Finding Optimal Steiner Trees

Zvi Margaliot
Alex Pruss
Patrick Surry
University of Western Ontario
London, Ontario
Canada N6A 5B9

Advisor: H. Rasmussen

Introduction

We find a minimum rectilinear Steiner tree for the network given. We prove or cite some important results concerning minimum spanning trees and Steiner trees, in particular:

- There is a lower bound on the tree length (**Theorem 1** below).
- There is a minimal Steiner tree that spans the network (**Theorem 2**).
- We can find a reduced set of lattice positions for phantom points; in our case, we need to consider only 31 positions out of a possible 936 lattice points (**Theorems 3** and **5**, and **Figure 1**).
- A network of n stations will require no more than $n-2$ phantom stations (**Theorem 4**).
- The general problem is NP-complete [Chung and Whang 1979].
- We use an algorithm to construct a minimum spanning tree from a given set of fixed and phantom points (see Minieka [1978]).

Using these results, we solve the problem with two basic general approaches:

- **Brute Force:** Our computer program cycled through all possible combinations of phantom point locations, coming up with the absolute optimal spanning tree in about 3.6 million iterations (17 hours of computing time).
- **Simulated Annealing:** A more elegant and efficient way of solving NP-complete problems, the simulated annealing program proved much faster (about 1.5 min for Part 1, 3 min for Part 2) than brute force and consistently (100 times out of 100 tries) converged to an optimal path found by the brute-force algorithm.

Both programs are completely general, in that any fixed-station network and any cost parameters can be used.

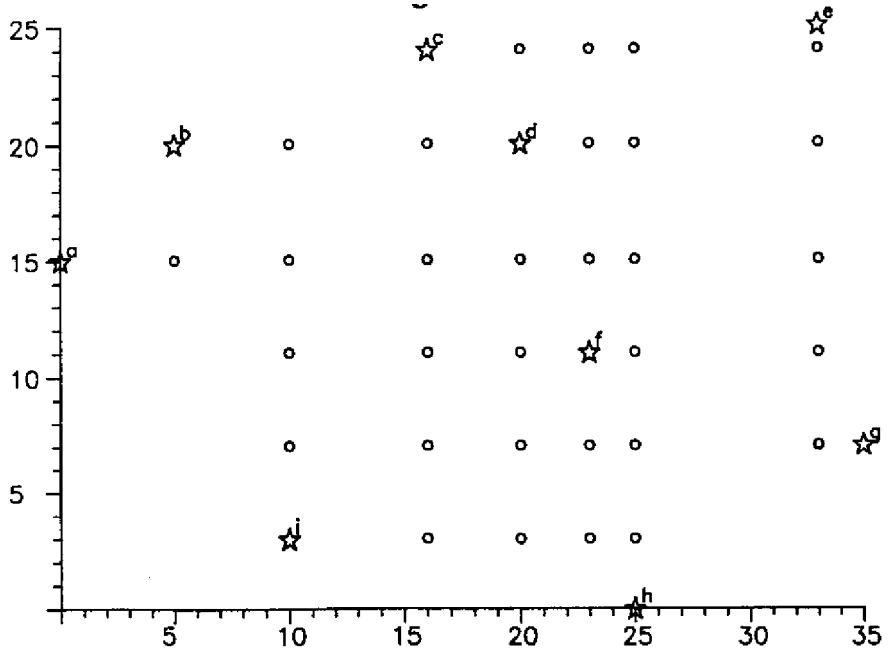

Figure 1. The reduced number of 31 positions that need to be considered for phantom stations. The original 9 points are denoted by lettered stars.

Results

- **Part 1:** We found five different optimal trees, each with either four or five phantom stations and having length exactly 94 (**Figure 2**).
- **Part 2:** We found two trees that are probably optimal, each with two phantom stations of degree three and each costing 135.89 (**Figure 3**).

General Discussion and Theory

Given a set of fixed stations in the taxicab metric $d(\mathbf{x},\mathbf{y}) = ||\mathbf{x}-\mathbf{y}||_1 = |x_1-y_1|+|x_2-y_2|$ on the lattice $\mathbb{Z}^2$, we are asked to find a minimum spanning graph to network all the stations together. We may add "phantom" stations in order to create a Steiner-type tree. As specified in the problem, the tree branches ("edges") and all station locations must lie along the lattice.

