

# The Feasible Solution Space for Steiner Trees

*Peter M. Joyce, F. D. Lewis, and N. Van Cleave*

Department of Computer Science  
University of Kentucky  
Lexington, Kentucky 40506

Contact: F. D. Lewis, lewis@cs.engr.uky.edu

## *Abstract*

*Since many optimization problems seem to require exponential time to find maxima or minima, much time and effort has been spent in attempting to diminish the size of feasible solution spaces for these problems in order to speed up the search for optimum solutions. We begin with a feasible solution space of size  $O(2^{2n(n-1)})$  for the rectilinear Steiner spanning trees over  $n$  points and show that there is a feasible solution space of size  $O(2^{n(2\log n-1)})$  which contains a shortest Steiner spanning tree by forcing trees to contain a minimum number of lines. Then by removal of some of the lines along the perimeter of the grid induced by the points, we bring the space size down to  $O(2^{n(2\log n-1)}/n^4)$ . In a simple example on four points this means reducing the size of the search space from over 16 million to 375.*

**1. Introduction.** Connecting a set of points in the plane with a minimum length collection of vertical and horizontal wires is called the *rectilinear Steiner problem*. These spanning trees have been used often in the placement and global routing phases of circuit design [Br77, LM84, NRT86] since timing considerations require routing paths of minimal length. Three shortest Steiner trees which span four points are provided in figure 1.

Optimum solutions to the problem have been investigated [Be90, YW72], but since the problem is NP-complete [GJ77], heuristics abound. Winter [Wi82] and Richards [Ri89] both provide excellent lists of heuristic solution efforts.

Cutting the size of a problem's feasible solution space often reduces search time to a large degree. This is the reason search space reduction has been

investigated for several NP-complete problems [HS74, La76, TT77]. Thus as the set of possible Steiner spanning trees over a set of points decreases, the chances of finding an optimum tree improve. This is true for exact as well as approximate algorithms. We intend to reduce this search space dramatically.

Our approach takes advantage of the very nature of shortest Steiner spanning trees. We first explore the layout shapes involved in their construction, and use this information to reorganize the lines used to construct Steiner spanning trees. This makes the trees simpler and reduces the feasible solution space which must be searched for optimum solutions.

First, we eliminate lines which are not connected to points. This is done when transforming the tree of figure 1a to that in 1b. It brings the number of lines needed for a Steiner spanning tree over  $n$  points down to  $2n$ . Then, when lines are arranged so that no two lines share a unique point as in figure 1c, Steiner spanning trees need only at most  $n$  lines. This is optimum and reduces the solution search space from nearly  $O(2^{2n(n-1)})$  to  $O(2^{n(2\log n-1)})$  trees. Eliminating some of the lines along the grid perimeter further reduces the space to a size of  $O(2^{n(2\log n-1)}/n^4)$ .

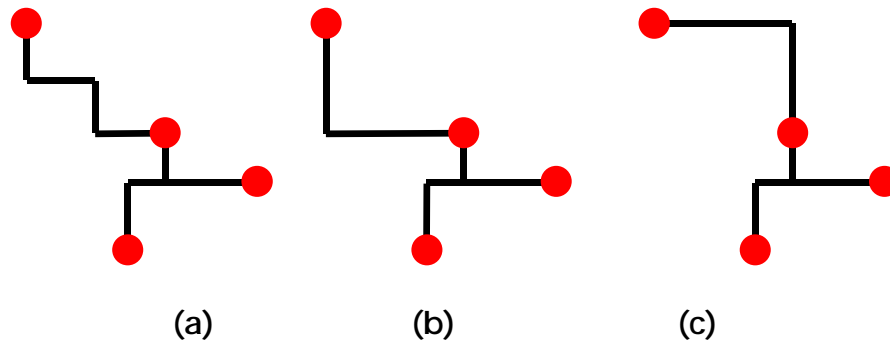


Figure 1 - Shortest Steiner Spanning Trees

## 2. Steiner Tree Components.

In order to find out exactly how to achieve the smallest search space for Steiner spanning trees, we first must precisely formulate their appearance. We begin with a definition.

**Definition.** *A **shortest rectilinear Steiner spanning tree** over a set of points is a connected collection of vertical and horizontal lines of minimum total length which spans the points.*

A grid can be induced by the set of points if lines are drawn in vertical and horizontal directions from each point. An example of such a grid over four points appears as figure 2.

