Two Heuristics for the Euclidean Steiner Tree Problem

Derek R. Dreyer[*] Michael L. Overton[†]

September 30, 2002

Abstract

The Euclidean Steiner tree problem is to find the tree with minimal Euclidean length spanning a set of fixed points in the plane, allowing the addition of auxiliary points to the set (Steiner points). The problem is NP-hard, so polynomial-time heuristics are desired. We present two such heuristics, both of which utilize an efficient method for computing a locally optimal tree with a given topology. The first systematically inserts Steiner points between edges of the minimal spanning tree meeting at angles less than 120 degrees, performing a local optimization at the end. The second begins by finding the Steiner tree for three of the fixed points. Then, at each iteration, it introduces a new fixed point to the tree, connecting it to each possible edge by inserting a Steiner point, and minimizes over all connections, performing a local optimization for each. We present a variety of test cases that demonstrate the strengths and weaknesses of both algorithms.

1 Steiner Trees

Given a set of fixed (or terminal) points $t_1, \ldots, t_n$ in $\Re^2$, the *Steiner tree problem* is to find the shortest network (in the Euclidean sense) connecting

[*]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA. E-mail: dreyer@cs.cmu.edu. The work of this author was conducted at New York University with support from National Science Foundation grant CCR-9625955.

[†]Computer Science Department, Courant Institute of Mathematical Sciences, New York University, New York, NY. E-mail: overton@cs.nyu.edu. The work of this author was supported in part by National Science Foundation grant CCR-9625955.

the points, allowing the addition of auxiliary points to the set for the purpose of minimizing the total length. These auxiliary points are called *Steiner points* and there need be at most $n-2$ of them [GP68]. Because of the ability to add Steiner points to the original graph, the Steiner tree problem is NP-hard [GGJ77], and thus good heuristics are perhaps the best hope.

An important aspect of Steiner trees is that the edges connected to the Steiner points all intersect at angles of 120 degrees. In fact, there is a theorem that states that "no two edges of a Steiner tree can meet at an angle less than 120 degrees" and "each Steiner point of a Steiner tree is of degree exactly three" [HRW92]. These facts are essential to an understanding of the problem and formed the basis for one of the heuristics we have developed.

The simplest example and one which occurs quite frequently inside larger problems is the 3-point case as seen in Figure 1, also known as the Fermat problem. The solution to this case, known as the Torricelli point, has a simple construction described in [Mel61] and [HRW92]. Figure 2 shows another elementary example consisting of 4 points forming a rectangle with sides 1 and 1.3. Here is a case where there exists a local minimal Steiner tree (the dashed tree) which is not the global minimal Steiner tree (the solid tree). In fact, the local minimal tree, which also has only 120-degree angles about the Steiner points, has length 3.252, whereas the global Steiner tree has length 3.032. Note that, in all the figures in this report, circles represent fixed points and stars represent Steiner points.

2 Heuristic Tools

Within our heuristics and in testing our heuristics we have utilized several algorithmic tools, some supplied by others and some of our own. Before moving on to a detailed description of our heuristics, it is first necessary to briefly discuss the general function of these tools and how we applied them.

2.1 Local Optimization Algorithms

There is a crucial step in both of our heuristics that involves finding locally optimal positions for the Steiner points given a fixed topology defining the edges of the graph. Originally, we used a primal-dual interior point method for minimizing a sum of Euclidean vector norms as described by Conn and