In finding a solution, we have three degrees of freedom:

- how many phantom stations are to be included,
- where each phantom station will be located, and
- how we construct a spanning graph of minimum cost.

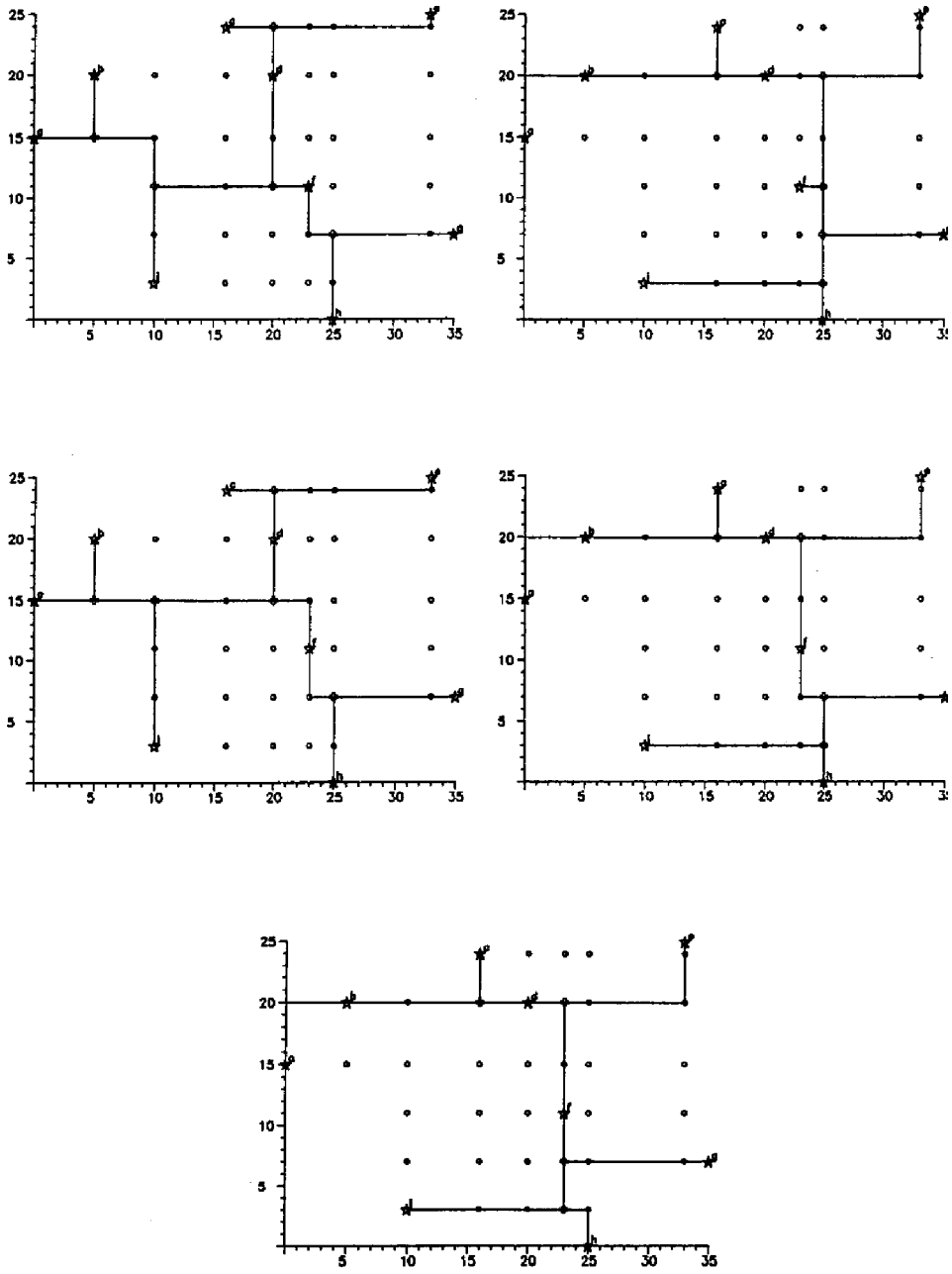

Figure 2. The five optimal solutions to Part 1, each with total edge length of 94. Stars indicate the original fixed points, and crosses indicate added phantom points.

