Open access • Proceedings Article • DOI:10.1109/ICCD.2005.45

Efficient rectilinear Steiner tree construction with rectilinear blockages — Source link

Z. Shen, Chris Chu, Ying-Meng Li

Institutions: Cadence Design Systems

Published on: 02 Oct 2005 - International Conference on Computer Design

Topics: Rectilinear Steiner tree, Steiner tree problem, Minimum spanning tree, Spanning tree and k-minimum spanning tree

Efficient Rectilinear Steiner Tree Construction with Rectilinear Blockages

Zion Shen — Cadence Design Systems, 555 River Oaks Parkway, San Jose, CA, 95134, zion@cadence.com

Chris C.N. Chu — Department of ECE, Iowa State University, Ames, Iowa, 50011, cnchu@iastate.edu

Ying-Meng Li — Atoptech, Inc., 2700 Augustine Drive, Santa Clara, CA 95054

Abstract

Given n points on a plane, a Rectilinear Steiner Minimal Tree ($RSMT$) connects these points through some extra points called steiner points to achieve a tree with minimal total wire length. Taking blockages into account dramatically increases the problem complexity. It is extremely unlikely that an efficient optimal algorithm exists for Rectilinear Steiner Minimal Tree Construction with Rectilinear Blockages ($RSMTRB$). Although there exist some heuristic algorithms for this problem, they have either poor quality or expensive running time.

In this paper, we propose an efficient and effective approach to solve $RSMTRB$. The connection graph we used in this approach is called spanning graph which only contains $O(n)$ edges and vertices. An $O(n \log n)$ time algorithm is proposed to construct spanning graph for $RSMTRB$. The experimental results show that this approach can achieve a solution with significantly reduced wire length. The total run time increased is negligible in the whole design flow.

1. INTRODUCTION

Given n points on a plane, a Rectilinear Steiner Minimal Tree ($RSMT$) connects these points through some extra points called steiner points to achieve a tree with minimal total wire length. Many works [1–12, 14–17] have been done on this fundamental problem in electronic design automation. However, most of them did not take blockages into consideration. In fact, today's design often contains many rectilinear routing blockages, e.g., macro cells, IP blocks, and pre-routed nets. Thus, rectilinear Steiner minimal tree construction with rectilinear blockages ($RSMTRB$) becomes a very practical problem.

Generally, $RSMT$ is used in initial net topology creation for global routing or incremental net tree topology creation in physical synthesis. It is also utilized to estimate total wire length, congestion and timing in early design stages, like block floorplanning and cell placement. The timing and congestion information obtained from $RSMT$ can be used as a criteria in following timing and congestion driven routing. Note that it is the problem applied hundreds of thousands times in each design flow and many times this problem comes with very large input size. $RSMT$ thus deserves much intensive research in VLSI CAD.

Unfortunately, $RSMT$ itself was first shown to be strongly NP-complete by Garey [7] in 1977. Taking blockages into account dramatically increases the problem complexity. Thus, it is extremely unlikely that an efficient optimal algorithm exists for $RSMTRB$. Although there exist some heuristic algorithms for this problem, they have either poor quality or expensive run time.

Existing heuristics for $RSMTRB$ can be classified into three categories.

The first category is maze routing [8] based approach. Maze routing can optimally route two pin nets. However, for multi-pin nets, designers need to introduce a multi-terminal variant [9–11], which incurs a solution far from optimal. In addition, since the run time and memory used in maze routing are proportional to the size of the routing area rather than the size of actual problem (i.e., the number of pins and blockages), maze routing algorithms are inefficient in terms of run time and memory.

The second category is called sequential approach which typically consists of two steps. These two steps are illustrated in Figure 1. Step 1 is to construct a tree T_1, which is either a minimum spanning tree (MST) or a Steiner minimal tree (SMT) with absence of blockages. In this example, T_1 is an MST. Step 2 is to transform T_1 to a $RSMT$ with blockages by substituting edges around the blockages for the edges overlapped by the blockages. Generally a simple line sweep technique is applied in step 2. For example, for pin pair (p_1, p_4), we have two possible L-shaped routes, r_1 and r_2 to interconnect p_1 and p_4 as illustrated in Figure $2(a)$ with absence of blockages. Then we sweep all blockages which overlap with these two routes and choose one route with smaller corresponding detour. As shown in Figure $2(b)$, route r_2 are selected since the detour length introduced by r_2 is smaller than that by r_1. This approach is commonly used in industry due to its simplicity and efficiency. However, since step 1 neglects the global view of blockages, step 2 can only locally remove overlap between T_1 and blockages. The quality(i.e., the total wire length) of resulting $RSMT$ can be much worse than expected in many cases.

Later on, Yang et.al [12] introduced a complicated 4-process heuristics to remove the overlaps in step 2 in a clever way. However the approach still cannot avoid bad solution for many cases. This is because a bad T_1 due to neglecting blockages in step 1 could introduce an unexpected routing detour in step 2.

