Generation of Optimal Obstacle-avoiding Rectilinear Steiner Minimum Tree

Liang Li
Department of Computer Science and Engineering
The Chinese University of Hong Kong
lli@cse.cuhk.edu.hk

Zaichen Qian
Department of Computer Science and Engineering
The Chinese University of Hong Kong
zcqian@cse.cuhk.edu.hk

Evangeline F. Y. Young
Department of Computer Science and Engineering
The Chinese University of Hong Kong
fyyoung@cse.cuhk.edu.hk

ABSTRACT

[1] In this paper, we present an efficient method to solve the obstacle-avoiding rectilinear Steiner tree problem optimally. Our work is developed based on the GeoSteiner approach, modified and extended to allow rectilinear blockages in the routing region. We extended the proofs on the possible topologies of full Steiner tree (FST) in [4] to allow blockages, where FST is the basic concept used in GeoSteiner. We can now handle hundreds of pins with multiple blockages, generating an optimal solution in a reasonable amount of time. This work serves as a pioneer in providing an optimal solution to this difficult problem.

1. INTRODUCTION

Construction of rectilinear Steiner minimum tree (RSMT) is an important problem in VLSI physical design. It is useful for both the detailed and global routing steps, and it is important for congestion, wire length and timing estimations during the floorplanning or placement step. The original RSMT problem assumes no obstacles in the routing region. In today's VLSI designs, there can be many routing blockages, like macro cells, IP blocks and pre-routed nets. Therefore, the RSMT problem with blockages, called obstacle-avoiding RSMT (OARSMT) problem, has become an important problem in practice.

This OARSMT problem has been widely studied [7, 8, 9, 10, 11]. More recently, Hu *et al.* [12] developed an efficient hierarchical heuristic called FORst. Their method can tackle large scale problems efficiently. Based on an ant colony optimization, Hu *et al.* [13] proposed another non-deterministic local search heuristic, called An-OARSMan, to handle small-scale OARSMT problems with complex obstacle shapes. Although An-OARSMan is flexible in handling complex obstacles, it takes extremely long running time for large scale designs. CDCTree proposed by Shi *et al.* [15] is based on a current driven circuit model and it can achieve shorter wire length than An-OARSMan. Shen *et al.* [14] proposed a connection graph based approach to solve the problem. An obstacle-avoiding spanning graph is first constructed and then transformed into an OARSMT. Feng *et al.* [16] proposed a method to construct obstacle-avoiding Steiner tree in an arbitrary λ-geometry by Delaunay triangulation.

Very recently, Wu *et al.* [17] presented another algorithm for constructing OARSMTs. Their approach first finds a minimum spanning tree of all the terminals. The segments intersecting with the obstacles will be removed, forming a set of sub-trees. An ant colony optimization based approach is then used to connect the sub-trees back into a single tree which is rectilinearized at the end to given an OARSMT. Hentschke *et al.* [18] presented AMAZE, a fast maze routing based algorithm to build Steiner trees. Another recent work on this problem is by Lin *et al.* [19]. They extended the connection graph based approach in [14] by identifying many "essential" edges which can lead to more desirable solutions in the construction of the obstacle-avoiding spanning graph. A recent work on this problem is by Li and Young [20]. In their work, a maze routing based approach is used. Multiple paths between the pins are kept until all the pins are reached. Then a MST based method is used to select between those paths to create an OARSMT. Another recent piece of work is by Long *et al.* [21]. They proposed an efficient four-step algorithm to construct an OARSMT. They presented a fast algorithm for the minimum terminal spanning tree construction. Very recently, Lin *et al.* [22] proposed a critical-trunk-based tree growth mechanism and applied it to construct an OARSMT with an objective to minimize sink delay and to maximize the worst negative slack of the Steiner tree.