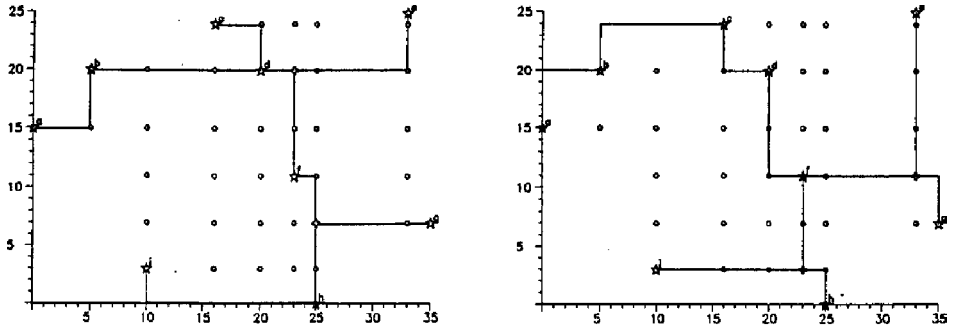

Figure 3. Two probably optimal solutions to Part 2, each with total cost 135.89.

Preliminary Results

- There is an optimal configuration for the given problem. Without one, we could be wasting our time. The result follows from **Theorem 2** below.
- Of the possible phantom-point locations—in our case, the finite set of lattice points within and on the bounding rectangle of the original nodes—only a few need to be considered for the minimal spanning tree (in our problem, only 31 out of a possible 936). Without such a reduction, the problem would be intractably large. The reduction follows from **Theorems 3** and **5**.
- We need at most $n-2$ phantom points and $n-1$ edges to minimize the spanning tree. This observation follows from **Theorem 4**.

Proofs of Preliminary Results

We are given a set of basic points $P = \{(x_i, y_i) | i = 1, 2, \ldots, n\}$. We define the sets of x- and y-coordinates of these points separately as $X = \{x_i\}$ and $Y = \{y_i\}$. Our goal is to form a connected graph on these points with lowest possible cost (dependent on length and station cost), where we are allowed to introduce "phantom stations" as we desire. Let the set of such new points in the optimal configuration be $P' = \{(x'_i, y'_i) | i = 1, 2, \ldots, m\}$, and as above put $X' = \{x'_i\}$ and $Y' = \{y'_i\}$.

Theorem 1. *Every configuration of connections and extra stations will always have a path length of at least* $d = x_{\max} - x_{\min} + y_{\max} - y_{\min}$, *where* $x_{\max} = \max x_i$, $y_{\max} = \max y_i$, *and so on.*

Proof: Since all of the basic points must be connected (possibly via extra stations), any vertical line $x = k$ with $x_{\min} \le k \le x_{\max}$ must cut the path in one or more points (otherwise we would have split a connected set into two

disjoint pieces, which is a contradiction). Hence, if we project the actual path onto the x-axis, its length is at least $x_{\max} - x_{\min}$. Similarly, the projection of the path onto the y-axis is at least $y_{\max} - y_{\min}$. Any projection of the path will be less than its actual length (regardless of the metric, as long as edges are continuous). Moreover, since the path is composed only of horizontal and vertical segments (using rectilinear distances), the vertical components contribute nothing to the x-axis projection and vice versa; so the total path length has at least the length indicated. **Λ**

Theorem 2. *There is always an optimal configuration.*

Proof: Given any set of n basic points, construct a spanning tree S by connecting P_1 to P_2 to P_3 ... to P_n; this tree has some finite length d. With the given definition of cost, this configuration will have cost less than $d+nC$, where C is the cost of a degree-2 station. Using the rectilinear metric, there are finitely many configurations with length less than $d + nC$, and each of these will have a cost of at least its length (depending on station cost). Therefore, there is only a finite number of paths whose total cost is less than the cost of S. Hence, there must be a minimal-cost configuration. **Λ**

Theorem 3. *For any optimal configuration (regardless of the cost of any extra stations), $X' \subset X$ and $Y' \subset Y$. (This says that all phantom stations must lie only on the same horizontal and vertical lines as the basic points.)*