The third category consists of connection graph [13] based approaches. Typically, the approach in this category is to first construct a connection graph by pins and blockage boundaries, which guarantees that at least one rectilinear Steiner minimal (or close to minimal) tree is embedded in the graph. Then, some graph searching tech-

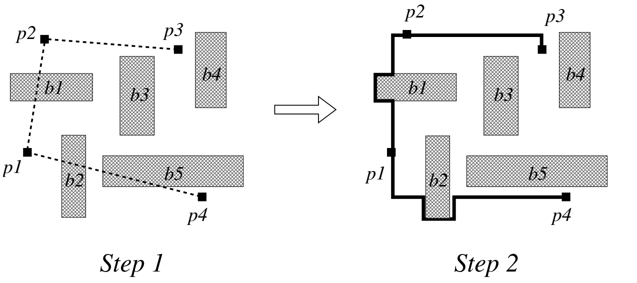


Figure 1: Sequential approach to solve $RSMTRB$.

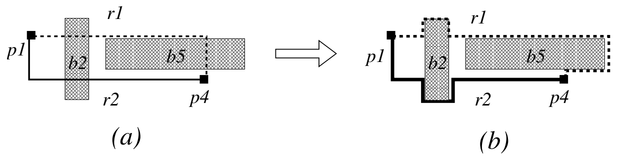


Figure 2: Overlap removal in sequential approach. (a) two L-shaped routes, r_1 and r_2, before overlap removal. (b) route r_1 and r_2 after overlap removal.

nique is used to find a $RSMT$ as a subgraph from the connection graph. Unlike sequential approach, this approach can globally catch the view of both pins and blockages in the design. Therefore, the connection graph based approach can generally achieve an optimal or near optimal $RSMT$.

The efficiency of connection graph based approach depends on the size of graph. While the accuracy or effectiveness of this approach depends on whether the graph contains a good Steiner minimal tree. Obviously, there is a trade-off between efficiency and accuracy in this approach.

In [14], a connection graph is called escape graph which is constructed by escape segments. The number of vertex in the escape graph can be $O(n^2)$ in the worst case, where n is the sum of pins and blockage boundaries. Even the size of a reduced escape graph is still quite huge. In [15], authors proposed a connection graph with $O(n \log n)$ vertices and edges and later on [16], they introduced an even smaller graph which contains $O(n\sqrt{\log n})$ vertices and $O(n \log^{3/2} n)$ edges. The construction of the connection graph takes $O(n \log^{3/2} n)$ time and memory usage.

In this paper, we will propose an efficient and effective connection graph. It is called spanning graph. We show that the spanning graph contains only $O(n)$ vertices and $O(n)$ edges, which is smaller than any previous connection graph. In addition, we show that our spanning graph can always produce a $RSMT$ with good quality. Due to the special property of the spanning graph, the construction takes only $O(n \log n)$ time and memory usage, which are also smaller than those in any previous connection graph construction.

We organize the rest of the paper as follows. In section 2, we will formally define the problem and explain the details of the basic component in the problem. In section 3, we will first demonstrate the drawback in the escape graph and then introduce the spanning graph as a connection graph in $RSMTRB$. Section 4 and 5 describe the details of construction of spanning graph for $RSMTRB$. The experimental results are shown in section 6. The paper is concluded in section 7.

2. PROBLEM FORMULATION

Let $P = \{p_1, p_2, p_3, \ldots, p_m\}$ be a set of pins for m pin net. Let $B = \{b_1, b_2, b_3, \ldots, b_k\}$ be a set of rectangular blockages. Let $V = \{v_1, v_2, v_3, \ldots, v_n\} = P \cup$ {corners in B} as the vertex set in the problem, where each v_i has coordinates (x_i, y_i). Note that each rectangular blockage has 4 corners, we have $n \leq m + 4k$. The rectilinear distance between v_i and v_j is given as $|x_i - x_j| + |y_i - y_j|$. A $RSMT$ connects all pins through some extra points (called Steiner points) to achieve a minimal total length, while avoiding the intersection with any blockage in the design.

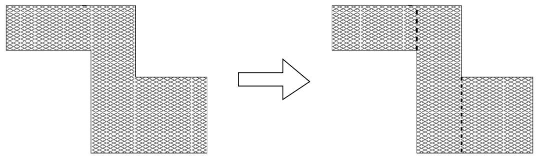


Figure 3: Dissect a rectilinear blockage into 3 rectangular blockages.

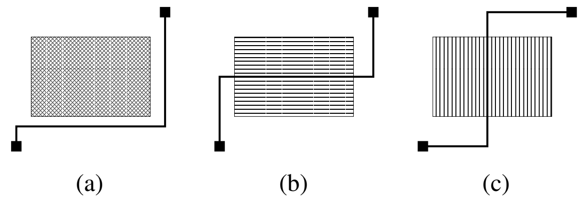


Figure 4: Three types of blockages: (a) Complete blockage (b) Vertical blockage (c) Horizontal blockage

.

