Research on Efficient Rectilinear Steiner Tree Construction with Rectangular Obstacles

Chiung-Yi Tseng
Department of Electrical Engineering
National Central University, Chung-Li, Taiwan
E-mail: 945001002@cc.ncu.edu.tw

Abstract—This thesis reviews the methods for rectilinear tree construction with rectangular obstacles with analysis on efficiency, and mainly focuses on RSMTRO (rectilinear Steiner minimal tree with rectangular obstacles). The thesis then lists the current research challenge of RSMTRO problem, and proposes an experimental method which is based on RSMTRO and extends the technique to 3D rectilinear tree construction with rectangular obstacles. Issues regarding 3D routing are proposed as future work.

Index Terms—Rectilinear Obstacles, Rectilinear Steiner Tree, Spanning Graph, 3D routing

I. INTRODUCTION

In VLSI/ULSI physical design [1], obstacle-avoiding rectilinear Steiner minimal tree (OARSMT) is an important problem [2-4]: with the evolution of technology, scheduling routing in 2D VLSI circuit with certain constraints by hand would be impractical. Design houses pursue shortest wire length with certain patterns, possibly large-scale power networks, prerouted nets, IP blocks, feature patterns for manufacturability etc., on the chip. OARSMT problem formulates the routing requirements: Given a set of n pins and a set of obstacles on a plane, an obstacle-avoiding rectilinear Steiner minimal tree (OARSMT) connects these pins, possibly through some additional points (called Steiner points), and avoids running through any obstacle to construct a tree with a minimal total wire length [2-4].

The formal RSMTRO problem can be stated as follows [9]. Let \( P = \{p_1, p_2, ..., p_m\} \) be a set of pins for \( m \) pins net, \( B = \{b_1, b_2, ..., b_k\} \) be a set of rectangular obstacles, and \( V = \{v_1, v_2, ..., v_n\} = P \cup \{\text{corners in } B\} \) as the vertex set, where each \( v_i \) has coordinates \((x_i, y_i)\). Note that each rectangular obstacle has four corners; we have \( n \leq m + 4k \). For example, there are 4 pins (\( m=4 \)), 5 rectangular obstacles (\( k=5 \)), and 24 vertices (\( v=24 \)). The input information for both pins is obstacles are listed in Figures 1(a) and 1(b), respectively. Each pin is assigned with its coordinate \((x, y)\), e.g., the pin A is located at \((2, 6)\); and each obstacle is defined by two pairs of points, i.e., the coordinates of the bottom-left and the top-right points of the obstacle, e.g., the two corners of obstacle B1 are \((1, 10)\) and \((6, 12)\), respectively. Based on the list of coordinates of pins n obstacles, Figure 1(c) shows the layout. The RSMTRO problem is to connect all pins through some extra points (Steiner points) to achieve a minimal total length, while avoiding the intersection with any rectangular obstacles in the design. Figure 1(d) shows a RSMTRO solution for the layout in Figure 1(c), where two Steiner points are needed for a conference. Figure 2 gives the definition of legal connection. An obstacle cannot overlap with another obstacle (Fig 2(a)). Fig 2(b) shows a legal connection. A vertices can only touch the edge of an obstacle (Fig 2(d)) but not within an obstacle (Fig 2(c)). The connection between two vertices can only touch the edge of an obstacle (Fig 2(e)) but not within an obstacle (Fig 2(f)).

II. LITERATURE REVIEW

The existence of obstacles increases the rectilinear Steiner minimal tree problem, which is proved to be a NP-complete problem. Thus, the previous heuristic algorithms target on OARSMT problem is mainly 3 categories [9]: (1) Maze routing; (2) Sequential approach; and (3) Connection graph based algorithm. Either suffers from poor quality and expensive running time.


Sequential approach first constructs a minimum spanning tree without obstacles, and then exploits simple line sweep techniques to replace the overlapping edges with the edges
around the obstacles. This approach is popular in industry due to its simplicity and efficiency. Several methods have been proposed to improve the edge-sweeping process; however, the first step for the tree construction may not have the global view of the obstacles, and thus the second step might only remove the overlaps locally around the obstacles. As a result, the solution quality may be limited.

Compared to sequential approach, connection graph based algorithm [9] has better global view and solution quality since it builds a connection graph by pins and obstacle boundaries, which guarantees at least one OARSMT is embedded. However, there is a trade-off between effectiveness and efficiency. The larger the connected graph, the more expensive computing time searching for a better OARSMT.

Our focus, RMSTRO falls in a new proposed category: spanning graph based approach which provides good solution quality and economical space usage, O(n), and running time, O(nlgn). The spanning graph construction, decides time complexity. RMSTRO eliminates the vertices which will not be used in the following process; thus, reduce the running time.

III. DEVELOPMENT OF RMSTRO

The development comprises two processes: (1) path construction: constructs a minimum spanning tree; (2) Path assignment: organize the local edges using specific patterns.