Proof: (by induction) We need only show the first of the two results; the second follows identically. Let $X^* = X \cup X' \backslash \{x'_m\}$. In the optimal configuration, there is a certain set of $k \leq m + n$ points connected to the point (x'_m, y'_m). Let $\hat{X}^* = \{\hat{x}_i | i = 1, 2, \ldots, k\}$, where $\hat{x}_1 \leq \hat{x}_2 \leq \cdots \leq \hat{x}_k$ are the x-coordinates of these points. The cost of the overall configuration can be viewed as being made up of a contribution that depends on the position of phantom point P'_m and another that doesn't depend on this position (connections not involving P'_m and station costs). Thus, we can write the cost as $f(x'_m, y'_m) + C$, where C is a constant. In particular, since we are using rectilinear distances, we have:

$$f = f_{\hat{X}^*} + f_{\hat{Y}^*} = \sum_{\hat{x} \in \hat{X}^*} |\hat{x} - x'_m| + \sum_{\hat{y} \in \hat{Y}^*} |\hat{y} - y'_m|.$$

Consider only the first term. If $x'_m \notin \hat{X}^*$, then there exists $l \leq k$ such that

$$\hat{x}_1 \leq \hat{x}_2 \leq \cdots \leq \hat{x}_l < x'_m < \hat{x}_{l+1} \leq \cdots \leq \hat{x}_k.$$

Thus,

$$f_{\hat{X}^*} = \sum_{i=l+1}^{k} (\hat{x}_i - x'_m) + \sum_{i=1}^{l} (x'_m - \hat{x}_i).$$

In the case that $l < k/2$, we can reduce the value of this sum by setting $x'_m = \hat{x}_{l+1}$, as doing so will increase the value of each of the $l < k/2$ terms in the first sum by a fixed amount Δx but will reduce each of the $k - l > k/2$ terms in the second sum by the same amount. In the same way, if $l > k/2$, we could lower the sum by setting $x'_m = \hat{x}_l$ (note that if $l = k/2$, we can set $x'_m = \hat{x}_l$ and leave the value unchanged). However, this is a contradiction to the optimality of the configuration (changing the x-coordinate of one point doesn't affect the cost in any other way). Hence, we can assume that $x'_m \in \hat{X^*} \subset X^*$ as required. Proceeding by induction on all of the "movable" points (x'_i, y'_i), we must have $X' \subset X$ and similarly $Y' \subset Y$. **Λ**

Corollary. *For n basic points, there are at most $n^2 - n$ locations for extra stations in an optimal configuration, given by $\{(x_i, y_j) | x_i \in X, y_j \in Y, 1 \leq i, j \leq n\}$. (There may be even fewer possibilities if some of the basic points lie on the same horizontal or vertical line.)*

Comment: We did not use the assumption that extra stations are at lattice points. Since all the basic points are lattice points, any extra points must be too.

Lemma 1. *For a network of n points (fixed or otherwise), the optimum spanning graph contains exactly $n - 1$ edges.*

Proof: We know that the optimum solution can have no closed cycles (if there were a loop, we could cut one of its edges, leaving a still connected graph but saving the cost of one whole edge). In other words, this graph is a tree. Therefore, there exists a point of degree one. We can remove this point and its connecting edge, leaving a connected set with one fewer edge and one fewer point, which again contains no closed loops. We continue in this way until we have one point left and no more edges. Hence, the tree contained only $n - 1$ edges. **Λ**

Theorem 4. *At most $n-2$ extra stations are required in an optimal configuration (the parallel of the given result for networks with a Euclidean metric).*

Proof: Suppose the optimal configuration (which exists by **Theorem 2**) has n basic points and m phantom stations. Each basic point must have degree at least one, since all points are connected. Further, each phantom station must have degree at least three or else there is no reason to have it in the configuration (a station of degree one doesn't connect anything, and a station of degree two just lies on a line). If we count all the edges emanating from each point in the graph, we will count all edges twice (since every edge has two ends). Thus, we arrive at the inequality $2e \geq 3m + n$, where e is the

number of edges. But the number of edges in the optimal configuration is $e = m+n-1$. Thus, we get the desired result. (Alternatively, we could apply Euler's relation $V + F - E = 2$ with $F = 1$ and $V = m + n$.) Substituting in the inequality, we get $2(m + n - 1) \geq 3m + n$, or $m \leq n - 2$, as required. **Λ**

Comment: Combining this result with the Corollary to **Theorem 3**, we see that we have at most $\binom{n^2-n}{n-2}$ possible combinations of the phantom stations. Since every configuration of basic points and extra points has a certain minimum spanning tree, we can actually use brute force to find the required configuration if we want. Note also that the given proof works for any metric—it doesn't use any specific measuring information, just connectivity.