2.1 Rectilinear blockages

If all boundaries of a blockage are either horizontal or vertical, we call this blockage as rectilinear blockage. Note that each rectilinear blockage can be dissected into a set of rectangular blockages (see Figure 3 as an example). In the rest of paper, for simplicity, we assume each blockage to be rectangular.

2.2 Directional blockages

In multi-layer routing, there exist three types of blockages. The first one is called complete blockage which blocks all vertical and horizontal metal layers in its obstructed area. A complete blockage requires all routes must detour around it. The second type is denoted as vertical blockage in which all vertical layers in obstructed area are blocked while a certain number of horizontal layers are still available for routing. The routes are allowed to horizontally pass through the obstructed area, while not allowed in vertical direction. The third type is called horizontal blockage, where routes can still vertically pass through the obstructed area. These three types of directional blockages are illustrated in Figure 4.

3. ESCAPE GRAPH VS. SPANNING GRAPH

3.1 Redundancy in escape graph

As we mentioned in Section 1, a connection graph can catch the global view of both blockages and pins. And the efficiency of this approach highly depends on the size of the connection graph. In other words, a good connection graph is able to describe all necessary geometrical relationship between pins and blockages using as few edges as possible. In [14], the connection graph is called escape graph, which is constructed by escape segments. Escape segments are formed by horizontal and vertical lines extending from pins and blockage boundaries, and ending with their abutment to either a blockage boundary or the internal perimeter of the routing region. The number of vertex in the escape graph can be $O(n^2)$ in the worst case. An example is shown in Figure 5. The collection of the escape segments (shown as dashed segments) composes the escape graph. And the graph preserves a good $RSMT$ for a multi-pin net. However, we notice that most of edges and vertices are redundant in the escape graph for finding a $RSMT$. For example, instead of using

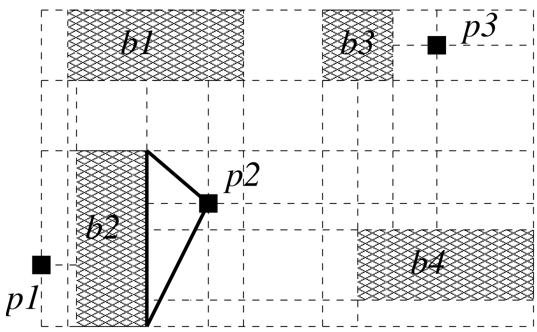


Figure 5: Escape graph

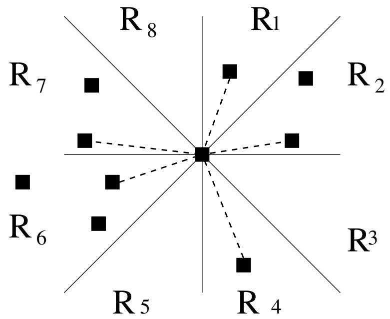


Figure 6: Eight regions defined for each point in spanning graph.

13 edges, 3 edges (shown as solid segments) are enough to represent the connection relationship between the blockage b_2 and pin p_2 in the corresponding connection graph.

3.2 Spanning graph

In [17], a spanning graph is introduced as an intermediate step in minimal spanning tree construction. Given a set of points on the plane, a spanning graph is an undirected graph over the points that contains at least one minimal spanning tree. The number of edges in the graph is called the cardinality of the graph.

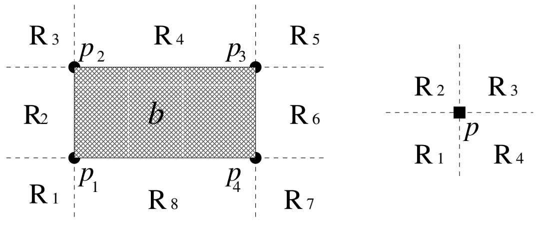
The construction of spanning graph is illustrated in Figure 6. From each point p, a plane can be divided into 8 regions by horizontal, vertical and $\pm 45^o$ lines through p. It can be proved that the rectilinear distance between any two points in one region is always smaller than the maximal distance from them to p. Due to the cycle property of a minimal spanning tree, that is, the longest edge on any cycle should not be included in any minimal spanning tree, which means only the closest point to p in each region needs to be connected to p. Considering all given points, the connections will form a spanning graph of cardinality $O(n)$. In other words, spanning graph is able to describe the relative geometrical relationship between points in the plane using $O(n)$ edges.

Enlightened by the spanning graph in MST construction, we borrow and revise this idea to solve $RSMTRB$. The details are presented in following section.