Path construction:

Sort pins by their x coordinates in ascending order first; if several pins have the same x coordinates, sort them by their y coordinates. In the same way, sort obstacle. Build a distance table (Figure 3) to records the Manhattan distance between any two pins without obstacles. Manhattan distance is defined as follow: let pi=(xi,yi) and pj=(xj,yj) be any two pins, the distance is defined as d(pi,pj)=|xi-xj|+|yi-yj|.

Observation 1:

Without the obstacles, the distance d(pi,pj) is a half length of the perimeter of a rectangle which takes the points pi and pj as either the diagonal or off-diagonal corners, as shown in Figure 5(b). The rectangle may become a line segment if either xi=xj or yi=yj. The rectangle has two paths traveling from pi to pj in the perimeter of a rectangle which takes the points pi and pj as either the diagonal or off-diagonal corners, as shown in Figure 5(b). The rectangle may become a line segment if either xi=xj or yi=yj. The rectangle has two paths traveling from pi to pj in the rectangle does not intersect any obstacles. As a result, the solution quality may be limited.

Compared to sequential approach, connection graph based algorithm [9] has better global view and solution quality since it builds a connection graph by pins and obstacle boundaries, which guarantees at least one OARSMT is embedded. However, there is a trade-off between effectiveness and efficiency. The larger the connected graph, the more expensive computing time searching for a better OARSMT.

Our focus, RMSTRO falls in a new proposed category: spanning graph based approach which provides good solution quality and economical space usage, O(n), and running time, O(nlgn). The spanning graph construction, decides time complexity. RMSTRO eliminates the vertices which will not be used in the following process; thus, reduce the running time.

III. DEVELOPMENT OF RMSTRO

The development comprises two processes: (1) path construction: constructs a minimum spanning tree; (2) Path assignment: organize the local edges using specific patterns.

Path construction:

Sort pins by their x coordinates in ascending order first; if several pins have the same x coordinates, sort them by their y coordinates. In the same way, sort obstacle. Build a distance table (Figure 3) to records the Manhattan distance between any two pins without obstacles. Manhattan distance is defined as follow: let pi=(xi,yi) and pj=(xj,yj) be any two pins, the distance is defined as d(pi,pj)=|xi-xj|+|yi-yj|.

Observation 1:

Without the obstacles, the distance d(pi,pj) is a half length of the perimeter of a rectangle which takes the points pi and pj as either the diagonal or off-diagonal corners, as shown in Figure 5(b). The rectangle may become a line segment if either xi=xj or yi=yj. The rectangle has two paths traveling from pi to pj in the absence of obstacles, and they may or may not intersect the obstacle(s).

To verify whether there exists a legal connection between two vertices, we give a definition to essential pair. If there was a legal connection, then there exists an essential pair; otherwise, we eliminate this pair of vertices since there is no legal connection between them.

Definition 1:

A pin-pair (pi,pj) is referred to as essential if at least one path in the rectangle does not intersect any obstacles. Otherwise, the pair is non-essential.

Path assignment

In this phase, paths are assigned according to certain rules given in Figure 6.

The selection is determined by which marked corners to be connected. For example, if the next corner to be connected is vc, then the left-up path is selected, as shown in Figure 6(b). On the other hand, when pi is connected to va, there exist two paths: right-up and upright. If the next corner to be connected is vd, the upright path is selected as given in Figure 6(c). If pi, va and vb are connected, then pi connecting vb chooses the up-left path and pi connecting va selects up-right path. As a result, these three points are connected with a Steiner point.
The development calls an end. Here, we analyze the time and space complexity of each step. All sorting process take Θ(nlgn) time complexity and Θ(n) space complexity with quick sort algorithm. The time and space complexity of building a distance table (step 1.) are O(n²). Deriving a MST with distance table (i.e. adjacency list) is O(n²) time complexity. Searching for non-essential pairs and removing them (step 2.2~2.4) takes O(n²) time complexity, since the algorithm has to examine pair wisely. Step 3 takes O(n) time complexity because the algorithm checks point wisely. The overall time complexity of ReSTRO algorithm is O(n²) and the time complexity is O(n²), which are satisfying compared with other approaches. On the level of NP problem, few algorithms could provide both quality and efficiency.

There are still several challenges in RSMTRO problem. First, with the evolution of technology, the size of components keeps shrinking with the ever rising frequency; eventually, the matching of wire length and wave length would be a problem. Under this circumstance, even a corner or a Steiner point has to be carefully planned in order not to incur undesired capacitors or inductors. Besides, the path delay would be a critical issue in the future. Second, there are vertices in a 2D space that connect to different voltage power supplies. One may try to connect the vertices sequentially, i.e. connect a group of vertices, set them as the obstacles for the groups which have not been processed, and then process another group in the same way. However, the priority problem bothers: once two lines overlap or intersect with each other, which one should be chosen? The existed methods connecting vertices to the same voltage may not be applicable. Therefore, limiting routing within a 2D space may result in a heuristic solution. Were a 3D space is allowed, a better global view with shorter wire length; nevertheless, increases the complexity. In some situations, overlapping and intersection occur in 2D routing. 3D routing seems to be an unavoidable choice. In the following text, we call vertices which result in overlapping or intersection "α vertices".