Theorem 5. *For any point $P^* = (x^*, y^*)$ such that all basic points P_i satisfy $x_i \leq x^*$ or $y_i \leq y^*$, then there can be no phantom point P_i' with $x_i' \geq x^*$ and $y_i' \geq y^*$ in an optimal configuration. (Note that by symmetry we can get three other similar results by swapping either or both of the x or y inequalities.)*

Proof: We proceed by contradiction. Assume there is such a phantom station (x_i', y_i') in the optimal configuration. Clearly, no path terminates in the region $x \geq x^*$, $y \geq y^*$, since any such terminating path could be removed, leaving all of the basic points still connected at lower cost. Hence, the paths inside the region connected to (x_i', y_i') must meet the lines $x = x^*$, $y = y^*$ in the points $A_1(a_1, y^*), A_2(a_2, y^*), \ldots, A_q(a_q, y^*)$ and $B_1(x^*, b_1), \ldots, B_r(x^*, b_r)$, where $x^* \leq a_1 < a_2 < \cdots < a_q$ and $y^* \leq b_1 < \cdots < b_r$. Because no path terminates in the region, we must have $q + r \geq 2$. For the case of $q \geq 2$ and $r = 0$, it is clear as in **Theorem 1** that the length of the tree connected to (x_i', y_i') is at least $a_q - a_1$ by projection onto the line $y = y^*$. Hence, we could replace all of these connections with the segment (a_1, y^*) to (a_q, y^*) and have an equal or lower cost (since we need introduce no new stations, keep connections as they are, but get a shorter path). In the case $q = 0$ and $r \geq 2$, we use the segment (x^*, b_1) to (x^*, b_r); and for $q > 0$ and $r > 0$, we use the two segments (x^*, b_r) to (x^*, y^*) to (a_q, y^*). In all cases, the cost is equal or reduced. **Λ**

Comment: This theorem allows us to reduce further the locations for phantom stations by chopping "corners" off the set of possible locations, as follows.

For each $y_k \in Y$ $(1 \leq k \leq n)$, set:

$$x_{00}(k) = \min_{\substack{(x_i,y_i)\in P \\ y_i < y_k}} \{x_i\} \qquad x_{01}(k) = \min_{\substack{(x_i,y_i)\in P \\ y_i > y_k}} \{x_i\}$$

$$x_{10}(k) = \max_{\substack{(x_i,y_i)\in P \\ y_i < y_k}} \{x_i\} \qquad x_{11}(k) = \max_{\substack{(x_i,y_i)\in P \\ y_i > y_k}} \{x_i\}.$$

Then by **Theorem 5** and the three symmetric results, there can be no phantom stations in any of the four regions:

$$x < x_{00}(k),\ y < y_k, \qquad x < x_{01}(k),\ y > y_k$$

$$x > x_{10}(k),\ y < y_k, \qquad x < x_{11}(k),\ y > y_k.$$

In our particular problem, we need only consider the 31 possible locations in **Figure 1** (p. 142), as opposed to the $9 \times 9 - 9 = 72$ points required using only the Corollary to **Theorem 3**.

Minimal Spanning Trees

Given a network of n points, we need to be able to construct the spanning tree of minimal total length. There are many algorithms to do this exactly, and we chose one that is fast and particularly adaptable for a computer. The minimum spanning tree algorithm of Minieka [1978] is guaranteed to find a tree (if one exists) in $\binom{n}{2}$ steps. Briefly, it works as follows:

> The set of all paths connecting every point to every other point in the network and their lengths is created. The algorithm then selects from the set the next path with the shortest length (i.e., lowest cost) and will do either of two things:
>
> - Reject the path if its inclusion will create a complete cycle with the rest of the paths already selected, or
> - Include the path in the tree if it is not part of a complete cycle.
>
> The tree is complete when it consists of $n-1$ separate paths. Since there are exactly $\binom{n}{2}$ possible such paths in the network, the algorithm is guaranteed to find a solution in that many steps. Thus, we can now extract a minimum spanning tree for *any* set of points.
>
> Note that this algorithm doesn't take into account station costs.