4. SPANNING GRAPH BASED APPROACH IN $RSMTRB$

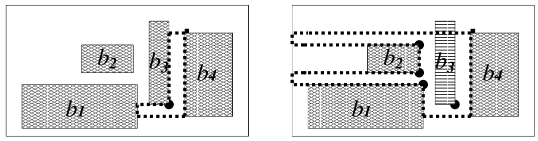
4.1 Search regions

In general spanning graph, each point p corresponds to 8 regions which are divided by the horizontal, vertical and $\pm 45^o$ lines going through p. Then we search each region and find the closest point in each region and connect it to p. While in $RSMTRB$, for each blockage b_i, we divide the whole plane into 8 regions as shown in Figure $7(a)$. Each corner of blockage b_i has 3 neighboring search



(a) (b)

Figure 7: Search regions for blockage and pin



(a) (b)

Figure 8: Visible points in search region for different blockages

regions which are adjacent to this corner. We call these three regions as neighboring search regions for a given corner point. For example, the neighboring search regions for the lowerleft corner p_1 in Figure $7(a)$ are R_8, R_1 and R_2. For each pin p, we divide the whole plane into 4 search regions. The corresponding neighboring search regions for p are R_1, R_2, R_3 and R_4 as shown in Figure $7(b)$.

For each $v \in V$, we connect the closest visible point in v's each neighboring search region to v. Note that the visible point in each search region could be different for complete blockage and directional blockage. For example, between Figure $8(a)$ and Figure $8(b)$, the only difference is that the blockage b_3 is a complete blockage in Figure $8(a)$ while a vertical blockage in Figure $8(b)$. The search region R_2 for blockage b_4 is denoted as the area enclosed by the dashed segments. Obviously, the search region in Figure $8(b)$ is larger than that in Figure $8(a)$. In Figure $8(a)$, there exists only 1 visible point in region R_2 of b_4. While in Figure $8(b)$, there are 4 visible points in R_2 of b_4.

A few natural questions may be raised as follows. First, why do we not apply $\pm 45^o$ lines in defining $RSMTRB$ search region? If $\pm 45^o$ lines are applied, each corner and pin has 6 and 8 search regions, respectively. We find that in practice, the total number of edges for spanning graph by our method is actually very close to the one by the method where $\pm 45^o$ lines are applied. However, ignoring these $\pm 45^o$ lines can greatly simplify the construction of spanning graph.

Second, why do we only consider 3 neighboring search regions for each corner p of blockage b? This is because for any visible point q in the rest 5 search regions of b, we can always find another corner p' of b such that q lies in one search region of p' and a shortest path from p to q can always be obtained by making p' as an intermediate point in the path.

In addition, for each point p, why do we only consider the connection with its visible points? The reason is that between the visible region and invisible region, there must exist at least one intermedi-

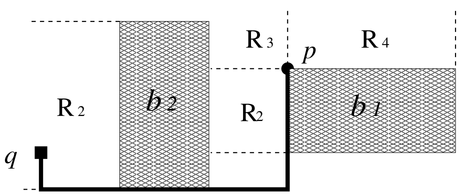


Figure 9: Example of invisible region for point p

ate blockage such that the point p can always reach the points in invisible region by making use of the boundaries of the intermediate blockage(s) as part of the shortest path. As an example in Figure 9, region R_2 of blockage b_2 is invisible to the upperleft corner p of b_1. However, any point in R_2 of b_2 can be reached from p by using boundaries of b_2 as part of the shortest path.

4.2 Spanning graph construction in $RSMTRB$

First, from blockage set B, we find out a subset B_s which includes each blockage lying within or intersected by the bounding box of net. Let V_s be a point set which includes all corners of B_s and pins of a given net. Our initial spanning graph G has V_s as its vertices and all blockage boundary segments of B_s as its edges. Then, we incrementally build the graph by applying a sweep line based edge connection among vertices in V_s. For any point $v \in V_s$, we only connect at most one visible point v' in each neighboring search region to v, where $v' \in V_s$ and v' is the closest point to v in the corresponding neighboring search region.

Based on an $O(n \log n)$ sweep line algorithm proposed in [17], we propose a revised sweep line algorithm to construct spanning graph in $RSMTRB$. Our construction algorithm consists of four passes. Each pass performs edge connection for a pair of search regions of V_s. For sake of simplifying the exposition, we only present the detail procedures of Pass one which performs edge connection for search region R_2 and R_6 of all corner points.

