An Improved Augmented Line Segment based Algorithm for the Generation of Rectilinear Steiner Minimum Tree

Vani V1, G. R. Prasad2

1Department of Information Science Engineering, Bangalore Institute of Engineering, Bangalore, India.
2Department of Computer Science and Engineering, BMSCE, Bangalore, India.

ABSTRACT
An improved Augmented Line Segment Based (ALSB) algorithm for the construction of Rectilinear Steiner Minimum Tree using augmented line segments is proposed. The proposed algorithm works by incrementally increasing the length of line segments drawn from all the points in four directions. The edges are incrementally added to the tree when two line segments intersect. The reduction in cost is obtained by postponing the addition of the edge into the tree when both the edges (upper and lower L-shaped layouts) are of same length or there is no overlap. The improvement is focused on reduction of the cost of the tree and the number of times the line segments are augmented. Instead of increasing the length of line segments by 1, the line segments length are doubled each time until they cross the intersection point between them. The proposed algorithm reduces the wire length and produces good reduction in the number of times the line segments are incremented. Rectilinear Steiner Minimum Tree has the main application in the global routing phase of VLSI design. The proposed improved ALSB algorithm efficiently constructs RSMT for the set of circuits in IBM benchmark.

Corresponding Author:
Vani V,
Department of Information Science and Engineering,
Bangalore Institute of Technology,
K R Road, V V Puram, Bangalore, Karnatake-560004, India.
Email: vanisrin@gmail.com

1. INTRODUCTION
Rectilinear Steiner Minimum Tree (RSMT) is the tree that connects the given set of points in a rectilinear fashion using only horizontal and vertical line segments. Few additional points called as Steiner points are added to the tree. Steiner points are added to connect the edges in rectilinear manner and to reduce the total cost of the tree. The rectilinear distance between two points \( P_1 = \{x_1, y_1\} \) and \( P_2 = \{x_2, y_2\} \) is given by \( |x_1-x_2| + |y_1-y_2| \). The main application of RSMT is in the global routing phase of the VLSI design. During global routing, the wiring channels (points) that have been placed at particular position using placement algorithm will be connected in rectilinear manner.

Figure 1 illustrates the RSMT constructed over the points \( P_1 \) to \( P_5 \) respectively. It can be identified that all the edges of the RSMT are the part of Hanan Grid. An edge can be either degenerate or non-degenerate. An edge is degenerate, if the two points it connects are in the same x-axis or y-axis. The non-degenerate edge for connecting two points is obtained from the enclosing rectangle and can be either the upper L-Shaped layout or the lower L-shaped layout. The cost or the total length of the RSMT can be reduced by carefully selecting an appropriate layout for each edge that overlaps with the layout of the other edges. For example in the Figure 1, the lower L-shaped layouts are selected for connecting the two edges \( P_1-P_2 \) and \( P_2-P_3 \) as they result in overlap and hence cost reduction. Rectilinear Steiner Minimum Tree construction is one of the fundamental problems that have many applications in VLSI design. It can be used...
for wire load estimation, identifying routing congestion and interconnect delay in early stages of VLSI design.

Rectilinear version of Steiner Tree was first studied by Hanan who gave exact solutions for the number of points \((N) \leq 5\) and also proved that the Steiner points lie on the Hanan Grid. Hanan grid is obtained by inducing horizontal and vertical line segments through the given set of points as shown in figure 2. The points at the intersection of these line segments are called Hanan Points [1]. Many approximation algorithms exist for constructing RSMT as the problem of constructing RSMT is shown to be NP complete by Garey and Johnson [2]. The ratio of the length of rectilinear version of Minimum Spanning Tree (RMST) to that of RSMT is proved to be \(\leq 3/2\) by Hwang [3].

Khang and Robins proposed Iterated 1- Steiner (I1S) algorithm that iteratively adds a Steiner point that results in cost reduction and a Batched version (Batched Iterated 1-Steiner algorithm) where a group of Steiner points are added during each iteration [4]. Borah, Owens and Irwin presented an edge-based heuristic algorithm which initially constructs a Minimum Spanning Tree and transforms it into a RSMT by iteratively connecting a point to the enclosing rectangular layout of the visible edge in the MST [5]. Zhou proposed a Rectilinear Spanning graph (RSG) algorithm [6] which applies Borah et al edge based heuristic algorithm on Spanning Tree constructed from the Zhou at al spanning graph algorithm which was constructed by connecting each point to the nearest point in eight octal regions [7].

