CHAPTER II

INITIAL RESULTS

The First Solution

A typical problem-solving approach is to begin with the simple cases and expand to a general solution. As we saw in Chapter I, the trivial three point problem had already been solved in the 1600's, so all that remained was the work toward a general solution. As with many interesting problems this is harder than it appears on the surface.

The method proposed by the mathematicians of the mid-seventeenth century for the three point problem is illustrated in Figure 1. This method stated that in order to calculate the Steiner Point given points A, B, and C, you first construct an equilateral triangle (ACX) using the longest edge between two of the points (AC) such that the third (B) lies outside the triangle. A circle is circumscribed around the triangle, and a line is constructed from the third point (B) to the far vertex of the triangle (X). The location of the Steiner Point (P) is the intersection of this line (BX) with the circle.

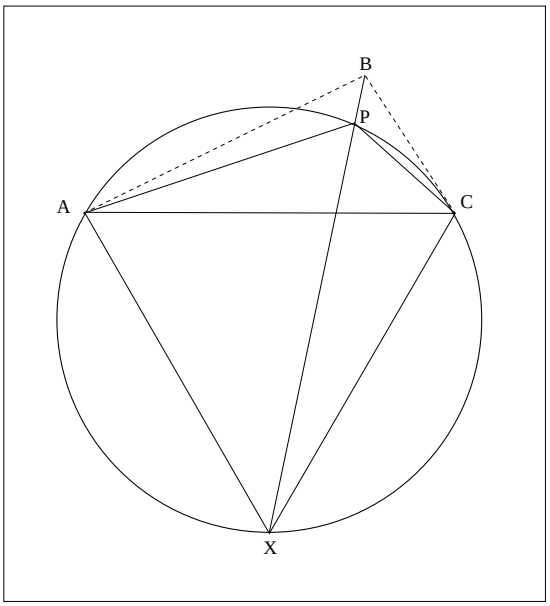

Figure 1. AP + CP = PX.

For the next thirty years after Kössler and Jarník presented the general form of the SMT problem, the only algorithms that existed were heuristics. The heuristics were typically based upon the Minimum-Length Spanning Tree (MST), which is a tree that spanns or connects all vertices whose sum of the edge lengths is as small as possible, and tried in various ways to join three vertices with a Steiner Point. In 1968 Gilbert and Pollak [6] linked the length of the SMT to the length of a MST. It was already known that the length of an MST is an upper bound for the length of an SMT, but their conjecture stated that the length of an SMT would never be any shorter than $\frac{\sqrt{3}}{2}$ times the length of an MST. This conjecture, was recently proved [7], and has led to the MST being the starting point for most of the heuristics that have been proposed in the last 20 years. This combination approach, starting with the MST, will serve as the basis for the heuristic we present later in this chapter.

In 1961 Melzak developed the first algorithm for calculating an SMT [8]. Melzak's Algorithm was geometric in nature and was based upon some simple extensions to Figure 1. The insight that Melzak offered was the fact that you can reduce an n point problem to a set of $n-1$ point problems. This reduction in size is accomplished by taking every pair of points, A and C in our example, calculating where the two possible points, X_1 and X_2, would be that form an equilateral triangle with them, and creating two smaller problems, one where X_1 replaces A and C, and the other where X_2 replaces A and C. Both Melzak and Cockayne pointed out however that some of these sub-problems are invalid. Melzak's algorithm can then be run on the two smaller problems. This recursion, based upon replacing two points with one point, finally terminates when you reduce the problem from three to two vertices. At this termination the length of the tree will be the length of the line segment connecting the final two points. This is due to the fact that $BP + AP + CP = BP + PX$. This is straightforward to prove using the law of cosines, for when P is on the circle, $\angle APX = \angle CPX = 60°$. This allows the calculation of the last Steiner Point (P) and allows you to back up the recursive call stack to calculate where each Steiner Point in that particular tree is located.

This reduction is important in the calculation of an SMT, but the algorithm still has exponential order, since it requires looking at every possible reduction of a pair of points to

a single point. The recurrence relation for an n-point problem is stated quite simply in the following formula:

$$T(n) = 2 * \left(\begin{array}{c} n \\ 2 \end{array} \right) * T(n-1).$$

This yields what is obviously a non-polynomial time algorithm. In fact Garey, Graham, and Johnson [9] have shown that the Steiner Minimal Tree problem is NP-Hard (NP-Complete if the distances are rounded up to discrete values).