First, all points in V_s are sorted by their x coordinates in non-decreasing order. Note that for each blockage, lowerleft corner shares the same x coordinate as upperleft corner and lowerright corner shares the same x coordinate as upperright corner. We only need to sort left boundary and right boundary segment of each blockage instead of point by point to speed up sorting process. Note that a fundamental operation of sweep line algorithm is to keep an active set A of v such that all points in A are visible to v. We thus build an active set A and dynamically keep A by adding and deleting points for set A. Starting with an empty set A, we check each point v_i in the sorted list V_{ss}. If v_i is not a lowerleft corner of a blockage (i.e., v is either a pin or one of the other three corners of a blockage), we just add v_i into set A, otherwise we perform edge connection for v_i as follows. Suppose v_i is the lowerleft corner of blockage b_j. We first pick the point v_{i+1} which is right after v_i in the list of V_{ss}. Note that v_{i+1} must be the upperleft corner point of b_j due to the attribute of our sorted list V_{ss}. Then, we check each point a in A. If point a lies in region R_2 of b_j, we add this point to another set A_s. If b_j is not a vertical blockage, we delete this point from A. After that, we find out two points q and q' from A_s which are closest to v_i and v_{i+1}, respectively, in rectilinear distance. Finally, we add two edges, (v_i, q) and (v_{i+1}, q'), into graph G. Note that q and q' could be the same point in A_s. Before we move to the next point in V_{ss}, we vacate set A_s.

An example is shown in Figure 10. The bold-faced segments are the edges added to graph G after edge connection is performed for R_2 of all corner points. Since search region R_6 has the reverse sweep sequence, we can make use of the same sorted list V_{ss} to perform edge connection for R_6 in Pass one. Similarly, in Pass two, we perform edge connection for search region R_4 and R_8 of blockages after sorting V_s in a non-decreasing order with their y coordinates. In Pass three, we perform similar edge connection for R_1 and R_5 of blockages and R_1 and R_3 of pins after sorting V_s in non-decreasing $x+y$. Similarly, in Pass four, edge connection is performed for search region R_3 and R_7 of blockages and R_2 and R_4 of pins after V_s is sorted in non-decreasing $y-x$.

In order to achieve $O(n)$ running time, the active sets A and A_s must be efficiently maintained so that searching, deletion, and insertion each can be done in $O(\log n)$ time. The spanning graph after four passes is shown in Figure 11. The algorithm of spanning graph construction in $RSMTRB$ is summarized in Algorithm 1. The detail procedure of edge connection for search region R_2 is summarized in Algorithm 2.

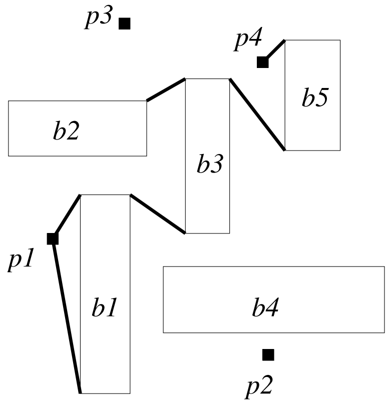


Figure 10: Edge connection for region R_2.

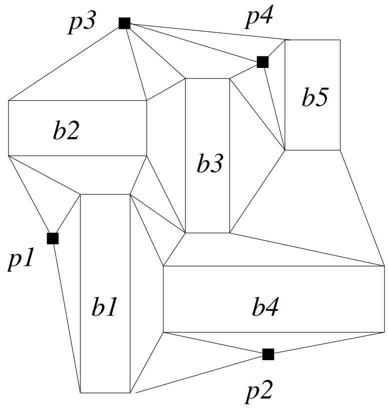


Figure 11: Complete spanning graph

Algorithm 1: Spanning graph construction in $RSMTRB$
Input: V_s
sort V_s by non-decreasing x;
perform edge connection for R_2 and R_6 of all corners;
sort V_s by non-decreasing y;
perform edge connection for R_4 and R_8 of all corners;
sort V_s by non-decreasing $x + y$;
perform edge connection for R_1 and R_5
of all corners and R_1 and R_3 of all pins;
sort V_s by non-decreasing $y - x$;
perform edge connection for R_3 and R_7
of all corners and R_2 and R_4 of all pins;
Return:spanning graph G for V_s

5. $RSMT$ CONSTRUCTION

After we complete the spanning graph $G = (V, E)$, we apply a heuristic to construct an $RSMT$ based on the graph G. The heuristic consists of following six steps. An example is shown from Figure 12 to Figure 17 to illustrate each step.

Step 1: Let P be a set of pins for a net. We construct a complete undirected graph $G_1 = (V_1, E_1)$ from G and P in such a way that $V_1 = P$ and for each edge $(v_i, v_j) \in E_1$, the length on the edge (v_i, v_j) is equal to the length of the shortest path from v_i to v_j in graph G. See Figure 12 as an illustration for step 1. The length of edge (p_1, p_3) in G_1 is the length of the shortest path from p_1 to p_3

Algorithm 2: edge connection for R_2
Input: a sorted V_{ss} with non-decreasing x.
$A = \phi$;
For each $v_i \in V_{ss}$ {
 if (v_i is a not a lowerleft corner of b_j){
  add v_i to A;
 else {
  if ($A! = \phi$){
   $A_s = \phi$;
   if (b_j is not a vertical blockage)
    delete points from A which are located in R_2 of v_i;
   add the points located in R_2 of v_i to A_s;
   if ($A_s! = \phi$){
    find point q and q' from A_s which are closest to
     v_i and v_{i+1}, respectively;
    add new edge (v_i, q) and (v_{i+1}, q') to graph G;
   }
  }
 }
}