Strategy

Lacking an algorithm for finding a minimal rectilinear Steiner tree, we can use either a slow exact algorithm or a faster nonexact one:

- **Brute force:** We simply try all possibilities for both the number of phantom points and their locations. In our problem, using 9 fixed points with up to 7 phantom points at 31 possible locations would require

$$1 + \binom{31}{1} + \binom{31}{2} + \cdots + \binom{31}{7} = 3,572,224$$

tries. With each try taking approximately 0.017 sec (to calculate the minimum spanning tree and its length) on a 25-MHz 386-based PC, this is approximately a 17-hour operation. This method will give an absolute minimum configuration and can be adapted to any finite starting network configuration. Despite being highly inefficient, it will do the job for the given dataset. We ran our program simultaneously on six workstations (MIPS, Irises, Suns, and 386s), each doing different tries (a poor mathematician's imitation of parallel processing), to confirm the results we got with the second, more elegant approach. [EDITOR'S NOTE: We omit the program.]

- **Simulated annealing:** This is a more elegant way of searching out all the feasible combinations of phantom points and locations. We start with a given tree of phantom points and locations and allow the annealing program to create a new configuration. For each new configuration, the routine determines the minimum spanning tree and calculates the tree length. The annealing program then decides whether to retain or reject the new configuration based on a given *cooling schedule*. Modifying a given configuration and evaluating the modification are both very easy. The approach is completely general: Any starting configuration may be used, and the algorithm does not directly depend on the set of fixed network points. Simulated annealing, as opposed to the brute-force method, may not produce the absolute minimum. The chance that it will settle on a local minimum different from the absolute minimum can, however, be made as small as we wish, while still providing a very considerable time improvement over the first approach.

Part 1: No Station Costs

Figure 1 (p. 142) is a graph of the starting network of basic (or fixed) points, along with the reduced set P' of possible phantom-point locations. The program for simulated annealing (written in C) takes as input fixed points and the possible phantom-point locations, along with a starting path. [EDITOR'S NOTE: We omit the program.] (We started the program with an optimal path that we found by hand: We divided the fixed network into groups of three points, optimized each group by adding one phantom point, and then optimized the groups. The resulting configuration turned up several times as the minimal path of the annealing routine.)

The current configuration changes in a way randomly chosen from:

- adding a new phantom point randomly to one of the allowed locations,
- removing an existing phantom point, and
- moving an existing phantom point to a new random location.

(Note that in these three ways, we can move between all possible network configurations. For doing so, the third way is actually unnecessary; but we included it to give the annealing procedure more freedom.)

With the new configuration in memory, a calculation routine is called to set up the minimal spanning tree and evaluate the cost of the new network. This cost is then used by the annealing part of the program (routine METROP from Press et al. [1988, 351]) to decide whether the new configuration should be retained or rejected, according to an agreed-upon cooling schedule.

Notes of Interest: The first few runs indicated a trend to add phantom points of degree two, which are useless. Even though in Part 1 of the problem stations don't cost anything, we decided to eliminate this excess by the program by introducing a small penalty for each phantom point. The penalty was made small enough not to discourage adding phantom points where necessary but to eventually dispose of unnecessary ones.

We also added a feature not normally present in annealing routines, whereby the program remembers the best solution encountered during the entire run (simulated annealing usually only returns the last configuration used).

Results

- **Brute Force**
  - Brute force ran about 3.5 million calculations, occupying two hours of searching on each of six workstations.
  - Five distinct optimal networks were found, all of length 94; three had five phantom stations and two had four (**Figure 2**) (p. 143).
- **Simulated Annealing**
  - It took an average of about 9,800 iterations of the annealing routine (approx. 1.5 min using a 25-MHz 386-based PC) for convergence to an optimal solution.
  - Different runs, with different seeds to the random number generator, gave all of the five different optimum solutions of length 94.
  - In more than 100 runs, simulated annealing always converged to one of the five optimal paths, proving its worth for more general instances of the problem (when brute force is impossible).

