Iterative and Constructive Models for Minimal Rectilinear Steiner Trees

Timothy McGrath
Monica Menzies
Christopher Smith
Beloit College
700 College St.
Beloit, WI 53511–5595

Advisor: Philip D. Straffin, Jr.

Introduction

Using the two major common approaches for solving the minimal rectilinear Steiner tree problem, we can calculate the set of optimal solutions in reasonable time for a small number n of vertices. Our iterative model uses a greedy algorithm and takes advantage of knowledge that limits the potential locations of Steiner points. Our constructive model calculates all possible Steiner tree solutions for an increasing number S of Steiner points, until we are unable to reduce the cost further. This model is a refined and improved version of an exhaustive search.

For the given dataset, both models generate the minimal rectilinear Steiner trees. For Part 1 of the problem, they reduce edge lengths from 110 (with $S = 0$) to 94 (with $S = 4$) (see Figure 1).

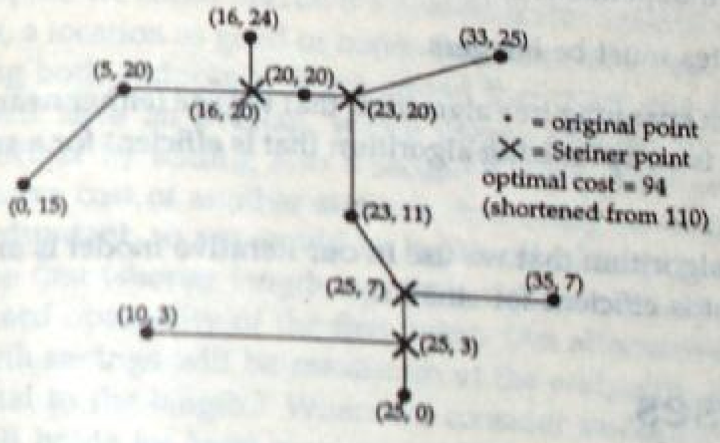


Figure 1. Optimal placement of Steiner points to minimize total length of edges.

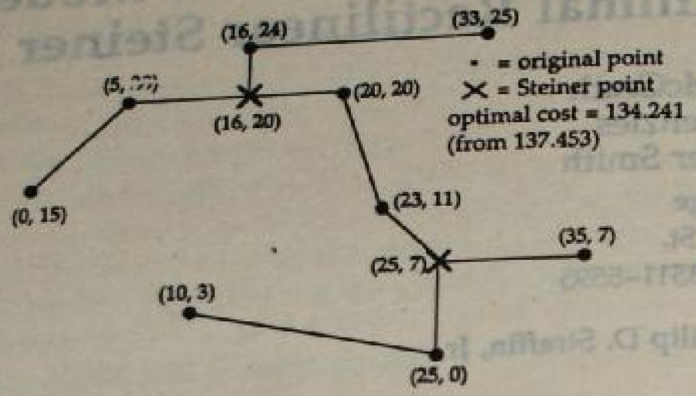


Figure 2. Optimal placement of Steiner points to minimize total cost, including cost of Steiner points.

For Part 2, they reduce total cost (including cost of Steiner points) from 137.453 (with $S = 0$) to 134.241 (with $S = 2$) (see **Figure 2**).

We generalize to handle changes in geometry, cost, dimension, and input, as well as some combinations of these. Then we discuss applications to related problems.

Assumptions

- The w of the station cost is in units of length.
- In analyzing the sensitivity of our model, we assume that the effects of changes are independent of each other.
- All coordinates must be integers.
- The minimum spanning tree algorithm that we use (either nearest-neighbor or Kruskal's) is an optimizing algorithm that is efficient for a small number n of vertices.
- The ranking algorithm that we use in our iterative model is an optimizing algorithm that is efficient for small n.

Hypotheses

We thought of the length of the line connecting two points as the length travelled in either direction along the rectilinear hull formed by those two points. When we drew the edges of the pairwise rectilinear hulls for the

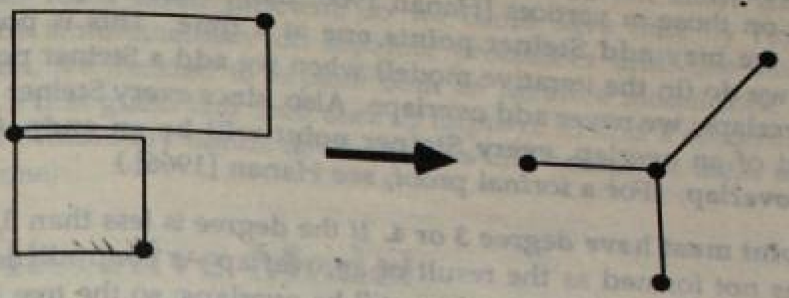


Figure 3. How to eliminate an overlap of pairwise rectilinear hulls.

given dataset, we realized that the only way to save on total edge length would be to shorten at least one *overlap* of these hulls (see Figure 3).