In 1967, just a few years after Melzak's paper, Cockayne [10] clarified some of the details from Melzak's proof. This clarified algorithm proved to be the basis for the first computer program to calculate SMTs. The program was developed by Cockayne and Schiller [11] and could compute an SMT for any placement of up to 7 vertices.

A Proposed Heuristic

Background and Motivation

By exploring a structural similarity between *stochastic Petri nets* (see [12] and [13]) and *Hopfield neural nets* (see [14] and [15]), Geist was able to propose and take part in the development of a new computational approach for attacking large, graph-based optimization problems. Successful applications of this mechanism include I/O subsystem performance enhancement through disk cylinder remapping [16, 17], file assignment in a distributed network to reduce disk access conflict [18], and new computer graphics techniques for digital halftoning [19] and color quantization [20]. The mechanism is based on maximum-entropy Gibbs measures, which is described in Reynold's dissertation [21], and provides a natural equivalence between Hopfield nets and the *simulated annealing* paradigm. This similarity allows you to select the method that best matches the problem at hand. For the SMT problem we will implement the Simulated Annealing approach.

Simulated Annealing [22] is a probabilistic algorithm that has been applied to many optimization problems in which the set of feasible solutions is so large that an exhaustive search for an optimum solution is out of the question. Although Simulated Annealing does not necessarily provide an optimum solution, it usually provides a good solution in a user-selected amount of time. Hwang and Richards [4] have shown that the optimal placement

of s Steiner Points to n original vertices yields a feasible solution space of the size

$$2^{-n} \begin{pmatrix} n \\ s+2 \end{pmatrix} \frac{(n-s-2)!}{s!}$$

provided that none of the original points have degree 3 in the SMT. If the degree restriction is removed they showed that the number is even larger. The SMT problem is therefore a good candidate for this approach.

Adding One Junction

Georgakopoulos and Papadimitriou [23] have provided an $\mathcal{O}(n^2)$ solution to the *1-Steiner problem*, wherein exactly one Steiner Point is added to the original set of points. Since at most $n-2$ Steiner Points are needed in an SMT solution, repeated application of the algorithm offers a "greedy" $\mathcal{O}(n^3)$ approach. Using their method, the first Steiner Point is selected by partitioning the plane into Oriented Dirichlet Cells, which they describe in detail. Since these cells do not need to be discarded and recalculated for each addition, subsequent additions can be accomplished in linear time. Deletion of a candidate Steiner Point requires regeneration of the MST, which Shamos showed can be accomplished in $\mathcal{O}(n \; log \; n)$ time if the points are in the plane [24], followed by the cost for a first addition ($\mathcal{O}(n^2)$). This approach can be regarded as a natural starting point for Simulated Annealing by adding and deleting different Steiner Points.

The Heuristic

The Georgakopoulos and Papadimitriou 1-Steiner algorithm and the Shamos MST algorithm are both difficult to implement. As a result, we have chosen to investigate the potential effectiveness of this Annealing Algorithm using a more direct, but slightly more expensive $\mathcal{O}(n^3)$ approach. As previously noted, all Steiner Points have degree 3 with edges meeting in angles of 120°. We consider all $\binom{n}{3}$ triples where the largest angle is less than 120°, compute the Steiner Point for each (a simple geometric construction), select that Steiner Point giving greatest reduction, or least increase in the length of the modified tree (increases are allowed since the Annealing Algorithm may go uphill) and update the MST accordingly. Again, only the first addition requires this (now $\mathcal{O}(n^3)$) step. We use the straightforward, $\mathcal{O}(n^2)$ Prim's algorithm to generate the MST initially and after each deletion of a Steiner Point.

The Annealing Algorithm can be described as a non-deterministic walk on a surface. The points on the surface correspond to the lengths of all feasible solutions, where two solutions are adjacent if they can be reached through the addition or deletion of one Steiner Point. The probability of going uphill on this surface is higher when the temperature is higher but decreases as the temperature cools. The rate of this cooling typically will determine how good your solution will be. The major portion of this algorithm is presented in Figure 2. This non-deterministic walk, starting with the MST has led to some very exciting results.