Part 2: Station Costs

The problem is the same as Part 1, except for a cost factor for each station of $w(\text{degree})^{3/2}$, with $w = 1.2$. Two competing assumptions are:

- **Only phantom stations cost:** This is not realistic but can provide useful insight. Since each phantom station is of degree at least three, it will have a cost of at least $3^{3/2} \times 1.2 = 6.24$. Now, the length of the minimum path with phantom stations is 94 (from Part 1), but the length of the minimum path with phantom stations is only 110. Hence, to keep the cost under 110, one can afford at most two phantom stations. A quick brute-force calculation showed that no gains can be made by adding phantom stations, so the optimum tree is the minimum spanning tree with no phantom points.
- **All stations cost:** If we wanted to make use of the annealing routine developed for Part 1, we could either:
  - use some type of brute-force approach;
  - anneal the location, number, and degree of phantom stations (nested annealing!); or
  - modify the minimum spanning tree algorithm to try find the least-expensive (as opposed to shortest) tree, and use annealing as in Part 1.

  It seemed that the first two options would be too time-consuming and inefficient, so we elected to try the third.

Modifying the Minimum-Spanning-Tree Algorithm

Given a set of points, we wish to construct a spanning tree of least cost, based on the cost formula given above. We base it on the routine described earlier, with the following changes:

1. All edges are calculated and their cost evaluated as $(\text{length} + 2w)$ (i.e., we assume for now a station of degree one at each end).
2. The least-cost edge that does not form a cycle joining points (A, B) is selected and put in the tree.
3. The cost of all edges containing either A or B is recalculated by incrementing the degree of the endpoint, since a station of degree one higher would now be created if we used such an edge. We then go back to Step 2.

As before, the routine ends when $(n-1)$ edges have been selected, in no more than $\binom{n}{2}$ steps. With the modification, we can use both annealing and brute force exactly as before, with minimal code changes everywhere except in the actual spanning-tree generator.

The modified routine is only heuristic; it may not generate the least-cost spanning tree. However, we expect it to give a good spanning tree, an expectation confirmed by examining some of the trees it generates.

Modification Shortcomings

The new minimum-cost heuristic has some definite shortcomings:

- It does not guarantee the overall minimum cost path.
- Moreover, since the general problem is NP-complete [Chung and Whang 1979], we cannot possibly always solve it exactly in $\binom{n}{2}$ steps, as our procedure would seem to suggest.
- The routine is considerably slowed (by as much as a factor of 2) by having to recalculate costs at every step.

What this procedure does seem to do is to provide a very good (usually dead-on) approximation to the minimum-cost spanning tree, and the shortcomings may be overcome by several restarts of the problem (though, as we shall see, this will not be necessary for the given problem).

Results

From several different annealing runs, we came up with two different trees with two phantom stations and cost 135.89 (see **Figure 3** on p. 144). (Note: The resulting network may sometime contain two edges that occupy the same lattice line. If such a result is not acceptable, one can always manually adjust the tree—without changing its total length or cost—to avoid it.)

We get a lower bound of 94 for the cost, from the minimum spanning tree with no station costs. Of the nine fixed points, at most eight can be of degree one (this is clearly underestimating the degree), for a total cost of 8×1.2, plus one additional one of degree two, costing $2^{1.5} \times 1.2 = 3.39$. Therefore, any spanning tree must cost at least $94 + 8(1.2) + 3.4 = 107$. Now, for every additional phantom point, we add at least the cost of a station of degree three, 6.24; hence, a network with m phantom points must cost at least $107 + 6.24m$. With our minimum from simulated annealing (clearly an upper bound for the absolute minimum) calculated at 135.89, we see that we can have at most four phantom stations with this cost scheme.

We stoop once more to the level of brute force and, using up to four phantom points, search out the minimum configuration (this search is much quicker than the previous brute-force calculation with up to seven). As before, the brute-force sledge-hammer verified our annealing results. Both the annealing and the brute-force results depend on the correctness of the modified minimum-spanning-tree routine but should in any case be very close to the truth.

Conclusions and Generalization

If we regard Part 1 as the special case $w = 0$ of Part 2, we then have two general procedures for optimizing a given spanning communication network. The simulated-annealing scheme produced verifiable results quickly, and it can accommodate any fixed-point network and any cost parameters. For a small number of points (or very fast vectorized/parallel computing machines), a brute-force approach is a not very elegant but absolute method of solution.