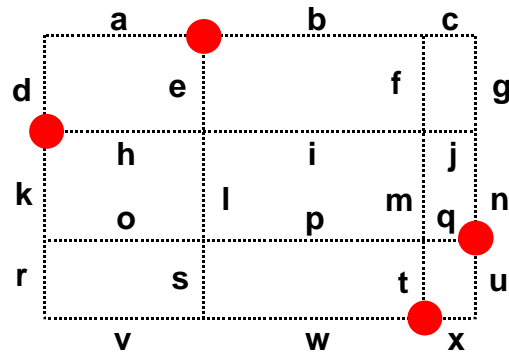


Figure 2 - Points and Their Induced Grid

This induced grid is very helpful since a theorem of Hanan [Ha66] uses it to provide a starting point for our search space.

**Theorem 1 (Hanan).** *A shortest rectilinear Steiner spanning tree over a set of points exists on the grid induced by the points.*

Now we have an upper bound of  $O(2^{2n(n-1)})$  on our search space since we need only consider combinations of grid edges. This bound for the grid of figure 2 comes out to be  $O(2^{24})$  or over 16 million possible configurations.

Whenever vertical and horizontal lines of a tree meet there is an *intersection*. Trees on the grid are made up of lines which begin and end with intersections, points, or both. Precise definitions of lines and their parts follow. (Our treatment below is for horizontal lines; that for vertical lines is similar.)

**Definition.** *A horizontal **line segment** of a tree is a sequence of adjacent horizontal grid edges within the tree which begins and ends with a point or an intersection and contains no other points or intersections.*

**Definition.** *A horizontal **line** of a tree is a sequence of adjacent horizontal line segments such that there exist no additional adjacent horizontal segments in the tree.*

Thus lines are composed of segments and segments are composed of grid edges in Steiner trees. The lowest horizontal line in the trees of figure one contains two segments, each of which is one grid edge.

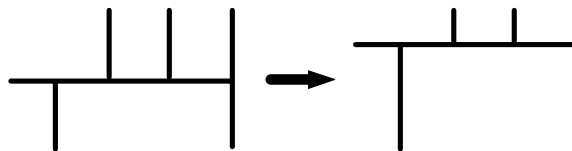
### 3. Pointless Line Elimination.

One of the major reasons that the search space for shortest Steiner spanning trees is so large is that almost any configuration of lines is contained within it. If we wish to go from one grid intersection to another, there are many possible equivalent ways. A particularly bad route is the staircase or zigzag route since it requires many lines and intersections. If we were able to only use direct routing when building Steiner spanning trees, we could avoid many configurations in favor of some canonical connections. Mandating that every line contain a point is the first step in this endeavor.

For some of the above reasons lines in a tree which do not contain points seem wasteful. They show up as intermediate lines in zigzag routes from one place to another on the grid. Our investigation of these *pointless lines* leads to a method for their consolidation and removal without harming the spanning tree. First we must explore the environment surrounding pointless lines.

**Theorem 2.** *A pointless line in a shortest Steiner spanning tree must have exactly as many lines intersecting it from one side as the other.*

**Proof.** Almost obvious. Assume that the tree fragment on the left in the picture below is part of a shortest Steiner spanning tree.



The horizontal line is pointless and has one more line intersecting it from above than from below. If we move the horizontal line up one grid edge (as in the right side of the above picture) we still have a tree which spans the original set of points. But we have shortened the tree by removing a portion of two lines while lengthening only one. This is impossible since we began with a shortest Steiner spanning tree.

We now need to note that if a pointless line has as many intersecting lines going to one side as the other, then it can be moved along the grid without changing the size of any tree it is in. And, since all pointless lines are of this type (due to

the last theorem), they all can be moved. We state this formally without proof in the next lemma.

**Lemma.** *Any pointless line in a shortest Steiner spanning tree can be moved to the end of its shortest intersecting line segment to either side without changing the size of the spanning tree.*

This means that we can move pointless lines around in a tree, and, if all goes well, combine them with others and make them disappear. Our next result provides the mechanism for their elimination.

**Theorem 3.** *There is a shortest rectilinear Steiner spanning tree over any set of points which contains no pointless lines.*

**Proof.** Our algorithm will be merely taking each pointless line in some shortest Steiner spanning tree, and, with the aid of the lemma, moving it the length of its shortest intersecting line segment. This does not change the tree size, so we still have a shortest spanning tree.