Results

Before we discuss large problems, a simple introduction into the results from a simple six point problem is in order. The Annealing Algorithm is given the coordinates for six points: (0,0), (0,1), (2,0), (2,1), (4,0), (4,1). The first step is to calculate the MST, which has a length of 7, as shown in Figure 3. The output of the Annealing Algorithm for this simple problem is shown in Figure 4. In this case the Annealing Algorithm calculates the exact SMT solution which has a length of 6.616994.

We propose as a measure of accuracy the percentage of the difference between the length of the MST and the exact SMT solution that the Annealing Algorithm achieves. This is a new measure which has not been discussed (or used) because exact solutions have not been calculated for anything but the most simple layouts of points. For the six point problem discussed above this percentage is 100.0% (the exact solution is obtained).

After communicating with Cockayne, data sets were obtained for exact solutions to randomly generated 100 point problems that were developed for [25]. This allows us to use the measure of accuracy previously described. Results for some of these data sets provided by Cockayne are shown in Table 1.

An interesting aspect of the Annealing Algorithm that cannot be shown in the table is the comparison of execution times with Cockayne's program. Whereas Cockayne mentioned that his results had an execution cut-off of 12 hours, these results were obtained in less than 1 hour. The graphical output for the first line of the table, which reaches over 96% of the optimal value, appears as follows: the data points and the MST are shown in Figure 5,

```
#define EQUILIBRIUM ((accepts>=100 AND rejects>=200) OR
     (accepts+rejects > 500))
#define FROZEN ((temperature < 0.5) OR ((temperature < 1.0)
     AND (accepts==0)))

while(not(FROZEN)){
     accepts = rejects = 0;
     old_energy = energy();
     while(not(EQUILIBRIUM)){
          operation = add_or_delete();
          switch(operation){
               case ADD:
                    ΔE = energy_change_from_adding_a_node();
                    break;
               case DELETE:
                    ΔE = energy_change_from_deleting_a_node();
                    break;
          }
          if(rand(0,1) < e^{min{0.0,-ΔE/temperature}}){
               accepts++;
               old_energy = new_energy;
          }else {
               /* put them back */
               undo_change(operation);
               rejects++;
          }
     }
     temperature = temperature*0.8;
}
```

Figure 2. Simulated Annealing algorithm.

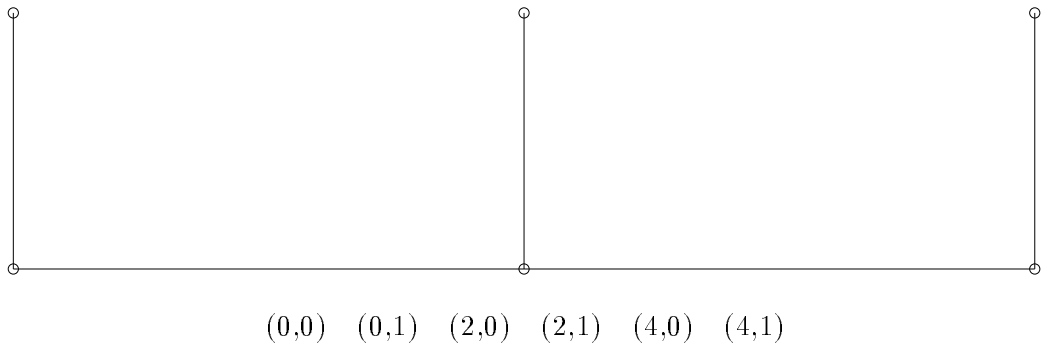

Figure 3. Spanning tree for 6 point problem.

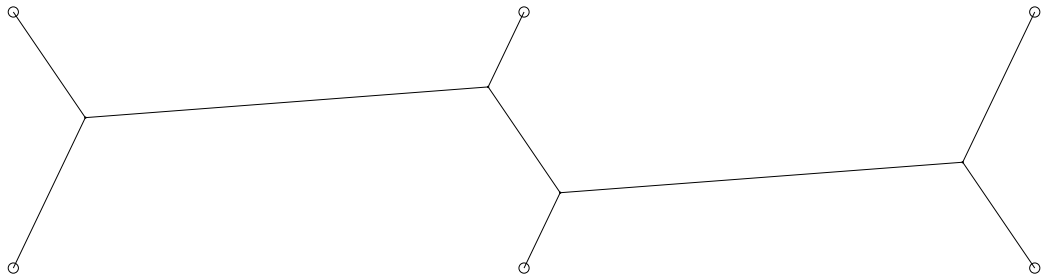

Figure 4. 6 point solution.

