The Construction of a Minimal-Cost Steiner Tree

Patrick J. Melody
Hall L. Moore III
Mary Wood
Mount St. Mary's College
Emmitsburg, MD 21727

Advisor: F.J. Portier

Summary

We develop two separate heuristics designed to find a minimal tree connecting an arbitrary number of points in rectilinear space, using Steiner nodes so as to minimize cost. We use a heuristic approach to the problem rather than an algorithmic approach, because the problem is NP-complete; an algorithmic approach is too time-consuming to be practical.

In the first part of the problem, cost is based exclusively on the length of the tree. From an analysis of a minimum spanning tree using rectilinear distances (an algorithmic process), we can determine a lower limit on the length of the best possible rectilinear Steiner tree. Therefore, we can compare our heuristic results to this limit. We also analyze another heuristic, developed by Hanan, for comparison purposes.

Then, for the second part of the problem, we adapt one of our heuristics to take into account the cost of the stations connected by the tree, and compare the results to the station-weighted costs of the previous three heuristics.

The third part of the problem asks us to attempt to generalize this problem. This we do by confining our work to methods applicable to an arbitrary point set. As a demonstration of this, we test all of our methods on four separate point sets that we generated.

We perform error analysis and discuss the models strengths and weaknesses.

Assumptions

- There is no limit on the number of Steiner nodes that may be added, nor is there any limit on the total length of the lines used.
- In our preliminary analysis, there is no cost associated with adding more Steiner nodes.

- All stations can be represented by zero-dimensional points, with no width or height.
- Corner turns do not require Steiner nodes.
- Steiner stations cost the same amount as original stations.

Analysis of Problem

Our task is to design a method that, given a set of vertices Z, produces a minimum-cost rectilinear tree spanning Z by incorporating a set of Steiner nodes S into the tree. The rectilinear Steiner problem (RSP) as presented is a special case of the more general Euclidean Steiner problem. In RSP, a grid may be generated by plotting a horizontal and vertical line through each vertex in Z. The intersection points of these lines (grid points) are the positions where points in S may lie [Bern and Graham 1989]. Furthermore, the edges of the final graph must lie along these horizontal and vertical lines. Since all of the given vertices have integer coordinates, this restriction also satisfies the requirement that Steiner nodes must lie at lattice points.

There are wo main differences between rectilinear and Euclidean space. The first is that the distance formula $d = |x_1 - x_2| + |y_1 - y_2|$ is used rather than the Euclidean distance formula $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. The other difference is the fact that in rectilinear space there are quite often two or more paths of equal length between any two points, while there is only one shortest path in Euclidean space.

There is an immediate obstacle in attempting a Steiner problem: the general problem has been shown to be NP-complete [Bern and Graham 1989]. This means that there is no known algorithm that gives the solution in polynomial time in the problem size (the number of vertices), nor are the prospects of finding such an algorithm at all good. For Steiner problems with few vertices, non-polynomial-time algorithms are practical; but as the vertex set increases in size, the time required to find an optimal solution explodes. So, for the general case, which we are asked to examine, algorithms that find optimal solutions are not practical. However, it is possible to find approximations of the optimal solution in polynomial time; such methods are feasible for the general case as well as for the nine-point case we are given.

With this background in mind, we have chosen a heuristic strategy to solve the problem: we use rules to generate a Steiner tree that, while not necessarily the best tree, is better than many.

We first look at algorithms for minimal spanning trees in Euclidean space. Two of the most common ones are *Kruskal's algorithm* and *Prim's algorithm*. In Kruskal's method:

The nodes of the graph are initially considered as n distinct partial trees with one node each. At each step of the algorithm, two distinct partial trees are connected into a single partial tree by an edge of the graph. When only one partial tree exists (after $n-1$ such steps), it is a minimum spanning tree.

[Tenenbaum and Augenstein 1981, 642]

The decision of which connecting arc to use next is determined by using the arc of minimum distance that connects two distinct trees.

Prim's solution for a minimal spanning tree is a greedy algorithm; it chooses an arbitrary node to begin with and connects to the next closest node. Successive nodes are added if they are the closest to a node already in the tree, until all nodes are used up [McEliece et al. 1989, 112–113].

To use the algorithms of Kruskal and Prim to attempt a solution to the RSP, we must adapt them to take into account the differences between rectilinear and Euclidean space.

