], and we proved that under the Manhattan metric: 1) in two dimensions the maximum MST degree of a newly added point can be made to be at most 4; and 2) in three dimensions the maximum MST degree of a newly added point can be made to be at most 14. These results have been recently extended [28] to show that for every pointset in the Manhattan plane there exists an MST with maximum degree 4, and for three-dimensional Manhattan space there exists an MST with maximum degree 14 (the best previously known bounds for two and three dimensions were 6 and 26, respectively). Aside from having independent theoretical interest, these results help reduce the running times of our algorithms.

Remaining open research questions include:

1. Finding an MRST heuristic with performance consistently higher than PB1S (or MB1S), but with a significantly better running time (note that it would not suffice to just find a heuristic with better performance but which runs more slowly, since increasing k in PBkS will achieve exactly that);
2. Determining if the maximum MST degree in three dimensional Manhattan space is 13 or 14;
3. Further tightening the bounds of [28] on the maximum MST degree in higher dimensions and under various L_p norms;
4. The exact planar MRST algorithm of Salowe and Warme [29] is by far the fastest currently

known; unfortunately, it relies heavily on the structure of planar MRSTs, and thus does not easily generalize to higher dimensions. Generalizing [29] to higher dimensions, or significantly improving its efficiency in two dimensions remains a major challenge.

7 Acknowledgments

We thank Professor Forbes Lewis, Chia-Chi Pong, and Nancy VanCleave [22] for making their local-improvement Steiner heuristic (UKy) code available to us. Many thanks go Professor Jeff Salowe and David Warme [29] for providing an efficient implementation of an exact Steiner tree algorithm (OPT).

References

[1] 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 Symposium on VLSI, Kalamazoo, MI, March 1993, pp. 90–94.

[2] 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.

[3] L. R. Foulds, *Maximum Savings in the Steiner Problem in Phylogeny*, J. Theoretical Biology, 107 (1984), pp. 471–474.

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

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

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

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

[8] L. J. Guibas and J. Stolfi, *On Computing all North-East Nearest Neighbors in the L1 Metric*, Information Processing Letters, 17 (1983), pp. 219–223.

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

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

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

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

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

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

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

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

[17] 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.

[18] ——, *On Performance Bounds for a Class of Rectilinear Steiner Tree Heuristics in Arbitrary Dimension*, IEEE Trans. on Computer-Aided Design, 11 (1992), pp. 1462–1465.

[19] J. W. Lavinus and J. P. Cohoon, *Routing a Multi-Terminal Critical Net: Steiner Tree Construction in the Presence of Obstacles*, Tech. Rep. CS-93-19, Computer Science Department, University of Virginia, April 1993.

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

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

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

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

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

[25] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, New York, 1985.

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

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

[28] G. Robins and J. S. Salowe, *On the Maximum Degree of Minimum Spanning Trees*, Tech. Rep. CS-93-22, Computer Science Department, University of Virginia, May 1993.

[29] J. S. Salowe and D. M. Warme, *An Exact Rectilinear Steiner Tree Algorithm*, in Proc. IEEE Intl. Conf. on Computer Design (to appear), Cambridge, MA, October 1993.

[30] 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.

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

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

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

8 Appendix: Performance Data

Average Perormance Results in Two Dimensions												
n	# nets	(1) UKy	(2) B1S	(3) MB1S	(4) PB1S	(5) I2S	(6) PI2S	Meta (2-3)	Meta (4-6)	Meta (2-6)	Meta (1-6)	OPT
4	10000	8.54	8.54	8.54	8.54	8.54	8.54	8.54	8.54	8.54	8.54	8.54
5	10000	9.44	9.34	9.34	9.39	9.44	9.45	9.37	9.46	9.46	9.47	9.47
6	10000	9.91	9.79	9.78	9.85	9.90	9.92	9.82	9.93	9.93	9.96	9.97
7	5000	10.15	10.04	10.03	10.12	10.15	10.19	10.08	10.21	10.21	10.25	10.27
8	5000	10.17	10.06	10.06	10.15	10.17	10.22	10.11	10.24	10.24	10.30	10.33
9	5000	10.26	10.16	10.14	10.25	10.27	10.33	10.20	10.34	10.34	10.42	10.47
10	5000	10.27	10.19	10.16	10.27	10.28	10.34	10.22	10.36	10.36	10.45	10.52
12	5000	10.25	10.24	10.24	10.34	10.33	10.39	10.29	10.41	10.41	10.50	10.58
14	5000	10.31	10.33	10.33	10.42	10.40	10.47	10.37	10.49	10.49	10.59	10.70
16	5000	10.29	10.33	10.32	10.42	10.40	10.47	10.37	10.49	10.49	10.60	10.73
18	4000	10.44	10.51	10.51	10.61	10.58	10.66	10.56	10.67	10.67	10.78	10.93
20	3000	10.39	10.51	10.51	10.61	10.56	10.64	10.55	10.66	10.66	10.76	10.92
25	2000	10.31	10.47	10.48	10.57	10.53	10.61	10.52	10.62	10.62	10.71	10.90
30	100	10.23	10.45	10.59	10.76	10.55	10.62	10.51	10.63	10.63	10.70	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 1: Performance statistics: the performance figures denote average percent improvement over MST cost.

