IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)
Volume 3, Issue 3 (Sep. – Oct. 2013), PP 61-68
e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197
www.iosrjournals.org

Performance Analysis of the Algorithms for the Construction of Rectilinear Steiner Minimum Tree

Vani V[1], G R Prasad[2]
[1](*Department of Information Science and Engineering, Bangalore Institute of Technology, Bangalore, India)*
[2](*Department of Computer Science and Engineering, B M S College of Engineering, Bangalore, India)*

Abstract : *The advances in VLSI technology have led to complex and larger circuits. As the circuits become complex and large, the amount of time required for the design of such circuits increases. The people in the VLSI industry are looking for faster EDA (Electronic Design Automation) tools so as to reduce the design time. Routing is a phase in the design (physical design) of electronic circuits, wherein pins of a net will be interconnected and this uses Rectilinear Steiner Minimum Trees. Rectilinear Steiner Minimum Tree problem is to find a minimum length tree connecting the given set of points using only horizontal and vertical line segments, with the additional set of points (Steiner points). Steiner points are introduced to reduce the total length of the tree and to connect in rectilinear manner. The problem of finding Rectilinear Steiner Minimum Tree is one of the fundamental problems in the field of electronic design automation. This paper provides a comprehensive analysis of the various Rectilinear Steiner Minimum Tree algorithms proposed till date and shows that there is a need for an algorithm or approach to produce better solution quality (reduced wire length) in less time. Rectilinear Steiner Minimum Tree is widely used in global routing phase of VLSI design and wire length estimation.*

Keywords: *Global Routing, Rectilinear Steiner Minimum Tree, Rectilinear Minimum spanning Tree, VLSI design*

I. Introduction

The main goal of routing in VLSI design is to interconnect the cells that have been assigned positions as a solution of the placement problem. Routing is generally performed in two different stages. The first stage, called the global routing will identify the wiring channels through which connections can run. The second stage, called detailed routing, fixes the exact paths that the wire has to take. Rectilinear Steiner Minimum Tree (RSMT) is used in the global routing phase of VLSI Design [1].

Given a set of points P(terminal points), Rectilinear Steiner Minimum Tree(RSMT) is the minimum length tree constructed over {P}U{S} using only horizontal and vertical line segments, where S is the set of additional points called Steiner Points (Fig. 1). The distance between two points is measured in the rectilinear metric. The rectilinear distance between a pair of points $p_i= (x_i, y_i)$ and $p_j=(x_j, y_j)$ is equal to $|x_i - x_j | + | y_i - y_j|$.

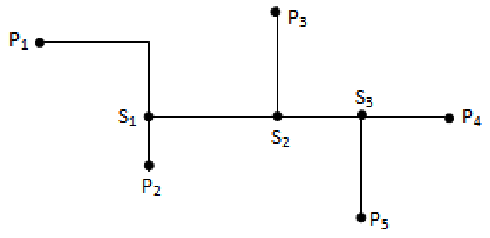

Fig. 1 Rectilinear Steiner Minimum Tree

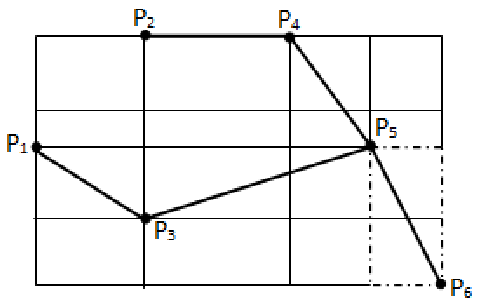

Fig. 2 Hanan Grid and Minimum Spanning Tree

Hanan[2] was the first to consider the rectilinear version of Steiner Tree. He gave exact solutions for n≤5 and also showed that the possible candidate Steiner points (CSp) lie on Hanan Grid (fig 2). Garey and Johnson [3] showed that the problem of constructing RSMT is NP-Complete and Hwang [4] proved that the ratio of cost of Rectilinear Minimum Spanning Tree (RMST) to the cost of RSMT is ≤ 3/2. Fig.1 shows a RSMT for connecting pins p_1, p_2, p_3, p_4 and p_5 along with the Steiner points s_1, s_2 and s_3.

