A New Approach to the Rectilinear Steiner Tree Problem

Jan-ming Ho
Department of EECS
Northwestern University
Evanston, Illinois

Gopalakrishnan Vijayan *and* **C. K. Wong**
IBM Research Division
Thomas J. Watson Research Center
Yorktown Heights, New York

ABSTRACT: We discuss a new approach to constructing the rectilinear Steiner tree (RST) of a given set of points in the plane, starting from a minimum spanning tree (MST). The main idea in our approach is to determine L-shaped layouts for the edges of the MST, so as to maximize the overlaps between the layouts, thus minimizing the cost (i.e., wire length) of the resulting RST. We describe a linear time algorithm for constructing a RST from a a MST, such that the RST is optimal under the restriction that the layout of each edge of the MST is an L-shape. The RST's produced by this algorithm have 8-33% lower cost than the MST, with the average cost improvement, over a large number of random point sets, being about 9%. The running time of the algorithm on an IBM 3090 processor is under 0.01 seconds for point sets with cardinality 10. We also discuss a property of RST's called stability under rerouting, and show how to stabilize the RST's derived from our approach. Stability is a desirable property in VLSI global routing applications.

1. Introduction

Given a set of points on the plane, the Rectilinear Steiner Tree (RST) problem is to find the rectilinear tree in the plane, of minimum total length, which connects the given set of points. Rectilinear Steiner Trees have many applications in VLSI physical design. They have been used to determine global routes for nets during the global routing phase [10]. Timing considerations make it desirable to minimize the cost (wire length) of the Steiner trees used for global routing of nets. Rectilinear Steiner trees have also been used in other VLSI physical design applications, such as estimating the wire length of nets during the placement phase of physical design [2,3].

The problem of constructing the minimum cost RST has been shown to be NP-complete [4]. The RST problem has been studied extensively and many heuristic algorithms have been proposed. Hwang [7] has shown that the ratio of the cost of a rectilinear MST to that of an optimal RST is no greater than 3/2. Therefore, the rectilinear MST is a suitable starting point for deriving good RST's. Many heuristic algorithms [5,6,9,10] take this approach. They start with a sequence of the input points and edges as given by a rectilinear MST algorithm, and insert points and edges from this sequence into a growing RST. Local optimization, such as finding the shortest path from the new point to the partial RST, is performed while inserting a new edge from the rectilinear MST sequence.

We introduce a global approach, in which the rectilinear MST is transformed into a RST, which is optimal under the condition that the layout in the RST of each edge of the MST is an L-shape.

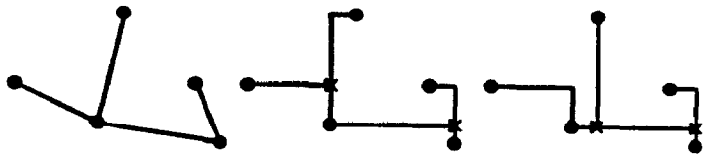

Figure 1. A MST and two L-RST's of the MST

Consider a rectilinear MST T of a set S of n points. An edge $e = (i,j)$ is said to be *nondegenerate* if the two points i and j do not lie on the same horizontal or vertical line. Each such nondegenerate edge has two distinct *L-shaped layouts*, which are *flips* of each other. Suppose we select a L-shaped layout for each MST edge, and merge the resulting overlaps among the layouts. The resulting structure is a RST, and since it is obtained using L-shapes for the MST edges, we define it to be a *L-RST*. Figure 1 shows an example of a rectilinear MST and two different L-RST's derived from it, as a result of selecting different sets of L-shaped layouts for the edges of the MST.

The main idea behind our approach is to select that set of L-shaped layouts for the nondegenerate edges that *maximizes the overlaps* between the layouts, thus *minimizing the cost* of the resulting L-RST. Note that there are 2^m L-RST's, where m is the number of nondegenerate edges of the MST. The RST algorithm described in [10], uses a similar approach by heuristically trying to maximize the overlaps among the layouts of the MST edges. Our approach is more global and yields optimal RST's under the restriction of L-shaped layouts. Given an input MST, our minimum cost L-RST algorithm runs in $O(n)$ time. We describe this algorithm and analyze its time complexity in section 4.