For the RSMT problem, there are ways to find optimal solutions effectively, e.g., using the GeoSteiner approach, with which one can compare with to get a better understanding of how far a proposed method is away from the optimum. For this OARSMT problem, there is not yet any good methods to generate optimal solutions effectively. The aim of this work is to study and provide such an optimal approach for the OARSMT problem. Our work is developed based on GeoSteiner, modified and enhanced to allow rectilinear blockages in the routing region. We extended the proofs on the possible topologies of full Steiner tree (FST) in [4] to consider blockages, where FST is the basic concept used in GeoSteiner. We can now connect more than 500 pins with multiple blockages, generating an optimal solution in a few minutes. This work serves as a pioneer in providing an optimal solution to this difficult problem.

In the rest of this paper, we first review the GeoSteiner approach. Then, we give an overview of our approach. After that, we discuss about the properties of FST, FST generation and FST concatenation when blockages exist. Experimental results will be shown in the last section, followed by a conclusion at the end.

[1]The work described in this paper was partially supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. 4184/07)

ICCAD'09, November 2–5, 2009, San Jose, California, USA.

2. PROBLEM DEFINITION

In the obstacle-avoiding rectilinear Steiner minimum tree problem, we are given a set V of n pins on the 2-D plane and a set B of m rectangular blockages, where the blockages cannot overlap with each other, and the objective is to give a shortest interconnection of those n pins using only horizontal and vertical lines without intersecting the blockages. This problem is well known to be NP-hard.

3. REVIEW ON GEOSTEINER

Rectilinear Steiner minimum trees (RSMTs) are unions of *full Steiner trees* (FSTs) in which every pin is a leaf. GeoSteiner is based on a framework in which subsets of pins are considered one after another. For each subset, all its FSTs are determined and the shortest one is kept. A number of checkings can be performed to identify and prune away some of those FSTs that cannot be in any RSMTs. Remaining FSTs are then selected and concatenated to obtain a RSMT with the shortest length. There are two major steps in this framework, FST generation and FST concatenation. For the FST generation step, an efficient algorithm by Zachariasen [6] is used that, based on some pre-processing information, grows FSTs while applying several optimality conditions to prune away those which will not be useful. The observed running time of this FST generation step is quadratic and approximately $4n$ FSTs will be generated on average for randomly generated problem instances. For the FST concatenation step, Warme [5] found that the problem is equivalent to the minimum spanning tree problem on a hypergraph with the set V of pins as vertices and subsets spanned by FSTs as hyperedges. It can be formulated as an integer linear program (ILP) and solved it using some branch-and-cut search.

4. OVERVIEW OF OUR APPROACH

For a given set of pins V and a set of blockages B, we want to find an OARSMT connecting all the pins in V. First of all, we add four virtual pins at the four corners of every blockage and we call the set of all these virtual pins V'.

Our algorithm is also based on the framework of GeoSteiner that is composed of the two major steps, FST generation and FST concatenation. After adding all the virtual pins to the original set of pins, we can show that the topologies of those FSTs with at least three vertices will follow a particular structure and the proof is very similar to that by Hwang [4]. For those FSTs with exactly two vertices, we have devised some interesting methods to generate those potentially useful ones effectively and this step is significantly different from that in GeoSteiner because of the existence of virtual pins. In our approach, we will generate all the FSTs after adding the virtual pins (one at each corner of a blockage) to the original set of pins. Notice that these virtual pins may or may not be used in the final optimal OARSMT. Following a similar fashion of GeoSteiner, we will then set up an ILP to select and concatenate some FSTs to form an OARSMT. The ILP is set up in such a way that those virtual pins can be included or not in the final tree as long as a shortest OARSMT is constructed. In the following sections, we will show some of the proofs[2] and explain each step in more details.

[2]We will not show all the proofs due to page limitation. A complete proof will appear in an extended version of this paper.

