Obstacle-avoiding Rectilinear Steiner Minimum Tree Construction: An Optimal Approach

Tao Huang
Department of Computer Science and Engineering
The Chinese University of Hong Kong
thuang@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*—In this paper, we present an efficient method to solve the obstacle-avoiding rectilinear Steiner minimum tree (OARSMT) problem optimally. Our work is a major improvement over the work proposed in [1]. First, a new kind of full Steiner trees (FSTs) called obstacle-avoiding full Steiner trees (OAFSTs) is proposed. We show that for any OARSMT problem there exists an optimal tree composed of OAFSTs only. We then extend the proofs on the possible topologies of FSTs in [2] to find the possible topologies of OAFSTs, showing that OAFSTs can be constructed easily. A two-phase algorithm for the construction of OARSMTs is then developed. In the first phase, a sufficient number of OAFSTs are generated. In the second phase, the OAFSTs are used to construct an OARSMT. Experimental results on several benchmarks show that the proposed method achieves 185 times speedup on average and is able to solve more benchmarks than the approach in [1].**

I. Introduction

Rectilinear Steiner minimum tree (RSMT) problem is one of the fundamental problems in integrated circuit computer-aided design (CAD). It has been widely used in the area of VLSI design automation. Finding an RSMT is useful for routing, global wire length estimation, and is also important for congestion and timing estimation in floorplanning and placement. The original RSMT problem assumes no obstacle in the routing region. However, in modern nanometer VLSI designs, there can be routing obstacles on the chip such as macro cells, IP blocks and pre-routed nets. Therefore, the obstacle-avoiding RSMT problem (OARSMT) has received a lot of research attentions.

The RSMT problem has been shown to be NP-complete [3], and the introduction of obstacles has made this problem even more complicated. In recent years, many works have been focusing on solving the OARSMT problem heuristically [4], [5], [6]. Shen *et al.* [7] introduced a connection graph called spanning graph. In their approach, an obstacle-avoiding spanning graph (OASG) was first constructed and then transformed into an OARSMT. Lin *et al.* [8] extended the approach in [7] by identifying many "essential" edges which can lead to more desirable solutions in the construction of the OASG. A maze routing based approach was proposed by Li *et al.* [9]. In their work, multiple paths between terminals were kept until all the terminals were reached. An MST based method was then used to select between those paths to create an OARSMT. Long *et al.* [10] proposed a four-step algorithm to construct an OARSMT. They presented a quadrant approach to construct an OASG and an edge-based heuristic to handle both global and local refinements. Very recently, Ajwani *et al.* [11] presented the FOARS, a top down approach for the OARSMT problem. They applied the OASG to partition a solution and constructed the OARSMT by using the obstacle-aware version of the fast lookup table based wire-length estimation.

In comparison with heuristics, few exact algorithms have been proposed. Maze routing [12] can give optimal solutions to two-terminal nets. Ganley *et al.* [13] proposed a topology enumeration algorithm to construct optimal three-terminal or four-terminal OARSMTs. Recently, Li *et al.* [1] proposed an exact algorithm for multi-terminal nets. They extended the GeoSteiner approach in [15] to an obstacle-aware version. By adding four virtual terminals to each obstacle, they proved that the FSTs are simple and can be constructed efficiently. Optimal solutions to problems with up to five hundred terminals are reported. However, empirical results show that their algorithm is severely affected by the number of virtual terminals required. Most of the test cases they used contain only ten obstacles but the running time is still very expensive. Based on the approach in [1], we propose a more efficient method to construct OARSMTs in this paper. The main contributions of this paper can be summarized as follows:

1) Different from the previous approach, the algorithm presented in this paper works by introducing only two virtual terminals to each obstacle. The essential idea is to reduce the total number of constraints in the integer program so that the performance of the algorithm can be improved significantly.
2) We improve the two-phase algorithm in [1] by using more powerful screening tests and more efficient separation algorithms.
3) We adopt an incremental way to handle obstacles, which can effectively reduce the running time for large scale problems.

The rest of this paper is organized as follows. Section II gives the OARSMT problem formulation. The definition and the structures of OAFSTs are discussed in section III. Section IV and V describe the iterative two-phase approach for the construction of OARSMT in detail. Experimental results and discussions are presented in section VI, followed by a

Fig. 1: Forbidden edges in an OAFST.

