On the Construction of Optimal Obstacle-Avoiding Rectilinear Steiner Minimum Trees

Tao Huang, Liang Li, and Evangeline F. Y. Young

***Abstract*—This paper presents an efficient method to solve the obstacle-avoiding rectilinear Steiner tree (OARSMT) problem optimally. Our work is developed based on the GeoSteiner approach in which full Steiner trees (FSTs) are first constructed and then combined into a rectilinear Steiner minimum tree (RSMT). We modify and extend the algorithm to allow obstacles in the routing region. For each routing obstacle, we first introduce four virtual terminals located at its four corners. We then give the definition of FSTs with blockages and prove that they will follow some very simple structures. Based on these observations, a two-phase approach is developed for the construction of OARSMTs. In the first phase, we generate a set of FSTs with blockages. In the second phase, the FSTs generated in the first phase are used to construct an OARSMT. Finally, experiments on several benchmarks are conducted. Results show that the proposed method is able to handle problems with hundreds of terminals in the presence of multiple obstacles, generating an optimal solution in a reasonable amount of time.**

***Index Terms*—Full Steiner tree, obstacle-avoiding, rectilinear Steiner minimum tree, routing.**

I. Introduction

CONSTRUCTION of rectilinear Steiner minimum tree (RSMT) is an important problem in very large-scale integrated circuit (VLSI) physical design. It is useful for both the detailed and global routing steps, and is important for congestion, wire length, and timing estimations during the floorplanning or placement step. The original RSMT problem assumes no obstacle 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.

The OARSMT problem has been widely studied in recent years. Since the problem is NP-complete [1], most of the previous papers focus on the development of heuristics. Some early approaches construct an obstacle-free Steiner tree first and then replace the edges that overlap with obstacles [2]. These approaches are simple, but may result in lower-quality solutions.

Nondeterministic approaches are also developed based on some meta-heuristics. Hu *et al.* [3] proposed an ant colony optimization-based 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.

Some of the recent works also extend the maze routing method to a multiterminal variant to handle multiterminal nets. Hentschke *et al.* [4] presented AMAZE, a fast maze routing-based algorithm to build Steiner trees. A recent work on this problem is proposed by Li and Young [5]. In their paper, multiple paths between the terminals are kept until all the terminals are reached. A minimum spanning tree (MST)-based method is then used to create an OARSMT. Although maze routing-based approaches can provide solutions with high quality, the space and time complexities are relatively higher which limit their applications to large-scale problems.

Most of the recent approaches on the OARSMT problem are graph-based algorithms where an OARSMT is built based on a connection graph. Hu *et al.* [6] proposed a 3-step heuristic called FORst for the OARSMT construction. In the first step, terminals are partitioned into small subsets. In the second step, terminals in the same subset are connected. In the final step, an OARSMT is constructed by combining all the subsets together. Feng *et al.* [7] proposed a method to construct obstacle-avoiding Steiner trees in arbitrary λ-geometry by Delaunay triangulation. Shen *et al.* [8] proposed another connection graph-based approach to solve the problem. An obstacle-avoiding spanning graph is first constructed and then transformed into an OARSMT. Lin *et al.* [9] extended the approach in [8] by identifying many "essential" edges which can lead to more desirable solutions in the construction of the obstacle-avoiding spanning graph. Another recent work is presented by Long *et al.* [10]. They proposed an efficient four-step algorithm to construct an OARSMT. Very recently, Ajwani *et al.* [11] presented an algorithm called FOARS, in which the obstacle-aware version of fast lookup table-based wire-length estimation is used to construct an OARSMT.

In comparison with heuristics, there has been relatively less research on exact algorithms for the OARSMT problem. Maze routing can give optimal solutions to two-terminal nets. Ganley *et al.* [12] proposed the concept of escape graph and proved that there is an optimal solution composed only of

Manuscript received July 7, 2010; revised October 8, 2010; accepted November 18, 2010. Date of current version April 20, 2011. This work was supported in part by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China, under Project 418908. Preliminary results of this work appeared in the Proceedings of IEEE/ACM International Conference on Computer-Aided Design, San Jose, CA, November 2009. This paper was recommended by Associate Editor C. J. Alpert.

T. Huang and E. F. Y. Young are with the Department of Computer Science and Engineering, Chinese University of Hong Kong, Shatin, Hong Kong (e-mail: thuang@cse.cuhk.edu.hk; fyyoung@cse.cuhk.edu.hk).

L. Li is with the Oracle Research and Development Center Company, Ltd., Shenzhen 518057, China (e-mail: liang.li@oracle.com).

Digital Object Identifier 10.1109/TCAD.2010.2098930

escape segments in the graph. A topology enumeration scheme is developed for the construction of optimal three-terminal and four-terminal OARSMTs. Note that by using escape graph, one can transform the geometric OARSMT problem to the Steiner tree problem in networks. Optimal solutions can be obtained using some exact algorithms for the network case [13].

For the RSMT problem, there are ways to find optimal solutions [15], [16]. A folk theorem states that any optimal RSMT can be unique partitioned into edge-disjoint full Steiner trees (FSTs). Based on this theorem, most of the exact algorithms use a two-phase approach, developed by Winter [17], to construct a RSMT. In the first phase, a sufficient number of FSTs is computed. In the second phase, a small subset of the FSTs with minimum total length is found and combined such that all terminals are interconnected. In the case of an obstacle-free plane, a FST has a topology, as characterized by Hwang [18], that consists of a backbone and alternating incident legs connecting the terminals. These particular shapes are simple and can be efficiently computed and examined based on some enumeration schemes [19], [20]. However, different from the aforementioned case, FSTs in the presence of obstacles can have various kinds of structures. It is usually difficult to directly construct FSTs which limits the application of this two-phase approach to the OARSMT problem.

The aim of this paper is to study the structures of FSTs when obstacles exist and provide an optimal approach for the OARSMT problem. Our work is developed based on GeoSteiner, modified and enhanced to allow non-overlapping rectangular blockages in the routing region. The contribution of this paper can be summarized as follows.

1) We propose the introduction of virtual terminals to each obstacle and prove that, with these virtual terminals, the structures of FSTs with blockages are the same as those of FSTs in the absence of obstacles. This provides theoretical support for using the two-phase approach to generate OARSMTs.
2) In the FST generation phase, we first develop a virtual terminal pruning process which can effectively reduce the number of resulting FSTs. Second, we analyze the screening tests and point out the differences in dealing with the existence of virtual terminals. Third, we propose an efficient method to construct two-terminal FST in the presence of obstacles.
3) In the FST concatenation phase, we propose a new formulation for the concatenation of FSTs with blockages. In the branch-and-cut search, we develop new separation algorithm to adapt to the presence of virtual terminals.

The rest of this paper is organized as follows. In Section II, we give the problem formulation of the OARSMT problem. In Section III, we present the definition of the FSTs with blockages and a comprehensive study on their possible structures. In Sections IV and V, we describe the two-phase approach in detail. Experimental results are shown in Section VI, followed by our conclusion in Section VII.

II. Problem Formulation

In the OARSMT problem, we are given a set T of terminals on the 2-D plane and a set O of rectangular obstacles. No obstacle can overlap with each other, but they can be line-touched at the boundary. A terminal cannot be located inside an obstacle, but it can be on the boundary of an obstacle. The objective of the problem is to give a tree with shortest length that connects all the terminals, possibly through some additional points, using only horizontal and vertical lines. This tree is known as an OARSMT of the problem. No edge on the OARSMT can intersect with an obstacle, but they can be point-touched at a corner or line-touched on the boundary of an obstacle. The OARSMT problem is well known to be NP-complete.

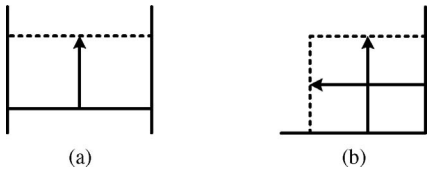


Fig. 1. (a) Shifting and (b) flipping.

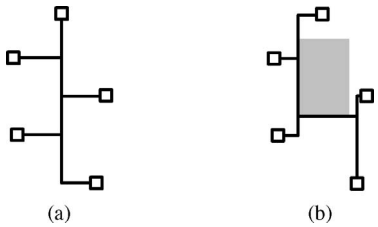


Fig. 2. FST structures (a) in the absence and (b) in the presence of obstacles.

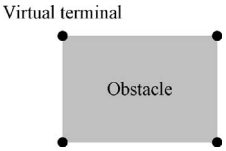


Fig. 3. Locations of virtual terminals of an obstacle.

III. FSTs with Blockages

Before the definition of FSTs, we first define the concept of equivalent trees as in [18]. Let s be a rectilinear Steiner tree. A tree s' is equivalent to s if and only if s' can be obtained from s by *shifting* [Fig. 1(a)] or *flipping* [Fig. 1(b)] some edges which have no nodes on them. A FST over a set P of terminals is a RSMT s of P such that in s and all its equivalent trees, every terminal $t \in P$ is a leaf node. In the absence of obstacles, a FST has a topology that consists of a backbone and alternating incident legs connecting the terminals. An example is shown in Fig. 2(a). However, the structures of FSTs in the presence of obstacles can be different. An example of a FST over the same set of terminals in the presence of one obstacle is shown in Fig. 2(b). The various structures make the construction of FSTs in the presence of obstacles difficult. In this section, we aim to show how this problem can be solved by introducing virtual terminals and the newly defined FSTs.

A. Definition

Given a set T of terminals and a set O of obstacles, we first add four virtual terminals to the four corners of each obstacle, as shown in Fig. 3. We use V to denote the set of all virtual

Fig. 4. Forbidden edges in a FST with blockages.

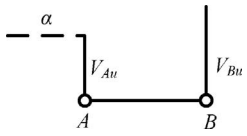


Fig. 5. Adjacent Steiner points.