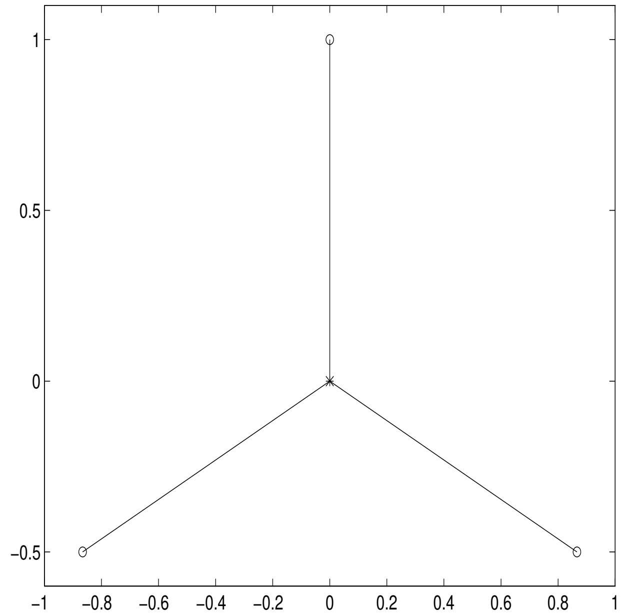

Figure 1: Steiner tree for simple 3-point case

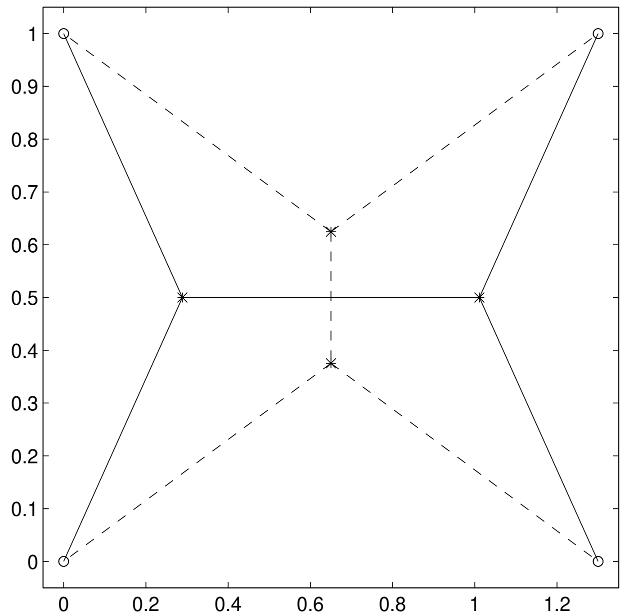

Figure 2: Global and local Steiner trees for simple 4-point case

Overton in [CO94] and written in MATLAB. This method was easily applicable to the Steiner tree problem and worked quite well. Currently, however, we are working with a similarly aimed, but considerably faster, code written in C by Knud Andersen[AA96]. Though naturally it takes longer to run large cases than small ones, the Andersen code is in general extremely efficient, and for the problems we have been dealing with (size 100 or less), the computation requires between 6 and 10 seconds on a Sparc 10 machine.

2.2 An Exact Steiner Algorithm

It has been immeasurably helpful in testing and comparing our heuristics to have access to a branch-and-bound algorithm, written by Xue, Dougherty, and Lillys in FORTRAN, that computes exact Steiner trees. Essentially, the algorithm begins by finding the Steiner tree for three of the fixed points. Then, at each iteration, it adds another fixed point to the graph, connecting it to the graph at each possible edge, and locally optimizing. Unfortunately, due to the exponential running time of the algorithm, we were not able to run it on cases of size 14 or greater. However, as we will describe in detail in Section 4, it provided the basic idea for our second heuristic.

2.3 The Minimal Spanning Tree

The *minimal spanning tree problem*, with edge cost defined by Euclidean distance, is essentially the Steiner tree problem without the ability to add Steiner points. In the worst case[1], specifically the simple 3-point case of Figure 1, the minimal spanning tree consists of two sides of an equilateral triangle with sides of length 1, while the exact Steiner tree consists of three edges of length $\frac{1}{\sqrt{3}}$ intersecting at the Torricelli point and bisecting each of the three angles of the triangle. So, the improvement of the exact Steiner tree over the minimal spanning tree is 13.4% in this case. On average, however, it is much less. Thus, the minimal spanning tree itself becomes a measuring-stick for Steiner heuristics. In addition, there exist a number of efficient algorithms for finding the minimal spanning tree (we have implemented a rather simple one), and so it can be a good starting point for building a somewhat better heuristic. In fact, finding the minimal spanning tree is the first step of the heuristic described in the next section.

[1] from the point of view of one who is trying to find a good approximation of the Steiner tree

3 The Steiner Insertion Heuristic

The first heuristic we have developed is the *Steiner insertion heuristic,* which is essentially an attempt to ensure the necessary, but not sufficient, condition that edges in Steiner trees meet at angles greater than or equal to 120 degrees. It was inspired by a suggestion of Thompson mentioned in the "Heuristics" chapter of [HRW92].