5. STRUCTURES OF FST WITH VIRTUAL PINS

5.1 Definition of FST

As defined in many previous works, a FST of a set P of pins is a rectilinear Steiner minimum tree T of P such that in T and all its *equivalent* trees, every pin is a leaf node. Using the same definition in [4], a tree T' is equivalent to another tree T if and only if T' can be obtained from T by *shifting* (Fig. 2(a)) or *flipping* (Fig. 2(b)) some edges which have no pins on them. After adding all the virtual pins to the original set of pins, an FST t with vertex set $V_t \subseteq (V + V')$ will have the following properties:

1. t is an OARSMT of the vertex set V_t and the degrees of the vertices in V_t are one in all the equivalent trees of t.
2. All the equivalent trees of t will not contain the types of edges (thickened) shown in Fig. 1, as splitting at the virtual pins will be done otherwise.
3. For any edge in an equivalent tree of t, the rectangular region covered by the two end points of the edge must contain no blockages, for property (2) will be violated (by shifting the edge to the boundary of the blockage) otherwise.

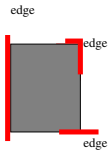

Figure 1: Forbidden edge

In the following, we will show that the structure and shape of a FST satisfying the above properties will follow some very simple forms.

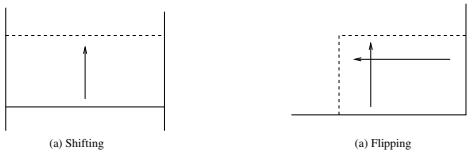

Figure 2: Shifting and flipping

5.2 Notations

In the following, we will use the same notations as in [4]. A *vertex* can be a pin (a pin refers to a real pin or a virtual pin unless stated specifically) or a Steiner point. An edge between two vertices is a sequence of alternating vertical and horizontal lines. We call each turning point a *corner*. A line has only one direction but may contain a number of vertices on it. V_{xu} (V_{xd}) denotes the vertical line at x which is above (below) x but excluding x itself. Similarly H_{xr} (H_{xl}) denotes the horizontal line at x which extends to the right (left) of x but excluding x itself. An example is shown in Fig. 3. If a line ends at a pin and contains no other vertices, we call it a *node line*. If it ends at a corner and contains no vertices, we call it a *corner line*. In the following figures for the proofs, an empty circle represents a vertex, which can be a pin or a Steiner point, and an empty square represents a pin.

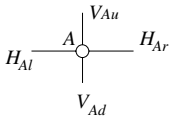

Figure 3: A point with four adjacent lines

5.3 Properties of FST with Virtual Pins

In this section, we want to derive the structure and topology of a FST in an $OARSMT(V+V')$. The steps of proof are similar to those in [4] but there are now blockages in the routing region. We will see that after the introduction of virtual pins, the structures of the FSTs are very similar to those without blockages [4]. In the following, we will present the proof of only one of the lemmas to illustrate the differences when blockages exist.

LEMMA 1. *Let A and B be two adjacent Steiner points in a FST. Suppose that AB is a horizontal line and both V_{Au} and V_{Bu} exist. Then $|V_{Bu}| \geq |V_{Au}|$ implies that V_{Au} is a corner line that turns away from V_{Bu}.*

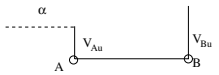

Figure 4: Property of adjacent Steiner points

PROOF. See Fig. 4. Suppose A is to the left of B. We can claim the following:

(i) V_{Au} contains no pins at the opposite end of A, for otherwise, we can shift AB to that vertex and obtain an equivalent tree in which a pin has degree more than one. If the line AB cannot be shifted due to some blockages, the FST (or its equivalent tree) will contain the structure as shown in Fig. 1.

(ii) No Steiner points on V_{Au} can have a line going right, for otherwise we can replace AB by extending that line to meet V_{Bu} and reduce the total length (thus not a FST, since FST is a Steiner minimum tree). If the line cannot be extended due to blockages, the FST (or its equivalent tree) will contain the structure as shown in Fig. 1 (again, by shifting AB to the blockage).