3D routing method simply maps some vertices to other layers according to some criteria in order to remove the overlapping and intersections or reduce the wire length in original layer, and then exploit ReSTRO algorithm in 2D layer with some vertices mapped from other layers. The overlapping and intersections are removed and the complexity of the original plane is transferred to another layer. Thus, a fair criterion must guarantee all the overlapping and intersections are removed and the vertices would not concentrate in certain layers. We us the notation: Xp as the x-coordinate of X and Yp as the y-coordinate. Here are the representation and definitions:

Definition 2:
An overlap happens when two pairs, say (A, B) and (C, D) are connected to different voltage supplies. Meanwhile, if (X_A ≤ X_C ≤ X_B and X_A ≤ X_D ≤ X_B) and Y_A=Y_B<Y_C and Y_A=Y_B<Y_D), then (A, B) and (C, D) overlap with each other.

Definition 3:
An intersection happens when two pairs, say (A, B) and (C, D) are connected to different voltage supplies. Meanwhile, if (X_A≤X_C≤X_B and X_A≤X_D≤X_B and Y_A=Y_B<Y_C and Y_A=Y_B<Y_D), then pair (C, D) intersects with pair (or (A, B) intersects with (C, D)) where (C, D) is the vertical line and (A, B) is the horizontal line. Definition for horizontal line is similar to vertical line.

Criteria 1:
The vertex which results in overlapping has the highest priority mapping to another layer. The second is the vertex which causes intersection. These two kinds of vertex must be mapped. The last are the non-α vertices.

To remove an overlap, one of the pair (2 vertices) must be mapped to another layer. Analogously, only 1 vertex has to be mapped in order to remove an intersection. To main the fair distribution of vertices among layers, a simple criterion is proposed to whether to map a vertex. Before exploit the criteria, we must consider several possible distributions of pins and obstacles. The most desired situation is that most of the pins concentrate on the center of the chip while the obstacles scatter apart. In this case, we can connect most of the pins without avoiding obstacles. The second case is having pins scatter apart and obstacles concentrate on the center. In this case, most of the pins can connect directly with each other with the paths on the outer region of the chip. The worst case is that all of the vertices are random distributed among the chip. In this case, the connection could be complex.
Using standard deviation with radius as the parameter, there are 3 cases. In the following context, we use the notation: \( P_i \) as the set of pins in layer \( i \), \( O_i \) as the set of obstacles in layer \( i \), and \( \sigma \) as the standard deviation of a specific set.

**case 1:** \( \sigma(P_i) < \sigma(O_i) \), which indicates obstacles are more concentrate than pins in layer \( i \).

**case 2:** \( \sigma(P_i) > \sigma(O_i) \), which indicates pins are more concentrate than obstacles in layer \( i \).

**case 3:** \( \sigma(P_i) \approx \sigma(O_i) \approx \sigma(P_i \cup O_i) \), which indicates all of the vertices are random distribution in layer \( i \).

Our goal is to have the vertex in every layer distribute like that in case 1, or case 2 (heuristic). Giving each case cost: 1, 2 and 3, we turn the goal into having lowest cost by mapping certain vertices to other layer. However, mapping vertex \( P \) from layer \( i \) to layer \( j \) may change the deviation of layer \( j \). A criterion is proposed for judge.

**Criteria 2:** Any mapping \( \alpha \) vertex to layer \( j \) that increases the total cost should be mapped to another layer, unless there is no other layer except \( i \) and \( j \).

At this moment, the proposed method could remove the overlapping and intersections. For other vertices which are non-\( \alpha \) vertices, our goal is to distribute them fairly. Another Criterion is stated as follows:

**Criteria 3:** For non-\( \alpha \) vertices, any mapping should not increase \( |\sigma(P_i) - \sigma(P_j)| \).

Lastly, the proposed method completes and mapping work. In each layer, ReSTRO algorithm is utilized to route the vertices and the task calls an end.

**CONCLUSION**

The 3D routing method may get a shorter wire length but higher complexity. Moreover, the cost of multilayer is higher than single layer. Besides, due to the evolution of technology, the chips become thinner, more fragile and easily cracked. Vias which connect layers may cause the chip cracked. Thus, there is a trade-off between the number of vias and total wire length.

**ACKNOWLEDGMENT**

The thesis is the final project of the course CAD for VLSI design, given by professor Chin-Long Wey.

**REFERENCES**


[10] C. L. Wey, "ReSTRO: Efficient Rectilinear Steiner Tree Construction with Rectangular Obstacles"