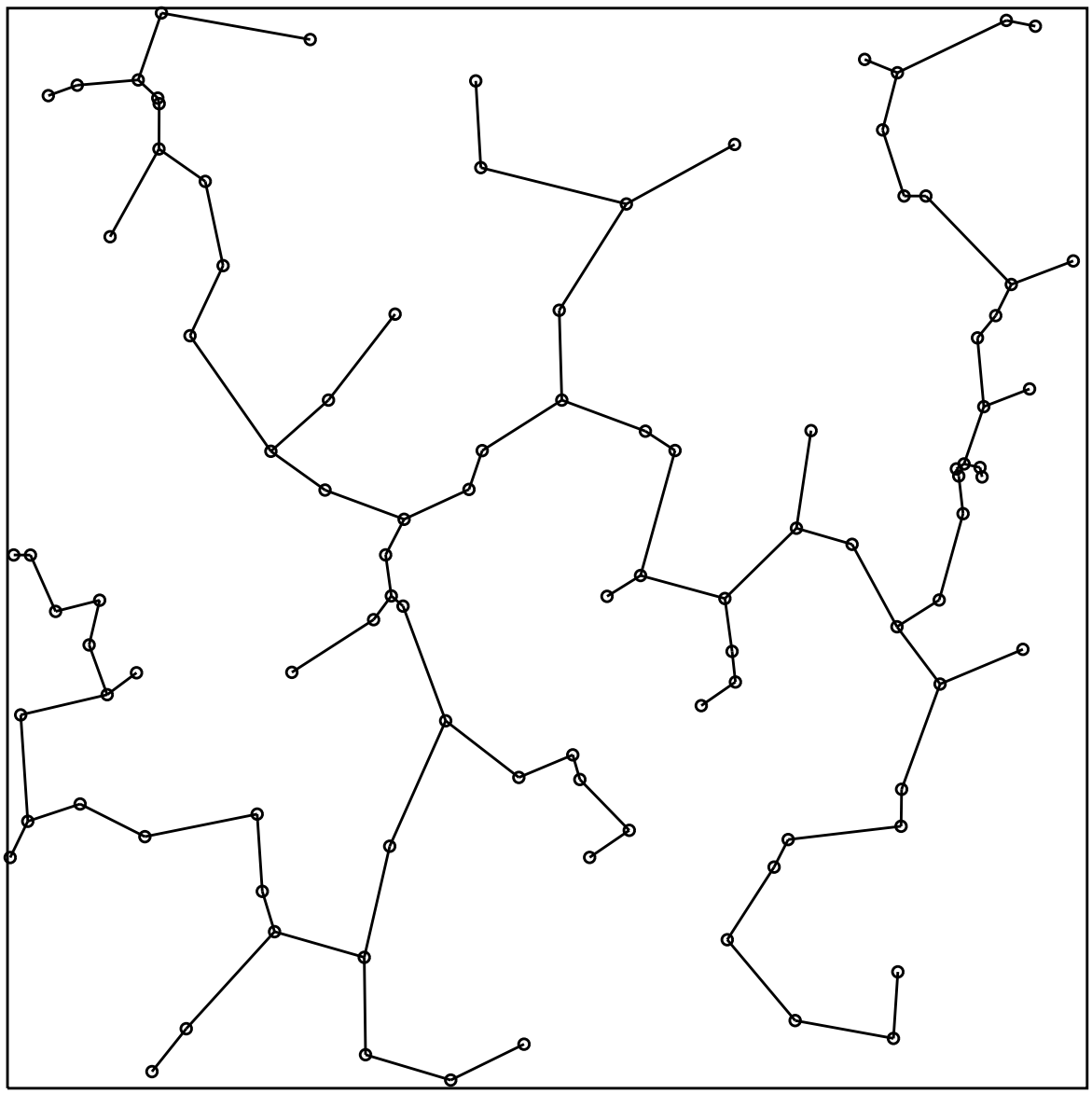

Figure 5. Spanning tree.

the Simulated Annealing Result is in Figure 6, and the Exact SMT Solution is in Figure 7. The solution presented here is obtained in less than $\frac{1}{10}th$ of the time with less than 4% of the possible range not covered. This indicates that we could hope to extend our Annealing Algorithm to much larger problems, perhaps as large as $1,000$ points. If we were to extend this approach to larger problems we would definitely need to implement the Georgakopoulos-Papadimitriou 1-Steiner Algorithm and the Shamos MST Algorithm.