II. BASIC DEFINITIONS

- ***Minimum Spanning Tree***

Given a set of points P, Minimum Spanning Tree (MST) is the minimum length tree over P. Fig 2 shows the MST over point's p_1 to p_6.

- ***Rectilinear Minimum Spanning Tree***

Rectilinear Minimum Spanning Tree (RMST) is a MST where the distances between two points are measured in rectilinear metric and the edges are restricted to be of horizontal and vertical line segments.

- ***Hanan Grid***

The Hanan grid [2] is the grid induced by the set of points P by drawing horizontal and vertical lines through each and every point in P (Fig.2).

- ***Hanan Points***

The points obtained at the intersection of horizontal and vertical lines drawn through the given set of points P are called as Hanan points [2].

- ***L-Shaped Layout***

For each non-degenerate edge (an edge is non-degenerate if the endpoints of the edge do not lie on the same horizontal or vertical line) , two L-shaped layouts can be formed from the enclosing rectangle. In Fig.2 the two L-Shaped layouts for the edge e (P_5, P_6) are indicated using dotted lines.

III. ALGORITHM FOR CONSTRUCTION OF RSMT

The analysis looks into the existing algorithms with respect to two perspectives: the execution time, i.e. the time complexity of the algorithm and the quality of the solution obtained, i.e. percentage of accuracy of optimal solution. It is still an open issue for the researchers to get optimal solution with reduced execution time. The existing algorithms are broadly classified as approximation and exact algorithms. Under this classification the different variants/approaches are grouped together and further expand on the research activity related to the construction of RSMT with respect to the time complexity and concept behind the algorithm.

3.1 Approximation Algorithms

3.1.1 Spanning Tree Embedded Algorithms

Spanning tree embedded algorithms are based on the concept of first constructing a MST over the given set of points P and then applying a particular strategy to generate RSMT from it. Most of the approximation algorithms for computing RSMT are based on this concept and are explained as follows:-

Ho, Vijayan and Wong [5] proposed a linear time algorithm (L-RST and Z-RST) for finding RSMT from MST. To construct a RSMT, the edges of the separable MST (the tree is separable if the enclosing rectangle layouts of non-adjacent edges do not overlap) are replaced by the L or Z-shaped layouts which have maximum overlap. The time to compute the MST is not considered while finding time to construct RSMT. Khang and Robins [6] have shown with example that the solution generated by the algorithm sometimes does not satisfy the condition of "the ratio of length Rectilinear Minimum Spanning Tree (RMST) to the length of Rectilinear Steiner Minimum Tree is $\leq 3/2$[4]".

Hassan, Vijayan and Wong [7] proposed an O(nlogn) algorithm, where the neighborhood structure (the sub tree inside the dotted triangle is the neighborhood structure of point P_1 in fig. 3) of the maximal independent set of the constructed MST points are replaced by their corresponding optimal RSTs. The condition for independency is that the neighborhood structure of the points should not have edges in common.