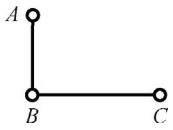


Fig. 6. Steiner chain structure.

terminals. These virtual terminals together with the real ones in T are used to construct the FSTs with blockages defined in this paper. With virtual terminals added to the obstacles, a FST with blockages (denoted by s) over a set of terminals $T_s \subseteq (V + T)$ has the properties as follows:

1) s is an OARSMT over the terminal set T_s;
2) every terminal $t \in T_s$ has degree one in s and all its equivalent trees;
3) all the equivalent trees of s cannot contain the types of edges as shown in Fig. 4. (Otherwise, we can further split s into two smaller FSTs at the virtual terminal.)

With the definition, we can easily verify that an OARSMT is a union of FSTs with blockages. Therefore, it is possible to construct an OARSMT by the concatenation of FSTs with blockages. In the following, we will show that the structure and shape of a FST satisfying the above properties will follow some very simple forms. For simplicity, we will use FSTs to denote FSTs with blockages in the rest of this paper.

B. Structures

The notations we are going to use in this section are the same as in [18]. A vertex can be a *node* (real terminal or virtual terminal) or a *Steiner point*. An edge between two vertices is a sequence of alternating vertical and horizontal lines and each turning point is a *corner*. A line has only one direction but may contain a number of vertices on it. $V_{xu}(V_{xd})$ denotes the maximal vertical line at point x which is above (below) x excluding x itself. Similarly, H_{xr} (H_{xl}) denotes the maximal horizontal line at point x which is on the right (left) of x excluding x itself. If a line ends at a node 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, we use an empty circle to represent a Steiner point and an empty square to represent a node.

The steps of proof to derive the structures of the proposed FSTs are similar to those in [18], but there are obstacles in the routing region now. In the following, we will present the proof for Lemma 2 to illustrate the differences when obstacles exist. The proofs for other lemmas are in Appendix A.

Lemma 1: All Steiner points either have degree three or degree four [22].

Lemma 2: Let A and B be two adjacent Steiner points in a FST. 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 ends at a corner turning away from V_{Bu}.

Proof: See Fig. 5. Suppose A is to the left of B.

1) V_{Au} contains no node on it, for otherwise we can shift AB to that node to obtain an equivalent tree in which the node has degree more than one. If the line AB cannot be shifted due to some obstacles, we can shift AB to the boundary of the obstacle obtaining an equivalent tree with forbidden structures as shown in Fig. 4, a contradiction to the definition of FSTs.

2) 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. If the line cannot be extended due to obstacles, an equivalent tree containing the structures as shown in Fig. 4 can be obtained by shifting AB to the obstacles.

3) Therefore, the other endpoint of V_{Au} cannot be a Steiner point since it has no lines going right or upward, and it hence must be a corner turning left.

4) 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 obstacles, an equivalent tree containing the structures in Fig. 4 can be obtained. ■

Corollary: Suppose V_{Bu} contains a vertex, then V_{Au} is a corner line that ends at a corner turning away from V_{Bu} and $|V_{Au}| < |V_{Bu}|$.

Lemma 3: Suppose V_{xu} (x is a vertex) is a corner line that ends at a corner 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 a FST, the Steiner points in s form a chain. We call the chain of Steiner points the *Steiner chain*.

Lemma 6: Suppose s is a FST. Then its Steiner chain cannot contain the subgraph shown in Fig. 6.

Define a staircase to be a continuous path of alternating vertical lines and horizontal lines such that their projections on the vertical and horizontal axes have no overlaps.

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

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

Lemma 9: Suppose s is a FST. 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 on the Steiner chain contains no Steiner point except at the end point, or vice versa.

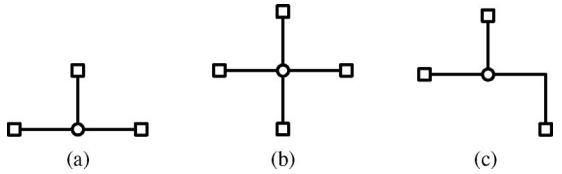


Fig. 7. Three possible tree structures when $m = 1$. (a) First type of tree structures when $m = 1$. (b) Second type of tree structures when $m = 1$. (c) Third type of tree structures when $m = 1$.

Note that the structure of a FST is not affected by 90° rotation. In the following lemmas and theorems, we assume that if s is a FST, the corresponding Steiner chain will consist of a set of vertical lines and adjacent vertical lines are connected by corners. We label the ith Steiner point on the chain counting from above by A_i.

Lemma 10: Suppose s is a FST. Every Steiner point on s must have a horizontal node line and the node lines alternate in the left-right direction.

Lemma 11: Suppose s is a FST. Then a corner connecting A_i and A_{i+1} can be transferred to one connecting A_{i-2} and A_{i-1}, or one connecting A_{i+2} and A_{i+3}, regardless of whether the place it transfers to has a corner or not.

Theorem 1: Suppose s is a FST and let m be the number of Steiner points. There exists a s' equivalent to s such that as follows:

1) if m is odd, the Steiner chain of s' is a straight line;
2) if m is even, all the Steiner points are on a straight line except possibly the last one.

To summarize, let s be a FST and let m be the number of Steiner points. s belongs to one of the three types as follows.

1) $m = 1$*: s is one of the trees as shown in Fig. 7.*
2) $m > 1$ *and the Steiner chain is a straight line:* the horizontal node line at the sequence of Steiner points must alternate in the left-right direction. Hence, each Steiner point has exactly one horizontal node line except A_1 and A_m. The tree structure is shown in Fig. 8(a).
3) $m > 1$ *and the Steiner chain is a straight line except that the last two Steiner points are connected by a corner:* without loss of generality, we assume that $H_{A_m l}$ exists. Then each Steiner point except A_m and A_1 has exactly one horizontal line alternate on the left-right direction and $V_{A_m d}$ is a node line. The tree structure is shown in Fig. 8(b).

As can be observed from the figures, with the introduced virtual terminals, the structures of FSTs with blockages are the same as those of FSTs in the absence of obstacles. This indicates that we can now use the two-phase approach to construct an OARSMT efficiently. In the first phase, we generate a sufficient set of FSTs. In the second phase, we identify and combine a subset of FSTs with minimum total length such that all real terminals are interconnected.

IV. FST Generation

To grow FSTs of a RSMT, Zachariasen [19] proposed an efficient algorithm in which some preprocessing information is applied to prune away those FSTs that are not required in any RSMTs. In this paper, we modify this algorithm for the generation of FSTs with blockages. Our FST generation algorithm differs from the previous one in the following aspects. First, we design a pruning process to reduce the number virtual terminals and the number of resulting FSTs. Second, we extend the screening tests to handle virtual terminals and blockages. Third, we develop an efficient approach to construct two-terminal FSTs when virtual terminals exist.

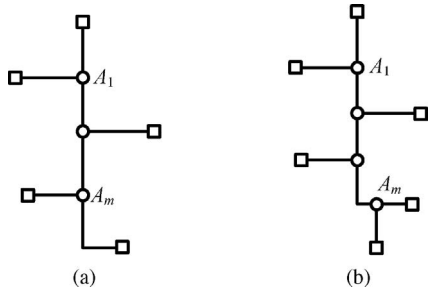


Fig. 8. Two possible structures when $m > 1$. (a) Tree structure when $m > 1$ and the Steiner chain is a straight line. (b) Tree structure when $m > 1$ and the Steiner chain is a straight line except that the last two Steiner points are connected by a corner.

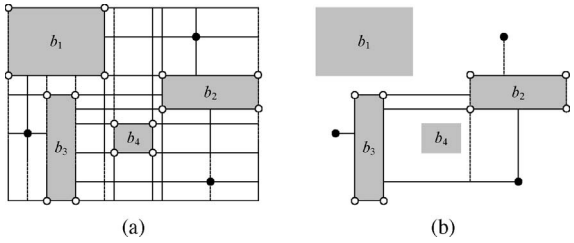


Fig. 9. (a) Escape graph. (b) Reduced escape graph.

A. Pruning of Virtual Terminals

As introduced in the previous section, prior to the implementation of the algorithm, we will add four virtual terminals at the four corners of each obstacle. These virtual terminals together with the real ones are used to construct the FSTs. However, it is often the case that a fraction of the virtual terminals can be removed from further consideration, while still maintaining the optimality of the solution. Therefore, we will apply pruning on virtual terminals to remove those that are not necessary.

First, an escape graph for the problem is constructed. The graph is composed of escape segments that extend from real terminals and virtual terminals, and end at an obstacle boundary or the boundary of the whole routing region. An example is shown in Fig. 9(a). Real terminals are represented by solid circles and virtual terminals are represented by empty circles. It has been proved that there is an OARSMT composed only of segments and vertices in the escape graph. Moreover, some reduction tests [12], [14] can be applied to eliminate many vertices from the graph to produce a reduced escape graph, as shown in Fig. 9(b), while still guaranteeing the existence of an optimal solution. Therefore, it is sufficient to consider only the virtual terminals that survive in the reduced escape graph. Those not in the graph are removed

from the list to cut down the number of resulting FSTs. In the example, all the virtual terminals of obstacle b_1 and b_4 will be removed. If all four virtual terminals of an obstacle are deleted, we will also remove the corresponding obstacle as it does not affect the routing. Empirical studies show that for the OARSMT problems consist of fewer terminals than obstacles, this pruning procedure can effectively reduce the number of virtual terminals as well as the produced FSTs.

B. Generation of FSTs with Three or More Terminals

The structures described in Theorem 1 will be used to identify FSTs. To reduce the number of resulting FSTs, we identify some necessary conditions for a FST to be a part of an OARSMT as in [19]. Most of the conditions in [19] are applicable to the proposed FSTs after some modifications. In the following, we will focus on the modifications made when obstacles and virtual terminals exist.