Table 1. Results from 100 point problems.

Exact Solution	Spanning Tree	Simulated Annealing	Percent Covered
6.255463	6.448690	6.261797	96.39%
6.759661	6.935189	6.763495	98.29%
6.667217	6.923836	6.675194	96.89%
6.719102	6.921413	6.721283	99.01%
6.759659	6.935187	6.763493	98.29%
6.285690	6.484320	6.289342	98.48%

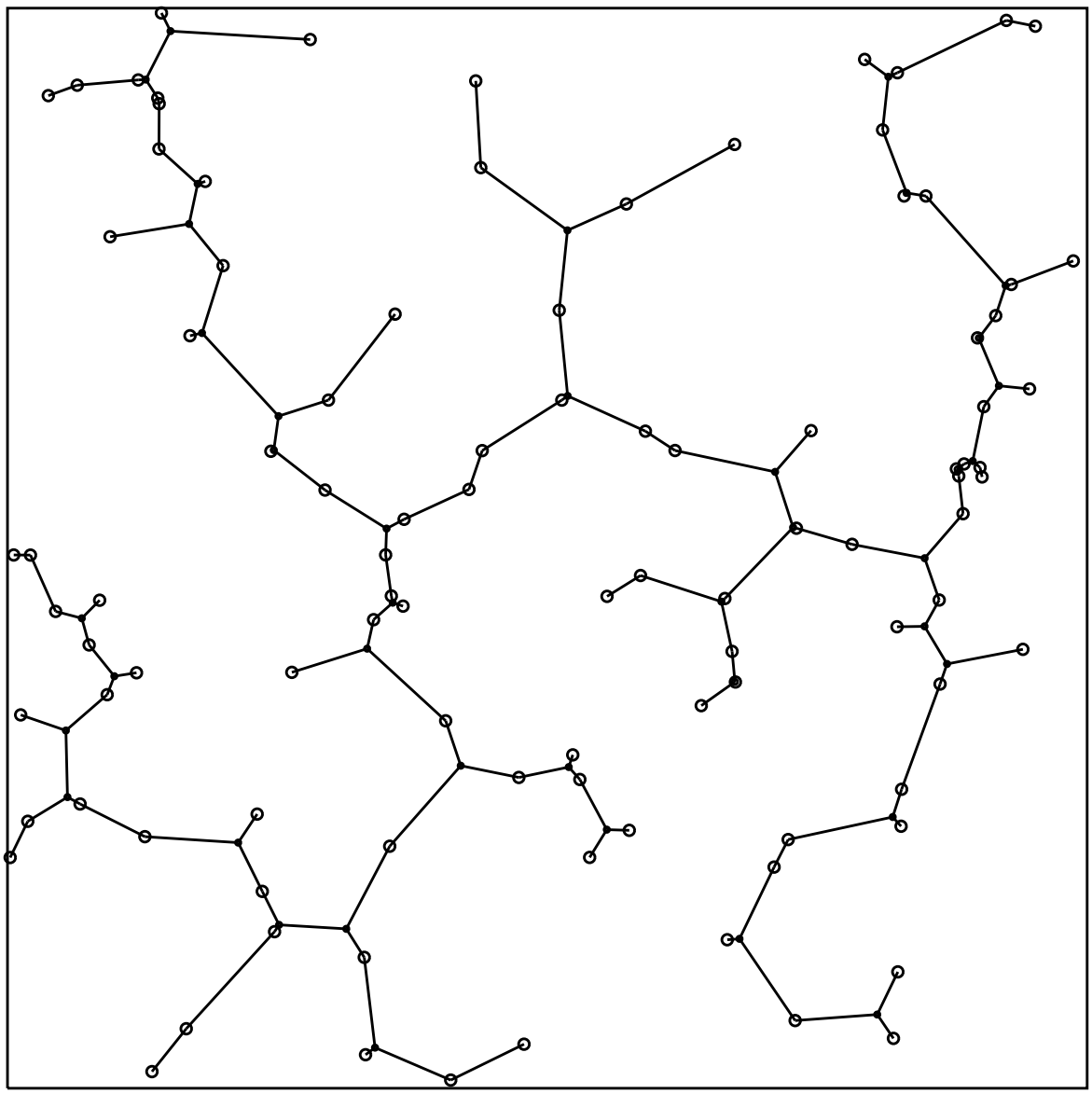

Figure 6. Simulated Annealing solution.

