The Rectilinear Steiner Tree Problem

Ben Wilfong

November 11, 2020

1 Introduction

The purpose of the minimum spanning Steiner tree problem is to minimize the total rectilinear distance between a set of nodes by adding additional "ghost nodes". This minimization is often done with the intention of decreasing cost, which can be measured in many different ways, but is assumed to be proportional to the total distance between nodes in the most basic statement of the problem. For this analysis, we will require rectilinear distances, which are defined as,

$$d(p_1, p_2) = |x_1 - x_2| + |y_1 - y_2|, \quad \text{where} \;\; p_i = (x_i, y_i).$$

By limiting the analysis to rectilinear distances and grid points, the problem becomes applicable to power plant and substation placement, cell tower placement, and other interconnected local networks largely constrained by the grids of cities and major roads. The problem is also applicable in electrical and computer engineering for electronic component layouts.

2 Notation

The following notation is guided by Zachariasen [1]. The n terminals in the plane we wish to connect will be collected in the set Z. In the rectilinear Steiner tree problem, these terminals are connected to other terminals and non-terminals via vertical and horizontal lines. There are three types of non-terminals, corner points (where exactly two segments intersect), T-points (where exactly three segments interest), and cross-points (where exactly four segments intersect). The number of segments that meet at a point is also refereed to as its order. Corner points, t-points, and cross-points have order two, three, and four respectively. T-points and cross-points are also referred to as Steiner points.

A line of segments is a sequence of one or more co-linear segments that meet at a non-terminal node but do not share any terminal nodes. A line of segments becomes a complete line when it is not properly contained in any other line of segments.

A corner point c is the intersection of two perpendicular complete lines. A complete corner at c is denoted (cu, cv), where cu and cv are the legs of the corner.

3 Hanan Grids

The Hanan grid $H(Z)$ for a set of points Z is the grid of horizontal and vertical lines through each terminal in Z. The set $I_{H(Z)}$ is the set of intersections in $H(Z)$, which is on the order of n^2 where n is the size of Z. Figure 1 shows an example of a Hanan Grid. Hanan grids serve as a strong starting point for exact solutions of the rectilinear Steiner tree problem.

An important result regarding Hanan grids is as follows:

> **Theorem 1:** "There exists a Steiner minimum tree for Z such that every non-terminal node belongs to $I_{H(Z)}$." [1]

Along with graph reduction in the Hanan grid, this theorem greatly decreases the number of optional solutions that exact algorithms must check.

3.1 Graph Reduction in the Hanan Grid

One graph reduction specific to Hanan Grids is the convex-hull reduction of Yang and Wing [2]. To apply this method, find a non-terminal node N with exactly two orthogonal edges e_1 and e_2. If the two edges that form a rectangle with e_1 and e_2 are present, the node N along with edges e_1 and e_2 can be removed. In some instances, the convex-hull reduction may have little effect, but for small random networks it typically reduces the grid quite well.

Further reduction can be achieved by removing second order non-terminal nodes that lie along a set of co-linear segments, keeping the end node, and replacing this set of segments with a single segment spanning its length. This method will later be referred to as line-reduction.

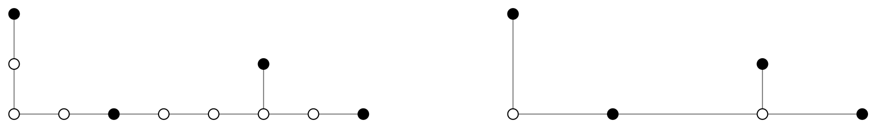
Figure 1: Example of Line Reduction

4 Exact Solutions

4.1 Spanning Tree Enumeration

One of the first algorithms for the rectilinear Steiner tree problem (rephrased as the graph Steiner tree problem) is the spanning tree enumeration algorithm presented by Hakimi [2]. This method utilizes the fact that the cheapest Steiner tree for a network with n terminal nodes never contains more than $n-2$ non-terminal nodes [3]. This algorithm simply creates all subsets S of at most $n-2$ non-terminals in the Hanan Grid, computes a minimal spanning tree of $Z \cup S$, and takes the minimum of these to be optimal.

While this algorithm is rather inefficient, it has the advantages that it requires no abstraction of the problem of interest. It is also guaranteed to yield an exact solution while being simple to implement. Due to it's inefficiencies, the spanning tree enumeration method can only be used for networks with a relatively small number of terminal nodes.

4.2 Dreyfus and Wagner

Drefus and Wagner's algorithm is a dynamic programming algorithm based on a decomposition theorem [2].

> **Theorem 2:** "An optimal Steiner tree of a set Z of terminals can be decomposed into three subsets A, B, and v such that from some nonterminal vertex u, the union of the optimal Steiner tress of $A \cup u$ and $B \cup u$ and a shortest path from v to u is an optimal Steiner tree for Z." [2]

This algorithm considers all subsets of terminals and considers every decomposition according to Theorem 1 [2]. By considering subsets of terminals in increasing size, the small optimal Steiner trees needed for decomposition have already been found and stored [2].