The bottleneck Steiner distance can be used to eliminate useless FSTs when obstacles exist. Let OARMST(T) be an obstacle avoiding rectilinear MST of the point set T and t_i, $t_j \in T$ be a pair of vertices. The bottleneck Steiner distance $\delta_{OARMST(t_it_j)}$ between t_i and t_j is equal to the length of the longest edge on the unique path between t_i and t_j in OARMST(T). Salowe *et al.* [15] proposed a theorem stating that if *MST* and *SMT*, respectively, are a MST and a Steiner minimal tree on a set of vertices T, then $\delta_{MST(t_it_j)} \geq \delta_{SMT(t_it_j)}$ for any $t_i, t_j \in T$. It can be easily verified that the property also holds for OARMST(T) and OARSMT(T). For a FST s to be part of an OARSMT, we require that $\delta_{MST(t_it_j)} \geq \delta_{s(t_it_j)}$ for any $t_i, t_j \in s \cap T$.

The empty diamond property proposed in [19] states that no other points of the RSMT can lie in $\mathscr{L}(u, v)$, where uv is a (horizontal or vertical) segment and $\mathscr{L}(u, v)$ is an area on the plane such that all the points in this area are closer to both u and v than u and v are to each other. However, when there are obstacles and virtual terminals, the points which cannot lie in $\mathscr{L}(u, v)$ are the real terminals in T only.

The empty corner rectangle property is also proposed in [19]. Let uw and vw denote two perpendicular segments sharing a common endpoint w. Then, no other points of the RSMT can lie in the interior of the smallest axis-aligned rectangle containing u and v. However, when there are obstacles and virtual terminals in the routing region, we only need to consider real terminals which can be projected on uw and vw without intersecting with any obstacles.

We also make use of the empty inner rectangle property proposed in [19]. A FST can be transformed to its corner-flipped version by shifting segments and flipping corners. The empty inner rectangle property states that no terminal (real or virtual) should be located between the backbone of the origin topology and that of the corner-flip topology.

Based on the above properties, we can generate all the required FSTs by growing them recursively as in [19].

C. Generation of FSTs with Two Terminals

For those FSTs with exactly two terminals, we will construct them by the following method. First of all, these FSTs can be divided into two types. The first type has its two end points both in T. The second type has at least one of its end points in V.

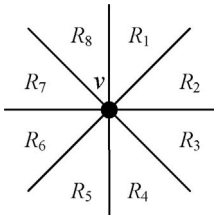


Fig. 10. Eight regions of a terminal.

For the first type, we can construct them according to the following lemma which is proposed by Fößmeier *et al.* [20].

Lemma 12: Let $G = (U, E)$ be a graph with edges assigned mutually distinct weights and let U' be a subset of U. Let L be an MST of G and L' be an MST of $G[U']$, the subgraph of G induced by U'. Then every edge (u, w) in L where both u and w are in U' will also appear in L'.

This lemma indicates that every two-terminal FST, in the OARSMT and with its two end points both in T, will also appear in the OARMST of T. In order to generate all possible type one two-terminal FSTs, we only need to construct an OARMST of T and include all the edges in it as candidates. In order to handle the requirement of mutually distinct weights, we arrange the edges with the same length by comparing their positions in the edge array. The one that has a smaller index is assumed to be "shorter." Note that this will not affect the optimality of the generated OARSMT.

For the second type, we will make use of a lemma proposed by Yao [23]. We know that at least one of the two end points of the FST under construction is in V and the rectangular area covered by the two end points is obstacle free (otherwise we can flip the edge to the boundary of the obstacle to obtain an equivalent tree with forbidden structures). For each virtual terminal $v \in V$, we divide its surrounding area into eight regions R_i for $i = 1, \ldots, 8$, as shown in Fig. 10. In every region R_i, we find the point $t \in T$ that has the shortest Manhattan distance (d_{vt}) from v and the rectangular area covered by v and t has no obstacle. Then, the edge connecting v and t is a two-terminal FST candidate. In this region R_i, we also find those points $u \in V$ with distance $d_{vu} \leq d_{vt}$ and the rectangular area covered by v and u is obstacle free. Then, the edge connecting v and u will also be included as a FST candidate. To verify the correctness of this approach, we assume on the contrary that there exist a two-terminal FST in the OARSMT, but not in our candidate set. Without loss of generality, we use $v \in V$ and $w \in T + V$ to denote the two end points of the FST. Assume that w is in the R_k region of v. Since the FST is not in our candidate set, there exist a terminal $t \in T$ in R_k such that $d_{vt} < d_{vw}$. According to [23], we have $d_{wt} < d_{vw}$, but this is impossible for otherwise we can delete (v, w) and connect either (v, t) or (w, t) to build a shorter tree. Therefore, the FST cannot exist which proves the correctness of our approach.

Based on the above methods, we can find all necessary two-terminal FSTs. The empty diamond and empty inner rectangle properties are also used to reduce the number of two-terminal FSTs.

V. FST Concatenation

The second phase of the algorithm is to use the FSTs generated in the first phase to construct an OARSMT spanning all real terminals with the minimum total length. In the construction of RSMTs, Warme [21] found that the FST concatenation problem is equivalent to the MST problem in hypergraph and formulated it as an integer program (IP). A branch-and-cut algorithm is used to solve this problem. In this section, we will show that the FST concatenation problem in this paper can also be formulated as an IP and solved by using the branch-and-cut search. Generally, our FSTs concatenation phase differs from the previous one in the following aspects. We modify the IP formulation for FST concatenation and the separation algorithm in [21] to handle virtual terminals. New features are introduced to accommodate the presence of virtual terminals. We also provide a theoretical proof to verify the correctness of the new separation algorithm.

A. IP Formulation

In the following, let S be the set of all FSTs found. Let T be the set of all real terminals and V be the set of all virtual terminals that survive after pruning. Let $|T|$ be the number of real terminals, $|S|$ be the number of FSTs in S, and $|V|$ be the number of virtual terminals. Each FST $s_i \in S$ is associated with a binary variable x_i indicating whether s_i is taken as a part of the OARSMT. Besides, there are binary variables y_i for $i = 1 \ldots |V|$ indicating whether virtual terminal $v_i \in V$ is connected in the OARSMT. We use $|s_i|$ to denote the size of s_i, i.e., the number of terminals (including virtual ones) connected by s_i, and use l_i to denote the length of s_i. The IP formulation is as follows:

Minimize:

$$\sum_{i=1}^{|S|} l_i \times x_i \tag{1}$$

Subject to:

$$\sum_{i=1}^{|S|} x_i(|s_i| - 1) = |T| - 1 + \sum_{i=1}^{|V|} y_i \tag{2}$$

$$2y_j \leq \sum_{i:v_j \in s_i} x_i \quad \forall v_j \in V \tag{3}$$

$$4y_j \geq \sum_{i:v_j \in s_i} x_i \quad \forall v_j \in V \tag{4}$$

$$\sum_{i:s_i \in (X:T+V-X)} x_i \geq 1$$

$$\forall X \subseteq T + V \text{ and } T \nsubseteq X \text{ and } X \cap T \neq \emptyset \tag{5}$$

$$\sum_{i:s_i \cap X \neq \emptyset} x_i(|s_i \cap X| - 1) \leq |X \cap T| + \sum_{i:v_i \in X} y_i - 1$$

$$\forall X \subset T + V \text{ and } X \cap T \neq \emptyset \text{ and } |X| \geq 2 \tag{6}$$

$$\sum_{i:s_i \cap X \neq \emptyset} x_i(|s_i \cap X| - 1) \leq \sum_{i:v_i \in X} y_i - \max_{i:v_i \in X}(y_i)$$

$$\forall X \subseteq V \text{ and } |X| \geq 2. \tag{7}$$

The notation $(X : T + V - X)$ in (5) means $\{s_i \in S : s_i \cap X \neq \emptyset \wedge s_i \cap (T + V - X) \neq \emptyset\}$. Constraint (2) is the total degree constraint. It requires the right amount of FSTs to construct an OARSMT. Constraints (3) and (4) bound the degree of any selected virtual terminal to be two, three, or four. Constraints (5) are the *cutset constraints*. The constraints require that a solution should be connected, i.e., for any cut with partitions X and $T + V - X$, there must be at least one selected FST to connect them. We require $X \cap T \neq \emptyset$ and $T \nsubseteq X$, because we do not need to ensure the connectivity of the virtual terminals. Constraints (6) and (7) are the subtour elimination constraints that are used to eliminate cycles. In (6), we consider those sets $X \cap T \neq \emptyset$. Since y_i tells whether v_i is selected, $|X \cap T| + \sum_{i:v_i \in X} y_i$ gives the exact number of selected terminals including virtual ones in X. In (7), we use $\sum_{i:v_i \in X} y_i$ to indicate the number of selected terminals in X. Since it is possible that the number of selected terminals in X is equal to zero, we do not simply subtract one from the right-hand side of the inequality. Instead, the term $\max_{i:v_i \in X}(y_i)$ is used to ensure that the inequality is not binding when the number of selected terminals in X is zero.

```
Algorithm branch-and-cut(S)
Input: S // The set of all FSTs
Output: OARSMT
 1: initialization
 2: add the first node to the node list
 3: while node list is not empty do
 4:     select a node from the node list
 5:     repeat
 6:         node processing
 7:         if LP feasible and objective value < best known objective value then
 8:             if the LP solution is integral and connected then
 9:                 save it as the best known integral solution
10:             end if
11:             separation
12:         end if
13:     until 1: LP infeasible, or
                2: objective value ≥ best known objective value,
                 or
                3: separation found no violation
14:     if case 1 or case 2 then
15:         delete the current node
16:     end if
17:     if case 3 then
18:         if the solution is fractional then
19:             branching
20:         end if
21:         if the solution is integral then
22:             delete the current node
23:         end if
24:     end if
25: end while
26: return the best integral solution // OARSMT
```

Fig. 11. Pseudocode of the branch-and-cut algorithm.

B. Branch-and-Cut

The IP described in the above section is solved via a branch-and-cut framework using lower bounds provided by the linear programming (LP) relaxation. We adopt the algorithm proposed by Warme [21] and extend it for solving the IP formulation of our FST concatenation problem. The pseudocode of the algorithm is shown in Fig. 11. In the following, we will give a brief overview of the algorithm, including initialization, node processing, and branching, and point out the differences in the separation algorithm in order to deal with our formulation. The readers may refer to [21] for more details.