We claim that with each move we have reduced the number of pointless lines by at least one. Recall that a line segment is bounded by a line, a point, or both. Moving our pointless line thus either combines it with another line or places a point upon it. Both events reduce the number of pointless lines in the tree.

Now to determine the new search space size. We only need consider a vertical and horizontal line from each point on the grid.  $2n$  lines in all. Let's examine the horizontal lines. A point on the grid perimeter has  $n$  possible horizontal lines. A point one edge in has  $2(n-1)$  possible lines. One another edge in has  $3(n-2)$ . Multiplying these together gives us:

$$\prod_{k=1}^n k(n-k+1) \text{ or } (n!)^2$$

combinations of horizontal lines. Putting these together with the vertical lines gives us a feasible solution search space size of  $(n!)^4$ . Applying Stirling's approximation gives:

$$\left( \sqrt{2\pi n} \left[ \frac{n}{e} \right]^n \right)^4 \text{ or } 4\pi^2 n^2 \left[ \frac{n}{e} \right]^{4n}$$

which comes out to be  $O(e^{4n(\log n - 1)})$ .

Turning to our sample problem of figure 2 we find the following lines possible from the four points.

Point	Lines							
top	a	ab	abc	b	bc	e	el	els
left	d	dk	dkr	k	kr	h	hi	hij
right	u	nu	gnu	n	gn	q	pq	opq
bottom	x	wx	vwx	w	vw	t	mt	fmt

Multiplying out the combinations of a vertical and a horizontal lines from each point gives us  $(5*3)^4$  or more than 50,000 possible configurations to examine in our new search space.

#### 4. Point to Line Correspondence.

Forcing lines to go through points means that the only shortest Steiner spanning trees we need to consider now contain at most  $2n$  lines (two for each point). This brings the feasible solution space down to  $O(2^{4n(\log n - 1)})$ . If we require that every line have its own special point, then we need only consider shortest Steiner spanning trees consisting of at most  $n$  lines thus diminishing the search space a bit more. This involves associating each point with exactly one line.

To do this, we need to force lines to move whenever two share the same point. Again, if all works out we shall consolidate lines and reduce the number of lines in the tree to at most  $n$ . Consider figure 3. If we add to these shapes the other mirror images of 3(a), (c), and (d), we have shown all of the possible situations where two lines share the same point.

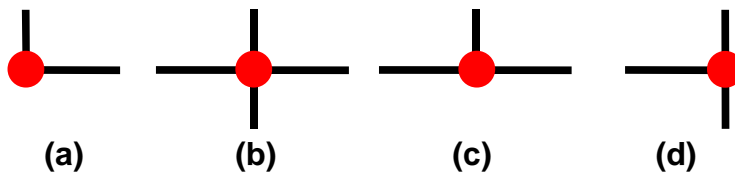


Figure 3 - Competition for a Point

Suppose that the horizontal lines in figure 3 do not contain any other point than that pictured. We would like to move these lines either up or down so that they each contain a different point than before. This is possible and the following lemma provides the translation algorithm.

**Lemma (Moving).** *If a line in a shortest Steiner spanning tree contains only one point and there is an intersecting line at that point, then the line can be moved at least one grid edge without changing the size of the tree.*

**Proof.** Again consider figure 3. Assume that the horizontal lines contain only the point pictured and extend to the left or right or both ways.

Much as with the pointless lines examined earlier, consider the number of vertical lines which intersect with one of the horizontal lines. Let *upcount* be the number of lines going up from the horizontal line and *downcount* equal the number going down. Note that the values of these cannot differ by more than 1. (Suppose that  $upcount = downcount + 2$ . Then we could shorten the spanning tree by moving the horizontal line up just as we did in the moving lemma for pointless lines.) For similar reasons the relationships between *upcount* and *downcount* in the following table are the only ones allowed for the various cases from figure 2 in shortest Steiner spanning trees.

case	relationship
(a) and (b)	$upcount = downcount$
(c), (d), and (e)	$upcount = downcount$ $upcount = downcount - 1$
(f), (g) and (h)	$upcount = downcount$ $upcount = downcount$

If  $upcount = downcount$  then the horizontal line may be moved along the vertical line in the picture without changing the size of the tree which contains it. If they differ by one, then the horizontal line may be moved in the direction opposite to the vertical line in the picture.

With this technical lemma we may now move lines which only contain one point and share it with another line. If after moving these lines now contain a point which they do not need to share, we may proceed easily to our desired result, restricting Steiner spanning trees to no more than  $n$  lines.