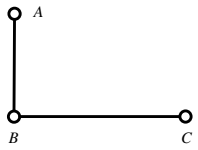

Fig. 2: A Steiner chain structure.

conclusion in section VII.

II. Problem formulation

In the OARSMT problem, we are given a set V of n terminals on the 2-D plane and a set B of m rectangular obstacles. No obstacle can overlap with each other, but they can be line-touched at the boundary. The objective of the problem is to give an obstacle-avoiding tree with the shortest length that connects all of those n terminals using only horizontal and vertical lines. This tree is known as an OARSMT.

III. OARSMT decomposition

In this section, we show how an OARSMT can be decomposed into small components called OAFSTs which are much simpler to generate.

A. OAFST definition

First, for each obstacle, we introduce two virtual terminals that are the two end points of a diagonal. Without loss of generality, in the rest of this paper, we assume that the two virtual terminals are located at the upper right and lower left corners of an obstacle, respectively. We use V' to denote the set of all virtual terminals. As defined in [2], if a tree T' can be obtained from another tree T by *shifting* or *flipping* edges, T' is equivalent to T. In the presence of obstacles, an OAFST T over a set of terminals $P \subseteq V + V'$ is defined as follows:

1) T is an OARSMT of P.
2) Every terminal $t \in P$ has degree one in T and in all its equivalent trees.
3) All the equivalent trees of T cannot contain edges that pass through a virtual terminal as shown in Fig. 1. Otherwise, we can split T into two smaller OAFSTs.

Theorem 1: Let V be a set of terminals, with $|V| \geq 2$, then V has an obstacle-avoiding rectilinear Steiner minimum tree that consists of only OAFSTs. These OAFSTs intersect only at real or virtual terminals of degree two or more.

Proof: Any OARSMT can be split into several small trees by cutting at real terminals of degree more than one. If any of these trees (or their equivalent trees) contain forbidden edges that pass through virtual terminals, we can further split it into smaller trees at these virtual terminals. As a result, an OARSMT can be partition into several trees each of which satisfies the definition of OAFSTs. ■

This theorem indicates a possible way to construct OARSMT by concatenation of OAFSTs. To prove the efficiency of this approach, the structures of OAFSTs are studied. We will show how the properties defined can help in shaping OAFSTs so that it can be generated easily.

B. OAFST structures

In this section, we use $V_{xu}(V_{xd})$ to denote the vertical line at point x which is above (below) x but excluding x itself. Similarly H_{xr} (H_{xl}) denotes the horizontal line at point x which is on the right (left) of x but excluding x itself. If a line ends at a node and contains no other vertex, we call it a *node line*. If it ends at a corner and contains no vertex, we call it a *corner line*. In the following figures, we use an empty circle to represent a Steiner point and an empty square to represent a terminal. The steps of proof to derive the topologies of OAFSTs are similar to those in [2]. Due to the limitation of space, the corresponding proofs are not shown here.

Lemma 1: All Steiner points either have degree three or degree four.

Lemma 2: Let A and B be two adjacent Steiner points in an OAFST. Suppose that AB is a horizontal line and both V_{Au}, V_{Bu} exist. Then $|V_{Bu}| \geq |V_{Au}|$ implies that V_{Au} is a corner line that turns away from V_{Bu}.

Corollary: Suppose V_{Bu} contains a vertex, then V_{Au} must be a corner line turning away from V_{Bu} and $|V_{Au}| < |V_{Bu}|$.

Lemma 3: Suppose V_{xu} (x is a vertex) is a corner line turning left (right), then H_{xl} (H_{xr}) does not exist.

Lemma 4: No Steiner point can have more than one corner line.

Lemma 5: If s is an OAFST, the Steiner points in s form a chain.

Lemma 6: Suppose s is an OAFST. Its Steiner chain cannot contain the subgraph shown in Fig. 2, where B is adjacent to A and C.

Lemma 7: Suppose s is an OAFST. The Steiner chain of s is then a staircase.

Lemma 8: Suppose s is an OAFST. The Steiner chain of s cannot contain a corner with more than one Steiner points on the two neighboring lines.