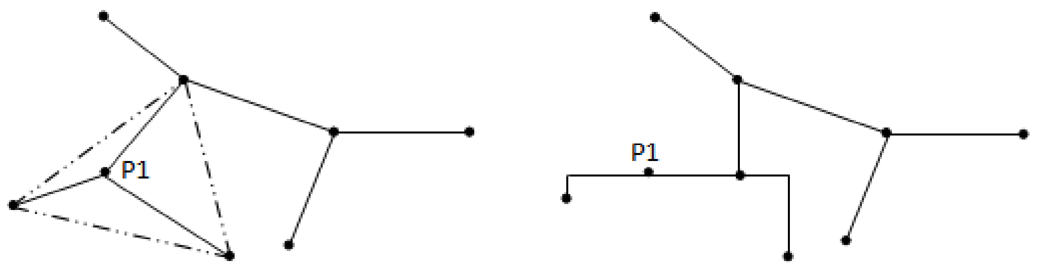

Fig. 3 Neighbourhood structure of point P1 and its RSMT

Khang and Robins [6] showed that all MST based algorithms have a worst case performance ratio of exactly 3/2. So, alternative RSMT approximation algorithms have been developed. Khang and Robins [8] presented a $O(n^3)$ Iterated 1-Steiner (I1S) algorithm which iteratively adds a Steiner point x and constructs a RMST over P U{x} until no more Steiner points which reduce the length can be found. Instead of adding a Steiner point iteratively, a batched variant (Batched Iterated 1-Steiner – BI1S) $O(n^4 \log n)$ algorithm adds a maximal independent set of Steiner points. The condition for independency is that a candidate Steiner point is not allowed to reduce the MST cost saving of another Steiner point.

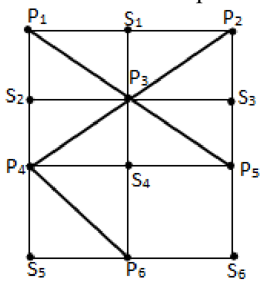

Fig. 4 MST over P and the Candidate Steiner Points (CSPs)

Chao and Hsu [9] proposed an $O(n^2 \log n)$ algorithm which incrementally adds Steiner points S to the point set P and generated the MST over {P}U{S}. The Steiner points are added in two stages. During first stage - local refinement, two types of local Steiner points (lSp – median of three points) are identified and included to the given point set by looking at the local structure of the tree. The essential leaf lSp is the unique lSp formed by a leaf node or point, its parent and the neighbors of the parent and if only one lSp is generated by the adjacent edges of the internal node or point, then lSp is called essential internal lSp. In fig 4, S_4 is the essential lSp for leaf P_6 and there is no essential leaf lSp for P_5. For the internal node P_4, S_4 is the essential internal lSp and there is no essential internal lSp for P_3. During second stage - global refinement, the point or the corner point of an edge(Candidate Steiner point –CSp) is connected to CSp of the another edge of the tree that has the shortest path as compared to CSp's of the other edges and the longest edge is removed in the formed loop i.e. the Steiner points are introduced by looking at the global structure of the tree.

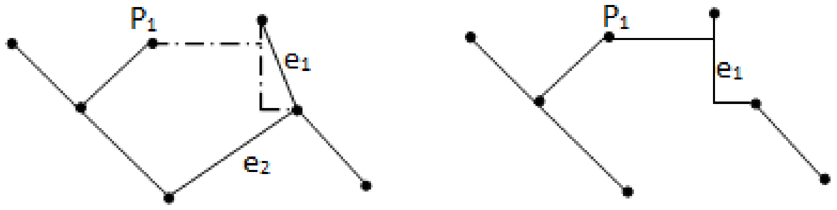

Fig. 5 Edge based heuristic

Borah, Owens and Irwin [10] proposed an $O(n^2)$ edge-based heuristic algorithm for RSMT construction, which starts by computing a MST, incrementally improves it by connecting a point to the enclosing rectangular layout of neighboring visible edge and removes the longest edge in the formed loop. The time complexity was reduced to $O(n \log n)$, but with complex programming strategies and data structures. In fig.5 point P_1 of the MST is connected to the enclosing rectangle layout of the visible edge e_1 and the edge e_2 is removed, as it is the longest edge in the formed loop.

Griffith, Jeff, et al [11] showed that the direct implementation of I1S requires $O(n^4 \log n)$ time. So, they proposed an algorithm for the implementation of BI1S based on dynamic MST update method with runtime of $O(n^3)$ per round. In dynamic MST update scheme, a new point is connected to each of the neighboring points in eight octants and the longest edge in the formed loop is removed. The parallel implementation using multiple processors was also done in near-optimal time.