**Theorem 4.** *For any set of  $n$  points there is a shortest Steiner spanning tree which contains at most  $n$  lines.*

**Proof.** Most of our work is done. Given a shortest Steiner spanning tree over a set of points where each line does pass through a point, we shall use the moving lemma to modify the tree and achieve a tree with no more than  $n$  lines.

We first select a leaf and do a depth-first search on our tree. As each line is encountered we mark it and a point upon it. (We are associating a point uniquely with each line.) This is carried out as in the algorithm below.

**match**(tree, ourline)

*PRE: ourline  $\in$  tree (a shortest Steiner spanning tree)  
ourline is unmarked*

*POST: subtree below ourline has  $|lines| \leq |points|$*

*if ourline contains no unmarked points then*

*move ourline one segment via the moving lemma*

*if ourline is still unmarked then*

*mark ourline and an unmarked point upon it*

*for each unmarked intersecting newline*

*do match(tree, newline)*

Since we are doing depth-first search in a tree, the only marked intersecting line on ourline is the parent of ourline. Thus the only problem which could arise is when ourline contains only one point, and it has been marked. When we use the moving lemma to translate ourline up or down one segment (the shortest one) there must be either be a line or a point or both at the intersection we moved to. If these are unmarked, then after assimilating them with ourline, we now have a point to associate with ourline since each line passes through a point in the original tree.

If we reach a marked line or point, we have traversed a segment on the parent of ourline. In this case we cannot reach only a marked point since the point associated with the parent of ourline was the original marked point on ourline. Joining ourline to the marked line at the intersection does solve our problem since we have associated ourline with another line's point.

Now we have a choice of one line from each point. A corner point has only  $2n$  possible lines, while a central point may have  $2(n/2)^2$  possible lines. So, the size of the feasible solution search space depends upon the data set itself. If for each point we sum the possible horizontal and vertical lines we come up with a

search space size of the sum of  $n^2$  products each containing either the number of horizontal or vertical lines for each of the  $n$  points. The worst case is  $(n!)^2 n^2$  or after applying Stirling's approximation,  $O(2^{n(2\log n - 1)})$ .

Life is easier with our four point example. This reduction to one line per point brings the search space for our example of figure 2 down to  $(5 + 3)^4$  or a little more than 4000 combinations.

## 5. Sideline Removal.

At this point we know that there are shortest Steiner spanning trees which contain no more than  $n$  lines and consequently have no more than  $n-1$  intersections. This brings our feasible solution space to size  $O(2^{n(2\log n - 1)})$ .

If we are dealing with  $n$  by  $n$  grids (which happens when no points share a coordinate) then we are able to reduce our search space a bit more. First we note that lines on the edge of the grid are restricted.

**Theorem 5.** *Lines along the grid perimeter in shortest Steiner spanning trees in which no points have coordinates in common must have a point at one end*

**Proof.** If a line along the grid perimeter spans a point then it must have lines going into the grid at each end. Moving this line an edge into the grid shortens the spanning tree. This of course is impossible.

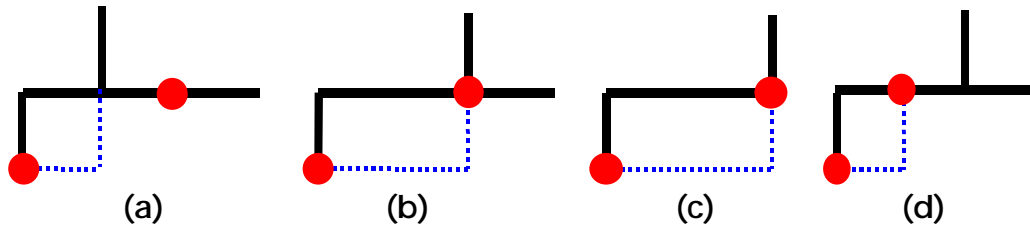
Not allowing perimeter lines to span points cuts the number of lines along the perimeter to  $n$ . Now we shall take this further by forcing one of the points on the perimeter of the grid to not have a line along the perimeter.

**Theorem 6.** *For any set of  $n$  points which have no coordinates in common there is a shortest Steiner spanning tree which does not have any lines along a particular side of the grid perimeter,*

**Proof.** Assume that we have a shortest Steiner spanning tree with one line through each point and no points with common coordinates. Consider the point on the left side. Because of theorem 5 we know that if there is a perimeter line then it begins at the point. Assume that the line from it goes up along the perimeter.