Design of the Model

One heuristic, proposed by Hanan, is specifically designed to take rectilinearity and the addition of Steiner nodes into account:

Z-vertices are ordered by increasing x-coordinate. Begin with the two Z-vertices having the smallest x-coordinate, (say, $i_1 = (x_1, y_1)$ and $i_2 = (x_2, y_2)$). Connect i_1 and i_2 by a minimum cost path in $G(Z)$. There are usually several such paths. Select the one having the largest portion coinciding with the line $x = x_2$. Suppose now that a tree T_k containing k Z-vertices with smallest x-coordinates has been constructed. $i_{k+1} = (x_{k+1}, y_{k+1})$ is added to T_k in the following manner. Select a vertex q in T_k (not necessarily a Z-vertex) closest to i_{k+1}. Connect i_{k+1} with q by a minimum cost path having the largest portion coinciding with the line $x = x_{k+1}$. The heuristic terminates as soon as T spans all Z-vertices.

[Winter 1987, 156]

Hanan's model can be directly applied to the problem. However, we found that modifications of Prim's method and Kruskal's method both created lower-cost minimal spanning trees than Hanan's heuristic.

In our modified Prim heuristic (MPH), we first calculated the distances between each pair of nodes and chose the pair with least distance between them: (x_i, y_i) and (x_j, y_j). If there existed more than one such pair, then the choice was made arbitrarily. If there was only one such shortest path available, then this "path" would be chosen. Otherwise, two paths were considered, the one containing (x_i, y_j) and the one containing (x_j, y_i). The choice of which path to take was made based on an analysis of the cost benefits associated with each path. At times, however, path choice was constrained by pre-existing tree branches. The cost was determined in the preliminary analysis by measuring the length of the paths. In the full analysis, the cost was determined by the combined length of the communication line and the cost of the stations created by the path. Once the path of least cost was chosen, this path became the current tree. Next, the point which was closest to any point in the tree was located, and the path of least cost from this point to the tree was found in the same manner as before. These last two steps were repeated until all of the original nodes were contained in the tree.

The modified Kruskal heuristic (MKH) began in the same way as the MPH, by choosing the two nodes which had the least distance between them, and determining the path to be taken by the aforementioned cost criterion. After the first path was chosen, however, the method differs in that it next analyzed the two nodes which had the next least distance between them, rather than the single node that was closest to any point already in the tree. If connecting a path between these two created a cycle, the pair of points was rejected, and the two nodes that had the next least distance between them were considered. In this way, fragmented subtrees were created, rather than one root tree that branched out (as in the MPH), until all nodes were connected in one main tree.

Appendix A contains definitions of terminology and precise formulations of the algorithms.

Testing of Model and Error Analysis

We used the given data to test the modified Prim heuristic, the modified Kruskal heuristic, and Hanan's heuristic against each other, and compared their results with the length of a minimal spanning tree without Steiner points. For the first part of the problem, which defined cost solely by the length of the communication lines, the MPH created the shortest tree of the three heuristics, with a length of 94 units. For the second part, in which cost includes both length and cost of stations, the MPH also produced the lowest-cost tree, with a cost of 134.8.

To further test our model, we created four sets (with 5, 9, 14, and 20 points) of "dummy" data points. Results for them also favored the MPH. The MKH did not produce trees with cost as low as the MPH trees, but the differences were in many instances small. Therefore, the MKH proved to be a feasible heuristic as well. The Hanan heuristic did not do very well at all, though it could be used to generate approximations of trees. The only advantage the Hanan heuristic has over the MPH and the MKH is that it can be converted into a computer program more easily.

In a rectilinear network, the use of Steiner points cannot reduce the length of the network's minimum spanning tree by more than one-third [Bern and

Graham 1989]. Therefore, from the length of the network's minimum spanning tree, we can find a lower limit for the length of its optimal Steiner tree, thus giving a criterion for judging our heuristics.

Appendix B contains a table of the data produced by the various test models and node sets. We also include there a chart comparing a minimal spanning tree without Steiner nodes with our MPH tree results. The corresponding trees for all the data sets and heuristics are in Appendix C.

Strengths and Weaknesses

The major weakness of our model is that it is not an algorithmic one, in the sense of being guaranteed to give the optimal Steiner tree. However, it is stronger than an algorithmic solution, in the sense that a good answer can be found within a reasonable amount of time, whereas—due to the NP-completeness of the problem—the best possible solution could take an impractical length of time to discover, particularly for large vertex sets.