Mandoiu [12] presented an algorithm similar to BI1S [8] where Steiner points are added based on bi-directed cut formulation [13] as compared to greedily in BI1S.

Areibi, Xie and Vannelli [14] used BI1S [8] as a basis. The main contribution was an algorithm for K-Rectilinear Steiner Tree (K-RST) where a set of Rectilinear Steiner Trees are generated for the given set of points by adding k Steiner points at a time. The gain of adding a Steiner point is computed by connecting it to the points in 4 quadrants and eliminating the cycles thus formed (dynamic update of spanning tree).

Khang, Mandoiu and Zelikovsky [15] proposed an $O(n \log^2 n)$ batched version of greedy triple contraction algorithm [16] - Batched Greedy Algorithm [BGA], where a batch of triples (triple is the optimal full Steiner tree for a set of three points where all the points are leaves) are added in each iteration to construct RSMT. Gain of adding a triple is computed by adding two edges for connecting the three points of a triple (addition of triple) and removing the longest edges in the formed loop (contraction). All the triples with positive gain are identified and added in a non-decreasing order of gain immediately followed by contraction. The Steiner points thus formed by the triple are added to the set of points P and MST over P is computed iteratively.

Zhou [17] proposed a O(nlogn) Rectilinear Spanning graph (RSG) algorithm based on Borah et al edge based heuristic [10] on the Zhou at al spanning graph algorithm [18] . A spanning graph is a graph that contains an embedded MST and is constructed by partitioning each point's plane into eight octal regions thereby connecting this point to the closest point in each region. Further a spanning tree is formed based on the cycle property by removing the longest edge in the formed loop. Edge based heuristic algorithm applied on the Spanning Tree thus formed generates RSMT.

Cinel and Bazlamacci [19] proposed a modified and distributed RSMT algorithm based on the idea of Khang et al Batched Greedy Algorithm (BGA) [15] and Zhou's Rectilinear Spanning graph (RSG) algorithm [17]. The modified algorithm uses RSG approach to generate initial sparse graph and to update the MST. BGA approach is used for longest edge computation. Distributed version is based on parallelization of major components of the modified RSMT.

3.1.2 *Two point and three point connection strategy based algorithms*

Two points connection strategy is based on an idea of connecting a new point to the constructed current tree by the L-shaped layout iteratively. In three point connection strategy, a new point is connected to the two points (terminal point or the Steiner point) of the current tree using optimal RSMT. Algorithms for computing RSMT based on these strategies are as follows:-

Lee, Bose and Hwang's [20] proposed an O(n^3) single-net algorithm that uses three point connection strategy and Hwang's [4] exact solution for three points to construct the RSMT. This algorithm starts by computing an optimal RSMT for every group of three elements and selects the one with minimum length. It proceeds by iteratively adding a new terminal point {x} that has the shortest distance to one of terminal point {y} in the current tree by an optimal RSMT of three points (x,y,z) where z is a Steiner or terminal point adjacent to y with shortest length. It was also shown that the RSMT constructed using this algorithm can actually be constructed in O(n^2) operations.

Hwang's[21] O(nlogn) suboptimal algorithm labels the given set of points first by applying a labeling algorithm (labeling helps in selecting the points in the order they are added to the MST), computes RSMT of first three points and then applies three point connection strategy such that the remaining points are iteratively added to form a RSMT.

An O(n^2) algorithm by Katsadas and Kinnen [22] starts with a RMST constructed from L1 Voronoi diagrams [23] instead of Prims algorithm. This algorithm uses the concept similar to the one in Prim's algorithm except that the point selected to be added to the current tree has shortest distance not only from terminal points in current tree but from the set of terminal points and Steiner points included in current tree. Thus the algorithm iteratively adds a point that has a shortest distance to one of the point (Steiner or terminal point) in the current constructed tree using one of the L-shaped layouts which is closer to the unconnected points.