The input MST to our L-RST algorithm must have a special property called *separability*. We describe this interesting property in section 2, and show that it is possible to construct a separable MST by only a slight modification of Kruskal's MST algorithm. In section 3, we prove that any given point in a rectilinear MST can have at most 8 neigbors, and that at most 6 of these neighbors form nondegenerate edges with the given point. This degree-boundedness property is used to derive the $O(n)$ time bound of the L-RST algorithm. Separbility and degree-boundedness are interesting properties in their own right, and may have other useful applications. In section 5, we discuss a useful property of RST's called *stability under rerouting*, and briefly describe how to stabilize an L-RST. We discuss experimental results in section 6.

2. Separability of Rectilinear MST's

Given any edge $e = (i,j)$ of a rectilinear MST, the *rectangle* with the points i and j on its opposite corners is defined as the *enclosing box of the edge e*. A rectilinear MST is said to be *separable* if the enclosing boxes of any two nonadjacent edges do not intersect or overlap. As a consequence of this definition, in a separable MST, the L-shaped layouts of two edges which do not share a common point of the MST cannot intersect or overlap. The removal of an edge from a separable MST results in two *separated* subtrees, in the sense, that an edge of one subtree cannot overlap or intersect an edge of the other subtree, no matter which L-shaped layouts are used for the edges. This property allows us to design a dynamic programming algorithm for constructing a minimum cost L-RST. It is also used in the stabilization algorithm of section 5.

Not all rectilinear MST's are separable. In Figure 2, we show a nonseparable and a separable MST of the same set of points. The first MST shown in the figure is nonseparable because the enclosing boxes of the two nonadjacent edges e_1 and e_2 intersect each other.

We give below a simple algorithm for contructing a separable rectilinear MST of a given point set S. We use $x(i)$ and $y(i)$ to denote the $x-$ and y-coordinates of a point i. We denote the rectilinear distance between two points i and j as $D(i,j)$.

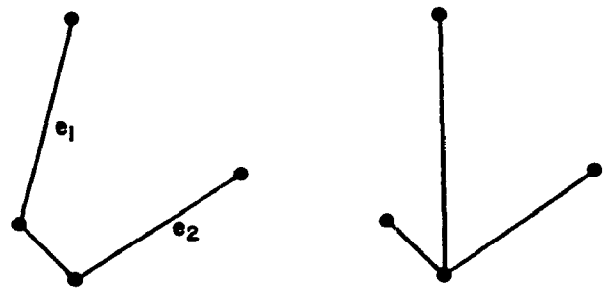

Figure 2. A Nonseparable MST and a Separable MST

Algorithm SMST:

1. Construct the complete graph G_c of the point set S.
2. For each edge (i,j) of G_c, construct $4tuple(i,j) = (D(i,j), -|y(i) - y(j)|, -\max(y(i), y(j)), -\max(x(i), x(j)))$ Assign $4tuple(i,j)$ as the weight of the edge (i,j).
3. Run Prim's MST algorithm [1] on the graph G_c, using the 4-tuples as the weights, to generate a minimum spanning tree T_s.

Note that in comparing two 4-tuples:
$(a_1, b_1, c_1, d_1) < (a_2, b_2, c_2, d_2)$, if and only if, (a_1, b_1, c_1, d_1) precedes (a_2, b_2, c_2, d_2) in nondecreasing lexicographic order. The last 3 elements of the 4-tuple weights are used only to break ties between two edges of the same rectilinear length. Therefore the following lemma is obvious.

Lemma 1: The rectilinear spanning tree T_s generated by Algorithm SMST is a minimum spanning tree.

It remains to show that T_s is separable.

Theorem 1: The rectilinear MST T_s generated by Algorithm SMST is separable.

Proof: The proof is by contradiction. Suppose there exists nonadjacent edges (i_1,j_1) and (i_2,j_2) in T_s whose enclosing boxes B_1 and B_2 respectively have a nonempty intersection. Since T_s is a tree, the edges (i_2,j_1) and (i_1,j_2) of the complete graph are not present in T_s. Let us examine the different intersection patterns between the boxes B_1 and B_2. Since the tree T_s is connected, assume without loss of generality that the points i_1 and i_2 have a path between them in T_s which does not contain the edges (i_1,j_1) or (i_2,j_2).