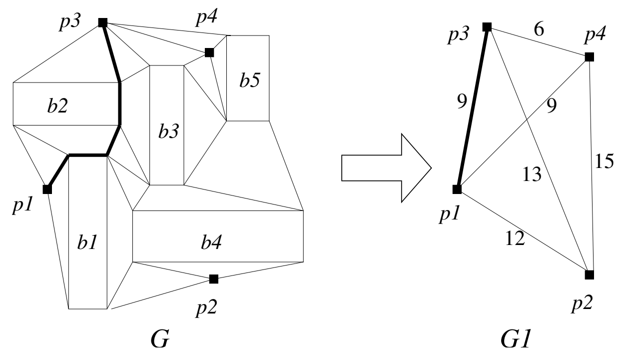


Figure 12: Step 1

in graph G, which is shown in bold-fased segments.

Step 2: Find the minimum spanning tree T_1 of G_1. If there exist several minimum spanning trees, pick an arbitrary one. In practice, we always pick the first one which we obtain. See Figure 13 as an illustration for step 2.

Step 3: Construct the subgraph G_s of G as follows. First we construct an intermediate subgraph $G'_s = (E', V')$ of G by replacing each edge in T_1 by its corresponding shortest path in G. If there are several shortest paths, pick an arbitrary one. In practice, we always pick the first one which we obtain. The edges of resulting graph G'_s are shown as bold-faced solid segments in Figure 14. Then we build G_s by adding edge (v_i, v_j) to G'_s, where $v_i \in V'$, $v_j \in V'$ and $(v_i, v_j) \in E$. The added edges are shown as bold-faced dashed segments in Figure 14.

Step 4: Find minimum spanning tree T_s of G_s. If there are several minimum spanning trees, pick an arbitrary one. See Figure 15 as an illustration for step 4.

Step 5: Construct a Steiner tree T_h from T_s by deleting edges in T_s, if necessary, so that all the leaves in T_h are pins. As illustrated in Figure 16, in this example, we could not find any leaf which is blockage corner point.

Step 6: Rectilinearize T_h to obtain a rectilinear steiner tree T_r. See Figure 17 as an illustration for step 6.