Souza, Carvalho and Ribeiro [24] devised an O(n^2) algorithm (H2PC) based on two point connection strategy. The algorithm starts by connecting the two nearest points by a straight line or by one of the two L-shaped layout using path selection criterion based on the delta value followed by connecting all the points in the eight octants to this path. The delta value is the difference in lengths of the two solutions obtained by connecting the two points using two paths (upper and lower L-shape layout). Similarly all the points are connected to the closest point in the current tree iteratively. Further they proposed an improved approximation algorithm (HAS) which uses H2PC as a building block. H2PC is successively applied by including the Steiner points obtained by the previous application of H2PC to the points set P.

3.1.3 Look-Up Table based algorithms

Look-up table based algorithms compute the solution for the smaller value of n and stores it in the table. For larger value of n, the problem is recursively divided until the value of n is such that the table can be applied. Look-up table based RSMT algorithms are as follows:-

Chu and Wong [25] proposed a fast look-up table based O(nlogn) algorithm (FLUTE) for RSMT construction. Tables for computing RSMT and their length are pre-computed for $n \leq 9$. For $n>9$, optimal net breaking algorithm is used to break the net recursively until $n \leq 9$. An accuracy parameter is introduced to indicate the level of accuracy the problem is handled with and its value indicates the different ways of net breaking that are tried. Increasing accuracy parameter value produced better solution but with increased runtime. FLUTE produced near-optimal solution similar to BI1S for $n<30$.

To overcome the problem of FLUTE, Wong, Yiu-Chung and Chu [26] provided an improvement (FLUTE 3.0) by modifying the net breaking algorithm, which efficiently partitions a net into smaller subnets that can be efficiently handled by original FLUTE algorithm. Experimental results showed that the quality of FLUTE 3.0 improved with accuracy parameter value and produced better solution than BI1S but with increased runtime.

3.1.4 Genetic algorithms

Genetic algorithms are based on the concept of natural evolution. They solve the optimization problems by relating the problem to natural evolution and apply similar techniques such as inheritance, generation, mutation etc. Genetic algorithms for RSMT construction are as follows:-

Julstrom[27] proposed a genetic algorithm that converts the list of spanning tree edges into RSMT. This algorithm searches the product space of the space of the MST on given set of points with the space of the choice of Steiner points for each and every MST edge to identify the RSMT. Since search space grows with n value, the performance deteriorates for large problem instances.

To improve the performance, Julstrom[28] reduced the search space by first constructing RMST for the given set of points and then searched only the space of Steiner point choices for each of the spanning tree edges. The candidate RSMT is encoded as binary strings and the two genetic operators (two-point crossover and position-by-position mutation) are applied.

3.1.5 Probabilistic algorithms

Probabilistic algorithms are the randomized algorithms where the result obtained depends on the chance and the problem space is searched based on probabilistic model of candidate solutions to the problem. Probabilistic based RSMT algorithms are as follows:-

Komlos and Shing's [29] O(f(t)n + nlogn) probabilistic partitioning algorithm recursively partitions a given set of n points in unit square into 4^k small rectangles where k is [$\log_4$ n/t] and t>0. The algorithm proceeds by recursively partitioning the rectangle into two sub-rectangles with a vertical line segment based on the median x co-ordinate. Further each of the sub-rectangles is partitioned with a horizontal line segment based on the median y co-ordinate. For each of the partition the dynamic algorithm of Dreyfus and Wagner [35] (the algorithm considers all subsets of given set of points P in order of increasing size and for each subset applies the following decomposition theorem - the optimal Steiner tree for a set of points P can be decomposed into three subsets R, S and {v} such that for some u not in P , the optimal Steiner tree of R U {u} and S U {u} and the shortest distance between u and v is the optimal Steiner tree for P) is applied. The second algorithm has the reduced time complexity of O(g(t)n) by avoiding the O(nlogn) time spent in partitioning the given set of points.