1) *Initialization:* Since there are an exponential number of constraints according to the problem formulation, we handle

them incrementally by using some separation methods. A constraint pool is used to keep all the currently processing constraints of the IP. At the beginning of the algorithm, the constraint pool is initialized with the total degree (2), constraints for virtual terminals (3) and (4), all one-terminal cutset constraints [(5) with $|X| = 1$], and all two-terminal subtour elimination constraints [(6) and (7) with $|X| = 2$]. Besides, an LP tableaux is constructed to store the constraints (which is a subset of the constraints retained in the constraint pool) being handled by the LP solver. The initial LP tableaux consists of all the constraints in the constraint pool except the two-terminal subtour elimination constraints.

2) *Node Processing:* The objective of the node processing procedure is to compute an optimal LP solution over the current constraint pool. The process begins with solving a linear relaxation with the constraints in the LP tableaux. If a solution exists, we will scan the constraint pool and check for violations. All violated constraints found will be added to the LP tableaux which is solved again in the next iteration. This operation terminates when the LP solution satisfies all the constraints in the pool (LP feasible) or a feasible solution does not exist (LP infeasible). If the result is LP infeasible or the objective value of the LP solution exceeds the objective value of the best-known integral solution, the processing of this node ends and the node will be deleted. If the objective value of the LP solution is better than that of the best-known integral solution, the integrality and connectivity of the LP solution is checked. If the solution is both integral and connected, it is saved as the best-known integral solution. A separation procedure will then be invoked. Note that after obtaining an optimum over the current pool, slack constraints[1] will be deleted from the LP tableaux (but are retained in the constraint pool).

3) *Separation:* The objective of the separation procedure is to find a set of constraints that is not present in the constraint pool, but is violated by the current solution. These constraints will be added to the constraint pool. There are mainly two sets of constraints to be considered, namely, the cutset constraints (5) and the subtour elimination constraints (6) and (7).

The first step in the separation procedure is to find the cuts $(X : T + V - X)$ with $\sum_{i:s_i \in (X:T+V-X)} x_i = 0$ that violate the cutset constraints (6). We first compute the connected components $D_1, D_2, D_3, \ldots, D_k$ of the solution. Since we do not need to ensure the connectivity of the virtual terminals, we require that $D_i \cap T \neq \emptyset, \forall 1 \leq i \leq k$. If $k > 1$, there exist cutsets of zero weight. If $k < 10$, we generate cutsets constraints for all the cuts induced by the connected components. If $k \geq 10$, we only generate the cutset constraints $(D_i : T + V - D_i)$ for $1 \leq i \leq k$. Notice that we do not consider those cuts with $0 < \sum_{i:s_i \in (X:T+V-X)} x_i < 1$ because they are too expensive to be identified while little improvement in the objective value can be made.

The second step is to find violations of the subtour elimination constraints (6) and (7). We define a function as follows:

$$f(X) = |X \cap T| + \sum_{i:v_i \in X} y_i - \sum_{i:s_i \cap X \neq \emptyset} x_i(|s_i \cap X| - 1). \quad (8)$$

Then finding violations of constraints (6) is equivalent to finding an $X \subset T + V$ such that $X \neq \emptyset$ and $f(X) < 1$. Before exactly solving this problem, we first apply problem reductions to speedup the process. In [21], the "congestion level" of a real terminal b_{t_j} is defined as follows:

$$b_{t_j} = \sum_{i:t_j \in s_i} x_i. \quad (9)$$

A real terminal t_j is uncongested if $b_{t_j} \leq 1$. In this paper, we define the "congestion level" of a virtual terminal as follows:

$$b_{v_j} = \sum_{i:v_j \in s_i} x_i. \quad (10)$$

We say that a virtual terminal v_j is uncongested if $b_{v_j} \leq y_j$. By the definition of "congestion level," we can have the following lemma.

Lemma 13: If a terminal u is uncongested and $f(X \cup \{u\}) < 1$, then $f(X) \leq f(X \cup \{u\}) < 1$.

The proof for the lemma is in Appendix B. According to Lemma 13, we can eliminate all uncongested terminals while looking for violations of the subtour elimination constraints. Since subtour elimination constraints are used to eliminate cycles, we can further confine our search to within several biconnected components. We use $C_1, C_2, C_3, \ldots, C_k$ to denote the biconnected components in which every terminal is congested. Now, the problem is reduced to identifying violations within $C_1, C_2, C_3, \ldots, C_k$. For each component C_i with less than ten terminals, we will enumerate all subsets X of C_i checking for violations of (6) and (7). For each remaining component C_i, we use a deterministic network flow method to find violations of (6) and (7). The deterministic flow network $G = (N, A)$ is defined as follows. Let $N = \{\sigma\} \cup Y \cup Z \cup \{\tau\}$ be the set of nodes in the graph, where $Y = \{s_i : s_i \cap C_i \neq \emptyset\}$, and $Z = \{t_j : t_j \in C_i\} \cup \{v_j : v_j \in C_i\}$. Let the arcs in the graph be $A = A_1 \cup A_2 \cup A_3$, where $A_1 = \{(\sigma, s_i)\}$, $A_2 = \{(s_i, t_j) : t_j \in s_i\} \cup \{(s_i, v_j) : v_j \in s_i\}$, and $A_3 = \{(t_j, \tau) : t_j \in C_i\} \cup \{(v_j, \tau) : v_j \in C_i\}$. Let the arcs A_1 have capacity x_i. Let arcs A_2 have infinite capacity. Let arcs $(t_j, \tau) \in A_3$ have capacity $b_{t_j} - 1$, and $(v_j, \tau) \in A_3$ have capacity $b_{v_j} - y_j$. The flow network is shown in Fig. 12. Note

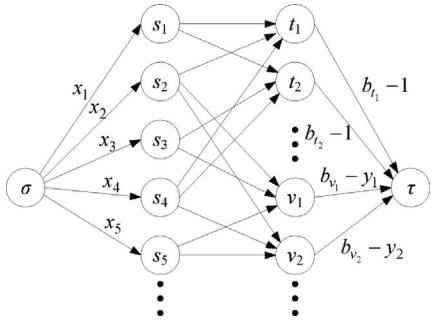


Fig. 12. Flow network formulation.

[1]Slack values of linear constraints are available to be queried from the LP solver.

TABLE I
COMPARISON OF THE FST GENERATION PHASE WITH AND WITHOUT THE PRUNING PROCESS

Test Cases	$\|T\|$	$\|O\|$	Without Pruning		With Pruning		FST Reduction (%)	Pruning Time	Test Cases	$\|T\|$	$\|O\|$	Without Pruning		With Pruning		FST Reduction (%)	Pruning Time
			$\|V\|$	$\|S\|$	$\|V\|$	$\|S\|$							$\|V\|$	$\|S\|$			
IND1	10	32	113	2120	20	75	96.46	<0.01	RT1_40	10	40	160	15263	30	234	98.47	0.05
IND2	10	43	147	494	62	229	53.64	<0.01	RT2_30	50	30	120	6529	80	2539	61.11	0.16
IND3	10	50	163	579	48	140	75.82	0.01	RT3_30	100	30	120	9366	100	8982	4.10	0.42
IND4	25	79	272	12442	79	190	98.47	0.02	RT4_30	100	30	120	3769	96	2496	33.78	0.25
IND5	33	71	249	1193	128	472	60.43	0.01	RT5_30	200	30	120	2837	108	2627	7.40	2.58
RC1	10	10	40	346	24	186	46.24	0.01	RC6_rand_40	100	40	160	6639	152	6273	5.51	0.28
RC2	30	10	40	330	24	280	15.15	0.01	RC7_rand_40	200	40	160	3904	160	3904	0.00	2.19
RC3	50	10	40	324	36	306	5.56	0.01	RC8_rand_30	200	30	120	4214	104	3800	9.82	2.96
RC4	70	10	36	350	36	350	0.00	0.03	RC9_rand_30	200	30	120	3002	112	2728	9.13	2.54
RC5	100	10	40	772	36	761	1.42	0.05	RC10_rand_30	500	30	120	2728	120	2728	0.00	16.12
RC6_40	100	40	160	7153	160	7153	0.00	0.34	RT1_rand_40	10	40	160	148236	32	418	99.72	0.03
RC7_40	200	40	160	4007	152	3858	3.72	1.68	RT2_rand_30	50	30	120	4024	84	1925	52.16	0.17
RC8_30	200	30	120	3397	108	3301	2.83	1.56	RT3_rand_30	100	30	120	3255	92	2558	21.41	0.49
RC9_30	200	30	120	2590	112	2537	2.05	2.60	RT4_rand_30	100	30	120	3828	96	3296	13.90	0.53
RC10_30	500	30	120	4055	116	4041	0.35	13.76	RT5_rand_30	200	30	120	3126	116	3113	0.42	2.21
									Average							29.30	1.70

$|T|$ denotes the number of real terminals. $|O|$ denotes the number of obstacles. $|V|$ denotes the number of virtual terminals. $|S|$ denotes the number of FSTs generated. CPU time is in seconds.