Algorithm 1 (SI) The Steiner Insertion Algorithm.

1. *Find the minimal spanning tree.*
2. *FOR each edge connecting fixed points* (t_x, t_y) *DO*
   (a) *Find the edge* (t_y, t_z) *that meets* (t_x, t_y) *at the smallest angle, where* t_z *can be either a fixed point or a Steiner point.*
   (b) *IF this angle is less than 120 degrees THEN*
      i. *Place a new Steiner point* s_n *on top of* t_y*.*
      ii. *Remove the edges* (t_x, t_y) *and* (t_y, t_z)*. These edges will no longer be considered for the loop of Step 2.*
      iii. *Add the edges* (t_x, s_n)*,* (t_y, s_n)*, and* (t_z, s_n)*.*
3. *Run the local optimization algorithm on the tree with its new topology.*

In implementing this algorithm, it is important to note that the topology switching and Steiner insertion of step 2b causes the formation of a zero-length edge. Such edges are ignored when considering edges for step 2a. In addition, in the FOR loop of step 2, one must be sure to consider each edge in both directions provided that, after processing it in one direction, it hasn't been removed. For instance, suppose we have a graph of three vertices and the minimal spanning tree has two edges (t_1, t_2) and (t_1, t_3). Then, suppose that step 2 examines both of these edges with t_1 in the place of t_x. Neither edge processing will result in any Steiner insertion because neither t_2 nor t_3 is connected to any other point but t_1. So, it becomes necessary to examine the edges from the reverse perspective, with t_2 or t_3 in the place of t_x. Lastly, it should be understood that, if the minimal spanning tree is not unique, different minimal spanning trees may produce different SI heuristic trees. Thus, since the initial ordering of the points can affect the outcome of the minimal spanning tree algorithm as well as the order in which the edges are examined in step 2 of SI, the SI heuristic tree is frequently dependent on the initial ordering of the points.

As a demonstration of how the algorithm works, we will step through its execution for the simple 4-point case. First, let's label the points $t_1 = (0, 0)$, $t_2 = (0, 1.0)$, $t_3 = (1.3, 1.0)$, and $t_4 = (1.3, 0)$. The minimal spanning tree consists of the edges (t_1, t_2), (t_2, t_3), and (t_3, t_4). The insertion heuristic would first notice that the edges (t_1, t_2) and (t_2, t_3) meet at an angle of 90 degrees, and therefore it would perform a Steiner insertion – discarding those edges, introducing the Steiner point s_1, and adding the edges (t_1, s_1), (t_2, s_1), and (t_3, s_1). Likewise, the heuristic would perform a Steiner insertion between the edges (t_4, t_3) and (t_3, s_1), thus producing a tree with edge list

$$\{(t_1, s_1), (t_2, s_1), (t_3, s_2), (t_4, s_2), (s_1, s_2)\}.$$

This is clearly the topology of the global minimal Steiner tree in Figure 2, and running the local optimization algorithm on it would produce the exact Steiner tree as seen there.

In theory, steps 1 and 2 of the Steiner insertion heuristic both have a worst-case running time of $O(n^3)$, but they turn out to be very fast in practice. The Andersen code for step 3, though very efficient, generally takes about as long as the other two, if not slightly longer, for cases of size 100 or less. So, for example, on a Sparc 10 running MATLAB (which makes external calls to Andersen's C code), the algorithm only takes 40 seconds on average to run for a 100-point problem, and a few seconds for small cases. More examples and results are given in Section 5.

4 The Incremental Optimization Heuristic

The second heuristic we have developed is the *incremental optimization heuristic.* As mentioned earlier, it is a greedy variant of the exact Steiner algorithm by Xue, Dougherty, and Lillys, but since it only performs a polynomial number of local optimizations, it does not obtain optimal solutions in general.

Algorithm 2 (IO) The Incremental Optimization Algorithm.

1. *Order the fixed points by the distance from their mean, the first being closest to it. (This ordering step greatly reduces the dependency of the*

final tree on the initial ordering of the points, although there can still be ties.)

2. *Insert a Steiner point between the first three fixed points, connect it to each, and locally optimize to obtain the Steiner tree for those three points. Call it the current tree.*

3. *FOR* $k = 4, \ldots, n$ *DO*

   (a) *Save the current tree as* oldtree *and set* besttree *to an artificial tree with length* ∞.