This algorithm is more efficient than the method of spanning tree enumeration mentioned before. While it does require some slight abstraction and is slightly more difficult to implement, its decrease in run time over spanning tree enumeration make it one of the most popular methods used in practice. The decrease time complexity of this algorithm allows it to be applied to networks with more terminal nodes, but it is still limited in comparison to heuristic algorithms. Dreyfus and Wager's algorithm is best suited for relatively small network problems in which an exact solution is desired.

5 Cost

In the most basic rectilinear Steiner tree problem, cost is equal to the sum of all distances in the minimum spanning tree, or a scalar multiple of this quantity. One addition to this metric is to have a non-terminal node cost $d^a w$, where d is the order of the non-terminal node and a and w are some constants [3]. Another modification could be that cost is non-linearly proportional to length. To relate this to application, a cable may need to be larger, and hence more expensive per unit length, if the distance it must cover is larger.

The methods presented in section four can both be easily modified to accommodate any set of these modifications. Instead of referencing the minimization on the size of each minimum spanning tree, the lengths of each segment or the degree of each non-terminal node, or any other cost metric, can be passed to a function that adds up the total cost of the tree. Minimization can then be made over this new cost function.

6 Example

The Hanan grid for the points

$$a(0,15),\ b(5,20),\ c(16,24),\ d(20,20),\ e(33,25),\ f(23,11),$$

$$g(35,7),\ h(25,0),\ i(10,3),$$

is

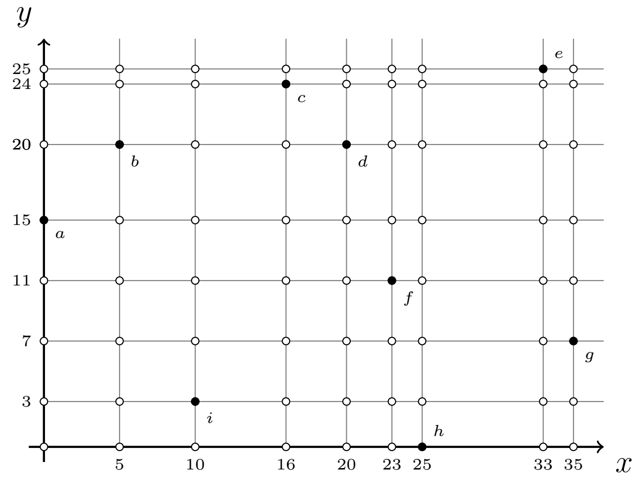

Figure 2: Hanan Grid

Reduction via the convex-hull method yields:

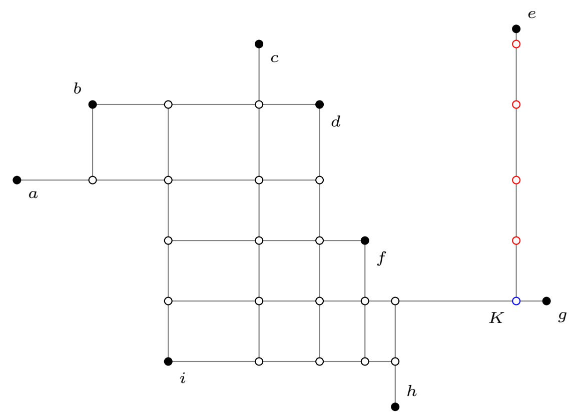

Figure 3: Results of Convex-Hull Reduction

The grid in Figure 3 can be further reduced by recognizing that the non-terminal nodes in red can be line reduced. Furthermore, the subsets S of size no more than $|Z|-2$ can be reduced to only those subsets that contain non-terminal node K. By inspection, it is necessary that this node is contained in the set of non-terminal nodes S to connect the terminal nodes e and g to the rest of the tree.

If it is assumed that stations are free, the optimal rectilinear Steiner tree for this problem is:

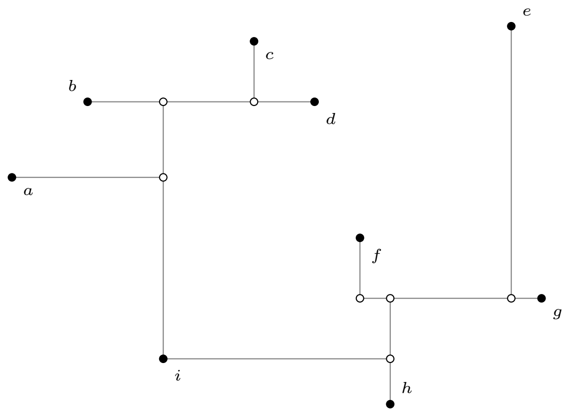

Figure 3: Solution Neglecting Non-Terminal Node Cost.