TABLE II
RESULTS OF THE FST GENERATION PHASE AND THE FST CONCATENATION PHASE

Test Cases	$\|T\|$	$\|O\|$	FST Generation		FST Concatenation				Total Time	Test Cases	$\|T\|$	$\|O\|$	FST Generation		FST Concatenation				Total Time
			$\|S\|$	Time	n_{node}	n_{cons}	L_{opt}	Time					$\|S\|$	Time	n_{node}	n_{cons}	L_{opt}	Time	
IND1	10	32	75	0.36	2	219	604	0.10	0.46	RT1_40	10	40	234	0.53	1	1336	1872	0.58	1.11
IND2	10	43	229	0.61	3	813	9500	2.83	3.44	RT2_30	50	30	2539	1.29	3	8329	44 294	43.65	44.94
IND3	10	50	140	0.47	3	615	600	0.84	1.31	RT3_30	100	30	8982	3.90	1	36 718	7580	175.33	179.23
IND4	25	79	190	2.38	1	1085	1086	0.77	3.15	RT4_30	100	30	2496	2.73	1	9261	7825	59.77	62.50
IND5	33	71	472	2.32	1	3076	1341	22.41	24.73	RT5_30	200	30	2627	10.59	1	7713	42879	29.14	39.73
RC1	10	10	186	0.21	4	751	25 980	0.37	0.58	RC6_rand_40	100	40	6273	5.06	2	25 597	76 840	533.17	538.23
RC2	30	10	280	0.27	1	912	41 350	0.28	0.55	RC7_rand_40	200	40	3904	14.17	2	13 311	105 358	139.89	154.06
RC3	50	10	306	0.40	1	1036	54 160	0.18	0.58	RC8_rand_30	200	30	3800	10.09	1	17 674	107 811	374.53	384.62
RC4	70	10	350	0.53	3	1033	59 070	0.57	1.10	RC9_rand_30	200	30	2728	10.44	5	9861	105 875	73.55	83.99
RC5	100	10	761	0.96	1	2117	74 070	1.13	2.09	RC10_rand_30	500	30	2728	76.93	11	15 350	162 470	655.59	732.52
RC6_40	100	40	7153	5.68	2	34 802	76 946	258.04	263.72	RT1_rand_40	10	40	418	0.36	3	1818	1817	1.66	2.02
RC7_40	200	40	3858	12.45	1	13 791	105 956	111.98	124.43	RT2_rand_30	50	30	1925	1.25	3	6804	44 358	22.11	23.36
RC8_30	200	30	3301	11.06	1	13 425	107 833	483.88	494.94	RT3_rand_30	100	30	2558	3.09	1	9071	7595	29.76	32.85
RC9_30	200	30	2537	10.05	2	10 636	106 139	163.55	173.60	RT4_rand_30	100	30	3296	3.25	2	12 017	7681	60.87	64.12
RC10_30	500	30	4041	73.87	4	16 024	163 050	1388.92	1462.79	RT5_rand_30	200	30	3113	10.70	1	10 793	42 821	86.66	97.36
										Average				9.20				157.40	166.60

$|T|$ denotes the number of real terminals. $|O|$ denotes the number of obstacles. $|S|$ denotes the number of FSTs generated. n_{node} denotes the number of branch-and-cut nodes. n_{cons} denotes the number of constraints. L_{opt} denotes the OARSMT length. CPU time is in seconds.

that different from the flow network formulation in [21], there are nodes that represent virtual terminals in our formulation.

We define a source to terminal cut $(W : N - W)$ of G such that $\sigma \in W$ and $\tau \in (N - W)$. The capacity of the cut $c(W)$ is the sum of the capacity of all arcs $(a, b) \in A$ such that $a \in W$ and $b \in (N - W)$. We have the following theorem.

Theorem 2: Let $(W : N - W)$ be a source to terminal cut of G that minimize $c(W)$. Let $X_m = \{u : u \in T + V \wedge u \in N - W\}$. Then X_m minimizes $f(X)$.

The proof for the theorem is in Appendix C. This theorem states that finding an X of C_i that violates (6) can be reduced to finding a minimum cut on the flow network G. This problem can be solved in polynomial time. Note that although the above procedure is not exact in finding violations of constraints (7), it can still provide good estimations.

4) *Branching:* If no violation can be found by separation and the node processing terminates with a fractional solution, branching on the current node occurs. A branch variable x_i (y_j) with non-integral value is selected. Two new nodes are generated by appending the constraints $x_i = 0$ or $x_i = 1$ ($y_j = 0$ or $y_j = 1$) to the current node. The processing of the current node terminates. New nodes are selected for processing until there is no node left in the node list.

VI. EXPERIMENTAL RESULTS

We implement our algorithm based on GeoSteiner-3.1 [24] and all experiments are conducted on a Sun Blade 2500 workstation with two 1.6 GHz processors and 2 GB memory. Note that although a dual processor machine is used, our program runs sequentially on a single processor. There are totally 20 benchmark circuits. Benchmarks IND1–IND5 are industrial test cases from Synopsys. Benchmarks RC1–RC10 are adopted from [7]. Benchmarks RT1–RT5 are random test

TABLE III
COMPARISON OF SOME RECENT HEURISTICS BASED ON THE OPTIMAL SOLUTIONS

Test Cases	L_{opt}	Lin [9]			Long [10]			Test Cases	L_{opt}	Lin [9]			Long [10]		
		H_1	D_1 (%)	T_1	H_2	D_2 (%)	T_2			H_1	D_1 (%)	T_1	H_2	D_2 (%)	T_2
IND1	604	632	4.43	<0.01	639	5.48	<0.01	RT1_40	1872	1872	0.00	<0.01	2040	8.24	0.01
IND2	9500	9600	1.04	<0.01	10 000	5.00	<0.01	RT2_30	44 294	45 221	2.05	0.01	45 983	3.67	0.01
IND3	600	613	2.12	<0.01	623	3.69	<0.01	RT3_30	7580	7829	3.18	0.03	7698	1.53	0.01
IND4	1086	1121	3.12	<0.01	1130	3.89	<0.01	RT4_30	7825	8088	3.25	0.02	8055	2.86	0.01
IND5	1341	1364	1.69	<0.01	1379	2.76	<0.01	RT5_30	42 879	44 109	2.79	0.05	43 711	1.90	0.01
RC1	25 980	26 900	3.42	<0.01	26 120	0.54	<0.01	RC6_rand_40	76 840	79 111	2.87	0.02	78 956	2.68	0.01
RC2	41 350	42 210	2.04	<0.01	41 630	0.67	<0.01	RC7_rand_40	105 358	108 446	2.85	0.06	107 159	1.68	0.01
RC3	54 160	55 750	2.85	<0.01	55 010	1.55	<0.01	RC8_rand_30	107 811	111 033	2.90	0.06	110 525	2.46	0.01
RC4	59 070	60 350	2.12	<0.01	59 250	0.30	<0.01	RC9_rand_30	105 875	109 649	3.44	0.06	108 115	2.07	0.01
RC5	74 070	76 330	2.96	0.01	76 240	2.85	<0.01	RC10_rand_30	162 470	168 400	3.52	0.27	165 720	1.96	0.03
RC6_40	76 946	79 118	2.75	0.02	78 805	2.36	0.01	RT1_rand_40	1817	1831	0.76	<0.01	2012	9.69	0.01
RC7_40	105 956	108 948	2.75	0.06	107 913	1.81	0.02	RT2_rand_30	44 358	45 797	3.14	0.01	45 968	3.50	0.01
RC8_30	107 833	110 731	2.62	0.05	111 073	2.92	0.01	RT3_rand_30	7595	7811	2.76	0.02	7726	1.70	0.01
RC9_30	106 139	109 101	2.71	0.05	108 027	1.75	0.02	RT4_rand_30	7681	7930	3.14	0.01	7847	2.12	0.01
RC10_30	163 050	169 200	3.63	0.27	166 070	1.82	0.04	RT5_rand_30	42 821	44 122	2.95	0.02	43 795	2.22	0.01
								Average			2.66	0.03		2.85	0.01

L_{opt} denotes the OARSMT length generated by us. H_i denotes the result of the corresponding heuristic. $D_i = (H_i - L_{opt})/H_i$. T_i is the running time of the corresponding heuristic in seconds.

cases used in [9]. For overlapping obstacles in the test cases, we dissect them into several rectangular blockages. Note that [9], [10] are modifying the test cases in the same way so that the comparison in this paper is fair. RC5–RC10 and RT1–RT5 are too large to be solved optimally. Therefore, we will only take some of the blockages in those cases. For RT1, RC6, and RC7, 40 blockages are taken. For the rest test cases, 30 blockages are taken. In order to clearly demonstrate the performance of the proposed algorithm, we use two different ways to select blockages. The first set of new test cases are generated by taking the first few blockages from the corresponding benchmarks. The set of new test cases are denoted by "benchmark_number," in which "benchmark" is the original benchmark and "number" is the number of blockages taken. The second set of new test cases are generated by taking the blockages randomly. We use "benchmark_rand_number" to denote them.

Table I illustrates the comparison of the FST generation phase with and without the pruning of virtual terminals. To show the efficiency of the pruning process, we compare the number of FSTs generated with and without the pruning process. On average, a 29.3% reduction on the number of FSTs can be achieved. The running time of the pruning process is 1.7 s, which is negligible in comparison with the total running time. We can also observe from the table that the pruning process is more powerful for the benchmarks with fewer terminals. Take IND1–IND5 for example, over 50% reduction can be achieved and the running time is around 0.01 s. For RT1_40 and RT1_rand_40, we can reduce 98.47% and 99.72% of the FSTs with the pruning process. Since a large number of nets in VLSI circuits have a small number of terminals, the pruning process can thus be very effective.

Table II illustrates the results of the proposed algorithm. For the FST generation phase, the average running time is 9.2 s. The running time of the FST generation phase is comparable to the FST concatenation phase for some small instances. However, for large cases, the FST concatenation phase dominates the total running time. The average running time of the second phase is 157.40 s which account for over 90% of the total running time. In the table we also list the number of constraints and the number of branch-and-cut nodes in the branch-and-cut search of each test case. As there are an exponential number of constraints (with respect to the number of terminals) in the IP formulation, by handling them incrementally, only a small subset of constraints are required. In all test cases except RC10_rand_30, the optimal solution can be achieved within five branch-and-cut nodes. Around half of the test cases achieve the optimum at the root node without branching. This indicates that the lower bound provided by

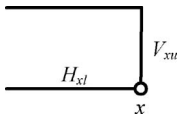


Fig. 13. Structure when V_{xu} is a corner line ended at a left-turn corner and H_{xl} exists.