Bern [31] proposed an algorithm similar to Kruskal's MST algorithm with following results i) Length of the generated RSMT is 29% shorter than MST. ii) At least .039n Steiner points are added in the optimal RSMT.

Hwang and Yao [32] applied the Bern's probabilistic result [31] on Exactly-N model with following results i) The length of generated RSMT is at least 0.0014n shorter than MST for large n. ii) At least .041n Steiner points are added in the optimal RSMT.

Chen, Zhao and Ahmadi [33] presented an algorithm based on probabilistic model. The probability of the patterns passing over individual edges is computed and only the edges with high probability are selected.

3.1.6 Other heuristic algorithms

Chung and Hwang [34] computed the length of RSMT with respect to the perimeter of the rectangle enclosing the given points for n≤ 10.

A heuristic presented by Beasley [35] considered all the connected sub graphs of MST which contains four vertices, finding the optimal Steiner trees for each of the sub graph.

Lim, Cheng and Wu [36] presented a new formulation of RST known as Performance Oriented RST which proceeds by identifying the edges and the direction for the edges to grow from source to destination.

Chen, Quio, Zhou and Cheng [37] proposed a more stable O(nlogn) Refined Single Trunk Tree (RST-T) algorithm, an improvement of a Steiner heuristic called Single Trunk Steiner Tree [38]. The points are either connected to the trunk (a line which goes horizontally or vertically through the median of the points) or the stem (edge that connects a point to the trunk), whichever is shorter (fig. 6).

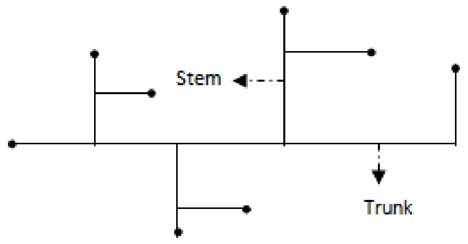

Fig. 6 Refined Single Trunk Tree

Elaheh, Kastner, and Sarrafzadeh [39] looked at the Steiner tree in a new perspective: flexibility. The algorithm takes a stable Steiner tree as an input and maps it into a more flexible Steiner tree which helps routing during congestion.

Hu et al [40] presented an O(n^2) algorithm based on ant colony optimization (ants are placed on the points and one ant is allowed to move leaving behind a trail for others to follow).

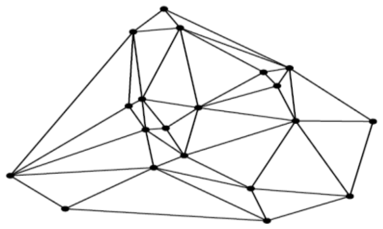
Fig. 7 Delaunay Triangulation

Wang et al [41] proposed an O(n^2logn) algorithm which first constructs a Bounded degree Neighborhood Graph (BNG)- a sub graph of Delaunay Triangulation(fig. 7), RMST over it and then applies polygon contraction to obtain RSMT.

3.1.7 Comparison of approximation algorithms

Spanning tree embedded algorithms are a good choice. BI1S is the champion with 11% improvement over the MST and only 0.5% away from the optimal. BGA produces output with less solution quality and runtime compared to BI1S. Edge-based heuristic gives solution similar to BI1S with better runtime but does scale well for large problem instances. RSG is based on edge-based heuristic with improved performance and runtime. Two point and three point connection strategy based algorithms generate RSMT with solution quality degrading for increasing n value. Probabilistic and genetic algorithms have high time complexity and hence not suitable for larger n value problem instances. Look-up table based algorithms have an accuracy parameter that allows the user to control the accuracy and the runtime. RST-T has the lowest runtime but with less solution quality.

3.2 Exact Algorithms

Exact algorithms are based on the Full Steiner Tree construction [42]. RSMT is called a Full Steiner Tree (FST) if all points are the leaves of the Tree. At every point with more than one degree, the tree can be split into edge-disjoint Full Steiner Trees that have in common only the split points. Those trees are called full components. Hwang [4] showed that the topologies of full components have only two restricted structure (fig 8) and every RSMT consisted of number of full components that intersect at points of degree 2 or more.