The cost of this tree is 116. If we take into account a station cost of $1.2d^{\frac{3}{2}}$, we get the optimal rectilinear Steiner tree:

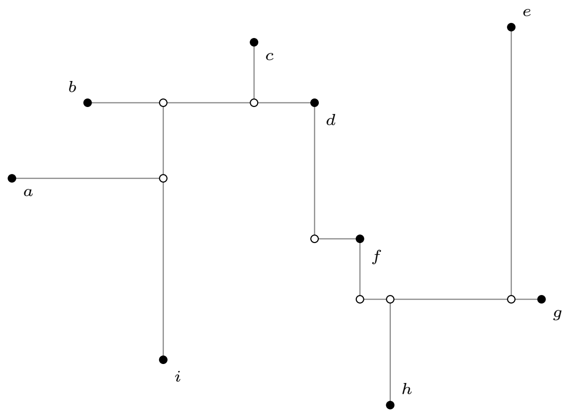

Figure 4: Solution Including Non-Terminal Node Cost.

The cost of this tree is 139.8. The cost of the solution optimized without regard to non-terminal node cost in Figure 3 including non-terminal node cost would be 140, only slightly greater than that of the tree in Figure 4 optimized to include this cost. The cost of the tree in Figure 4 ignoring non-terminal node cost is 117, only slightly more than that of the tree optimized to ignore this cost in Figure 3. Interestingly enough, the inclusion of non-terminal node cost does not play a large role in this application. However, in other instances of the problem, it very well could. The MATLAB implementation of the spanning tree enumeration algorithm used to find these solutions is provided as Appendix B.

7 Conclusion

The rectilinear Steiner tree problem is highly applicable problem in applied mathematics. Despite being an NP-Hard problem, exact solutions to small networks can be found rather easily. For larger networks, approximations through genetic programming and other heuristic methods are required.

A References

[1] X.Chen and D.Z. Du, Steiner Trees In Industry, Kluwer Academic Publishers, (2001).

[2] J. Ganley, Computing Optimal Rectilinear Steiner Trees: A Survey and Experimental Evaluation, Discrete Applied Mathematics 90, (1999).

[3] B. Fusary, 1991: The Steiner Tree Problem, (1991).

B MATLAB Implementation

```matlab
clc;clear;close all;

%%%  Terminal and Non-Terminal Nodes  %%%
T = {[0,15],[5,20],[16,24],[20,20],[33,25],[23,11],[35,7],[25,0],[10,3]};
NT = {[5,15],[10,7],[10,11],[10,15],[10,20],[16,3],[16,7],[16,11],...
    [16,15],[16,20],[20,3],[20,7],[20,11],[20,15],[23,3],[23,7],[25,3],...
    [25,7],[33,7]};

%%%  Initilizations
del = [0];
Cmin = 98754321;
Cmin2 = 987654321;
Amin = speye(3);
Amin2 = speye(3);

%%%  Application
for i=3:length(T)-2
    %%%  Create all subsets and filter other those not containing (33,7)
    clear del
    Si = cell2mat(nchoosek(NT,i));
    p=1;
    for j = 1:size(Si,1)
        for k = 1:2:size(Si,2)
            if ismember(33,Si(j,k:k+1)) ~= 1 || ismember(7,Si(j,k:k+1)) ~=...
                1
                del(p) = 1;
            else
                del(p) = 0;
            end
        end
        p = p+1;
    end
    idx = find(del==1);
    Si(idx,:) = [];
    %%%  Calculate the minium spanning tree for each subset
    for i=1:length(Si)
        C = 0;
        nodes = [cell2mat(T),Si(i,:)];
        nodes = reshape(nodes,[2,length(nodes)/2]);
        A = speye(size(nodes,2),size(nodes,2));
        for j=1:size(nodes,2)
            for k=1:size(nodes,2)
                if nodes(1,j) == nodes(1,k) | nodes(2,j) == nodes(2,k)
                    A(j,k) = 1;
                else
                    A(j,k) = 0;
                end
            end
        end
        [tree,pred] = graphminspantree(A);
         if any(isnan(pred))
             a=2;
         else
            for i = 1:size(tree,2)
                for j = 1:i
                    if tree(i,j) == 1
                        O = tree + tree';
                        Corder = sum(O(10:end,:)*ones(size(A,2),1).^...
                            (3/2))*1.2 + sum(O(1:9,1:9)*ones(9,1).^...
                            (3/2))*1.2;
                        C = C + abs(nodes(1,i)-nodes(1,j))+...
                            abs(nodes(2,i)-nodes(2,j));
                        C2 = C + Corder;
                    end
                end
            end
            %%%   Compare against previous best
            if C < Cmin
                Cmin = C;
                Amin = tree;
                Nmin = nodes;
            end
            if C2 < Cmin2
                Cmin2 = C2;
                Amin2 = tree;
                Nmin2 = nodes;
            end
         end
    end
end
```