Lemma 9: Suppose s is an OAFST. If the number of Steiner points is greater than two, either every vertical line (on the Steiner chain) contains more than one Steiner points (except possibly the first and the last vertical lines) and every horizontal line contains exactly one Steiner point, or vice versa.

Lemma 10: Suppose s is an OAFST. Every Steiner point on s must have a horizontal node line.

Lemma 11: Suppose s is an OAFST and A_i is the i^{th} Steiner point on the Steiner chain. Then a corner connecting A_i and A_{i+1} can be transferred to one connecting A_{i+2} and A_{i+3}, regardless of whether the place it transfers to has a corner or not. If the corner cannot be transferred due to some obstacles, then A_{i+3} is the last Steiner point on the chain. Similarly, this corner can also be transferred to one connecting A_{i-1} and A_{i-2}.

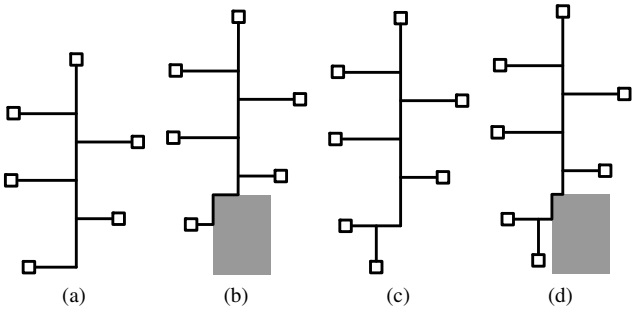

Fig. 3: (a) OAFST structure type I. (b) OAFST structure type II. (c) OAFST structure type III. (d) OAFST structure type IV.

If the corner cannot be transferred due to some obstacles, then A_{i-2} is the first Steiner point on the chain.

Lemma 12: Suppose s is an OAFST. There exists an s' equivalent to s such that all the Steiner points are on a straight line except possibly the last one.

To summarize, if the Steiner chain is a vertical line, the horizontal node lines at the Steiner points must alternate in the left-right directions. Hence, each Steiner point has exactly one horizontal node line except the first and the last one. By putting all of the above lemmas together, we can have the following conclusion:

Theorem 2: Suppose s is an OAFST over a set P of terminals. Then s is in the form of one of the four structures in one direction as shown in Fig. 3.

As we can observe from the figures, the structures of OAFSTs are very similar to those of FSTs as described in [2] and [1], except that OAFSTs have two additional structures, namely type II and type IV.

IV. Two-phase algorithm

The similarity between OAFSTs and FSTs indicates that we can utilize the method for the construction of FSTs to generate OAFSTs efficiently. The resulting OAFSTs are then combined into an OARSMT.

A. OAFST generation

The first phase of the algorithm is to generate a sufficient number of OAFSTs. In our approach, we modify the FST generation algorithm in [14] for the generation of OAFSTs with three or more terminals. By using the bottleneck Steiner distance, empty diamond, empty corner rectangle and empty inner rectangle properties, the algorithm recursively grows the OAFSTs for every terminal $v \in V + V'$ in both the horizontal and vertical directions. For OAFSTs with exactly two terminals, we construct them by using the same method as in [1].

B. OAFST concatenation

The second phase of the algorithm is to use the OAFSTs generated in the first phase to construct an OARSMT. We show that the OAFST concatenation can be formulated as an integer program (IP) and solved by using the branch-and-cut algorithm extended from [15].

Let V be the set of real terminals, V' be the set of virtual terminals and T be the set of all OAFSTs generated in the first phase. Let $n = |V|$, $n' = |V'|$, and $m = |T|$. Let I be the index set of T such that t_i ($i \in I$) is an OAFST in T spanning terminal set S_i. Let c_i be the length of t_i. We use variable x_i to denote whether t_i is taken as a part of the OARSMT solution. $x_i = 1$ for the OAFSTs that are in the OARSMT, while $x_i = 0$ for the OAFSTs that are not parts of the OARSMT. Besides, we use variable y_i to denote whether a virtual terminal $v'_i \in V'$ is connected in the OARSMT. In the following, $(A : B)$ means $\{i \in I : (S_i \cap A \neq \emptyset) \wedge (S_i \cap B \neq \emptyset)\}$. The IP formulation of the OAFST concatenation problem is as follows.

Minimize:
$\sum_{i=1}^{m} c_i x_i$ (1)
Subject to:
$\sum_{i \in I} x_i(|S_i| - 1) = n - 1 + \sum_{j=1}^{n'} y_j$ (2)
$2y_j \leq \sum_{v'_j \in S_i} x_i$ for all $v'_j \in V'$ (3)
$y_j \geq x_i$ for all v'_j and t_i such that $v'_j \in S_i$ (4)
$\sum_{i \in (S:V+V'-S)} x_i \geq 1$ (5)
for all $S \subseteq V + V'$ and $V \nsubseteq S$ and $S \cap V \neq \emptyset$
$\sum_{i \in I} \max\{|S_i \cap S| - 1, 0\} x_i \leq |S \cap V| + \sum_{v'_j \in S} y_j - 1$ (6)
for all $S \subseteq V + V'$ and $S \cap V \neq \emptyset$ and $2 \leq |S| < n + n'$
$\sum_{i \in I} \max\{|S_i \cap S| - 1, 0\} x_i \leq \sum_{v'_j \in S} y_j - \max_{v'_j \in S} y_j$ (7)
for all $S \cap V = \emptyset$ and $|S| \geq 2$

Constraint (2) requires the right number of edges to construct a spanning tree. Constraints (3) ensure that if a virtual terminal is selected, at least two OAFSTs connecting it are selected as well. Constraints (4) ensure that if a certain OAFST containing a virtual terminal is selected, the corresponding virtual terminal is also selected. Constraints (5) require that a solution should be connected, that is, for any cut $(S : V + V' - S)$, there must be at least one selected OAFST to connect them. Constraints (7) and (8) are used to eliminate cycles. The IP described above is solved via a branch-and-cut framework. We adopt Warme's [15] algorithm and extend it for solving the OAFST concatenation problem. Details of the algorithm are omitted due to space limitation.

V. Incremental construction

In order to avoid explosion of OAFSTs, we adopt an incremental way to construct an OARSMT. An obstacle list is maintained during the generation of the OARSMT. The list is responsible for keeping track of the obstacles we need to avoid during the construction. Initially, the OARSMT problem with an empty list of obstacles is solved resulting in an RSMT. We then check for obstacles that overlap with the solution and add them to the obstacle list. A new iteration then starts again by solving the OARSMT problem with the obstacles in the renewed list. This procedure repeats until no overlapping obstacle can be found. This approach is effective as in most cases only a fraction of the obstacles will affect the final OARSMT.

TABLE I: Results of our approach in comparison with the approach in [1].

Bench	m	k	Ours		Li [1]		ω
			L_1	t_1	L_2	t_2	
IND1	10	32	604	1	-	-	-
IND2	10	43	9500	1	-	-	-
IND3	10	50	600	1	-	-	-
IND4	25	79	1086	1	-	-	-
IND5	33	71	1341	2	-	-	-
RC1	10	10	25980	1	25980	130	130×
RC2	30	10	41350	1	41350	101	101×
RC3	50	10	54160	1	54160	4	4×
RC4	70	10	59070	1	59070	3	3×
RC5	100	10	74070	2	74070	27	13×
RC6	100	500	79714	5033	-	-	-
RC7	200	500	108740	12328	-	-	-
RC8	200	800	112564	127822	-	-	-
RC9	200	1000	111005	96831	-	-	-
RC10	500	100	164150	944	-	-	-
RC11	1000	100	230837	27131	-	-	-
RT1	10	500	2146	639	-	-	-
RT2	50	500	45852	208	-	-	-
RT3	100	500	7964	874	-	-	-
RT4	100	1000	9693	228558	-	-	-
RT5	200	2000	-	-	-	-	-
IND1′	10	20	604	1	604	498	498×
IND2′	10	22	9300	1	9300	365	365×
IND3′	10	20	587	1	587	43	43×
IND4′	25	20	1078	1	1078	474	474×
IND5′	33	11	1295	1	1295	179	179×
RC6′	100	15	76436	1	76436	250	250×
RC7′	200	10	105305	14	105305	1647	178×
RC8′	200	10	107652	48	107652	1314	28×
RC9′	200	10	105712	4	105712	82	20×
RC10′	500	10	162230	82	162230	552	7×
RT1′	10	15	1858	1	1858	124	124×
RT2′	50	15	44294	1	44294	335	335×
RT3′	100	10	7579	1	7579	913	913×
RT4′	100	10	7678	4	7678	149	37×
RT5′	200	10	42748	7	42748	35	5×
Average							185×

m denotes the number of real terminals. k denotes the number of obstacles. L_1 and L_2 denote the length of the resulting OARSMT. t_1 and t_2 denote the CPU time in seconds. $\omega = t_2/t_1(\times)$ is the speedup. "-" denotes no solution after 72 hours.