In testing a heuristic, it is useful to compare its performance with other heuristics, which we have done. Although the modified Prim heuristic consistently yielded results better than the others, this does not mean that the other heuristics should be disregarded, as they may work well for other cases we have not tried them on.

Appendix A

Definitions

Z = set of all nodes originally given

G = set of all grid intersection points

T = set of all nodes contained in the tree at the current time

$S = G - Z$

P = any x in T such that $\deg(x) > 2$ and x in S

distance = $|x_1 - x_2| + |y_1 - y_2|$

cycle = path whose starting node and finishing node are the same node

Modified Prim Heuristic

1. Find the two nodes z_i, z_j in Z, with coordinates (x_i, y_i) and (x_j, y_j), which have the least distance between them. If there is more than one such pair, choose arbitrarily.

2. If there is one path between them, then connect them, adding z_i, z_j, and all g in G along the path to T; else

   (a) Construct path$_1$ that goes through (x_i, y_j), thus adding z_i, z_j, and all g in G along path$_1$.

   (b) Find z in Z but not in T which is the closest to the tree (dist1), then delete path$_1$.

   (c) Construct path$_2$ through (x_j, y_i), adding z_i, z_j, and all g in G along path$_2$.

   (d) Find z in Z but not in T which is closest to the tree (dist2), then delete path$_2$.

   (e) If dist1 < dist2, then add path$_1$; if dist2 < dist1, then add path$_2$; if dist1 = dist2, then reiterate 2(a)–2(e) for next closest point until get preference or else run out of nodes (in which case, choose arbitrarily).

3. Find $z_i \in Z \cap (G - T)$ and z_j in T such that z_i and z_j as close as possible.

4. Reiterate 2 until all nodes of Z are in T.

Modified Kruskal Heuristic

1. Find the two nodes z_i, z_j in Z, with coordinates (x_i, y_i) and (x_j, y_j), which have the least distance between them. If there is more than one such pair, choose arbitrarily.

2. If one path connects them and does not create a cycle, then connect them; else

   (a) Construct path$_1$ that goes through (x_i, y_i), thus adding z_i, z_j, and all g in G along path$_1$.

   (b) If path$_1$ creates a cycle, go to 2(d).

   (c) Find z in Z but not in T which is the closest to the tree (dist1).

   (d) Delete path$_1$ and construct path$_2$ through (x_j, y_i), adding z_i, z_j, and all g in G along path$_2$.

   (e) If path$_2$ creates a cycle, go to 2(g).

   (f) Find z in Z but not in T which is closest to the tree (dist2).

   (g) Delete path$_2$.

(h) If path_1 creates a cycle and path_2 creates a cycle, then reject z_i, z_j, in $Z \cup T$ and find the next two nodes z_i, z_j in $Z \cup T$ which have the next least distance between them, and go to 2.

(i) If path_1 does not create a cycle and dist1 < dist2, then add path_1; if dist2 < dist1, then add path_2; if dist1 = dist2, then reiterate 2(a)–2(e) for the next closest point, until dist1 $\neq$ dist2 or else run out of nodes (in which case, choose arbitrarily).

3. Find the next two nodes z_i, z_j in $Z \cup T$ which have the least distance between them.

4. Repeat steps 2 and 3 until all nodes of Z are in T.

Appendix B

Table 1.

Results of applying the heuristics to the original data set and the additional "dummy" data sets.

Data set	Hanan	Length-weighted modified Kruskal	Length-weighted modified Prim	Node-weighted modified Prim
		Length Cost		
Original (9 nodes)	98	94	94	104
Dummy set 1 (9 nodes)	79	79	79	82
Dummy set 2 (14 nodes)	106	102	97	107
Dummy set 3 (5 nodes)	63	60	60	64
Dummy set 4 (20 nodes)	132	125	121	140
		Length Cost + Station Cost		
Original (9 nodes)	144.365	136.324	136.324	134.847
Dummy set 1 (9 nodes)	117.283	117.283	117.282	116.880
Dummy set 2 (14 nodes)	172.383	169.377	164.377	156.635
Dummy set 3 (5 nodes)	87.706	80.665	80.664	80.624
Dummy set 4 (20 nodes)	239.948	232.948	232.989	212.871

Table 2.

Lengths of Steiner trees by Prim's heuristic, on the original given data set and on the authors' "dummy" data sets.