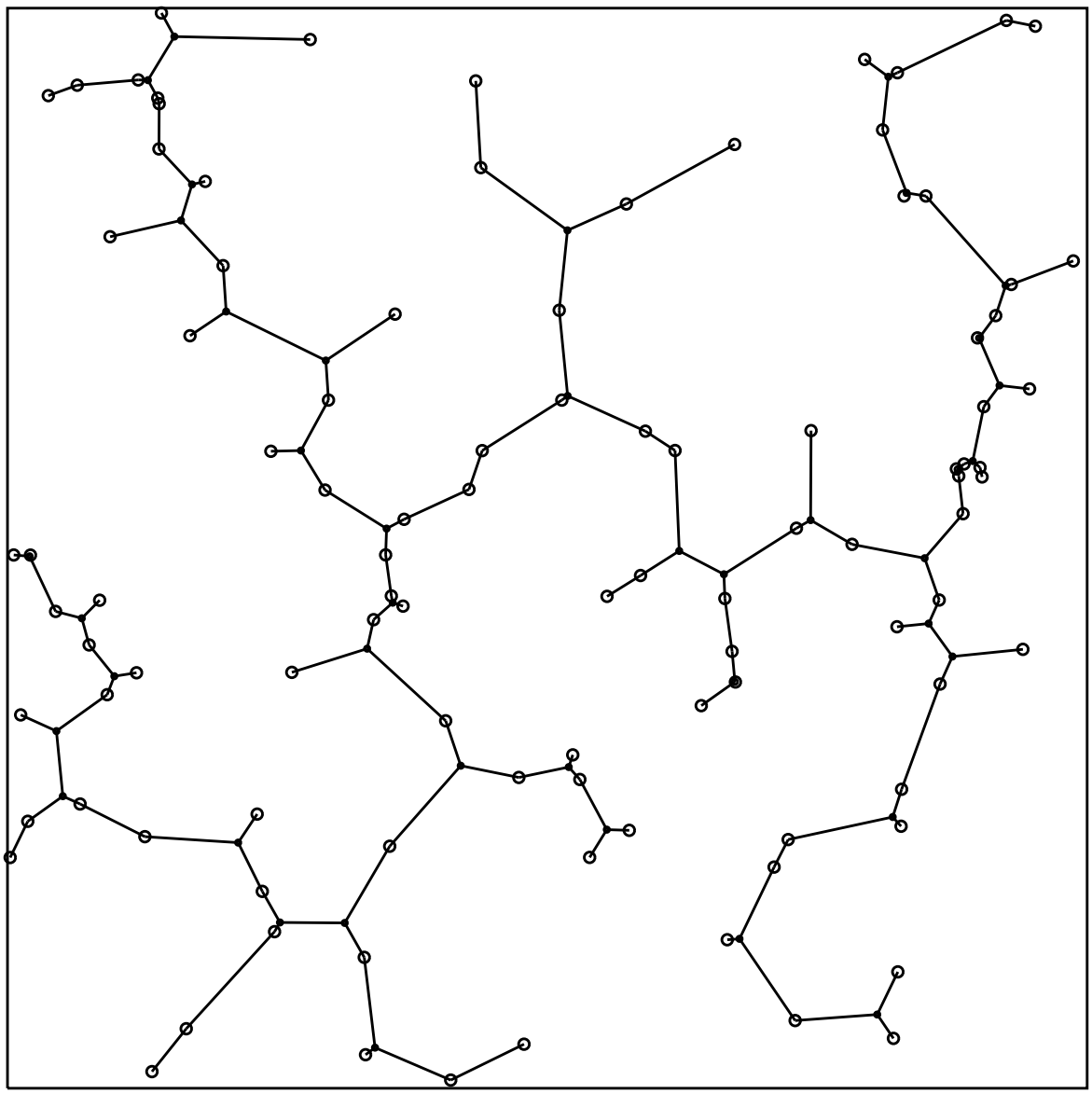

Figure 7. Exact solution.