VI. Experimental results

We implement our algorithm based on GeoSteiner-3.1 and all the experiments are conducted on a Sun Blade 2500 workstation with two 1.6GHz processors and 2GB memory. We test our algorithm using 21 benchmark circuits. Benchmarks IND1-IND5 are industrial test cases from Synopsys. Benchmarks RC1-RC11 are adopted from [5]. Benchmarks RT1-RT5 are random test cases used in [8]. Table I shows the results of our method in comparison with the approach in [1]. We execute the algorithm in [1] on our platform. In order to show the efficiency of our approach, we also tabulate the results of fifteen additional test cases (IND1′-IND5′, RC6′-RC10′, and RT1′-RT5′) which are used in [1]. These test cases are obtained by selecting the first few obstacles from the corresponding benchmarks. We run each test case for 72 hours at most. As can be observed from the table, the time required for the OARSMT generation has been improved a lot by our method. Comparing with the approach in [1], our method can solve problems with up to one thousand obstacles, while the approach in [1] can only deal with cases with about ten obstacles. Among those solvable cases, our approach is 185 times faster than the approach in [1] on average. However, the proposed method still cannot solve RT5 in which there are two thousand obstacles.

VII. Conclusion

In this paper, we presented an optimal approach to solve the OARSMT problems. In the proposed approach, OARSMTs are constructed by concatenation of OAFSTs. Experiment results show that our approach can handle problems with hundreds of terminals in the presence of up to one thousand obstacles. Our future work is to enhance the performance of the proposed method in solving large cases.

References

[1] L. Li, Z. Qian, and Evangeline F. Y. Young, "Generation of Optimal Obstacle-avoiding Rectilinear Steiner Minimum Tree," in *Proc. Int. Conf. Comput.-Aided Des.*, pp.21-25, 2009.

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

[3] M. R. Garey and D. S. Johnson, "The Rectilinear Steiner Tree Problem is NP-Complete," *SIAM Journal of Applied Mathematics*, pp. 826C834, 1977.

[4] Y. Hu, Z. Feng, T. Jing, X. Hong, Y. Yang, G. Yu, 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.

[5] Z. Feng, Y. Hu, T. Jing, X. Hong, X. Hu, G. Yan, "An O($n\log n$) Algorithm for Obstacle-avoiding Routing Tree Construction in the λ-geometry Plane," in *Proc. Int. Symp. Phys. Des.*, pp.48-55, 2006.

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

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

[8] C.W. Lin, S. Y. Chen, C. F. Li, Y. W. Chang, and C. L. Yang, "Efficient Obstacle-avoiding Rectilinear Steiner Tree Construction," in *Proc. Int. Symp. Phys. Des.*, pp.380-385, 2007.

[9] L. Li and Evangeline F. Y. Young, "Obstacle-avoiding Rectilinear Steiner Tree Construction," in *Proc. Int. Conf. Comput.-Aided Des.*, pp.523-528, 2008.

[10] J. Y. Long, 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.

[11] G. Ajwani, C. Chu, and W. K. Mak, "FOARS: FLUTE Based Obstacle-Avoiding Rectilinear Steiner Tree Construction," in *Proc. Int. Symp. Phys. Des.*, pp.27-34, 2010.

[12] C. Y. Lee, "An Algorithm for Connections and Its Application," *IRE Trans. on Electronic Computer*, pp. 346-365, 1961.

[13] J. L. Ganley and J. P. Cohoon, "Routing a Multi-terminal Critical Net: Steiner Tree Construction in the Presence of Obstacles," in *Proc. of IEEE ISCAS, London, UK*, pp.113-116, 1994.

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

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