Griffith, Jeff, et al. proposed a variant of B1S using dynamic MST update scheme where a point is connected to the nearest points in eight octants and the longest edge is removed in the formed loop[8]. Khang, Mandoiu and Zelikovsky proposed a batched version of greedy triple contraction algorithm[9] called Batched Greedy Algorithm where RSMT is constructed by iteratively adding a batch of triples (optimal full Steiner tree for a set of 3 points with all the points in the leaves position) [10]. Wong, Yiu-Chung and Chu proposed a Fast Look-Up Table based algorithm with a pre-computed table for constructing RSMT for \(N<9\). For \(N>9\), a net breaking algorithm is iteratively used until \(N<9\) and the pre-computed table can be used [11]. RSMT was also constructed by connecting the trees that have been constructed for the computed clusters of given points [12]. The existing algorithms for the construction of RSMT have been extensively surveyed [13].

2. RESEARCH METHOD

The proposed algorithm works by drawing incremental four line segments through each and every point. The edges are iteratively added when two line segments intersect. For each of the edge, two L-shaped layouts can be identified. The L-shaped layout which has an overlap with other edges should be cleverly selected to reduce the overall cost of the RSMT. If a decision in selecting a layout cannot be made or if both are of same length, the process of adding the edge to the RSMT will be delayed until proper decision cannot be made. The enhancement to Augment Line Segment Based (ALSB) Algorithm is carried out by doubling the size of line segments in each iteration unless they cross the intersection point [14]. If they cross intersection point, the length of the line segments will be reduced back to the previous value and augmenting starts will value of 1. The procedure carried out is as follows:-

1. Identify the boundary- Boundary is computed by identifying the minimum and maximum x and y values. The length of the line segments are incremented until they touch the boundary or until the RSMT is constructed.

Figure 1. Rectilinear Steiner Minimum Tree

Figure 2. Hanan Grid

An Improved Augmented Line Segment based Algorithm for the Generation of Rectilinear …. (Vani V)
2. Augment the line segments- The length of the line segments are doubled each time until it crosses the intersection point of any two line segments else the length of all the line segments will be set to the previous values and again starts augmenting with the step_size of one.

3. Construct RSMT- RSMT will be constructed by incrementally adding edges when two line segments intersect and if that edge does not form a loop in the partially constructed RSMT. Adding an edge requires selecting one of the two L-Shaped layout which results in cost reduction. If both the edges are of same length, then both edges will be marked as temporary edges until a decision can be made.

Finally when no more edges can be added and if temporary edges exist then the L-shaped layouts are checked for overlap with the constructed RSMT and the corresponding Layout will be added.

2.1. Algorithm

The improved ALSB algorithm takes as an input a set of points and computes the RSMT along with its cost or total length.

\[ n --- \text{number of points} \]

\[ \text{Current\_length}=0 \]

\[ \text{step\_size}=0 \]

\[ \text{while (num\_edges} < n-1) \]

\[ \text{begin} \]

\[ \text{for i}=1 \text{ to n --- do in parallel} \]

\[ \text{previous\_length} = \text{Current\_length of line segments} \]

\[ \text{if (step\_size} = 0) \]

\[ \text{Current\_length}= \text{previous\_length}+1 \]

\[ \text{step\_size}=1; \]

\[ \text{else} \]

\[ \text{Current\_length}=\text{previous\_length}+2*\text{step\_size} \]

\[ \text{for i}=1 \text{ to n --- do in parallel} \]

\[ \text{begin} \]

\[ \text{if (two line segments intersect \\&
\& doesn't form loop in constructed RSMT)} \]

\[ \text{begin} \]

\[ \text{compute length of upper L-shaped layout (length1) and lower L-shaped layout (length2)} \]

\[ \text{if (length1} = \text{length2)} \]

\[ \text{begin} \]

\[ \text{if (degenerate edge)} \]

\[ \text{add edge to RSMT} \]

\[ \text{else} \]

\[ \text{indicate both as temporary} \]

\[ \text{end} \]

\[ \text{else} \]

\[ \text{begin} \]

\[ \text{add the shorter edge to RSMT} \]

\[ \text{if (length reduction because of temporary edge)} \]

\[ \text{make that edge permanent and add to RSMT} \]

\[ \text{Remove the other contemporary temporary edge} \]

\[ \text{end} \]

\[ \text{end} \]

\[ \text{if (line segments cross the intersection point)} \]

\[ \text{step\_size}=0 \]

\[ \text{Current\_length}=\text{previous\_length} \]

\[ \text{end} \]

\[ \text{if (there are temporary edges that need to be added to the tree)} \]

\[ \text{begin} \]

\[ \text{Compute the length of upper L-shaped layout (length1) and lower L-shaped layout (length2)} \]

\[ \text{if}\text{(length1} = \text{length2)} \]

\[ \text{end} \]

\[ \text{end} \]

\[ \text{end} \]

\[ \text{end} \]

\[ \text{end} \]
randomly add any one L-shaped layout
else
    add the L-shaped layout with reduced length
end

The RSMT length reduction is obtained by further checking for overlap when the temporary edges are to be added to the constructed tree rather than randomly selecting any one as in ALSB algorithm. The number of line segment increment is reduced by doubling the step_size until the line segments cross the intersection point.

3. RESULTS AND ANALYSIS

The proposed algorithm has been implemented in C. Figure 3 shows that the improved ALSB algorithm shows good improvement over the ALSB algorithm with respect to number of times the line segments are augmented on various set of points that were randomly generated. When the points are less and sparsely located, good reduction can be identified. Table 1 demonstrates the cost reduction obtained by the improved ALSB algorithm.

![Comparison between ALSB and improved ALSB algorithm](image)

**Figure 3. Comparison between ALSB and improved ALSB algorithm**

<table>
<thead>
<tr>
<th>Number of points</th>
<th>Total length of Rectilinear Minimum Spanning Tree</th>
<th>Total length of RSMT using ALSB algorithm</th>
<th>Total length of RSMT using improved ALSB algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>832</td>
<td>791</td>
<td>786</td>
</tr>
<tr>
<td>20</td>
<td>1344</td>
<td>1305</td>
<td>1294</td>
</tr>
<tr>
<td>30</td>
<td>1916</td>
<td>1787</td>
<td>1773</td>
</tr>
<tr>
<td>40</td>
<td>2196</td>
<td>1975</td>
<td>1958</td>
</tr>
<tr>
<td>50</td>
<td>2497</td>
<td>2336</td>
<td>2321</td>
</tr>
<tr>
<td>100</td>
<td>3257</td>
<td>3156</td>
<td>3132</td>
</tr>
<tr>
<td>200</td>
<td>4682</td>
<td>4376</td>
<td>4345</td>
</tr>
<tr>
<td>300</td>
<td>5849</td>
<td>5198</td>
<td>5173</td>
</tr>
<tr>
<td>400</td>
<td>6679</td>
<td>6056</td>
<td>6026</td>
</tr>
<tr>
<td>500</td>
<td>7407</td>
<td>6594</td>
<td>6559</td>
</tr>
<tr>
<td>600</td>
<td>8054</td>
<td>7130</td>
<td>7105</td>
</tr>
</tbody>
</table>

The algorithm was tested on IBM ISPD08 benchmark for global routing where the placement was carried out by Dragon 1.0 [15]. The benchmark information is listed in the Table 2 with the details of the number of nets in each circuit. It can also be identified that most of the nets in each circuit have degree < 10.

Table 3 shows the results obtained by the application of improved ALSB algorithm on IBM benchmarks. RSMT was effectively constructed for all the 10 circuits and the maximum RSMT length computed for a net in each circuit is as shown in Table 3.
Table 2: IBM benchmark for global routing

<table>
<thead>
<tr>
<th>Circuit</th>
<th># of nets</th>
<th># of nets with degree \leq 10</th>
<th># of nets with degree &gt;10</th>
<th>Average degree</th>
<th>Maximum degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>ibm01</td>
<td>11508</td>
<td>10897</td>
<td>611</td>
<td>3.847</td>
<td>42</td>
</tr>
<tr>
<td>ibm02</td>
<td>18430</td>
<td>17211</td>
<td>1219</td>
<td>4.052</td>
<td>134</td>
</tr>
<tr>
<td>ibm03</td>
<td>19735</td>
<td>18759</td>
<td>976</td>
<td>3.195</td>
<td>53</td>
</tr>
<tr>
<td>ibm04</td>
<td>26164</td>
<td>25679</td>
<td>485</td>
<td>3.424</td>
<td>46</td>
</tr>
<tr>
<td>ibm05</td>
<td>27778</td>
<td>24124</td>
<td>3654</td>
<td>4.480</td>
<td>17</td>
</tr>
<tr>
<td>ibm06</td>
<td>33355</td>
<td>31014</td>
<td>2341</td>
<td>3.726</td>
<td>35</td>
</tr>
<tr>
<td>ibm07</td>
<td>44395</td>
<td>41671</td>
<td>2724</td>
<td>3.702</td>
<td>25</td>
</tr>
<tr>
<td>ibm08</td>
<td>47495</td>
<td>44719</td>
<td>3226</td>
<td>4.133</td>
<td>75</td>
</tr>
<tr>
<td>ibm09</td>
<td>50394</td>
<td>47542</td>
<td>2852</td>
<td>3.728</td>
<td>39</td>
</tr>
<tr>
<td>ibm10</td>
<td>64228</td>
<td>59583</td>
<td>4645</td>
<td>4.188</td>
<td>41</td>
</tr>
</tbody>
</table>

Table 3: Application of improved ALSB algorithm on IBM ISPD08 benchmark

<table>
<thead>
<tr>
<th>Circuit</th>
<th># of nets</th>
<th>Total RSMT length of all nets</th>
<th>Max RSMT length of a net in the circuit</th>
</tr>
</thead>
<tbody>
<tr>
<td>ibm01</td>
<td>11508</td>
<td>61967</td>
<td>118</td>
</tr>
<tr>
<td>ibm02</td>
<td>18430</td>
<td>172778</td>
<td>366</td>
</tr>
<tr>
<td>ibm03</td>
<td>19735</td>
<td>138681</td>
<td>176</td>
</tr>
<tr>
<td>ibm04</td>
<td>26164</td>
<td>167795</td>
<td>269</td>
</tr>
<tr>
<td>ibm05</td>
<td>27778</td>
<td>438022</td>
<td>329</td>
</tr>
<tr>
<td>ibm06</td>
<td>33355</td>
<td>300093</td>
<td>295</td>
</tr>
<tr>
<td>ibm07</td>
<td>44395</td>
<td>378053</td>
<td>209</td>
</tr>
<tr>
<td>ibm08</td>
<td>47495</td>
<td>428319</td>
<td>414</td>
</tr>
<tr>
<td>ibm09</td>
<td>50394</td>
<td>429567</td>
<td>264</td>
</tr>
<tr>
<td>ibm10</td>
<td>64228</td>
<td>597643</td>
<td>333</td>
</tr>
</tbody>
</table>

4. CONCLUSION

The proposed improved ALSB algorithm provides good improvement over the ALSB algorithm in terms of number of line segment increment and cost reduction. The algorithm was also efficiently tested on IBM ISPD08 benchmark as shown in table 3. Future efforts would be directed towards further cost reduction of the tree and to implement the above algorithm on FPGA to improve the performance.

REFERENCES

An Improved Augmented Line Segment based Algorithm for the Generation of Rectilinear .... (Vani V)


BIOGRAPHIES OF AUTHORS

Vani V is an Assistant Professor in the Department of Information Science and Engineering, Bangalore Institute Of Technology, Bangalore. She received B.E degree in Computer Science and Engineering from Visveshwaraiah Technological University in 2003 and M.Tech degree in Computer Network Engineering from VTU in 2007. She is currently pursuing Ph.D. degree in VTU in the area of Reconfigurable Computing.

Prasad G R is an Associate Professor in Department of Computer Science & Engineering, BMSCE, Bangalore. He holds a Ph.D from National Institute of Technology, Karnataka, Surathkal, INDIA. He received his M.Tech degree in Computer Science & Engineering from Bangalore University in 1999 and B.E Degree in Computer Science & Engineering from Bangalore University in 1995. His research interests include Reconfigurable computing.