To eliminate some of the patterns, note that the points i_2 and j_2 cannot lie either on the boundary or in the interior of B_1, and that the points i_1 and j_1 cannot lie either on the boundary or in the interior of B_2. Suppose i_2 is on the boundary or interior of B_1. Then replacing the edge (i_1,j_1) by the edge

(i_2, j_1) , results in a rectilinear spanning tree whose cost is strictly less than that of T_s. This is a contradiction that T_s is a MST. A similar argument holds for the other cases.

Assume that $y(i_1) > y(i_2)$. The case $y(i_1) \leq y(i_2)$ is symmetrical. It now suffices to consider the five intersection patterns shown in Figure 3.

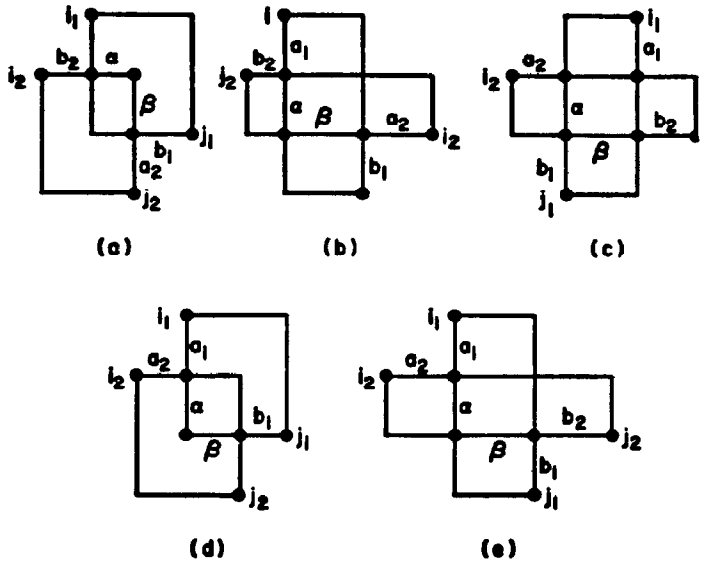

Figure 3. Five Intersection Patterns of Enclosing Boxes - I

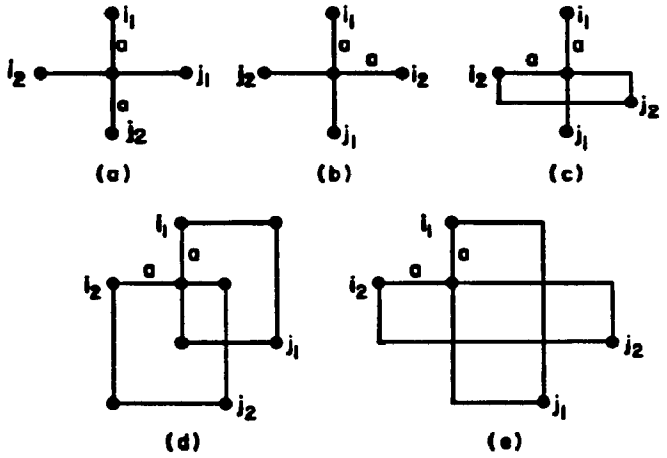

Figure 4. Five Intersection Patterns of Enclosing Boxes - II

The symbols a, b, α and β are used to denote the rectilinear lengths marked in the figure. Since T_s is a MST, it must be that $D(i_1, j_2) \geq D(i_2, j_2)$, and $D(i_2, j_1) \geq D(i_1, j_1)$. In cases (a) and (b) of Figure 3, simple arithmetic shows that these two inequalities imply that $a_1 \geq a_2 + \alpha + \beta$, and $a_2 \geq a_1 + \alpha + \beta$, which in turn implies that $a_1 = a_2 = a$, and $\alpha = \beta = 0$. Therefore cases (a) and (b) of Figure 3 reduce respectively to cases (a) and (b) of Figure 4. Similarly it is easy to show that in case (c) of Figure 3, we must have $a_1 = a_2 = a$ and $\beta = 0$. In cases (d) and (e), we have only $a_1 = a_2 = a$. These five cases of Figure 3 thus reduce to the corresponding five cases of Figure 4. In all five cases, we now have $D(i_1, j_2) = D(i_2, j_2)$, and $D(i_2, j_1) = D(i_1, j_1)$.

Consider case (a) of Figure 4. Compare the 4-tuple weights of the two edges (i_1, j_2) and (i_2, j_2) of G_c. We have $D(i_1, j_2) = D(i_2, j_2)$, $- |y(i_1) - y(j_2)| = - |y(i_2) - y(j_2)|$ but $- \max(y(i_1), y(j_2)) < - \max(y(i_2), y(j_2))$ Hence, $4tuple(i_1, j_2) < 4tuple(i_2, j_2)$. Replacing the edge (i_2, j_2) by the edge (i_1, j_2) , we get a spanning tree of G_c of lesser 4-tuple weight than T_s, which is a contradiction. A similar argument holds for the remaining cases (b) - (e) of Figure 4. This completes the proof of the theorem. ▪

The time complexity of Algorithm SMST is the same as that of running Prim's Algorithm on the complete graph G_c, which is $O(n^2)$. We note here that, it is possible to derive a $O(n \log n)$ algorithm for constructing separable MST's, using the notion of L_1 Voronoi diagrams [8]. However, the $O(n^2)$ algorithm presented in this paper, is more practical for point sets of cardinality ≤ 100.

3. Degree-Boundedness of MST's

We now show that the degree of any point in a rectilinear MST is bounded by a constant.

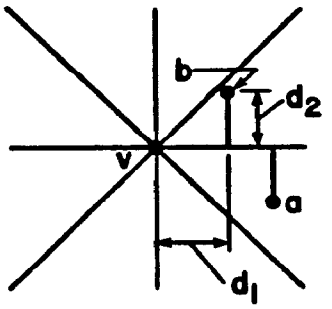

Figure 5. Proof of Lemma 2

Lemma 2: Let v be a point in a rectilinear MST T. Consider the *4* regions defined by the two $\pm$ 45° lines passing through the point. There can be at most one neighbor of v (in the MST T) located in the *interior* of each of these regions.

Proof: Suppose there are two neighbors a and b of the point v in the interior of the same region, say the region shown in Figure 5. Note that the interior does not include the two $\pm$ 45° lines. Without loss of generality, let the x-coordinate of the point a be $\geq$ the x-coordinate of the point b, as shown in Figure 5. Consider the two distances d_1 and d_2 marked n the figure. Clearly $d_2 < d_1$, and thus $D(a,b) < D(a,v)$. This

means that if we remove the edge (a,v) from the tree T and add the edge (a,b), we will get a new spanning tree of the same set of points, which has a smaller total cost than T. But this contradicts that T is a MST. ▪

Theorem 2: (1) There can be at most 8 neighbors for any point v in a rectilinear MST T. (2) At most 6 of these neighbors can form nondegenerate edges with v.

Proof: From the previous lemma there can be at most 4 neighbors of v located in the interiors of the four regions defined by the $\pm 45°$ lines. In addition, neighbors can be located on the $\pm 45°$ lines. Suppose a and b are two neighbors of v located on the same side of v on one of the two $\pm 45°$ lines, with b being the closer point to v. Then $D(a,b) < D(a,v)$, which contradicts that (a,v) is an edge of the MST. Therefore at most one neighbor of v can be located on each of the 4 segments of the $\pm 45°$ lines. Thus v can have at most 8 neighbors. We skip the proof of the second part of the lemma, that at most 6 neighbors can form nondegenerate edges with v. ▪

Theorem 2 essentially says that any point in a rectilinear MST has constant degree. This result is used in bounding the time complexity of the L-RST algorithm described in the next section.

4. Algorithm for Mimimum Cost L-RST

We now present our linear time algorithm for obtaining a minimum cost L-RST from a given separable MST T of a point set S. The algorithm has to select for each nondegenerate edge one of its two possible L-shaped layouts, so as to maximize the total amount of overlap, thus minimizing the total cost of the L-RST that results from merging the overlaps.

Suppose we root the input separable MST T at any point r, resulting in the rooted tree T_r. For each point $v \in S$, denote the subtree rooted at v as T_v, and the subtree (a star) induced by v and its neighbors as T_v^S. For a nonroot point v, we denote the edge incident from its parent on v as e_v.

For each nonroot point v, we let T_v^+ denote the subtree $T_v \cup e_v$. If e_v is nondegenerate, then we let $\Phi_u(v)$ denote the L-RST of the subtree T_v^+, which has the *minimum cost* among all L-RST's of T_v^+, *in which the layout of the edge e_v is constrained to be the upper L-shape of the enclosing box of e_v*. Similarly, let $\Phi_l(v)$ denote the *constrained minimum cost L-RST* of the subtree T_v^+, in which the layout of e_v is *constrained* to be the *lower L-shape* of the enclosing box. If e_v is degenerate, or if v is the root r (in which case there is no incoming edge), we let $\Phi_l(v) = \Phi_u(v) = \Phi(v)$.

We now show how to recursively compute $\Phi_u(v)$ and $\Phi_l(v)$.

For a leaf point v, $\Phi_u(v)$ is simply the the upper L-shape of e_v, and $\Phi_l(v)$ is the lower L-shape of e_v.

For a nonroot nonleaf point v, $\Phi_u(v)$ can be computed recursively as follows: Let w_i: $i = 1,2, \ldots, d$ be the d children of the point v in the rooted tree T_r. Recursively compute the minimum cost L-RST's $\Phi_u(w_i)$ and $\Phi_l(w_i)$ of the subtrees $T_{w_i}^+$, $i = 1,2, \ldots, d$. Suppose we arbitrarily select either $\Phi_u(w_i)$ or $\Phi_l(w_i)$ for each $T_{w_i}^+$, $i = 1,2, \ldots, d$. We fix the layouts of the edges $e_{w_i} = (v, w_i)$ to be upper L-shapes or the lower L-shapes as given by the selection of the L-RST's. We also fix the layout of e_v to be its upper L-shape. The result is a L-RST of the subtree T_v^+, in which the layout of e_v is constrained to be its upper L-shape. Since the input MST T has the separability property, the selected L-RST's of the subtrees $T_{w_i}^+$, $i = 1,2, \ldots, d$, do not overlap or intersect each other. Therefore the total amount of overlap occurring in the resulting L-RST of T_v^+ is the sum of the following:

1. the amount of overlap in the star T_v^S,
2. for $i = 1,2, \ldots, d$, the amount of overlap in the selected L-RST $\Phi_l(w_i)$ or $\Phi_u(w_i)$.

The cost of the resulting L-RST of T_v^+ is simply the sum of the rectilinear lengths of the edges of the tree minus the total amount of overlap. Therefore $\Phi_u(v)$ can be computed by fixing the layout of e_v to be its upper L-shape, and enumerating the 2^d different combinations of selecting one of $\Phi_u(w_i)$ or $\Phi_l(w_i)$ for each subtree $T_{w_i}^+$. Each combination corresponds to a L-RST of T_v^+, in which the layout of e_v is constrained to be its upper L-shape. Select $\Phi_u(v)$ to be the *minimum cost L-RST* among these 2^d enumerated L-RST's of T_v^+. $\Phi_l(v)$ is similarly computed.

The procedure is similar at the root r, except there is no incoming edge, and therefore there is only one constrained L-RST to compute at the root, which we denote as $\Phi(r)$. The algorithm outputs $\Phi(r)$ as the minimum cost L-RST of the separable MST T.

The following procedure computes the minimum cost L-RST $\Phi_x(v)$, where x is either the upper L-shape u or the lower L-shape l.

Procedure MinCost-L-RST ($\Phi_x(v)$):

1. If v is a leaf in T_r, return $\Phi_x(v)$ as the x L-shape of e_v.
2. If v is not a leaf and not the root, then fix the layout of e_v to be the x L-shape.
3. Recursively compute $\Phi_u(w_i)$ and $\Phi_l(w_i)$ for the children w_i, $i = 1,2, \ldots, d$ of v.
4. For each of the 2^d different ways of selecting either the L-RST $\Phi_u(w_i)$ or $\Phi_l(w_i)$, compute the total amount of overlap in the resulting L-RST of T_v^+, as the sum of the overlaps in the star T_v^S and the amount of overlaps in the selected L-RST's of the children. Return $\Phi_u(v)$ to be that L-RST that maximizes the total amount of overlap.