- **If there is overlap, we do not have minimal length.** Adding a Steiner point at one end of the overlap or the other will reduce the overall length.
- **Each Steiner point will be on an original overlap,** the rectilinear hull of a pair of original points. This is the translation, in terms of overlaps, of the second lemma below. If we went beyond the end of an overlap, we would be creating additional overlap without incurring extra savings.
- **Each Steiner point will be an *endpoint* of an overlap.** Consider a single overlap. Suppose we could find on it a non-endpoint optimal location for a Steiner point, a location as good or better than any other point on the overlap, including both endpoints. If we added that point as a Steiner point, we would still have an overlap, which means that we could minimize the length further by adding also a second Steiner point on the overlap (but at the extra cost of another station). But doing so would make our first point redundant, so we would be better off adding the second point instead of the first (shorter length, no extra station cost)—a contradiction of the supposed optimality of the first point. (An alternative argument is that the length savings will be maximum at the endpoint, as the savings is proportional to the length.) When we consider multiple overlaps, this argument still holds for local maxima; and therefore, over the longest of the multiple overlaps, one of the potential Steiner points at the end of some overlap (perhaps not the one we are examining) must be the global maximum (because any global maximum must be a local maximum). If

we examine all overlaps, then we must discover that the overall global maximum is at the endpoint of an overlap.

- For n points in the plane, let G be a minimal rectilinear Steiner tree. If G' is a connected subgraph of G with m vertices, then G' is a minimal rectilinear Steiner tree on those m vertices [Hanan 1966, 258]. This is equivalent to saying that **we may add Steiner points one at a time.** This is possible because all we do (in the iterative model) when we add a Steiner point is eliminate overlaps; we never add overlaps. Also, since every Steiner point is at the end of an overlap, **every Steiner point will be an endpoint of an *original* overlap.** (For a formal proof, see Hanan [1966].)

- **A Steiner point must have degree 3 or 4.** If the degree is less than 3, then the point was not formed as the result of an overlap—a contradiction. If the degree is greater than 4, there must still be overlaps; so the tree is not optimal, and this is a contradiction. (This argument also shows that no point in the minimal Steiner tree will have degree greater than four.)

- For Part 2, with added station (Steiner-point) costs: For a fixed number of Steiner points of fixed degree, we could minimize total cost by positioning the Steiner points so as to minimize total length. **For the specific given dataset and station cost, in order to be able to reduce cost by adding another Steiner point, the tree would need to have a vertex of degree at least five** (justified by the search conducted by the computer program in the appendix [EDITOR'S NOTE: Omitted here.]). In our solution, each point has degree at most four.

Lemmas

- The minimal rectilinear Steiner tree on n vertices has at most $n-2$ Steiner points [Hanan 1966, 258].
- A Steiner point must share its x-coordinate with one vertex in the original graph and also share its y-coordinate with another [Hanan 1966, 258].

The Two-Model Approach

According to Hanan, the two main heuristic approaches to this problem use either iterative algorithms or constructive algorithms. Although these approaches are for suboptimal solutions, we believe the same two approaches should be usable for optimal solutions. "Iterative algorithms seek to improve a solution by repeated modification of it" while "constructive algorithms ... sequentially build a solution." Hanan states that the current (as of 1975) literature showed that both iterative and constructive algorithms are better than manual efforts [Hanan 1975, 85–86].

As the two sorts of algorithms will not necessarily run in similar time, they could be helpful in different situations. We do not know which would be quicker for a given problem, so we could save time by running both algorithms simultaneously on separate processors until one finishes. For this reason, we decided to explore both an iterative model (one that would generate a first guess and then seek to improve it) and a constructive model (one that continually builds new solutions and compares them to determine the best one).