Data set	Minimal spanning tree with no Steiner points	Steiner tree by Prim's heuristic	Percentage decrease
Original data (9 nodes)	110	94	14.5
Dummy set 1 (9 nodes)	87	79	9.2
Dummy set 2 (14 nodes)	112	97	13.4
Dummy set 3 (5 nodes)	68	60	11.8
Dummy set 4 (20 nodes)	147	121	17.7

Appendix C

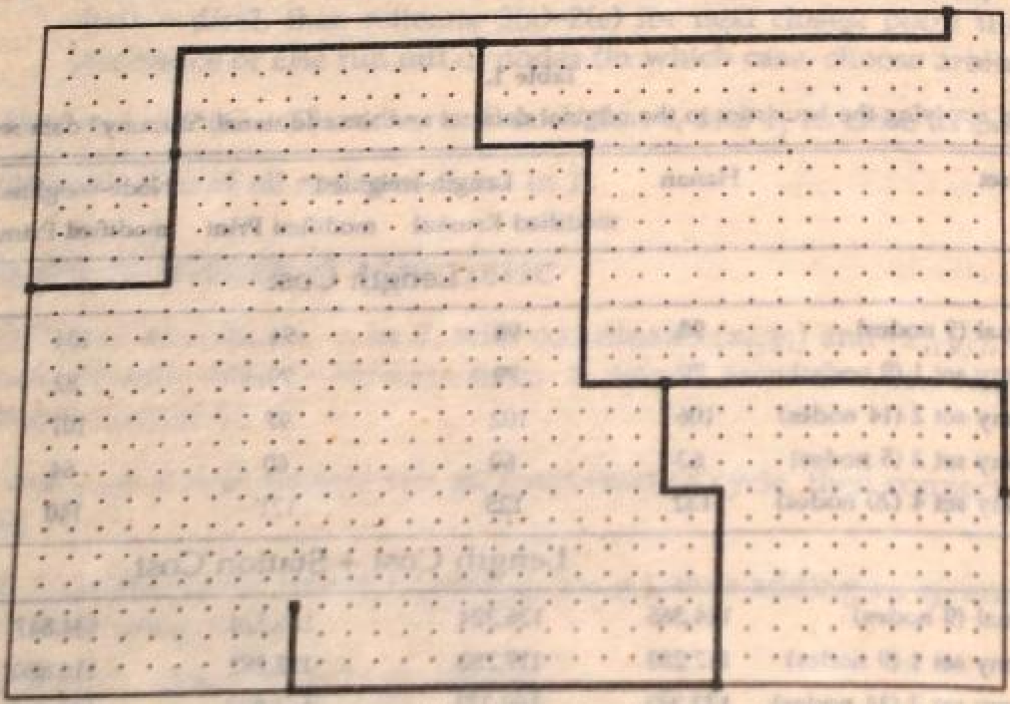

Figure 1. Original data set: minimal spanning tree without Steiner points.

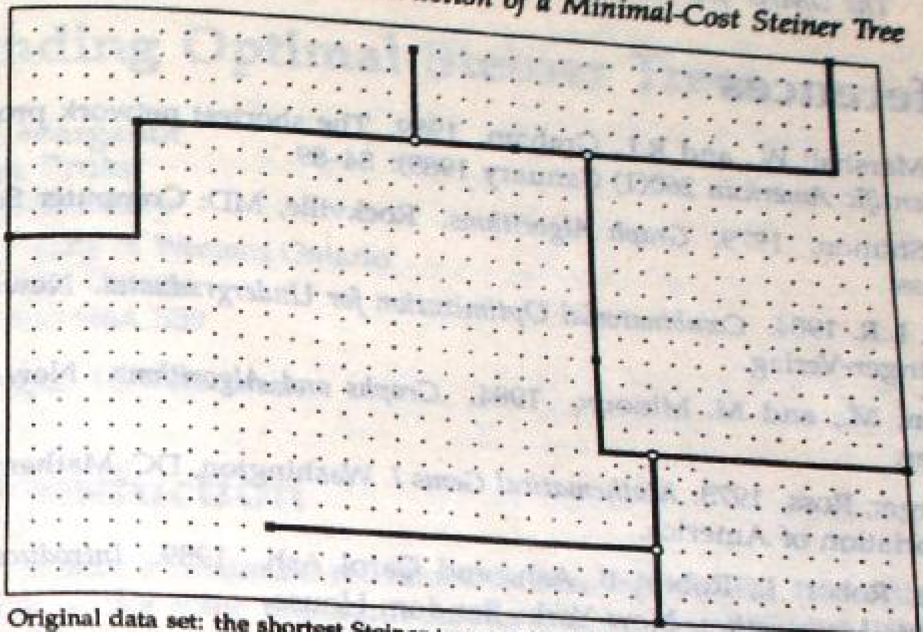

Figure 2. Original data set: the shortest Steiner tree produced by the heuristics for the case of costless Steiner points—produced by modified Kruskal heuristic.