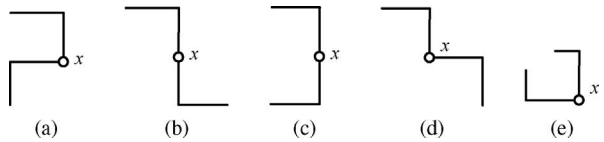


Fig. 14. Five possible structures when a Steiner point has two corner lines. (a) Case when H_{xl} exists and ends at a corner turning down. (b) Case when V_{xd} exists and ends at a corner turning right. (c) Case when V_{xd} exists and ends at a corner turning left. (d) Case when H_{xr} exists and ends at a corner turning down. (e) Case when V_{xl} exists and ends at a corner turning up.

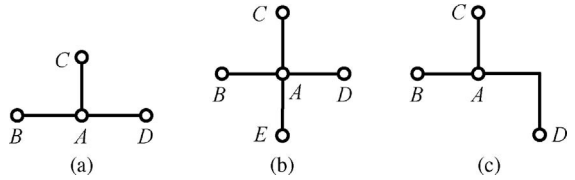


Fig. 15. Three possible structures when a Steiner point is adjacent to more than two other Steiner points. (a) First type of structures when a Steiner point is adjacent to more than two other Steiner points. (b) Second type of structures when a Steiner point is adjacent to more than two other Steiner points. (c) Third type of structures when a Steiner point is adjacent to more than two other Steiner points.

the LP relaxation is very tight and the separation procedure is very efficient in finding violated constraints. The resulting OARSMT length of each test case is tabulated in the table.

Table III shows the comparison of some recent heuristics [9], [10] based on the optimal solutions generated by us. The results are obtained by running the executables provided by the authors. 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.

VII. Conclusion

In this paper, we extended the well-known GeoSteiner method to solve the OARSMT problem optimally. In the proposed approach, OARSMTs are constructed by the concatenation of FSTs with blockages. Detailed implementation of the algorithm is presented. Experimental results showed that our approach can handle problems with hundreds of terminals in the presence of multiple obstacles, generating an optimal solution in a reasonable amount of time.

Appendix A
Proofs for Section III

Corollary: Suppose V_{Bu} contains a vertex, then V_{Au} is a corner line that ends at a corner turning away from V_{Bu} and $|V_{Au}| < |V_{Bu}|$.

Proof: By Lemma 2, if $|V_{Au}| \geq |V_{Bu}|$, V_{Bu} must be a corner line and can have no vertex on it. Therefore, $|V_{Au}| < |V_{Bu}|$. Again from Lemma 2, V_{Au} is a corner line that ends at a corner turning away from V_{Bu}. ■

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

Proof: See Fig. 13. If H_{xl} exists, we can shift the line V_{xu} to the left and reduce the total length. If the line cannot be shifted due to some obstacles, the tree will contain the structures as shown in Fig. 4, a violation of the FST definition. ■

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

Proof: Consider a Steiner point x with two corner lines. Without loss of generality, we assume V_{xu} exists and ends at a corner turning left. The second corner line can be V_{xd}, H_{xl}, or H_{xr} and ends at a corner turning to two different directions. The case when H_{xr} exists and ends at a corner turning up is equivalent to the case when H_{xl} exists and ends at a corner turning down, and thus can be removed. Therefore, there are totally five possible cases as shown in Fig. 14. Fig. 14(a) and (e) cannot exist according to Lemma 3. Fig. 14(b) is impossible because the third line at the Steiner point cannot exist by Lemma 3. Fig. 14(c) and (d) can be transformed into Fig. 14(a) and (b) by flipping a corner. If the corner cannot be flipped due to some obstacles, we can flip the corner to the boundary of the obstacle to obtain an equivalent tree with forbidden structures as shown in Fig. 4, a violation of the definition of FSTs. ■

Lemma 5: If s is a FST, the Steiner points in s forms a chain.

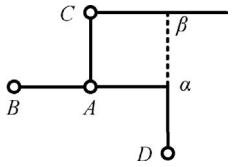


Fig. 16. Special structure of one Steiner point with more than two neighboring Steiner points.

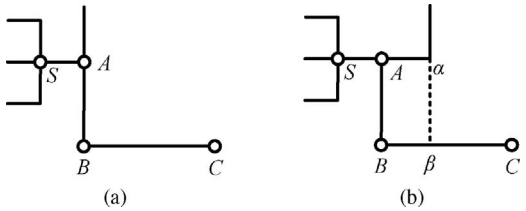


Fig. 17. (a) Topology when V_{Au} exists. (b) Topology when H_{Ar} exists.

Proof: First of all, if s is a FST, the Steiner points in s are connected, for otherwise some Steiner points have to be connected by terminals of degree two or more. Therefore, we only need to prove that no Steiner point in s is adjacent to more than two other Steiner points. Suppose the contrary, and let A be such a Steiner point. Then from Lemma 3 and Lemma 4, the connection between A and its adjacent Steiner points must be in one of the three forms as shown in Fig. 15.

First, consider Fig. 15(a) and (b). Suppose H_{Cl} exists. Then, from the corollary of Lemma 2, H_{Cl} must be a corner line that ends at a corner turning up. Similarly, if H_{Cr} exists, it is also a corner line ends at a corner turning up. Since C is a Steiner point, at least two of the three lines H_{Cl}, H_{Cr}, and V_{Cu} must exist. However, regardless of which two (or all three) exist, we end up a contradiction to either Lemma 3 or Lemma 4.

Next, consider Fig. 15(c). The argument on H_{Cl} is the same as that in Fig. 15(a) and (b). If H_{Cr} exists and $|H_{Cr}| \leq |H_{Ar}|$, the argument on H_{Cr} is again the same. Thus, we only need to discuss the case that $|H_{Cr}| > |H_{Ar}|$ (see Fig. 16).

Let α be the corner on the edge connecting A and D. Shift AC to α and let the new line meet H_{Cr} at β. Now, the tree contains a Steiner point α that is adjacent to three other Steiner points β, B, and D in the form of Fig. 15(a), which has already been shown to be impossible. If AC cannot be shifted due to some obstacles, we can shift AC to the boundary of the obstacle and achieve an equivalent tree with forbidden edges as shown in Fig. 4, a contradiction to the assumption that s is a FST. ■

Lemma 6: Suppose s is a FST. Then its Steiner chain cannot contain the subgraph shown in Fig. 6.

Proof: Suppose H_{Ar} exists. Then from the corollary of Lemma 2, H_{Ar} is a corner line. Since H_{Ar} and V_{Au} cannot both exist by Lemma 3, H_{Al} must exists for A is a Steiner point. If H_{Ar} exists, we can simply shift AB to $\alpha\beta$, as shown in Fig. 17(b), and obtain a similar structure as Fig. 17(a). Therefore, in the following, we can just consider the case when V_{Au} exists.

From Lemma 3, H_{Al} cannot be a corner line. Besides, H_{Al} cannot contain any Steiner points. If H_{Al} contains a Steiner point S, V_{Au} cannot contain any Steiner points by Lemma 5.

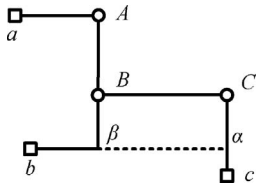


Fig. 18. Topology when V_{Bd} exists.

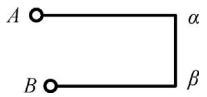


Fig. 19. Structure of the Steiner chain when it bends back.

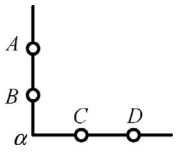


Fig. 20. Corner with more than one Steiner point on each line.

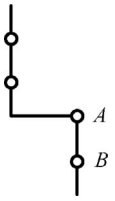


Fig. 21. Possible structure of the Steiner chain.

If V_{Au} is a corner line, it must be a corner turning right by Lemma 3. We can flip the corner and repeat the operation as shown in Fig. 17(b). Therefore, we can assume without loss of generality that V_{Au} is a node line. Since S is a Steiner point, two of the lines H_{Sl}, V_{Su}, and V_{Sd} must exist. By the corollary of Lemma 2, V_{Su} and V_{Sd} must be corner lines and the corners must turn away from AB. As a result, by Lemma 3 and Lemma 4, H_{Al} cannot contain any Steiner point. Moreover, H_{Al} cannot contain more than one node for the tree is a FST. Therefore, H_{Al} is a node line. By symmetry, V_{Cd} exists and is a node line. Since B is a Steiner point, at least one of the lines H_{Bl} or V_{Bd} exists. We first assume that V_{Bd} exists and V_{Bl} does not exist. Since V_{Cd} contains a vertex (see Fig. 18), by the corollary of Lemma 2, V_{Bd} must be a corner line that ends at a corner (denoted by β) turning left and connects to a node b by Lemma 5. But this is impossible, for otherwise we can shift BC to $\beta\alpha$ to obtain a tree in which both H_{Al} and $H_{\beta l}$ are node lines, a contradiction to Lemma 2. If the line cannot be shifted due to some obstacles, the tree or its equivalent will contain the structures as shown in Fig. 4, an absurdity. Similarly, H_{Bl} cannot exist. As a result, B cannot be a Steiner point which is contradictory to the assumption. ■

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

Proof: Suppose that the Steiner chain bends back as shown in Fig. 19, where A and B are Steiner points that are closest to the turning points α and β. There must be at least two Steiner points on $\alpha\beta$, for otherwise we can shift $\alpha\beta$ to the left and reduce the total length. If the line cannot be shifted due to some obstacles, the tree will contain the structures as shown in Fig. 4. From Lemma 6, neither α nor β can be a Steiner point. From Lemma 3 and Lemma 5, the horizontal line of any Steiner point on $\alpha\beta$ must be a node line and the first one below $A\alpha$ must be a line going right. From the corollary of Lemma 2, the adjacent Steiner points on $\alpha\beta$ cannot have horizontal lines going in the same direction. Therefore, $\alpha\beta$ must have more left lines (including $A\alpha$ and $B\beta$) than right lines, which implies that we can shift $\alpha\beta$ to the left and reduce the total length, an absurdity. ■

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

Proof: Suppose to the contrary that s contains the subgraph shown in Fig. 20.