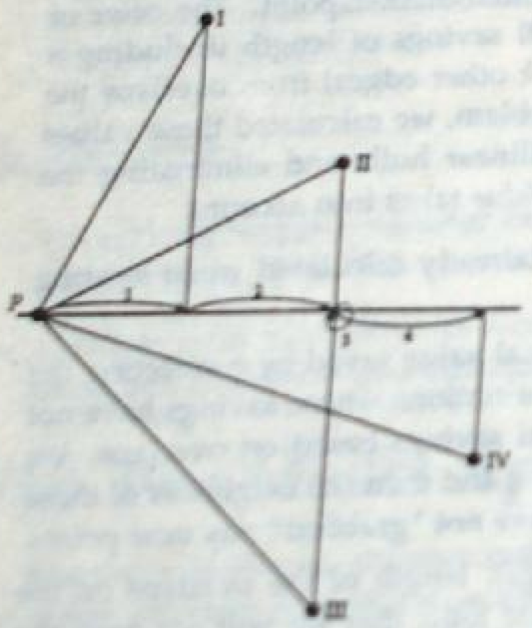
The Iterative Model

We used the following algorithm to compute the minimal rectilinear Steiner tree for the problem involving edge costs and vertex costs.

1. Input the coordinates of the vertices.
2. Compute the minimum rectilinear spanning tree (with no Steiner points) using a known algorithm. (We used Kruskal's greedy algorithm.)
3. Find overlaps and the possible Steiner points at their ends.
4. Calculate the length saved by each potential Steiner point. The *value* of a possible Steiner point is the sum of all savings of length (including a multiplier of k if an edge is covered by k other edges) from overlaps the point would eliminate. For the given problem, we calculated these values by hand, by drawing the pairwise rectilinear hulls and eliminating the overlaps that we found (see Figure 3). *Value* takes into account:
   - savings based on overlaps avoided (already calculated when locating the point).
   - "grabbed" point values: the additional value saved by connecting the possible Steiner point with any of the stations whose savings have not already been accounted for as initial savings based on overlaps. We look at all neighbors of the neighbors, and then the neighbors of those that we have "grabbed," until we have not "grabbed" any new points.
   - foregone savings (a negative savings): length of the overlaps on the other sides of the pairwise rectilinear hulls that we will not have the option of saving later and which we do not save now.
   - savings incurred because of the decreased degree of some neighbors or some of the "grabbed" points, or some of the neighbors of the "grabbed" points.
   - the degree cost of the possible Steiner point (a negative saving).

5. Add the Steiner point of greatest value, treat it as a given point, and treat the new tree as a rectilinear minimum spanning tree over all the stations (including the added Steiner points). Reiterate from Step 3 until there are no possible Steiner points with positive value.

For large n, solving a problem in this manner would be too awkward. Therefore, we have to be more descriptive to mechanize this procedure. First, it is necessary to rank the x- and y-coordinates into two separate arrays, where the array also contains information about which vertex the coordinate represents and which vertices are connected to it. For each vertex, check in the positive and negative x- and y-directions to determine how many vertices are located in that direction which are also connected to that vertex. Then go out to the first coordinate and give that intersection a multiplier value of k, as well as details about the overlap, including the length of the overlap and the name of the vertex that is located at the other end. Repeat this process until the end of the ranked list. Then step out to the next coordinate and give it information, being careful to decrease k by the number of coordinates passed unless another coordinate is the same as the coordinate in question (see Figure 4). Use the same value of k that would be associated with the first coordinate encountered in the ranked list which has the same coordinate as the coordinate in question. (Note: It may be possible to accomplish the same objective more efficiently for large n, but this is not our goal. We need only come up with an efficient algorithm for small n.)



After Step	Vertex	k-Value
1	I	3
2	II	2
3	III	2
4	IV	0

Figure 4. Details of associating a value saved with each Steiner point.

The Constructive Model

We considered brute force: try the first Steiner point at every vertex of the graph, checking each possibility for minimum cost of the tree; and add succeeding Steiner points in the same manner. With p vertices in the rectilinear hull of the original graph, we would need to try $\binom{p}{k}$ combinations of Steiner points, for $k = 1$ up to $n-2$ Steiner points.

We found this approach to be both cumbersome and unnecessary when we discovered that each Steiner point has to share coordinates with points in the original graph. The number of combinations to try drops to $\binom{n\times n}{k}$ for each value of k. This was a great improvement but still not practical.

Then we discovered that when a minimum-cost tree was found for k Steiner points, the $(k+1)$-point minimum Steiner tree must have the k-point Steiner tree as a subtree. This final observation (for which we later found a proof by Hanan [1966, 258]) led to our algorithm, which is efficient for small n.