(iii) Therefore, the other end point of V_{Au} cannot be a Steiner point since it has no lines going right or up (a Steiner point has degree more than two), and it hence must be a corner turning left.

(iv) V_{Au} can have no Steiner point on it, for let C be such a Steiner point which is nearest to the corner point, then H_{Cr} does not exist and hence H_{Cl} must exist. We can then shift the line between point C and the corner point to the left to reduce the total length. If the line cannot be shifted due to some blockages, the FST (or its equivalent tree) will contain the structure as shown in Fig. 1. □

THEOREM 1. *Suppose s is a FST of a set P of pins and $|P| = p$. Then s is in the form of one of the following two structures in one direction as shown in Fig. 5.*

6. GENERATION OF FST WITH VIRTUAL PINS

6.1 Generation of FST with Two Pins

For those FSTs with exactly two vertices, we will construct them separately for efficiency purpose by the following method. First of all, these FSTs can be divided into two types. The first type is that the two end points are both in V. The second type is that at least of the two end points is in V'.

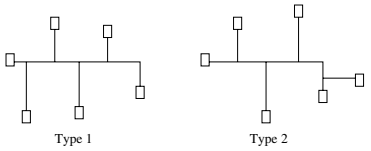

Figure 5: Possible FST structures in one direction

For the first type, we can construct them according to a lemma by Fößmeier *et al.* [2]. Let $G = (V,E)$ be a graph with edges assigned mutually distinct weights and let U be a subset of V. Let T be a MST of G and T' be a MST of $G[U]$, the subgraph of G induced by U. Then, every edge (u,v) in T where both u and v are in U will also appear in T'. We can show that the above property will also hold for RMST with obstacles. In order to generate all possible type one FSTs with two end points in V, we only need to construct an obstacle avoiding RMST of V and include all the edges in it as potential candidates.

For the second type, we will make use of a similar idea proposed by Yao *et al.* [1]. We know that at least one of the two end points of the edge under construction is in V' and the rectangular area covered by the two end points of the edge is obstacle free. For each virtual pin $v'_i \in V'$, we will divide its surrounding area into eight regions R_i for $i = 1 \ldots 8$. In every region R_i, we will find the point $v_j \in V$ which has the shortest rectilinear distance ($d_{v'_i v_j}$) from v'_i and the rectangular area covered by v'_i and v_j has no obstacles. Then, the edge containing v'_i and v_j is a potential candidate. In this region R_i, we will also find those points $u \in V'$ where the distance $d_{v'_i u} \leq d_{v'_i v_j}$ and the area covered by v'_i and u is obstacle free. Then, the edge containing v'_i and u will also be a potential candidate.

Based on the above methods, we can find all the potentially useful two pin FSTs. The number of this kind of edges is very large. Therefore we adopt some techniques to remove redundancy. Firstly, we will remove an edge if the rectangular area covered by its two end points is not obstacle free. This can be easily understood according to the definition of our FST with virtual pins. Secondly, if the rectangular area covered by the two end points is obstacle free but contains some pin points from V, we will also remove the edge. This technique has also been adopted by Zachariasen in [6].

6.2 Generation of FST with Three or More Pins

Actually, we are only interested in the FSTs that can be part of an OARSMT. Thus we should identify some necessary conditions for a FST to be a part of an OARSMT as in [6] to improve efficiency. Most of the conditions in [6] are applicable to us with some modifications. In the following, we will just focus on the modifications made when blockages and virtual pins exist.

For example, the bottleneck Steiner distance can also be used to eliminate useless FSTs when blockages exist. Let OARMST(V) be an obstacle avoid rectilinear minimum spanning tree of the point set V and v_i, $v_j \in V$ be a pair of vertices. The bottleneck Steiner distance $\delta_{OARMST(v_i v_j)}$ between v_i and v_j is equal to the length of the longest edge on the unique path between v_i and v_j in OARMST(V). Salowe *et al.* [3] proposed a theorem stating that if *MST* and *SMT* are respectively a minimum spanning tree and a Steiner minimal tree on a set of vertices V, then $\delta_{MST(v_i v_j)} \geqq \delta_{SMT(v_i v_j)}$ for any $v_i, v_j \in V$. We can show that the property will also hold for OARMST(V) and OARSMT(V). However, we can only compare the edges of

which both end points are in V in the OARSMT with the bottleneck Steiner distance computed from the OARMST.