6. EXPERIMENTAL RESULTS

We implemented the spanning graph based $RSMTRB$ algorithm

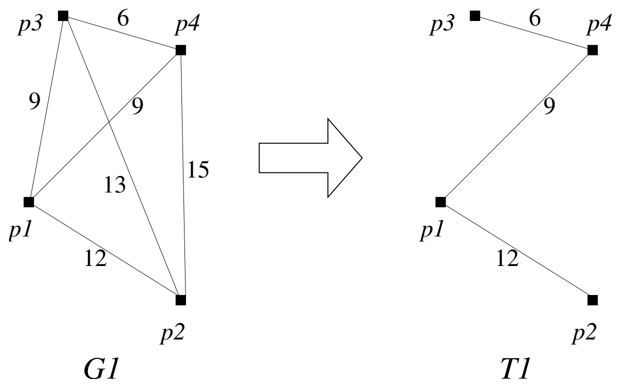


Figure 13: Step 2

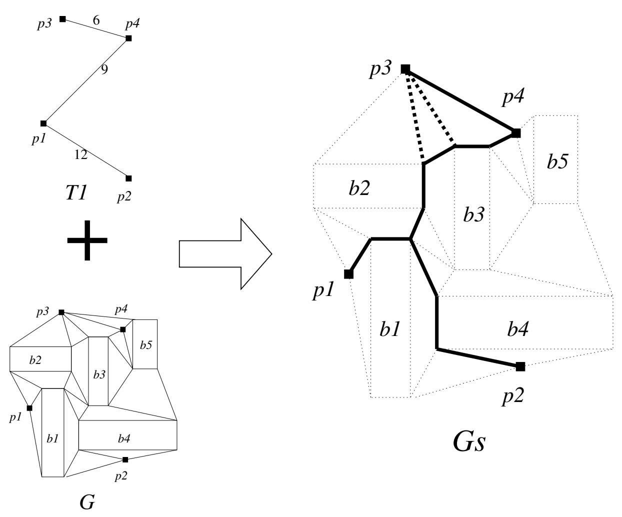


Figure 14: Step 3

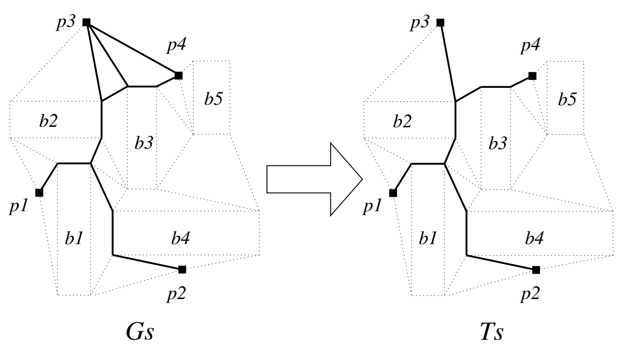


Figure 15: Step 4

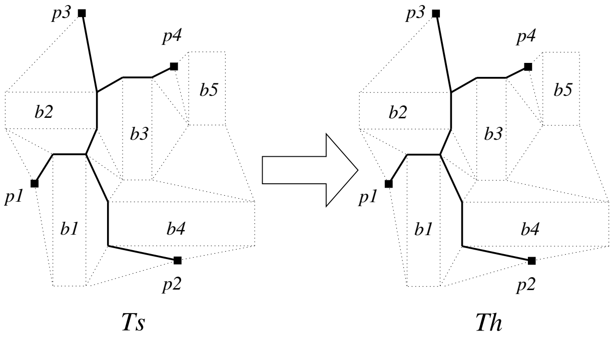


Figure 16: Step 5

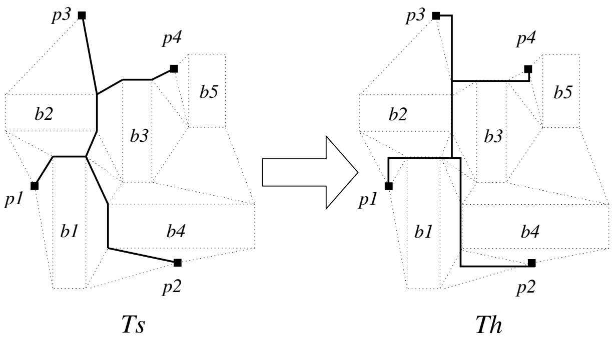


Figure 17: Step 6

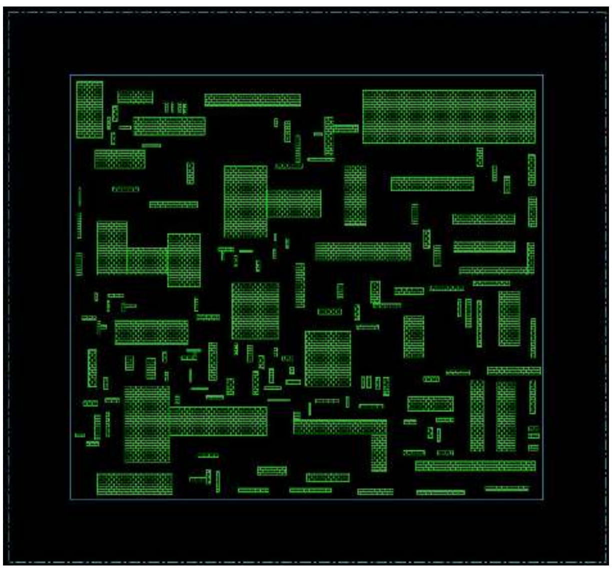


Figure 18: Blockage placement in test case 1

in C++ language. We compile and run the program on Intel Pentium 4 machine with 2.80GHz frequency and 1.5GB RAM. We pick 5 industrial test cases and randomly create blockages for these test cases. In case a source pin is inside a blockage, we move this pin to upper-left corner of this blockage. If sink pin is inside a blockage, we move this pin to lowerright corner of this blockage. The statistic data of test cases are listed in Table 1. The blockage placement for test case 1 is shown in Figure 18 as an example. We compare our approach with the traditional sequential approach illustrated in Figure 1.

The total wire length and run time comparison is given in Table 2. The results show that our spanning graph based approach can reduce 12.081% (on average) wire length of $RSMT$, comparing to sequential approach. And the run time is only increased by 48.440% on average.

Table 3 shows the wire length reduction percentage of different pin nets for each test case. It is not hard to find that our approach has most significant wire length reduction on multi-pin nets, especially while the pin number exceeds 50. This will benefit timing closure for the whole design. Since in general, multi-pin nets or high fanout nets contribute most significant portion of interconnect delay in the most timing critical path of a circuit.

Test cases	1	2	3	4	5
num of inst	158672	35601	437444	277356	450367
num of I/O pins	865	201	1774	1453	1276
num of nets	169243	36244	477380	285556	451250
2 pin net (%)	52.2	63.4	57.4	71.4	76.1
3-10 pin net (%)	42.9	32.8	21.2	16.3	16.8
11-50 pin net (%)	4.5	3.2	17.6	10.1	6.7
51-100 pin net(%)	0.34	0.47	3.50	2.01	0.36
$\geq$ 101 pin net(%)	0.06	0.13	0.30	0.20	0.03
num of blkgs	145	37	136	539	487

Table 1: Statistics of test cases

7. CONCLUSION AND DISCUSSION

In this paper, we propose an efficient and effective approach to construct rectilinear steiner minimum tree with rectilinear blockages. The connection graph we used in this approach is called spanning graph which only contains $O(n)$ edges and vertices. An $O(n \log n)$ time algorithm is proposed to construct spanning graph for $RSMTRB$. The experimental results shows that this approach can achieve a solution with significantly reduced wire length. The total run time increased is negligible in the whole design flow.