The first thing to note is that we must have a configuration such as one of those below since if there were another intersection on the perimeter line we could move the perimeter line in and shorten the spanning tree. Thus

the line from the leftmost point goes up along the perimeter and then into the grid (to the right). The first intersection on this line must occur before, at, or after the point on that line.



In cases (a), (b), and (c) we can flip the line over to the dotted line and have a shortest spanning tree with exactly  $n$  lines as before. In case (d) we may flip over the corner as indicated and apply the algorithm of theorem 4 to move lines until we again have only one line per point.

Now we have a shortest Steiner spanning tree with exactly  $n$  lines and no lines on the left edge of the grid.

Let us examine what we have accomplished for shortest Steiner spanning trees over points which do not share coordinates. Theorem 6 allows us to omit the lines on one side of the grid from consideration. In our example of figure 2 we eliminate the left edge. This means getting rid of  $d$ ,  $dk$ ,  $dkr$ ,  $k$  and  $kr$ . We may also discard lines containing  $a$ ,  $o$ , and  $v$ . (That is  $a$ ,  $ab$ ,  $abc$ ,  $opq$ ,  $vw$  and  $vw$ .)

In addition we may employ theorem 5 to omit perimeter lines which cross or span a point. Again, in figure 2 these add  $nu$ ,  $gnu$ , and  $wx$  to our discard list. This completes our reduction of the feasible solution space for rectilinear Steiner spanning tree. The general bound is at last  $O(2^{n(2\log n-1)}/n^4)$ . In our example of figure 2 we now only have  $3(5)^3$  or 375 line combinations.

## 6. Conclusion.

By confining the lines used to construct rectilinear Steiner spanning trees to those which contain the points to be spanned and limiting the number of these lines to the number of points, we have made a dramatic reduction in the size of the feasible solution space. In fact, this reduction was from Hanan's bound of  $O(2^{2n(n-1)})$  to  $O(2^{n(2\log n-1)})$ . Careful examination of the perimeter lines required in shortest Steiner spanning trees brought this bound down even further, to  $O(2^{n(2\log n-1)}/n^4)$ . This makes it possible to find Steiner spanning trees much more quickly using heuristic as well as exact methods.

## References

- Be90** Bern, M. "Faster Exact Algorithms for Steiner Trees in Planar Networks." *Networks*, 20 (1990) 109 - 120.
- Br77** Breuer, M. A. "Min-Cut Placement." *Design Automation and Fault-Tolerant Computing*, 1 (1977), 343 - 362.
- GJ77** Garey, M. R., and D. S. Johnson. "The Rectilinear Steiner Tree Problem is NP-Complete." *SIAM Journal of Applied Mathematics*, 32 (1977), 855 - 859.
- Ha66** Hanan, M. "On Steiners Problem with Rectilinear Distance." *SIAM Journal of Applied Mathematics*, 14:2 (1966), 255 - 265.
- HS74** Horowitz, E. and S. Sahni. "Computing partitions with applications to the knapsack problem." *Journal of the Association for Computing Machinery*, 21 (1974), 277 - 292.
- La76** Lawler, E. L. "A note on the complexity of the chromatic number problem." *Information Processing Letters*, 5 (1976), 66 - 67.
- LM84** Li, J. T., and M. Marek-Sadowska. "Global Routing for Gate Arrays." *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, CAD-3:4 (1984), 298 - 308.
- NRT86** Ng, A. P-C, P. Raghavan, and C. D. Thompson. "Experimental Results for a Linear Program Global Router." *Proceedings of the 1986 ACM Design Automation Conference*, (1985).
- Ri89** Richards, D. "Fast Heuristic Algorithms for Rectilinear Steiner Trees." *Algorithmica*, 4 (1989), 191 - 207.
- TT77** Tarjan, R. E. and A. E. Trojanowski. "Finding a maximum independent set." *SIAM Journal on Computing*, 6 (1977), 537 - 546.
- Wi87** Winter, P. "Steiner Problem in Networks: A Survey." *Networks*, 17 (1987), 128 - 167.
- YW72** Yang, Y. Y. and O. Wing. "Optimal and Suboptimal Solution Algorithms for the Wiring Problem." *Proc. IEEE Int. Symp Circuit Theory*, (1972), 154 - 158.