   (b) *FOR each edge* (a, b) *of the current tree DO*

      i. *Place a Steiner point* s *on the edge* (a, b).

      ii. *Remove the edge* (a, b).

      iii. *Add the edges* (t_k, s), (a, s), (b, s).

      iv. *Run the local optimization routine.*

      v. *If the resulting tree is shorter than* besttree, *then set* besttree *to this new tree.*

      vi. *Set the current tree to* oldtree.

   (c) *Set the current tree to* besttree.

4. *Set the final tree to the current tree.*

If we were to run this heuristic on the simple 4-point case, we would first place a Steiner point between t_1, t_2, and t_3, connect each of those points to the Steiner point, and locally optimize. Then, we would try connecting t_4 to each of the three edges of the current tree, and for each connection perform a local optimization. Connecting t_4 to (t_3, s_1) produces the globally optimal solution.

Since at each iteration two more edges are added to the tree, the total number of calls to the local optimization routine (which does most of the work) is $n^2 - 4n + 4$, and so the running time of this heuristic is $\Theta(n^2)$ times the running time of one local optimization. Therefore, working on a Sparc 10 in MATLAB (with external calls to Andersen's C code), we find that, on average, a 10-point test takes 10 minutes, a 40-point test 3 hours, and a 100-point test 1 day. These times are quite reasonable, considering that the algorithm is frequently very accurate, as we will discuss in the next section.

5 Test Results

Through experimentation, we have found that each of our heuristics for the Steiner tree problem has its own advantages and disadvantages. While the Steiner insertion heuristic is always much faster, the incremental optimization heuristic gets better results in most cases. However, for many large randomly generated cases, the latter cannot surpass the former and there does not seem to be much room for improvement anyway. In this section we provide the results of different kinds of test runs for both heuristics in order to give a well-rounded view of their performance.

Figure 3 and Figure 4 show the results of the two heuristics on a test case of nine points, grouped in three triangular-shaped groups of three, with one point from each group the vertex of a larger triangle. (Note that, in these figures, instances of stars and circles overlapping signify that a Steiner point coincided with a fixed point, i.e. was deemed unnecessary in the current topology by the local optimization routine.) Here, the trouble with the Steiner insertion heuristic is that, although it creates a tree with no angles less than 120 degrees, it fails to insert the critical Steiner point in the center of the large triangle. On the other hand, the more consistently accurate incremental optimization heuristic *does* manage to insert it. The importance of the critical Steiner point here is reflected in the heuristics' percent improvement over the minimal spanning tree: IO achieves 7.9% improvement, while SI only achieves 3.4%. In fact, in this case, the IO heuristic tree *is* the exact Steiner tree.

Similar results can be seen in this next example. Figure 5 and Figure 6 depict the results of both heuristics on a point set of 40, grouped into four 10-point clusters occupying the four corners of a square region. Again, though the SI tree satisfies the 120-degree rule, it fails to recognize the underlying structure of the point set, while the IO tree succeeds. The percent improvement for the SI is 0.7%, and for the IO 5.4%. We do not know if the IO tree *is* the exact Steiner tree because the point set is too large for us to use the exact algorithm, but it is most definitely very close. However, it should be noted that, whereas the SI tree was computed in a matter of seconds, the IO heuristic took over three hours.

Our third test is the well-known "ladder" test of Chung and Graham with $n = 10$ [CG78], and its exact Steiner tree is displayed in Figure 7. The results of our heuristics are shown in Figure 8 and Figure 9. While SI scores an expected 3.0% improvement over the minimal spanning tree, IO does quite