$\Phi(r)$ computed at the root yields the minimum cost L-RST of the separable MST T. An inductive proof of correctness can be easily established using the observation about the sum of overlaps, which itself is a consequence of the separability of the tree T. From Theorem 2, the degree of any point in the MST T is at most 8, and at most 6 of the incident edges are nondegenerate. Therefore the total number of combinations enumerated in the computations of $\Phi_l(v)$ and $\Phi_u(v)$ together is at most 2^6 which is a constant. Note that this observation is true even if we use the degree bound of 8 from part 1 of Theorem 2, instead of the degree bound of 6 from part 2. Summing this constant over all the points of the MST T, we conclude that the time complexity of the algorithm is $O(n)$.

Theorem 3: Procedure MinCost-L-RST ($\Phi(r)$) computes the minimum cost L-RST of the separable MST T in linear time.

5. Stability under Rerouting

A RST is said to be *stable under rerouting*, if there is no pair of degenerate or non-degenerate L-shaped segments, whose enclosing boxes intersect or overlap, except touching at a common end point (if any) of the two segments.

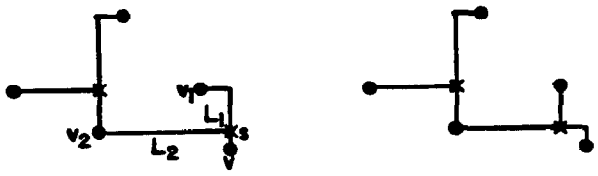

Figure 6. An Unstable L-RST and a Stable L-RST

In Figure 6, we show a RST of the MST of Figure 1, which is unstable because the enclosing boxes of non-degenerate L-shape L_1 and the degenerate L-shape L_2 overlap, However, if we flip the L-shape L_1 and merge the resulting overlap, the result is the second RST shown in the figure, which is stable.

No matter how we reroute a L-shaped segment, in a stable RST, within its enclosing box, no overlaps or crossings will occur, and the RST will essentially remain unchanged. A stable RST corresponds to a local minimum under the rerouting operation. Stability is a useful property to have in many applications. In [10], the *switchable* property of a non-degenerate L-shaped segment is used to explore alternate global routes for L-shaped segments of nets. Here, it would be very useful to have a stable RST for two reasons. First, a stable RST is locally optimal under rerouting. Second, a stable RST guarantees no further crossings or overlaps, when exploring the alternate routes for an L-shaped segment, within its enclosing box. This avoids complications in the rerouting phase of the global router.

A RST may be stabilized by repeatedly picking pairs of L-shapes which can be made to cross or overlap by flipping either or both of them, and by eliminating the crossing or the overlap. When a crossing is eliminated or an overlap is merged, the two old L-shapes get replaced by new L-shapes, and it is not obvious whether this process will eventually converge to result in a stable RST. Even if it does, the sequence of pairs of L-shapes that we may have to process, may be very long. We briefly describe an alternative procedure that can, in linear time, stabilize a L-RST. The key ideas used by the algorithm are again the separability and bounded degree properties of the original MST.

The stabilization algorithm processes the neighborhood of each original MST point in the L-RST. When processing a point, we examine all the non-degenerate L-shaped segments on the paths from this point to all its neighbors in the rectilinear MST. Note that there may be several Steiner points along these paths to the neighbors. For example, in the first RST of Figure 6, the neighborhood of point v contains the Steiner point s, the non-degenerate L-shape L_1 , the degenerate L-shape L_2, and the two neighbors v_1 and v_2. Using part (1) of Theorem 2, it is possible to show that there are at most 6 non-degenerate L-shaped segments in the neighborhood of a point v. We locally stabilize this neighborhood, by case analysis on the interactions among these L-shaped segments. For example, the neighborhood of point v in the first RST of Figure 6, can be stabilized by simply flipping the L-shaped segment L_1, and merging the resulting overlap. In general, we have to perform a case analysis on at most a constant number of L-shaped segments, and therefore the local stabilization step can be carried out in constant time. We omit the details of the case analysis.

We execute the local stabilization step separately on the L-shapes in the neighborhood of each point of the original MST. Since our rectilinear MST has the separability property, we know that two L-shapes in different neighborhoods cannot overlap or cross each other. Thus, local stabilization of the neighborhood of each point of the MST implies stabilization of the entire L-RST. Each local stabilization step takes only constant time, therefore the stabilization algorithm runs in $O(n)$ time.

6. Experimental Results:

Our overall RST algorithm can be summarized as follows: (1) Construct a separable rectilinear MST of the given set of points, (2) compute the minimum cost L-RST, and (3) derive a stable RST from the L-RST. We have implemented this RST algorithm as a C program in a VM system running on an IBM 3090 processor.