We developed two computer programs to solve the problem using a constructive approach [Editor's Note: The computer programs are omitted.] The first program finds a minimal-length rectilinear Steiner tree on n vertices. The algorithm goes as follows:

1. Find the minimum spanning tree on those n vertices and store its cost.
2. Try a Steiner point at a possible vertex and find the minimum spanning tree for the new graph.
3. If this tree's cost is less than the minimum cost so far, save this tree as potentially optimal.
4. Repeat Steps 2 and 3 until either we reach the maximum number of Steiner points $(n-2)$, or until a new Steiner point is added which gives no savings.

The second program is a simple modification of the first. In Step 3, where we calculate the minimum cost, we take into account both the length cost and the node cost and minimize their sum.

Testing the Models

We will now test the sensitivity to the assumption that the coordinates must be integers. If they are not, then the calculations for rectilinear length-minimizing spanning trees can be off by at most $2n-2$: we calculate $n(n-1)/2$ edges but only use $n-1$ of them, and each one's maximum possible error is 2 (twice the lattice size), as a result of the way we calculate distances.

Also, the calculations for the rectilinear length-minimizing Steiner trees can be off by at most $4n$: we have n points, each of which can have an error in either the x- or y-direction, and the error in any particular direction can be at most 2 (twice the lattice size).

Therefore, the maximum possible error in savings is no greater than $6n-2$. In the given problem, the maximum possible error in savings is $6(9)-2=52$, which is 325% of our actual savings, a large error to arise from potential rounding errors. If the input values were different, we might be able to construct a very different Steiner tree with huge further savings (in fact, just moving one point to a diagonal neighbor could save up to 5 percentage points). For our problem, we are only guaranteed to have found a solution 70–75% of optimal. We could check just how much by executing the algorithm on all datasets whose x-coordinates and y-coordinates vary by 1 from those of the given one.

When station costs are added, the complexity of the problem is dramatically increased. A relatively efficient way to conduct error analysis is to determine the maximum possible error for a given configuration, as above. Fortunately, we also know that the maximum possible savings we can obtain by adding Steiner points in a rectilinear problem is 1/3 [Hwang 1976]. Therefore, in all cases, we can further limit the error to 33%.

Critical to finding an optimal solution is the assumption that the w in the station cost is in units of length, because we need to know how to compare and/or add edge costs and station costs.

The assumption that the algorithm for the minimum spanning tree is optimal and efficient is critical. If a subalgorithm is not necessarily optimal or efficient, then we cannot hope for the larger algorithm to be.

The assumption that the ranking algorithm is optimal and efficient is critical for the iterative model, for the same reason. The constructive model, however, does not use the ranking algorithm.

Generalizations and Applications

Geometry

Rectilinear distance is of special interest because of printed circuit technology, "where it is desired to electrically interconnect a set of points using the shortest possible total wire length with the wires restricted to run in just horizontal and vertical directions" [Hwang 1976, 104].

For a different metric, Steiner points need not be located at the endpoint of an overlap, so the modified exhaustive search would no longer be exhaustive. Every grid point between two points that share either an x- or a y-coordinate with other points must be checked (rather than just the possible Steiner points in the iterative model, and the trial Steiner locations in the constructive model).

Our model could accommodate rotations of the grid system or a change in the grid system. For example, we can obtain the standard Steiner solution for an equiangular triangular grid.

Cost

Rather than actual cost or distance, the quantity to be minimized may be time of travel or the amount of materials used for the construction of telephone lines, computer circuits, roads, etc. This change is easily handled by our model, because time and material may be made proportional to the length of the network that links the points. Thus, the cost function of our models is the only component that may need to be changed.

Clearly, we could accommodate changes in the cost function such as a different power for d, a different value of w, or a linear combination of powers of d.

Dimension

Other applications, such as mechanical and electrical systems in buildings, may require that we examine the three-dimensional version of the rectilinear Steiner problem. The constructive model may need to be changed, but the iterative model would only need to be modified slightly. The iterative model can handle higher dimensions through simple adaptations, basically by constructing the pairwise rectilinear hulls within n-dimensional space. (Dimensions zero (a point) and one (a segment) are both trivial.)

Input

The computer programs for the models accept, as input, planar points with integer coordinates; but our models could handle a wider range of input. The program for the iterative model can handle rational numbers without change, although some changes (in global constants and data types) would need to be made in the constructive model; in both cases, we use the common denominator of all the rational-number coordinates as the grid size. If the coordinates are multiples of integers by a real number, we first factor out this number from each coordinate.