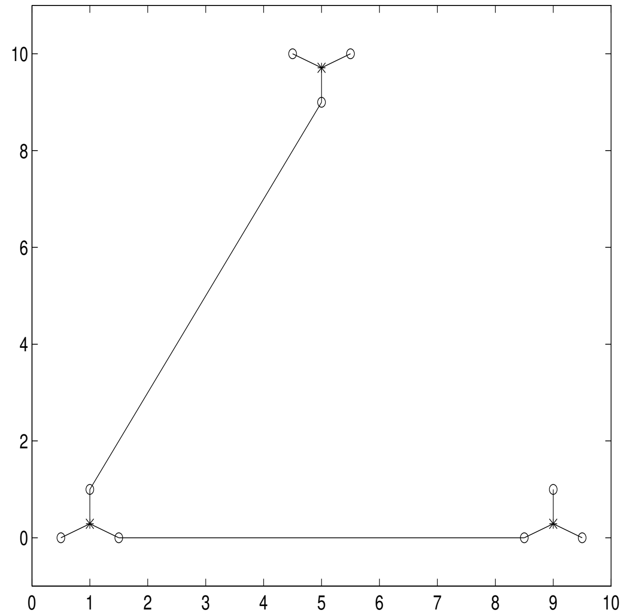

Figure 3: SI heuristic tree for first case

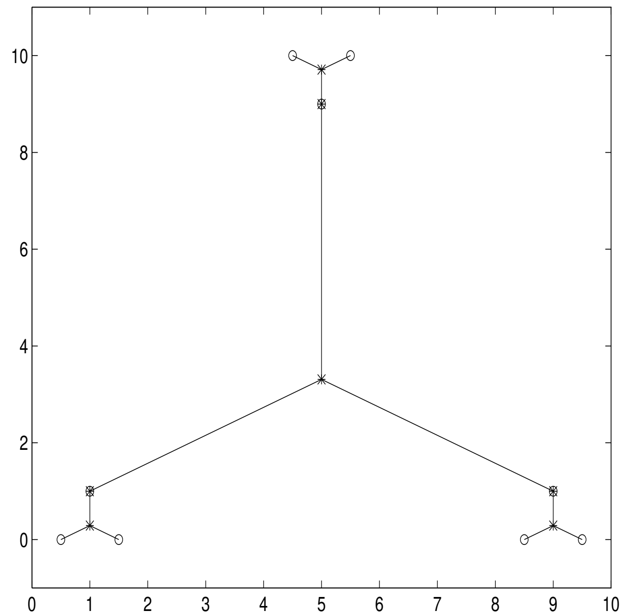

Figure 4: IO heuristic tree for first case

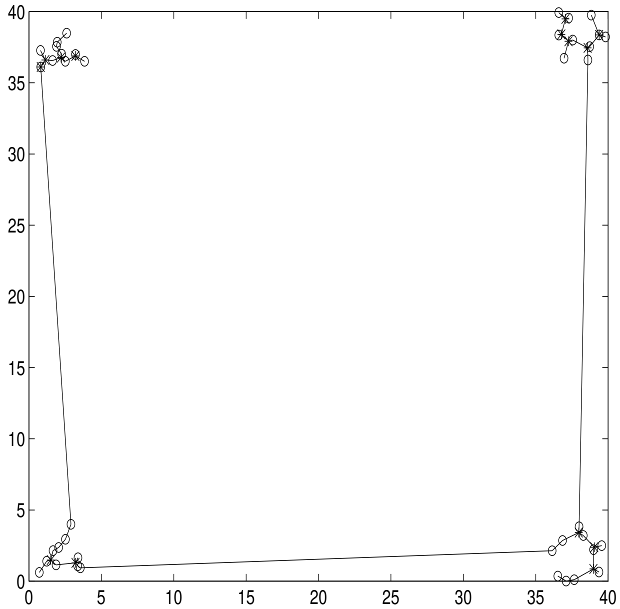

Figure 5: SI heuristic tree for second case

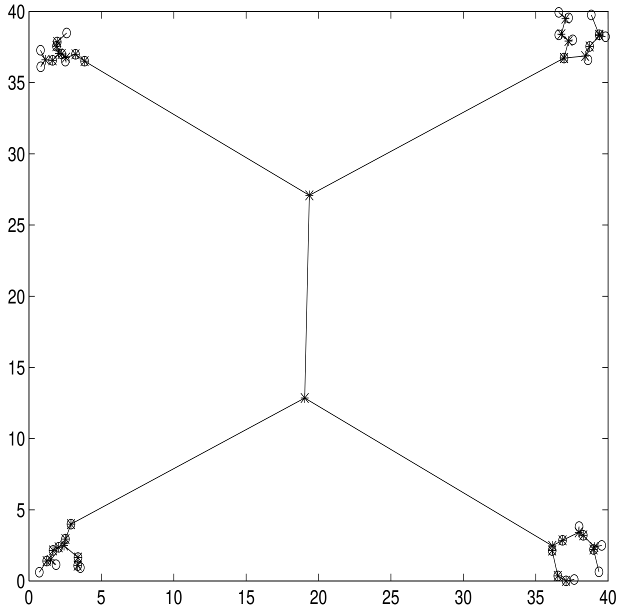

Figure 6: IO heuristic tree for second case

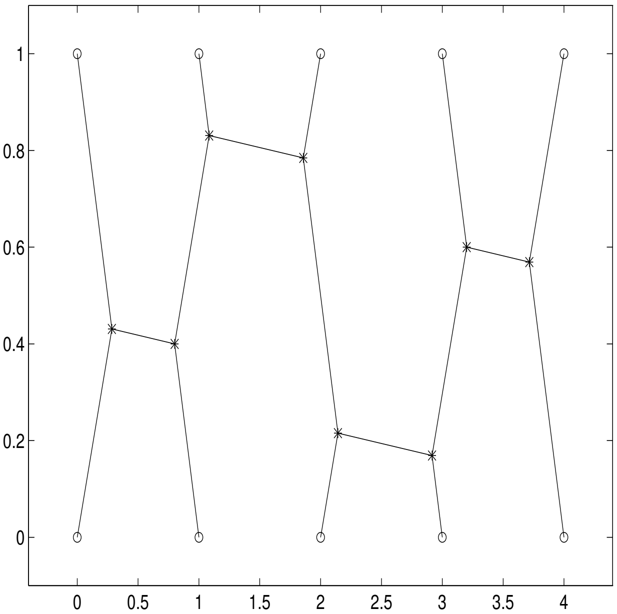

Figure 7: Steiner tree for “ladder” case

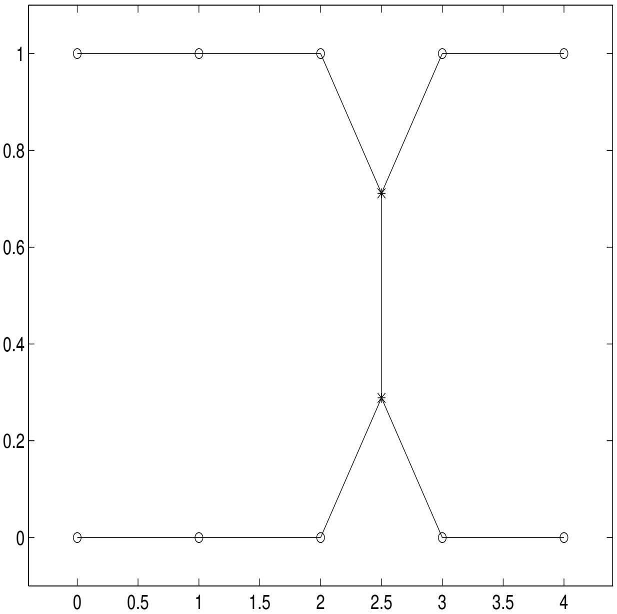

Figure 8: SI heuristic tree for “ladder” case

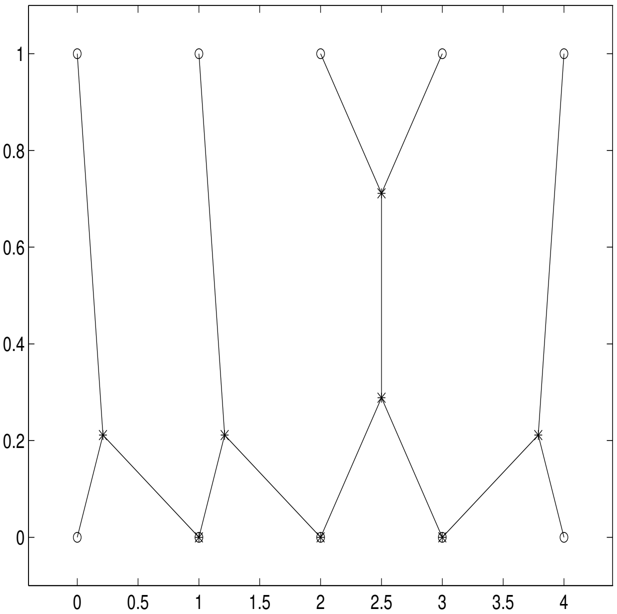

Figure 9: IO heuristic tree for “ladder” case

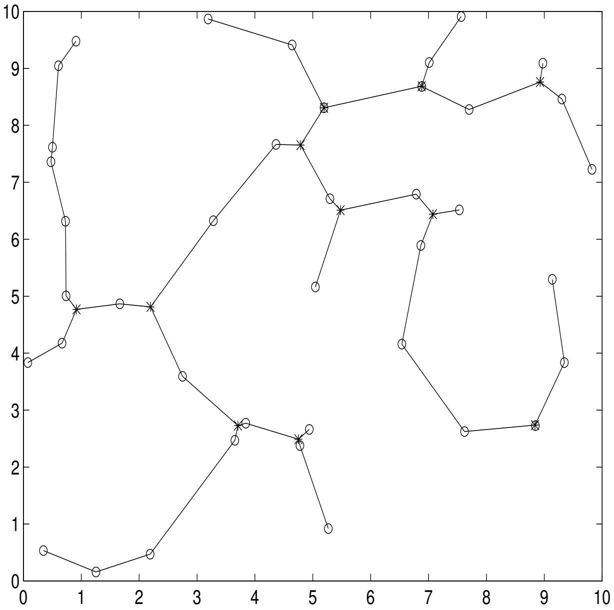

Figure 10: SI heuristic tree for last case

well, with 5.2%. The improvement of the exact Steiner tree is 7.3%.

In our last 40-point example, the point set is completely random within a square with side length 10. As reflected here, our experience with these completely random cases is that the IO heuristic doesn't improve much (if any) on the SI, and thus these are the best kinds of problems for the latter. In addition, at least for many small cases, the exact Steiner tree is the same as or not much better than the SI tree. The reason for this seems to be that any tree that connects a set of closely spaced points covering an entire region simply can't be that much shorter than the minimal spanning tree, and thus the small improvement of the SI tree is all that can be expected. Figure 10 shows the SI tree for this example.

6 Concluding Remarks

We would first like to make special note of our major reference throughout our work on Steiner trees, the monograph on the subject by Hwang, Richards, and Winter [HRW92]. The "Heuristics" chapter of this book contains many suggestions for possible heuristics, one of which we actually used for our Steiner insertion algorithm. The difficulty we found with some of them, however, is that they were either too vague in description or we could not see how to integrate them with our local optimization routine, an essential part of both of our heuristics. Nevertheless, we found the book to be a vital resource.

We would also like to thank Knud Andersen for the use of his local optimization code and Guo-Liang Xue for the use of his branch-and-bound code. The second author also thanks Bill Pulleyblank for several helpful conversations. In conclusion, we hope that these new heuristics and the knowledge gained from them will prove useful to those studying Steiner trees and may be applicable to more concrete problems in the near future.

References

[AA96] E.D. Andersen and K.D. Andersen. *APOS user's manual for QMSN problems ver 1.71.* EKA consulting, June 1996.

[CG78] F.R.K. Chung and R.L. Graham. Steiner trees for ladders. In B. Alspach, P. Hell, and D.J. Miller, editors, *Annals of Discrete Mathematics 2*, pages 173–200, Amsterdam, 1978. North-Holland.

[CO94] A.R. Conn and M.L. Overton. A primal-dual interior point method for minimizing a sum of Euclidean vector norms, July 1994. Draft copy of incomplete manuscript.

[GGJ77] M.R. Garey, R.L. Graham, and D.S. Johnson. The complexity of computing steiner minimal trees. *SIAM Journal on Applied Mathematics*, 32:835–859, 1977.

[GP68] E.N. Gilbert and H.O. Pollak. Steiner minimal trees. *SIAM Journal on Applied Mathematics*, 16:1–29, 1968.

[HRW92] F.K. Hwang, D.S. Richards, and P. Winter. *The Steiner Tree Problem.* Annals of Discrete Mathematics. Elsevier Science Publishers B.V., Amsterdam, 1992.

[Mel61] Z.A. Melzak. On the problem of Steiner. *Canadian Mathematical Bulletin*, 4:143–148, 1961.