Percent of the Time Solution Was Optimal												
n	# nets	(1) UKy	(2) B1S	(3) MB1S	(4) PB1S	(5) I2S	(6) PI2S	Meta (2-3)	Meta (4-6)	Meta (2-6)	Meta (1-6)	OPT
4	10000	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
5	10000	98.43	94.29	93.96	96.53	98.42	99.04	95.30	99.26	99.27	99.86	100.00
6	10000	95.52	90.34	90.00	93.68	94.92	96.91	91.98	97.46	97.46	99.24	100.00
7	5000	90.68	85.20	84.75	89.96	90.72	93.77	87.45	94.75	94.75	98.02	100.00
8	5000	85.00	79.62	78.77	85.26	84.73	89.87	82.13	90.94	90.95	96.39	100.00
9	5000	79.96	75.80	74.88	82.23	81.70	86.86	78.64	87.92	87.92	94.38	100.00
10	5000	73.34	72.03	70.44	79.12	77.01	82.92	75.09	84.49	84.49	91.88	100.00
12	5000	61.46	65.15	63.98	73.01	70.41	77.17	68.49	78.72	78.72	87.12	100.00
14	5000	52.49	57.86	58.73	66.69	62.84	70.73	62.31	72.33	72.33	80.86	100.00
16	5000	43.97	50.36	49.92	60.17	54.84	63.92	54.99	65.73	65.73	75.50	100.00
18	4000	36.30	44.90	45.02	55.65	49.33	58.85	49.45	61.02	61.02	70.47	100.00
20	3000	30.13	42.87	42.00	53.20	45.70	55.47	47.00	58.23	58.23	66.87	100.00
25	2000	18.15	31.25	31.70	41.45	34.15	44.30	35.60	45.90	45.90	52.85	100.00
30	100	6.00	27.00	26.00	41.00	32.00	45.00	31.00	47.00	47.00	50.00	100.00

Table 2: Optimality Percentages: the figures denote what percent of the time the various heuristics found the optimal solution.

Average CPU Time Per Net (in Seconds)											
n	(1) UKy	(2) B1S	(3) MB1S	(4) PB1S	(5) I2S	(6) PI2S	Meta (2-3)	Meta (4-6)	Meta (2-6)	Meta (1-6)	OPT
4	0.005	0.002	0.001	0.054	0.001	0.072	0.003	0.127	0.130	0.135	0.006
5	0.010	0.004	0.002	0.107	0.003	0.068	0.006	0.178	0.184	0.194	0.010
6	0.016	0.006	0.004	0.184	0.006	0.128	0.010	0.318	0.328	0.344	0.018
7	0.025	0.009	0.006	0.287	0.012	0.231	0.015	0.530	0.545	0.570	0.031
8	0.036	0.013	0.009	0.432	0.019	0.384	0.022	0.835	0.857	0.893	0.046
9	0.077	0.019	0.012	0.571	0.030	0.634	0.031	1.235	1.266	1.343	0.066
10	0.122	0.025	0.016	0.806	0.046	0.951	0.041	1.803	1.844	1.966	0.090
12	0.284	0.043	0.026	1.324	0.096	1.967	0.069	3.387	3.456	3.740	0.158
14	0.606	0.066	0.041	2.127	0.180	3.623	0.107	5.930	6.037	6.643	0.268
16	1.415	0.096	0.059	3.288	0.317	6.372	0.155	9.977	10.13	11.55	0.405
18	3.474	0.134	0.081	4.216	0.540	9.700	0.215	14.46	14.68	18.15	0.774
20	6.005	0.181	0.115	6.440	0.833	15.87	0.296	23.14	23.44	29.45	1.618
25	24.71	0.341	0.220	11.94	2.050	48.09	0.561	62.08	62.64	87.35	13.88
30	66.08	0.569	0.375	18.93	4.184	86.10	0.944	109.2	110.1	176.2	495.8
50	852.9	2.732	1.694	79.78	35.91	764.6	4.426	880.3	884.7	1738	
70	3064	6.664	4.236	255.4	150.3	5668	10.90	6074	6085	9149	

Table 3: Average execution times, in seconds, for each of the heuristics.

Average Perormance of PB1S in Three Dimensions									
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 4: Average percent improvement over MST cost of PB1S 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 5: 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.