The empty diamond property proposed in [6] states that no other points of an RSMT can lie in $\mathscr{L}(u,v)$, where uv is a (horizontal or vertical) segment in the RSMT and $\mathscr{L}(u,v)$ is an area on the plane where all the points in this area are closer to both u and v than u and v from each other. However, when there are blockages and virtual pins, the points which cannot lie in $\mathscr{L}(u,v)$ are the points in V only. The empty corner rectangle property is also proposed in [6]. Let uw and vw denote two perpendicular segments sharing a common end point w. Then no other points of the RSMT can lie in the interior of the smallest axis-aligned rectangle $\mathscr{R}(u,v)$ containing u and v. However, when there are blockages and virtual pins, we only need to consider points in V and can project on uw and vw without intersecting with any blockages.

Based on all of the above properties, we can generate all the potentially useful FSTs by growing them recursively as in [6]. We have also implemented some efficient methods to find the shortest distance between two points with blockages, the bottleneck Steiner distance, the long leg terminal candidates and the short leg terminal candidates to accelerate the process.

7. CONCATENATION OF FSTS WITH VIRTUAL PINS

After generating all the potentially useful FSTs, we can set up an ILP as in the GeoSteiner approach to select and concatenate a subset of FSTs to construct an OARSMT, connecting all the points in V with the minimum total length.

In the following, let S be the set of all FSTs found, V be the set of all real pins, T be the set of virtual pins (note that there is one virtual pin on each corner of the blockages), $n = |V|$, $m = |S|$ and $t = |T|$. Each FST $s_i \in S$ is associated with a binary variable x_i indicating whether s_i is taken. Besides, there are binary variables y_i for $i = 1 \ldots t$ indicating whether virtual pin $t_i \in T$ is connected in the tree. We use $|s_i|$ to denote the size of s_i, i.e., the number of pins (including virtual pins) connected by s_i, and use $len(s_i)$ to denote the length of s_i. Then the ILP is as follows:

$$\text{Minimize: } \sum_{i=1}^{m} len(s_i) \times x_i \quad (1)$$

Subject to:

$$\sum_{i=1}^{m} x_i(|s_i|-1) = n-1+\sum_{i=1}^{t} y_i \quad (1a)$$

$$y_j \le \sum_{t_j \in s_i} x_i \quad \forall t_j \quad (1b)$$

$$y_j \ge x_i \quad \forall t_j \forall s_i \text{ s.t. } t_j \in s_i \quad (1c)$$

$$\exists s_i \in S \text{ s.t. } s_i \in (X : V+T-X) \text{ and } x_i = 1$$
$$\forall X \subseteq V+T \text{ and } V \nsubseteq X \text{ and } X \cap V \neq \emptyset \quad (1d)$$

$$\sum_{s_i : s_i \cap X \neq \emptyset} x_i(|s_i \cap X|-1) \le |X-T| + \sum_{t_i \in X} y_i - 1$$
$$\forall X \subseteq V+T \text{ and } X \cap V \neq \emptyset \text{ and } 2 \le |X| \le n+t-1 \quad (1e)$$

$$\sum_{s_i : s_i \cap X \neq \emptyset} x_i(|s_i \cap X|-1) \le |X|-1 \quad \forall X \subseteq T \text{ and } X \neq \emptyset \quad (1f)$$