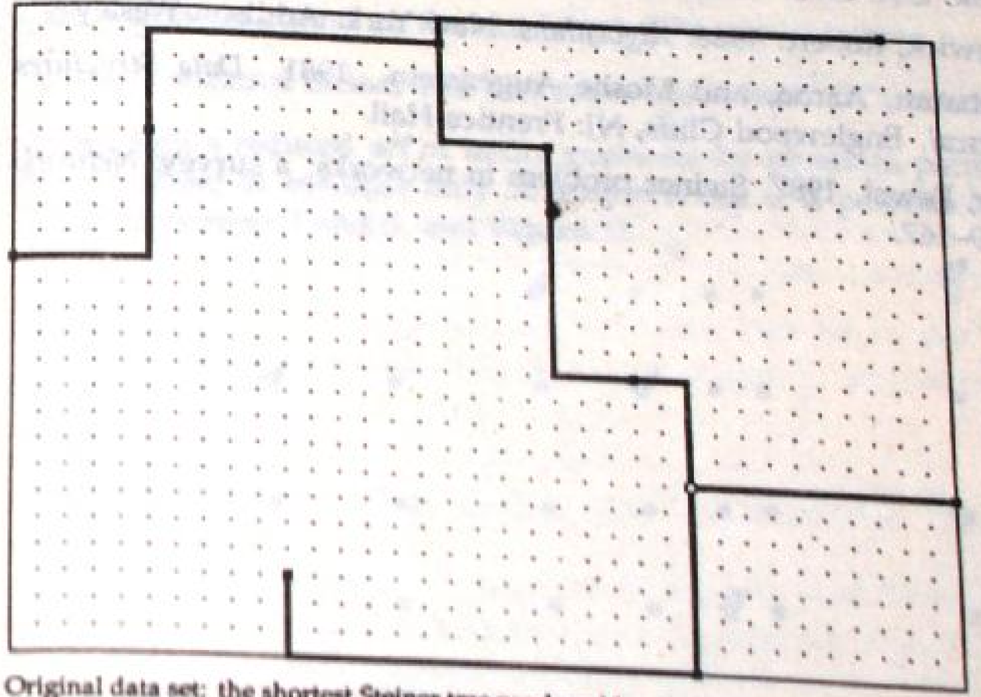

Figure 3. Original data set: the shortest Steiner tree produced by the heuristics when the cost of Steiner points is included—produced by modified Prim heuristic.

[EDITOR'S NOTE: Twenty-two other figures, including those for the authors' "dummy" data sets, are omitted for space reasons.]

References

Bern, Marshall W., and R.L. Graham. 1989. The shortest network problem. *Scientific American* 260(1) (January 1989): 84–89.

Even, Shimon. 1979. *Graph Algorithms*. Rockville, MD: Computer Science Press.

Foulds, L.R. 1984. *Combinatorial Optimization for Undergraduates*. New York: Springer-Verlag.

Gondran, M., and M. Minoux. 1984. *Graphs and Algorithms*. New York: Wiley.

Honsberger, Ross. 1973. *Mathematical Gems I*. Washington, DC: Mathematical Association of America.

McEliece, Robert J., Robert B. Ash, and Carol Ash. 1989. *Introduction to Discrete Mathematics*. New York: Random House.

Melzak, Z.A. 1973. *Companion to Concrete Mathematics*. New York: Wiley.

Sedgewick, Robert. 1988. *Algorithms*. New York: Addison-Wesley.

Tenenbaum, Aaron, and Moshe Augenstein. 1981. *Data Structures Using Pascal*. Englewood Cliffs, NJ: Prentice-Hall.

Winter, Pawel. 1987. Steiner problem in networks: a survey. *Networks* 17(2): 129–167.