Since our heuristic is graph-based, it can be easily modified to handle other metrics. A possible extension is to construct timing aware routing tree topology for a net among routing blockages.

8. REFERENCES

[1] P. Berman, U. Fossmeier, M. Kaufmann, M. Karpinski and A. Zelikovsky. Approaching the 5/4-approximation for rectilinear Steiner trees. *Proc. European Symp. on Algorithms (ESA)*, Springer Verlag Lecture Notes in Computer Science 762, pages 533-542, 1994.

[2] M. Borah, R. M. Owens, and M. J. Irwin. A fast and simple Steiner routing heuristic. , Springer Verlag Lecture Notes in Computer Science 762, pages 533-542, 1994.

[3] M. Hanan. On Steiner's problem with rectilinear distance. *SIAM Journal on Applied Mathematics*, 14, pages 255-265, 1966.

[4] F. K. Hwang and D. S. Richards and P. Winter. *The Steiner tree problem*. North-Holland, annals of Discrete Mathematics 53, 1992.

[5] A. B. Kahng and G. Robins. A new class of iterative Steiner tree heuristics with good performance. *IEEE Transaction on CAD*, 11, pages 1462-1465, 1992.

[6] A. Zelikovsky. An 11/6-approximation for the network Steiner tree problem. *Algorithmica*, 9, pages 463-470, 1993.

[7] M. Garey and D. Johnson. The rectilinear Steiner tree problem is NP-complete. *SIAM J. Appl. Math*,32, pages 826-834, 1977.

[8] C. Y. Lee. An algorithm for path connections and its application. *IRE Transaction on Electronic Computers*,V.EC-10, pages 346-365, 1961.

[9] S. B. Akers. A modification of Lee's Path Connection Algorithm. *IEEE Transaction on Electronic Computer*,16(4), pages 97-98, 1967.

[10] J. Soukup. Fast Maze Router. *Proceeding. of 15th Design Automation Conference*, pages 100-102, 1978.

[11] F. Rubin. The Lee Connection Algorithm. *IEEE Transaction on Computer*, 23, pages 907-914, 1974.

[12] Y. Yang, Q. Zhu, T. Jing, X. Hong and Y. Wang. Rectilinear Steiner minimal tree among obstacles. *ASIC 5th Intl. Conf.*, Vol. 1, 21-24 pages 348-351, Oct. 2003.

[13] T. Lengauer. *combinatorial Algorithms for Integrated Circuit Layout*. Wiley, England, 1990.

[14] J. Ganley and J. P. Cohoon. Routing a Multi-Terminal Critical Net: Steiner Tree Construction in the Presence of Obstacles. *Intl. Symp. on Circuits and Systems*, Vol. 1, pages 113-116, 1994.

[15] K. L. Clarkson, S. Kapoor and P. M. Vaidya. Rectilinear shortest paths through polygonal obstacles in $O(n \log^2 n)$ time. *ACM Symp. on Computational Geometry*, pages 251–257, 1987.

[16] K. L. Clarkson, S. Kapoor and P. M. Vaidya. Rectilinear shortest paths through polygonal obstacles in $O(n \log^{3/2} n)$ time. Unpublished manuscript.

[17] H. Zhou, N. Shenoy, and W. Nicholls. Efficient spanning tree construction without delaney triangulation. *Information Processing Letter*, 81(5), 2002.

	Total wire length (μm)			Total run time (s)		
Test case	$Sequential$	Ours	decreased (%)	$Sequential$	Ours	increased (%)
1	15174595	13855164	8.695	9.531	12.193	27.926
2	1995	1820	8.778	1.404	2.361	68.112
3	2854122	2400499	15.890	18.671	27.339	46.425
4	2452867	2051254	16.371	14.998	20.668	37.805
5	1022723	913584	10.670	13.576	21.984	61.933
On average			12.081			48.440

Table 2: Totoal wire length and run time comparison.

Test case	2 pin net(%)	3-10 pin net(%)	11-50 pin net (%)	51-100 pin net (%)	$\geq$ 101 pin net (%)
1	4.366	14.042	15.301	22.923	29.989
2	4.619	13.990	14.935	23.494	30.492
3	6.984	19.831	21.404	27.230	32.103
4	6.972	19.983	22.398	24.506	33.840
5	4.581	15.720	18.591	20.383	31.729
On average	5.492	16.713	18.526	23.707	31.631

Table 3: Wirelength reduction percentage on nets.