From Lemma 3, V_{Cu} does not exist. Thus, V_{Cd} exists and is a node line by Lemma 3 and Lemma 5. Suppose V_{Dd} exists. Then from the corollary of Lemma 2, V_{Dd} is a corner line. As a result, H_{Dr} does not exist by Lemma 3. Therefore, V_{Dd} and H_{Dr} cannot both exist, and hence V_{Du} must exist and is a node line by Lemma 3 and Lemma 6. Flip the corner α between B and C to form a new corner β. If the corner cannot be flipped due to obstacles, we can flip it to the boundary of the obstacle to obtain a tree with edges as shown in Fig. 4. If $|V_{Du}| \leq |C\beta|$, we can shift DC to the node on V_{Du} and obtain an tree in which a node has degree 2. If the line cannot be shifted, the tree will contain the structures as shown in Fig. 4. If $|V_{Du}| > |C\beta|$, we can shift CD to β making β a Steiner point. But the induced subgraph between $AB\beta$ cannot exist by Lemma 6. Again, the line can be shifted, or else it cannot be a part of a FST. ■

Lemma 9: Suppose s is a FST. 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 on the Steiner chain contains no Steiner point except at the end point, or vice versa.

Proof: By Lemma 4, a Steiner point cannot have two corner lines. Hence, at least two of the first three Steiner points (counting from either end) are collinear. Without loss of generality, suppose the first collinearity occurs on a vertical line. Let A be the first Steiner point (if any) not on the vertical line. Then A is connected to its preceding Steiner point through a corner as shown in Fig. 21 by Lemma 6. Let B be the Steiner point (if any) succeeding A. Then A and B must be on the same vertical line, for otherwise either Lemma 8 is contradicted (if A and B are on the same horizontal line), or Lemma 4 is contradicted for A has two corner lines (if A and B are connected through a corner). If there are more Steiner points after B, we repeat the above argument to prove Lemma 9. ■

Lemma 10: Suppose s is a FST. Every Steiner point on s must have a horizontal node line and the node lines alternate in the left-right direction.

Proof: Note that a horizontal line not on the Steiner chain cannot contain any Steiner points, nor can it contain more than

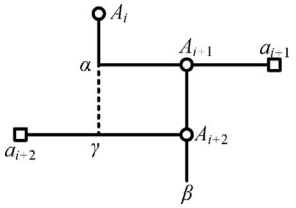


Fig. 22. Graph when A_i and A_{i+1} are connected by corner.

one node since s is a FST. Therefore, it suffices to show that there exists a horizontal line and it cannot be a corner line.

1) If the Steiner point is connected to its preceding Steiner point through a corner (A in Fig. 21), the third line of the Steiner point cannot be a vertical line according to Lemma 3. Therefore, the third line must be horizontal and it cannot be a corner line by Lemma 4.
2) If the Steiner point is on the vertical line of the Steiner chain (B in Fig. 21), the third line must be horizontal and it cannot be a corner line by Lemma 3.

By the corollary of Lemma 2, two adjacent Steiner points on the same line cannot have node lines on the same side. For the Steiner points connected through corner, it is also easy to prove this (Lemma 3). Hence, if s is a FST, the node line on the Steiner chain must alternate in the left-right direction. ■

Lemma 11: Suppose s is a FST. Then a corner connecting A_i and A_{i+1} can be transferred to one connecting A_{i-2} and A_{i-1}, or one connecting A_{i+2} and A_{i+3}, regardless of whether the place it transfers to has a corner or not.

Proof: From Lemma 9 and Lemma 10, when A_i and A_{i+1} are connected by a corner, the graph must be the one given in Fig. 22. Here a_i denotes a node and β is either a Steiner point, a node, or a corner.

Necessarily $|a_{i+2}A_{i+2}| > |\alpha A_{i+1}|$, since otherwise we can shift $A_{i+1}A_{i+2}$ to a_{i+2} to obtain a tree in which the node a_{i+2} has degree more than one. Now, shift $A_{i+1}A_{i+2}$ to α and suppose this line meets $a_{i+2}A_{i+2}$ at γ. If the line cannot be shifted, the tree will contain the structures as shown in Fig. 4. Now, α and γ replace A_{i+1} and A_{i+2} as Steiner points. We can flip the corner A_{i+2} between γ and β. Then the corner connecting A_i and A_{i+1} is transferred to one connecting A_{i+2} and A_{i+3}. If the corner A_{i+2} cannot be flipped due to some obstacles, the tree or its equivalent will contain forbidden edges as shown in Fig. 4. Similarly, we can use the same method to transfer the corner connecting A_i and A_{i+1} to one connecting A_{i-2} and A_{i-1}. Note that if A_{i+3} does not exist, i.e., β is either a node or a corner turning right to connect a node, the above operation eliminates the corner between A_i and A_{i+1}. ■

Theorem 1: Suppose s is a FST and let m be the number of Steiner points. There exists a s' equivalent to s such as follows:

1) if m is odd, the Steiner chain of s' is a straight line;
2) if m is even, all the Steiner points are on a straight line except possibly the last one.

Proof: By pushing the corners along the direction according to Lemma 11, there will be at most one corner connecting the last two Steiner points. If m is odd, the corner can be eliminated. ■

Appendix B
Proof for Lemma 13

Lemma 13: If a terminal u is uncongested and $f(X \cup \{u\}) < 1$, then $f(X) \le f(X \cup \{u\}) < 1$.

Proof: Let

$$A = \{s_i \in S : |s_i \cap X| \ge 1 \text{ and } u \in s_i\} \text{ and}$$
$$B = \{s_i \in S : |s_i \cap X| \ge 1 \text{ and } u \notin s_i\}.$$

Then

$$\begin{aligned} f(X \cup \{u\}) - f(X) = {} & |(X \cup \{u\}) \cap T| \\ & + \sum_{i:v_i \in X \cup \{u\}} y_i - \sum_{i:s_i \in A} |s_i \cap X| x_i \\ & - \sum_{i:s_i \in B} (|s_i \cap X| - 1) x_i \\ & - |X \cap T| - \sum_{i:v_i \in X} y_i + \sum_{i:s_i \in A} (|s_i \cap X| - 1) x_i \\ & + \sum_{i:s_i \in B} (|s_i \cap X| - 1) x_i. \end{aligned}$$

If u is a real terminal, then

$$\begin{aligned} f(X \cup \{u\}) - f(X) = {} & |X \cap T| + 1 - \sum_{i:s_i \in A} |s_i \cap X| x_i \\ & - |X \cap T| + \sum_{i:s_i \in A} |s_i \cap X| x_i - \sum_{i:s_i \in A} x_i \\ = {} & 1 - \sum_{i:s_i \in A} x_i \ge 1 - b_u \ge 0. \end{aligned}$$

If u is a virtual terminal and its index is k, then

$$\begin{aligned} f(X \cup \{u\}) - f(X) = {} & |X \cap T| + y_k \\ & - \sum_{i:s_i \in A} |s_i \cap X| x_i - |X \cap T| \\ & + \sum_{i:s_i \in A} |s_i \cap X| x_i - \sum_{i:s_i \in A} x_i \\ = {} & y_k - \sum_{i:s_i \in A} x_i \ge y_k - b_u \ge 0. \end{aligned}$$

In conclusion, we have $f(X) \le f(X \cup \{u\}) < 1$. ■

Appendix C
Proof for Theorem 2

Theorem 2: Let $(W : N - W)$ be a source to terminal cut of G that minimize $c(W)$. Let $X_m = \{u : u \in T + V \wedge u \in N - W\}$. Then X_m minimizes $f(X)$.

Proof: Let $W = \{\sigma\} \cup I \cup J$ be such a minimum cut of G, where $I \subseteq Y$ and $J \subseteq Z$. According to [21], I is completely determined by J.

Let $w_{t_j} = 1$ if $t_j \in W$ and $w_{t_j} = 0$ otherwise. Let $w_{v_j} = 1$ if $v_j \in W$ and $w_{v_j} = 0$ otherwise. Then $c(W)$ can be written as

$$\begin{aligned} c(W) = {} & \sum_{i:s_i \in S} \left(1 - \prod_{j:t_j \in s_i} w_{t_j} \prod_{j:v_j \in V} w_{v_j}\right) x_i + \sum_{j:t_j \in T} (b_{t_j} - 1) w_{t_j} \\ & + \sum_{j:v_j \in V} (b_{v_j} - y_j) w_{v_j} \end{aligned}$$

$$= \sum_{i:s_i \in S} \left(-x_i \prod_{j:t_j \in s_i} w_{t_j} \prod_{j:v_j \in s_i} w_{v_j} \right) + \sum_{j:t_j \in T} (b_{t_j} - 1) w_{t_j} + \sum_{j:v_j \in V} (b_{v_j} - y_j) w_{v_j} + \sum_{i:s_i \in S} x_i.$$