Ex.	#Pts	MST Cost	L-RST Cost	Stab Cost	Impr
1	5	371	294	281	24%
2	5	318	278	256	20%
3	10	352	291	291	17%
4	10	593	494	487	18%
5	15	676	563	556	18%
6	15	619	543	524	15%
7	20	807	702	677	16%
8	20	660	559	556	16%
9	25	1400	1226	1193	15%
10	25	1225	1090	1063	13%
11	30	1374	1215	1203	12%
12	30	1349	1225	1199	11%
13	50	1794	1642	1621	10%
14	50	1796	1603	1582	12%
15	100	8411	7585	7498	11%
16	100	8931	7931	7869	12%

Figure 7. Table of Examples

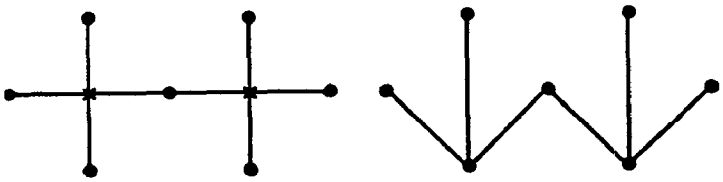

Figure 8. Hwang's 2/3 Example and its Mincost L-RST

The table in Figure 7 lists examples ranging from 5 input points to 100 input points. The last column shows the improvement in the cost of the stabilized minimum cost L-RST over the cost of the MST. We ran our algorithm on 500 randomly selected sets of points, for number of points ranging from 5 to 100. Interestingly, the average improvement in the cost of the stabilized minimum cost L-RST over the rectilinear MST, was seen to be about 9.1% for each one of these different cardinalities. Our algorithm produces optimal RST's for each member of the class of point sets which have the maximum possible improvement of 2/3 [7]. One such example is illustrated in Figure 8. Our method takes about 0.01 seconds on an IBM 3090 processor, to produce the stabilized minimum cost L-RST of 10 input points. Therefore it is suitable for applications such as global routing.

Acknowledgements

We wish to thank Kai-Win Lee and Majid Sarrafzadeh for several discussions about this work.

References:

1. A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *Data Structures and Algorithms*, Addison-Wesley Publishing Company, 1983.
2. M. A. Breuer, "Min-Cut Placement," *Design Automation and Fault-Tolerant Computing*, Vol. 1, 1977, pp 343-362.
3. A. E. Dunlop, and B. W. Kernighan, "A Procedure for Placement of Standard-Cell VLSI Circuits," *IEEE Transactions on Computer-Aided Design*, Vol. CAD-4, 1985, pp 92-98.
4. M. R. Garey, and D. S. Johnson, "The Rectilinear Steiner Tree Problem is NP-complete," *SIAM Journal of Applied Mathematics*, Vol. 32, No. 4, 1977, pp 37-58.
5. F. K. Hwang,. "An $O(n \log n)$ Algorithm for Rectilinear Minimal Spanning Trees," *Journal of the Association for Computing Machinery*, Vol 26, No. 2, April 1979, pp 177-182.
6. F. K. Hwang, "An $O(n \log n)$ Algorithm for Suboptimal Rectilinear Steiner Trees," *IEEE Transactions on Circuits and Systems*, Vol. CAS-26, No. 1, January 1979, pp 75-77.
7. F. K. Hwang, "On Steiner Minimal Trees with Rectilinear Distance," *SIAM Journal of Applied Mathematics*, Vol. 30, No. 1, January 1976, pp 104-114.
8. D. T. Lee and C. K. Wong, "Voronoi Diagrams in L_1, L_∞ Metrics with 2-Dimensional Storage Applications," *SIAM Journal of Computing*, Vol. 9, No. 1, February 1980, pp 200-211.
9. J. H. Lee, N. K. Bose, F. K. Hwang, "Use of Steiner's problem in sub-optimal routing in rectilinear metric," *IEEE Transactions on Circuits and Systems*, Vol. CAS-23, July 1976, pp 470-476.
10. K-W. Lee, and C. Sechen, "A New Global Router for Row-Based Layout," *Proceedings of IEEE International Conference on Computer-Aided Design*, Santa Clara, November 1988, pp 180-183.