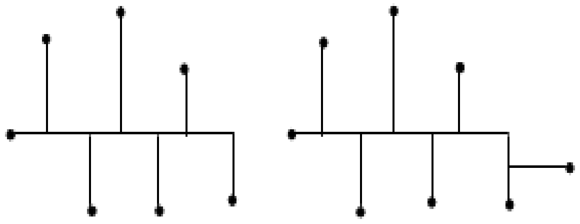
Fig. 8 Two topologies of Full component

The exact algorithms (Geo-Steiner[43, 44]) for RSMT work in two stages. In first stage, candidate full components are identified. In the second stage, subset of the identified full components whose union is a RSMT are selected and concatenated. The existing exact algorithms for RSMT construction are based on three approaches: backtrack searching, dynamic programming and integer programming.

Salowe and Warne [45] proposed an algorithm which finds the candidate full components that can be a part of RSMT by applying tests (coarse tests and set-specific tests), to judge whether the point set can be accepted or rejected as candidate full components and then backtrack search is used to obtain the subset of the full components from which FST can be extracted. Backtrack search starts with a single full component and recursively adds full component to the current solution until FST over a given set of points is generated. If adding a full component results in a cycle, then the search backtracks and adds some other full component.

Ganley and Cohoon[46,47] gave two dynamic programming algorithms. The first is Full-set Dynamic Programming(FDP) algorithm, which was based on the idea that optimal RSMT for a set of points P is either a FST and hence can be computed using Hwang's theorem[4] or else is composed of two smaller RSMT's A and B such that $|A\cup B|=|P|$ and $|A\cap B|=1$. The algorithm enumerates all possible subsets of P in order of increased cardinality with time complexity of O($n3^n$). The second algorithm- Screened Full-set Dynamic programming(SFDP) with time complexity O($n^2 2.62^n$) was based on the idea of full-set screening, where number of tests were conducted to reduce the number of full components that could be considered as candidate full components.

Fößmeier and Kaufmann [48] also used dynamic programming to combine the smaller number of full components and reduced the runtime to $O(n2.38^n)$ as compared to $O(n^2 2.62^n)$ by Ganley and Cohoon [47]. It was shown that for the given set of points P, there can be a RSMT containing only tree stars as full components and hence the number of full components to be considered was reduced. In a tree star the triangles defined by 45^o diagonals on both sides of each segment do not contain any other point of the given point set P.

Warme [49] used a branch and cut algorithm based on integer programming. This method was much faster than backtrack search or dynamic programming.

3.2.1 Comparison of Exact algorithms

Exact algorithm based on backtrack search has a very steep runtime growth and hence suitable only medium sized problem instances. Dynamic programming based exact algorithms produced better worst case bound but with a huge memory requirement. The exact algorithm based on integer programming is faster than the dynamic programming or backtrack search.

TABLE I
Comparison of Important RSMT Algorithms

Sl. No	Algorithm	Underlying Concept	Time Complexity	Advantages	Disadvantages
1	Batched Iterated Steiner Tree (BI1S)	Approximation algorithm, Spanning Tree embedded algorithm	$O(n^4 \log n)$	Produces near-optimal solution	High runtime
2	Batched Greedy Algorithm(BGA)	Approximation algorithm, Spanning Tree embedded algorithm	$O(n \log^2 n)$	Produces near-optimal solution, good runtime	Solution quality inferior to BI1S
3	Refined Single Trunk (RST-T) algorithm	Approximation algorithm, Other heuristic algorithm	$O(n \log n)$	Fast (good runtime)	Solution is far away from optimal for larger n
4	Rectilinear Spanning Graph Algorithm (RSG)	Approximation algorithm, Spanning Tree embedded algorithm	$O(n \log n)$	Fast (good runtime)	Solution is not optimal.
6	Edge-based heuristic	Approximation algorithm, Spanning Tree embedded algorithm	$O(n^2)$	Produces near-optimal solution	Does not scale well.
5	Fast Look-Up Table based algorithm(FLUTE)	Approximation algorithm, Look-up table based algorithm.	$O(n \log n)$	Fast, near-optimal solution for n<30	Solution is far away from optimal for n>30
6	GeoSteiner	Exact algorithm	-	Optimal solution	High runtime
7	FLUTE-3.0	Approximation algorithm, Look-up table based algorithm	$O(n \log^2 n)$	Fast and accurate	Increasing accuracy parameter value results in good solution but with increased runtime