Some Combinations of the Above

If we use the ordinary Euclidean metric and allow the initial stations and the "phantom" stations to be anywhere, we obtain a problem that has not been solved analytically. This problem is known as the *Steiner problem*: Link n points in the plane using the shortest path. A simple algorithm (Kruskal's greedy algorithm) will construct minimal spanning trees; but if Steiner points are allowed, no efficient algorithm exists for the construction of minimal Steiner trees to connect arbitrary points in the plane [Gardner 1986, 21–22].

Although our models cannot be modified to accommodate so general a problem (e.g., our models cannot handle arbitrary irrational coordinates), we are aware of an analog solution to this problem. We build a physical model consisting of two parallel plates of glass joined by perpendicular pins that correspond to the points to be spanned in a given network and dip the model into a soapy solution. The soap films that are formed link all of the pins; and because soap films form the (relative) minimum surface area for a given boundary, an approximate solution may be obtained rather quickly. This property is justified by Isenberg [1976], who uses an argument from thermodynamics. If the quantity that we wish to minimize can be made proportional to the surface area of a soap film in some physical model, then the equilibrium positions of the soap film will give the local minima solutions to the original problem, and one of these local minima will be the absolute minimum [Hoffman 1979, 378]. All of the minimum solutions must be determined in order to pick out the shortest one. The problem is that "there is no way to determine the total number of minimum configurations" once the number of points is large [Isenberg 1976, 515]. Nevertheless, the soap-bubble "computer" finds a good suboptimal solution to the Steiner problem.

Strengths and Weaknesses

Because both our models obtain the same solutions for the given dataset, we are confident that we have found an optimal solution for each of Parts 1 and 2.

For small n (definitely for $n \leq 9$), both of our models provide a fast optimal solution. The iterative model gave quick optimal solutions by hand, and the constructive model ran in less than one minute on a Macintosh II. Our models use algorithms that are efficient for small n but may not be the most efficient algorithms available.

However, the problem cannot be solved quickly for all n, since it involves solving the minimal rectilinear spanning tree problem, which is known to be NP-complete (see Garey [1979]).

Both of our models are robust, and we can generalize them in many different ways (though not all at once). Some generalizations slow down the constructive model's algorithm tremendously.

Running both of our algorithms in parallel will yield a solution in the shorter of the two running times for the specific problem at hand.

References

Gardner, Martin. 1986. Mathematical games: Casting a net on a checkerboard and other puzzles of the forest. *Scientific American* 254(6) (June 1986): 16, 20–23, 128.

Garey, Michael R., and David S. Johnson. 1979. *Computers and Intractability.* New York: Freeman.

Graham, R.L., D.E. Knuth, and O. Patashnik. 1989. *Concrete Mathematics.* Reading, MA: Addison-Wesley.

Hanan, M. 1966. On Steiner's problem with rectilinear distance. *SIAM Journal of Applied Mathematics* 14: 255–265.

Hanan, M. 1975. Layout, interconnection, and placement. *Networks* 15 (1975): 413–423.

Hoffman, Dale T. 1979. Smart soap bubbles can do calculus. *Mathematics Teacher* 72(5) (May 1979): 377–385, 389.

Hwang, F.K. 1976. On Steiner minimal trees with rectilinear distance. *SIAM Journal of Applied Mathematics* 30 (January 1976): 104–114.

______. 1979. An $O(n \log n)$ algorithm for suboptimal rectilinear Steiner trees. *IEEE Transactions on Circuits and Systems* CAS–26 (January 1979): 75–77.

Isenberg, Cyril. 1976. The soap film: an analogue computer. *American Scientist* 64(5) (September-October 1976): 514–518.

Koffman, E.B. 1985. *Problem Solving and Structured Programming in Pascal.* 2nd ed. Reading, MA: Addison-Wesley.

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

Lee, Jerry H., N.K. Bose, and F.K. Hwang. 1976. Use of Steiner's problem in suboptimal routing in rectilinear metric. *IEEE Transactions on Circuits and Systems* CAS–23 (July 1976): 470–475.

Manber, Udi. 1989. *Introduction to Algorithms: A Creative Approach.* Reading, MA: Addison-Wesley.

Naps, T.L., and B. Singh. 1986. *Introduction to Data Structures with Pascal.* St. Paul, MN: West.

Parker, S.P., ed. 1989. *McGraw-Hill Dictionary of Scientific and Technical Terms.* 4th ed. New York: McGraw-Hill.