For $w \neq 0$, we can bound the number of phantom stations by using the optimal or near-optimal tree found for the case $w = 0$, and thus find upper and lower bounds for the minimum cost of any configuration using n or fewer phantom stations.

For the problem at hand, we found optimum solutions. We know these solutions to be absolutely optimal (for Part 1) or very likely to be optimal (Part 2), from brute-force checking.

While simulated annealing cannot does not always produce the absolute optimum solution, it did give the absolute minimum in every one of the more than 100 trial runs we did, indicating an acceptably high success rate.

Other Generalizations

If, instead of generalizing our point set or the value of w, we wish to make larger generalizations, then we can take one of several routes. First, we can generalize the station-cost function. Our near-optimal tree generation routine can be applied with just the cost function being changed. The brute-force technique, with the bound on the number of phantom stations, can also be applied.

On the other hand, we could generalize to networks located in more general spaces than $\mathbb{Z}^2$ with the taxicab metric. Our techniques will work only for spaces in which we can find a bounding box with finitely many feasible sites for the phantom stations. This includes all of the spaces S^n with the taxicab metric, for $S \subset \mathbb{R}$, since **Theorems 3** and **5** can be generalized. We retain the results restricting feasible points to having coordinates from the projections of the set of stations, as well as the restriction to a bounding box with corners removed (as in the comment following **Theorem 5**).

We may generalize to any metric derived from $||\cdot||_p$, for $1 \leq p < \infty$; but then we require S to be discrete. In this case, not having the taxicab metric, we will have to check many more points in our bounding box, since we can no longer confine our attention to points whose coordinates lie in projections of the initial stations. In particular, we can easily generalize to $\mathbb{R}^n$ in the taxicab metric, as well as to $\mathbb{Z}^n$ in any metric derived from a norm $||\cdot||_p$.

However, even with no greater number of basic points, still we will have many more feasible points than before. In the taxicab metric, with m initial stations and a space in $\mathbb{R}^n$, we have now $\mathcal{O}(m^n)$ feasible locations for phantom stations; while for $\mathbb{Z}^n$ in other metrics, the number of feasible locations is proportional to the hypervolume of the bounding box of the points. Thus, in some cases, the brute-force approach will be much more impractical. While simulated annealing can be expected to take longer to run, it should nonetheless work, although—since the state-space is much larger—it may no longer be as likely to give the optimal solution.

We could also generalize by adding extra cost criteria, such as minimal radius or diameter of the (weighted) graph representing the network (which may be useful if we wish to make the network respond faster), or having some preferential network routings. Unfortunately, such changes would render ineffective our heuristic approach to finding near-optimal spanning graphs for a fixed set of phantom stations and a fixed node-cost function based on degree. We would need either to develop a new heuristic approach for these specific cost criteria or to use a general method (such as simulated annealing) to find the near-optimal spanning graph for fixed phantom stations. As before, the phantom stations can be found by an outer annealing procedure or brute force, although if the spanning tree computations take too long, brute force would be completely impossible. Thus, we would have to resort to either a heuristic or an annealing procedure to find an optimal tree with fixed stations, and then an outer annealing to find the best configuration.

Impermeable or costly-to-cross obstacles, around which we must route the network, can be handled by modifying our metric (i.e., the metric function would route around impermeable obstacles and add the crossing cost to the distance; if done properly, doing so will still give a metric function). We would need to derive a bounding box for all the feasible network configurations; but this can be done fairly easily, since if we can find any spanning graph, a box of radius equal to the cost of this graph can be constructed. As long as we work in a discrete space such as $\mathbb{Z}^n$, we can use all of our machinery, though brute-force might be inefficient.

References

Chung, F.R.K., and F.K. Whang. 1979. The largest minimal rectilinear Steiner trees for a set of n points enclosed in a rectangle with given perimeter. *Networks* 9 (1979): 19–36.

Minieka, Edward. 1978. *Optimization Algorithms for Networks and Graphs.* New York: Dekker.

Press, William H., et al. 1988. *Numerical Recipes in C: The Art of Scientific Computing.* New York: Cambridge.