Note that the last term in the equation does not depend on w_{t_j} or w_{v_j}, and, therefore, is a constant. Now consider the function $f(X)$. Let $z_{t_j} = 1$ if $t_j \in X$ and $z_{t_j} = 0$ otherwise. Let $z_{v_j} = 1$ if $v_j \in X$ and $z_{v_j} = 0$ otherwise. Let $\bar{z}_{t_j} = 1 - z_{t_j}$ and $\bar{z}_{v_j} = 1 - z_{v_j}$ be the complementary variables. We can rewrite $f(X)$ as

$$\begin{aligned}
f(X) &= |X \cap T| + \sum_{i:v_i \in X} y_i - \sum_{i:s_i \cap X \neq \emptyset} x_i (|s_i \cap X| - 1) \\
&= \sum_{j:t_j \in T} z_{t_j} + \sum_{j:v_j \in V} z_{v_j} y_j \\
&\quad - \sum_{i:s_i \in S} \left[\left(\sum_{j:t_j \in s_i} z_{t_j} + \sum_{j:v_j \in s_i} z_{v_j} \right) - 1 \right. \\
&\qquad \left. + \prod_{j:t_j \in s_i} (1 - z_{t_j}) \prod_{j:v_j \in s_i} (1 - z_{v_j}) \right] x_i \\
&= \sum_{j:t_j \in T} (1 - \bar{z}_{t_j}) + \sum_{j:v_j \in V} (1 - \bar{z}_{v_j}) y_j \\
&\quad - \sum_{i:s_i \in S} \left[\left(\sum_{j:t_j \in s_i} (1 - \bar{z}_{t_j}) + \sum_{j:v_j \in s_i} (1 - \bar{z}_{v_j}) \right) \right. \\
&\qquad \left. - 1 + \prod_{j:t_j \in s_i} \bar{z}_{t_j} \prod_{j:v_j \in s_i} \bar{z}_{v_j} \right] x_i \\
&= |T| - \sum_{j:t_j \in T} \bar{z}_{t_j} + \sum_{j:v_j \in V} y_j - \sum_{j:v_j \in V} \bar{z}_{v_j} y_j \\
&\quad - \sum_{i:s_i \in S} \left(|s_i| - \sum_{j:t_j \in s_i} \bar{z}_{t_j} - \sum_{j:v_j \in s_i} \bar{z}_{v_j} \right. \\
&\qquad \left. - 1 + \prod_{j:t_j \in s_i} \bar{z}_{t_j} \prod_{j:v_j \in s_i} \bar{z}_{v_j} \right) x_i \\
&= |T| - \sum_{j:t_j \in T} \bar{z}_{t_j} + \sum_{j:v_j \in V} y_j - \sum_{j:v_j \in V} \bar{z}_{v_j} y_j - \sum_{i:s_i \in S} (|s_i| - 1) x_i \\
&\quad + \sum_{i:s_i \in S} \left(x_i \sum_{j:t_j \in s_i} \bar{z}_{t_j} \right) + \sum_{i:s_i \in S} \left(x_i \sum_{j:v_j \in s_i} \bar{z}_{v_j} \right) \\
&\quad - \sum_{i:s_i \in S} \left(x_i \prod_{j:t_j \in s_i} \bar{z}_{t_j} \prod_{j:v_j \in s_i} \bar{z}_{v_j} \right) \\
&= |T| - \sum_{j:t_j \in T} \bar{z}_{t_j} + \sum_{j:v_j \in V} y_j - \sum_{j:v_j \in V} \bar{z}_{v_j} y_j - \sum_{i:s_i \in S} (|s_i| - 1) x_i \\
&\quad + \sum_{j:t_j \in T} \bar{z}_{t_j} b_{t_j} + \sum_{j:v_j \in V} \bar{z}_{v_j} b_{v_j} - \sum_{i:s_i \in S} \left(x_i \prod_{j:t_j \in s_i} \bar{z}_{t_j} \prod_{j:v_j \in s_i} \bar{z}_{v_j} \right) \\
&= \sum_{i:s_i \in S} \left(-x_i \prod_{j:t_j \in s_i} \bar{z}_{t_j} \prod_{j:v_j \in s_i} \bar{z}_{v_j} \right) + \sum_{j:t_j \in T} \bar{z}_{t_j} \left(b_{t_j} - 1 \right) \\
&\quad + \sum_{j:v_j \in V} \bar{z}_{v_j} \left(b_{v_j} - y_j \right) - \sum_{i:s_i \in S} (|s_i| - 1) x_i + \sum_{j:v_j \in V} y_j + |T|.
\end{aligned}$$

The last three terms do not depend on $\bar{z}_{t_j}$ or $\bar{z}_{v_j}$, and, therefore, are constants. By setting $\bar{z}_{t_j} = w_{t_j}$ and $\bar{z}_{v_j} = w_{v_j}$, we can see that $c(W)$ and $f(X)$ differ only by a constant. Therefore, minimizing $c(W)$ is equivalent to minimizing $f(X)$. Let $(W : N - W)$ be a source to terminal cut of G such that $c(W)$ is minimized, then $X_m = \{u : u \in T + V \wedge u \in N - W\}$ is a minimum of $f(X)$. ■

REFERENCES

[1] M. Garey and D. Johnson, "The rectilinear Steiner tree problem is NP-complete," *SIAM J. Appl. Math.*, 1977, pp. 826–834.

[2] Y. Yang, Q. Zhu, T. Jing, X. Hong, and Y. Wang, "Rectilinear Steiner minimal tree among obstacles," in *Proc. IEEE ASICCON*, Oct. 2003, pp. 348–351.

[3] Y. Hu, T. Jing, X. Hong, Z. Feng, X. Hu, and G. Yan, "An-OARSMan: Obstacle-avoiding routing tree construction with good length performance," in *Proc. ASP-DAC*, 2005, pp. 7–12.

[4] R. Hentschke, J. Narasimham, M. Johann, and R. Reis, "Maze routing Steiner trees with effective critical sink optimization," in *Proc. Int. Symp. Phys. Des.*, 2007, pp. 135–142.

[5] L. Li and E. F. Y. Young, "Obstacle-avoiding rectilinear Steiner tree construction," in *Proc. Int. Conf. Comput.-Aided Des.*, 2008, pp. 523–528.

[6] Y. Hu, 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," *J. Inform. Computat. Sci.*, 2004, pp. 107–116.

[7] Z. Feng, Y. Hu, T. Jing, X. Hong, X. Hu, and 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.*, 2006, pp. 48–55.

[8] Z. Shen, C. Chu, and Y. Li, "Efficient rectilinear Steiner tree construction with rectilinear blockages," in *Proc. ICCD*, 2005, pp. 38–44.

[9] 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.*, 2007, pp. 380–385.

[10] J. Y. Long, H. Zhou, and S. O. Memik, "EBOARST: An efficient edge-based obstacle-avoiding rectilinear Steiner tree construction algorithm," *IEEE Trans. Comput.-Aided Des.*, vol. 27, no. 12, pp. 2169–2182, Dec. 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.*, 2010, pp. 27–34.

[12] J. L. Ganley and J. P. Cohoon, "Routing a multi-terminal critical net: Steiner tree construction in the presence of obstacles," in *Proc. IEEE ISCAS*, May–Jun. 1994, pp. 113–116.

[13] F. K. Hwang, D. S. Richards, and P. Winter, *The Steiner Tree Problem*. Amsterdam, The Netherlands: North-Holland, 1992.

[14] Y. Y. Yang and O. Wing, "Optimal and suboptimal solution algorithms for the wiring problem," in *Proc. IEEE Int. Symp. Circuit Theory*, Apr. 1972, pp. 154–158.

[15] 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.

[16] D. M. Warme, P. Winter, and M. Zachariasen, "Exact algorithms for plane Steiner tree problems: A computational study," in *Advances in Steiner Trees*, D. Z. Du, J. M. Smith, and J. H. Rubinstein, Eds. Boston, MA: Kluwer, 2000, pp. 81–116.

[17] P. Winter, "An algorithm for the Steiner problem in the Euclidean plane," *Networks*, vol. 15, no. 3, pp. 323–345, 1985.

[18] F. K. Hwang, "On Steiner minimal trees with rectilinear distance," in *Proc. SIAM J. Appl. Math.*, vol. 30, no. 1, pp. 104–114, 1976.

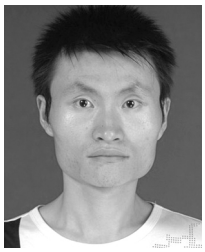
[19] M. Zachariasen, "Rectilinear full Steiner tree generation," *Networks*, vol. 33, no. 2, pp. 125–143, 1999.

[20] U. Fößmeier and M. Kaufmann, "On exact solutions for the rectilinear Steiner tree problem part I: Theoretical results," *Algorithmica*, vol. 26, no. 1, pp. 68–99, 2000.

[21] D. M. Warme, "Spanning trees in hypergraphs with applications to Steiner trees," Ph.D. thesis, Dept. Comput. Sci., Univ. Virginia, Charlottesville, 1998.
[22] M. Hanan, "On Steiner's problem with rectilinear distance," in *Proc. SIAM J. Appl. Math.*, vol. 14. 1966, pp. 255–265.
[23] A. C. C. Yao, "On constructing minimum spanning trees in k-dimensional spaces and related problems," *SIAM J. Comput.*, vol. 11, no. 4, pp. 721–736, 1982.
[24] *GeoSteiner: Software for Computing Steiner Trees* [Online]. Available: http://www.diku.dk/geosteiner

Tao Huang received the B.E. and M.E. degrees in electronic engineering from Sun Yat-sen University, Guangzhou, China, in 2007 and 2009, respectively. He is currently pursuing the Ph.D. degree from the Department of Computer Science and Engineering, Chinese University of Hong Kong, Shatin, Hong Kong.

His current research interests include computer-aided design of very large-scale integrated circuits, physical design, interconnect optimization, and combinatorial optimization.



Liang Li received the Bachelors degree from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, and the Masters degree from the Chinese University of Hong Kong, Shatin, Hong Kong.

He is currently with Oracle Research and Development Center Company, Ltd., Shenzhen, China. His current research interests include very large-scale integrated circuits computer-aided design, algorithms, and combinatorial optimization. He focuses on routing problems, especially the problem in the presence of obstacles.

Evangeline F. Y. Young received the B.S. and M.Phil. degrees in computer science from the Chinese University of Hong Kong (CUHK), Shatin, Hong Kong, and the Ph.D. degree from the University of Texas at Austin, Austin, in 1999.

Currently, she is an Associate Professor with the Department of Computer Science and Engineering in CUHK. Her current research interests include algorithms and computer-aided design of very large-scale integrated circuits. She is now working actively on floorplanning, placement, routing, and algorithmic designs.

Dr. Young has served on the technical program committees of several major conferences including ICCAD, ASP-DAC, ISPD, and GLSVLSI, and has also served on the Editorial Board of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS.