Using Partitioning and Clustering Techniques to Generate Rectilinear Steiner Trees

Extended Abstract

Linda L. Deneen
Jeanne B. Dezell

Department of Computer Science
University of Minnesota, Duluth
Duluth, Minnesota 55812

Keywords: Steiner tree, rectilinear Steiner tree, clustering, computational geometry, minimal spanning tree.

Abstract

We present several new heuristic algorithms for computing rectilinear Steiner trees based on partitioning and clustering. Using partitioning and clustering techniques, we divide the set of sites into subsets, compute the minimal rectilinear Steiner tree on each subset using an optimal algorithm, and then combine the solutions to obtain a rectilinear Steiner tree on the original set of sites. This work can be viewed as an extension of the work of Komlos and Shing [10]. We are able to show significant empirical and theoretical improvements on their work while retaining comparable run times and probabilistic results on the length of the solution.

1 Introduction

Given a set of n points in the plane, called *sites*, a minimal spanning tree on the sites is a minimal-length tree whose vertices are the sites. The length of a minimal spanning tree on a set of sites can usually be reduced by adding extra vertices, called *Steiner points*, to the set of sites and recomputing the edges. A tree with Steiner points is called a Steiner tree, and the problem of finding the minimal-length Steiner tree is called the Steiner problem, after the nineteenth century geometer, Jacob Steiner. The difficulty of the Steiner problem lies in determining where to place the Steiner points. This difficulty is so great as to make the problem NP-complete [6].

A special case of the Steiner problem is the *rectilinear Steiner problem*, in which the rectilinear, or Manhattan, metric is used in place of the usual Euclidean metric to determine the length of the edges. The rectilinear Steiner problem has particular application to VLSI design, where a set of terminals is to be connected by wires running along channels that are either vertical or horizontal. The rectilinear distance between two points is equal to the length of an edge between them consisting of segments that are either vertical or horizontal. The rectilinear Steiner problem is also NP-complete [7].

Algorithms for solving Steiner problems are divided into two categories: optimal algorithms that are guaranteed to produce minimal trees with exponential run times and heuristic algorithms with faster run times that are not guaranteed to produce the minimal

tree. A number of optimal algorithms exist, including Dreyfus and Wagner's divide-and-conquer algorithm [3] and Yang and Wing's branch-and-bound algorithm [18]. Heuristic algorithms are more numerous and include [1, 2, 8, 10, 13, 14, 17].

In this paper we pay particular attention to Komlos and Shing's work [10], in which they use two partitioning schemes to divide up the set of sites, use an optimal algorithm to find the minimal rectilinear Steiner tree on each partition, and then paste the results back together to get a rectilinear Steiner tree on the original set. In Section 2 we describe an improvement on their first algorithm made by changing the method of pasting the subsolutions back together. In Section 3 we describe some new algorithms that use clustering techniques to partition the sites. Finally, in Section 4 we give some empirical results.

2 An Improvement on Komlos and Shing's Algorithm

Komlos and Shing [10] proposed two algorithms for solving the rectilinear Steiner problem based on the probabilistic approach of Karp [9]. The basic approach for the first algorithm is this: divide the set of sites up into small partitions, run an optimal algorithm on each partition, take the union of the resulting Steiner trees, run a minimal spanning tree algorithm on the union to find a Steiner tree on the original set of sites, and finally remove any Steiner points that are leaves in the tree. The second algorithm is similar, except that it uses a bucketing technique to partition the points, and the union of the subtrees produced is itself a Steiner tree, so there is no need to run the minimal spanning tree algorithm. Their algorithms exhibit a trade-off between time and optimality of the solution: the larger the partitions, the more time needed to compute the subtrees, and the better the solution. Their first algorithm runs in time $O(f(t)n + n \log n)$, where t is the partition size, n is the number of sites, and $f(t)$ is an exponential function related to the run time of the optimal algorithm. The ratio of the length of the solution to the length of the optimal solution is bounded above by $1 + O(\sqrt{1/t})$ with probability approaching 1 as n approaches infinity.

We propose a simple improvement of their first algorithm. Rather than using the union of the subtrees obtained from the optimal algorithm as the graph on which the minimal spanning tree algorithm is run, we throw away all of the edges produced on the partitions and simply keep the Steiner points and the original sites. We then compute the Delaunay triangulation on this enhanced set of points, which can be done in $O(n \log n)$ time [11, 15, 16]. Since the Delaunay triangulation is guaranteed to contain a minimal spanning tree on a set of points [12, 15], completing the algorithm with this graph instead of the union of the subtrees is guaranteed to produce a solution at least as short as the solution produced by Komlos and Shing's algorithm. Preliminary empirical test results given in Section 4 indicate that this approach yields a significant length improvement over Komlos and Shing's algorithm. The additional step of computing the Delaunay triangulation does not change the theoretical run time.

3 Variations Using Clustering Techniques

Komlos and Shing's first algorithm [10] partitions the set of sites by dividing the set in half iteratively, alternating between vertical and horizontal divisions. We propose a class of new algorithms based on our improvement of the Komlos and Shing algorithm, where we use clustering techniques instead of simple subdivision to find the subsets on which we will run the optimal algorithm. That is, we divide the set of points into clusters of size

t, run an optimal algorithm on each cluster to get a set of Steiner subtrees, throw away the edges of the subtrees but retain the sites and Steiner points, compute the Delaunay triangulation on the union of these sets, find a minimal spanning tree on the graph, and remove any Steiner points that are leaves. The theoretical run time of this class of algorithms is $O(f(t)n + n\log n + c(n))$, where $f(t)$ is the function in Komlos and Shing's run time, and $c(n)$ is the run time for the clustering algorithm.

The fundamental idea in this approach is that if we can find a good set of Steiner points, then the minimal spanning tree on the Delaunay triangulation of the set of sites and Steiner points will be a good Steiner tree. We wanted to see whether by varying the subsets sent to the optimal algorithm we could find better sets of Steiner points. We devised two clustering algorithms based on the graph theoretic approach described in [4, 5]. We omit a detailed discussion of these algorithms in this extended abstract. The first algorithm tends to give very tight clusters, whereas the second algorithm gives algorithms that are quite spread out. The empirical tests given in the next section indicate that these two clustering techniques show no improvement over the simple partitioning scheme of Komlos and Shing. Further investigation of this class of algorithms is underway.

4 Empirical Results

The table below gives some preliminary empirical comparisons of our algorithms with Komlos and Shing's first algorithm [10]. All algorithms were implemented in C and tested on a Sun 4C. All implementations use Dreyfus and Wagner's algorithm [3] as the optimal algorithm run on the subsets. Length comparisons are given as a percentage improvement over the length of the rectilinear minimal spanning tree $RMST(S)$ as given by the formula

$$\frac{RMST(S) - Approx(S)}{RMST(S)} \times 100\%.$$

Each number represents the average over 50 trials where the partition size is approximately 8. Sites are generated randomly on the vertices of a square grid whose size is given in the column titled "Range." KS is the original Komlos and Shing algorithm; KS-DT is the improved Komlos and Shing algorithm that uses the Delaunay triangulation; Cluster1 and Cluster 2 are our algorithms based on the clustering techniques. Run times are not greatly different and are omitted from this extended abstract. It is interesting to note that the original Komlos and Shing algorithm produces trees that are longer that the minimal spanning tree for the tests we have run. Although this algorithm is asymptotically optimal as the partition size increases, its practical use is questionable, because the run time is exponential in the partition size.

Algorithm Comparisons

Points	Range	KS	KS-DT	Cluster1	Cluster2
100	1000	-1.63	6.82	4.23	4.47
500	5000	-3.05	6.58	4.20	3.92

References

[1] M. W. Bern. Two probabilistic results on rectilinear Steiner trees. *Algorithmica*, 3:191–204, 1988.

[2] L. L. Deneen and G. M. Shute. A plane-sweep algorithm for rectilinear Steiner trees with deferred connections. unpublished manuscript, 1989.

[3] S. E. Dreyfus and R. A. Wagner. The Steiner problem in graphs. *Networks*, 1:195–207, 1972.

[4] R. C. Dubes and A. K. Jain. *Algorithms for Clustering Data.* Prentice Hall, Englewood Cliffs, 1988.

[5] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* John Wiley and Sons, Inc., New York, 1973.

[6] M. R. Garey, R. L. Graham, and D. S. Johnson. The complexity of computing Steiner minimal trees. *SIAM Journal of Applied Mathematics*, 32(4):835–859, 1977.

[7] M. R. Garey and D. S. Johnson. The rectilinear Steiner tree problem is NP-complete. *SIAM Journal of Applied Mathematics*, 32(4):826–834, June 1977.

[8] F. K. Hwang. An $O(n \log n)$ algorithm for suboptimal rectilinear Steiner trees. *IEEE Transactions on Circuits and Systems*, CAS–26(1):75–77, January 1979.

[9] R. M. Karp. Probabilistic analysis of partitioning algorithms for the traveling-salesman problem in the plane. *Mathematical Operations Research*, 2:209–224, 1977.

[10] J. Komlos and M. T. Shing. Probabilistic partitioning algorithms for the rectilinear Steiner tree problem. *Networks*, 15:413–423, 1985.

[11] D. T. Lee and C. K. Wong. Voronoi diagrams in L_1 and L_∞ metrics with 2-dimensional storage applications. *SIAM Journal of Computing*, 9(1):200–211, February 1980.

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

[13] D. Richards. Fast heuristic algorithms for rectilinear Steiner trees. *Algorithmica*, 4:191–207, 1989.

[14] M. Servit. Heuristic algorithms for rectilinear Steiner trees. *Digital Processes*, 7:21–32, 1981.

[15] M. I. Shamos. Computational geometry. Ph. D. Thesis, Yale University, Department of Computer Science, 1978.

[16] G. M. Shute, L. L. Deneen, and C. D. Thomborson. An $O(n \log n)$ plane-sweep algorithm for L_1 and L_∞ Delaunay triangulations. To appear in *Algorithms*, 1987.

[17] J. M. Smith, D. T. Lee, and J. S. Liebman. An $O(n \log n)$ heuristic algorithm for the rectilinear Steiner minimal tree problem. *Engineering Optimization*, 4:179–192, 1980.

[18] Y. Y. Yang and O. Wing. Optimal and suboptimal solution algorithms for the wiring problem. In *IEEE International Symposium on Circuit Theory*, pages 154–158, 1972.