The notation $s_i \in (X : V+T-X)$ in (1c) means that $s_i \cap X \neq \emptyset$ and $s_i \cap (V+T-X) \neq \emptyset$. Constraint (1a) requires the right amount of edges. Each selected FST s_i contributes $|s_i|-1$ edges for the tree. Since we do not know the exact total number of pins (because of the virtual pins), y_i is added. Constraint (1b) ensures that if all FSTs containing a virtual pin are not selected, that virtual pin will not be in the tree. Constraint (1c) ensures that if a certain FST containing a virtual pin is selected, the corresponding virtual pin is also selected. Constraint (1d) requires that a solution should be connected – for any cut with partitions X and $V+T-X$, there must be at least one selected FST linking it up. We require $X \cap V \neq \emptyset$, because we do not care about the connectivity of the virtual pins. We require $V \nsubseteq X$ for the same reason that we do not need to connect to virtual pins if V itself has been all connected. Both constraints (1e) and (1f) are used to eliminate cycles. Consider a set X of two pins only. We can allow to have at most one FST which are selected and contain both pins of X. A cycle will be formed otherwise. Similar concept can be extended for sets of larger size and this results in the formulation of constraints (1e) and (1f). In (1e), we consider those sets $X \cap V \neq \emptyset$. Since y_i tells whether t_i is selected, $|X - X \cap T| + \sum_{t_i \in X} y_i$ gives the exact number of pins (including virtual pins) in X. In (1f), we can just use $|X|$ for the number of pins as $X \subseteq T$. From the above formulation, we can know the number of variables in the ILP is the summation of the number of FSTs and the number of virtual pins. From the experiment results, we observed that the number of FSTs is approximately a constant times the number of pins (pins and virtual pins) when the number of virtual pins is small. We can also see from the above formulation that the number of constraints is exponential.

8. EXPERIMENTAL RESULTS

We implemented our algorithm based on geosteiner-3.1 and all the experiments were performed in a computer with a 2.2GHz Intel Pentium processor and 2GB memory running in the Linux environment. There are totally 22 benchmark circuits. All these test cases are obtained from [19] by selecting some blockages. (If we allow k blockages, we will take the first to the k^{th} ones.)

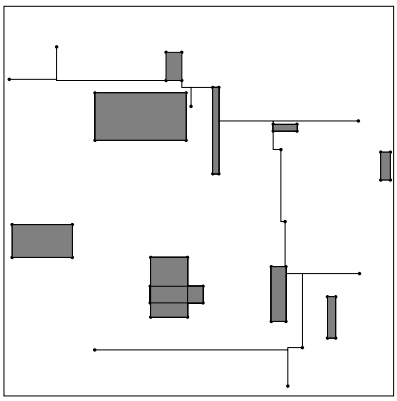

Figure 6: The result of our algorithm with 10 points and 10 blockages

In table 1, we list the running time of the algorithm, and the length of the optimal OARSMT constructed by our algorithm. The running time includes the time of generating all the FSTs and the time of solving the ILP. From the results, we can see that the time of generating all the FSTs is very short and solving the ILP dominates the total running time. We also list the lengths of the OARSMTs constructed by some recently published heuristics [19], [20] and [21]. As we do not have the executable of the approach in [19], we have just listed the corresponding results appeared in their paper (others are shown as N/A). For the other two approaches [20, 21], we run the experiments with the executables provided by the authors. For the test case ind3_10_20, the method in [21] cannot give a solution and it is indicated with an N/A in the table. With our optimal method, we can easily compare the performance of different approaches and see how far a heuristic solution is away from the optimum.

We show an optimal result for the test case rc1 generated by our algorithm in Fig. 6. The points which are not connected is the virtual points we added at the corners of the blockages.

Test Cases	CPU Time (s)		$L(O)$	Other Heuristics $D_i = (H_i - O)/H_i$ (%)					
name_n_m	FST	ILP		[19] (H_1)	D_1	[20] (H_2)	D_2	[21] (H_3)	D_3
ind1_10_20	0.05	318.68	604	N/A	N/A	619	2.42	648	6.82
ind2_10_22	0.04	206.59	9300	N/A	N/A	9300	0.00	9600	3.13
ind3_10_20	0.02	23.81	587	N/A	N/A	590	0.51	N/A	N/A
ind4_25_20	0.05	380.03	1078	N/A	N/A	1088	0.92	1100	2.00
ind5_33_11	0.04	225.07	1295	N/A	N/A	1304	0.69	1326	2.34
rc1_10_10	0.06	240.76	25980	26900	3.42	25980	0.00	26120	0.54
rc2_30_10	0.06	108.99	41350	42210	2.04	42010	1.57	41630	0.67
rc3_50_10	0.10	2.35	54160	55750	2.85	54390	0.42	55010	1.54
rc4_70_10	0.10	2.06	59070	60350	2.12	59740	1.12	59250	0.30
rc5_100_10	0.23	9.61	74070	76330	2.96	74650	0.78	76240	2.85
rc6_100_15	0.40	120.67	76436	N/A	N/A	77720	1.65	77715	1.65
rc7_200_10	0.67	301.67	105305	N/A	N/A	106951	1.54	106891	1.48
rc8_200_10	0.75	139.67	107652	N/A	N/A	109161	1.38	110260	2.37
rc9_200_10	0.63	24.63	105712	N/A	N/A	107644	1.79	107630	1.78
rc10_500_10	4.59	106.04	162230	N/A	N/A	165580	2.02	165460	1.95
rt1_10_15	0.06	71.28	1858	N/A	N/A	1858	0.00	1963	5.36
rt2_50_15	0.15	198.72	44294	N/A	N/A	45796	3.28	45263	2.14
rt3_100_10	0.28	553.32	7579	N/A	N/A	7736	2.03	7664	1.11
rt4_100_10	0.20	81.79	7678	N/A	N/A	7820	1.82	7851	2.20
rt5_200_10	0.58	28.64	42748	N/A	N/A	43680	2.13	43481	1.69
Average					2.68		1.30		2.21

n is the number of pins, m is the number of blockages and L denotes the length of the OARSMT.

Table 1: Optimal OARSMT lengths and comparisons with some recent heuristics

9. CONCLUSION

In this paper, we extended the well known GeoSteiner method to solve the obstacle-avoiding RSMT problem optimally. We can now handle hundreds of pins with multiple blockages, generating an optimal solution in a reasonable amount of time. Our future work is to improve further the efficiency of this method to handle data with even more pins and obstacles. This work serves as a pioneer in providing an optimal solution to this important obstacle-avoiding RSMT problem.

10. ACKNOWLEDGMENT

The authors want to thank Professor Chris Chu and Hai Zhou for their kind help.

11. REFERENCES

[1] Andrew C.C. Yao, "On Constructing Minimum Spanning Trees in k-dimensional spaces and related problems", *SIAM Journal on Computing*, Vol.11, No.4, pp.721-736, 1982.

[2] U. Fößmeier and M. Kaufmann, "On Exact Solutions for the Rectilinear Steiner tree problem", *Technical Report WSI-96-09*, 1996.

[3] J.S. Salowe and D.M. Warme, "Thirty-Five Point Rectilinear Steiner Minimal Trees in a Day", *Networks*, Vol.25, No.2, pp.69-87, 1995.

[4] F.K. Hwang, "On Steiner Minimal Trees with Rectilinear Distance", *Proceedings SIAM Journal on Applied Mathematics*, Vol.30, pp.104-114, 1976.

[5] D.M. Warme, "A New Exact Algorithm for Rectilinear Steiner Minimal Trees", *Technical Report, System Simulation Solutions, Inc.*, Alexandria, VA 22314, USA, 1997.

[6] M. Zachariasen, "Rectilinear Full Steiner Tree Generation", *Networks*, Vol.33, pp.125-143, 1999.

[7] K.L. Clarkson and S. Kapoor and P.M. Vaidya, "Rectilinear Shortest Paths through Polygonal Obstacles in $O(nlog^2n)$", *Proceedings ACM Symposium on Computational Geometry*, pp.251-257, 1987.

[8] Y.F. Wu and P. Widmayer, M.D.F. Schlag and C.K. Wong, "Rectilinear Shortest Paths and Minimum Spanning Trees in the Presence of Rectilinear Obstacles", *IEEE Transaction on Computer-Aided Design*, Vol.36, No.3, pp.321-331, 1987.

[9] J. Ganley and J.P. Cohoon, "Routing a Multi-terminal Critical Net: Steiner Tree Construction in the Presence of Obstacles", *Proceedings International Symposium on Circuits and Systems*, pp.113-116, 1994.

[10] S.Q. Zheng and J.S. Lim and S.S. Iyengar, "Finding Obstacle-avoiding Shortest Paths using Implicit Connection Graphs", *IEEE Transaction on Computer-Aided Design*, Vol.15, No.1, pp.103-110, 1996.

[11] Y. Yang and Q. Zhu and T. Jing and X. Hong and Y. Wang, "Rectilinear Steiner Minimal Tree Among Obstacles", *Proceedings IEEE ASICCON*, pp.348-351, 2003.

[12] Y. Hu and Z. Feng and T. Jing and X. Hong and Y. Yang and G. Yu and X. Hu and G. Yan, "FORst: a 3-step Heuristic for Obstacle-avoiding Rectilinear Steiner Minimal Tree Construction", *Journal of Information and Computational Science*, pp.107-116, 2004.

[13] Y. Hu and T. Jing and X. Hong and Z. Feng and X. Hu and G. Yan, "An-OARSMan: Obstacle-avoiding Routing Tree Construction with Good Length Performance", *Proceedings ASP-DAC*, pp.7-12, 2005.

[14] Z. Shen and C. Chu and Y. Li, "Efficient Rectilinear Steiner Tree Construction with Rectilinear Blockages", *Proceedings ICCD*, pp.38-44, 2005.

[15] Y. Shi and T. Jing and L. He and Z. Feng and X. Hong, "CDCTree: Novel Obstacle-avoiding Routing Tree Construction based on Current Driven Circuit Model", *Proceedings ASP-DAC*, 2006.

[16] Z. Feng and Y. Hu and T. Jing and X. Hong and X. Hu and G. Yan, "An $O(nlogn)$ Algorithm for Obstacle-avoiding Routing Tree Construction in the λ-geometry Plane", *Proceedings ISPD*, pp.48-55, 2006.

[17] P.C. Wu and J.R. Gao and T.C. Wang, "A Fast and Stable Algorithm for Obstacle-avoiding Rectilinear Steiner Minimal Tree Construction", *Proceedings ASP-DAC*, pp.262-267, 2007.

[18] R. Hentschke and J. Narasimham and M. Johann and R. Reis, "Maze Routing Steiner Trees with Effective Critical Sink Optimization", *Proceedings ISPD*, 2007.

[19] C.W. Lin and S.Y. Chen and C.F. Li and Y.W. Chang and C.L. Yang, "Efficient Obstacle-avoiding Rectilinear Steiner Tree Construction", *Proceedings ISPD*, 2007.

[20] L. Li and Evangeline F.Y. Young, "Obstacle-avoiding Rectilinear Steiner Tree Construction", *Proceedings ICCAD*, 2008.

[21] J.Y. Long and H. Zhou and S.O. Memik, "EBOARST: An Efficient Edge-Based Obstacle-Avoiding Rectilinear Steiner Tree Construction Algorithm", *IEEE Transaction on Computer-Aided Design*, Vol.27, No.12, pp.2169-2182, 2008.

[22] Y.H. Lin and S.H. Chang and Y.L. Li, "Critical-Trunk Based Obstacle-Avoiding Rectilinear Steiner Tree Routings for Delay and Slack Optimization ", *Proceedings ISPD*, 2009.