IV. CONCLUSION

This paper presents a comprehensive analysis of various algorithms for computing the Rectilinear Steiner Minimum Tree. Table I provides the comparison of the important existing algorithms for the construction of RSMT. The selection of an algorithm is based on various parameters, the important ones being good solution quality (wire length) and runtime (time complexity). If good solution quality (reduced wire length) is the main aim GeoSteiner [43,44], BI1S[8], BGA[15], edge-based heuristic and RSG[17] are the good choice and if runtime is the main concern, then RST-T[37] and FLUTE[25] are the best options. FLUTE 3.0[26] produce a near-optimal solution by controlling the accuracy parameter. But, the decision regarding the selection of the algorithm for RSMT is a trade-off between efficiency and runtime. Therefore, there is a need for an algorithm or approach for the construction of RSMT to address the above challenge.

REFERENCES

[1] Sherwani, Naveed A. *Algorithms for VLSI Physical Design Automation*. Kluwer Academic Publishers, 1998.
[2] Hanan, Maurice, On Steiner's problem with rectilinear distance, *SIAM Journal on Applied Mathematics* 14.2 (1966): 255-265.
[3] Garey, Michael R., and David S. Johnson, The rectilinear Steiner tree problem is NP-complete, *SIAM Journal on Applied Mathematics* 32.4 (1977): 826-834.
[4] Hwang, Frank K, On Steiner minimal trees with rectilinear distance, *SIAM journal on Applied Mathematics* 30.1 (1976): 104-114.
[5] Ho, J-M., Gopalakrishnan Vijayan, and C. K. Wong, New algorithms for the rectilinear Steiner tree problem, *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 9.2 (1990): 185-193.
[6] Kahng, Andrew B., and Gabriel Robins, On the performance bounds for a class of rectilinear Steiner tree heuristics in arbitrary dimension, *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 11.11 (1992): 1462-1465.
[7] Hasan, N., G. Vijayan, and C. K. Wong, A neighborhood improvement algorithm for rectilinear Steiner trees, *Circuits and Systems, 1990., IEEE International Symposium on*. IEEE, 1990.
[8] Kahng, Andrew B., and Gabriel Robins, A new class of iterative Steiner tree heuristics with good performance, *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 11.7 (1992): 893-902.
[9] Ting-Hai, Chao, and Hsu Yu Chin., Rectilinear Steiner tree construction by local and global refinement, *Computer-Aided Design, 1990. ICCAD-90. Digest of Technical Papers., 1990 IEEE International Conference on*. IEEE, 1990.

[10] Borah, Manjit, Robert Michael Owens, and Mary Jane Irwin, An edge-based heuristic for Steiner routing, *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 13.12 (1994): 1563-1568.

[11] Griffith, Jeff, et al. "Closing the gap: Near-optimal steiner trees in polynomial time." *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 13.11 (1994): 1351-1365.

[12] Mandoiu, Ion I., Vijay V. Vazirani, and Joseph L. Ganley, A new heuristic for rectilinear Steiner trees, *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 19.10 (2000): 1129-1139.

[13] Rajagopalan, Sridhar, and Vijay V. Vazirani, On the bidirected cut relaxation for the metric Steiner tree problem, *Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 1999.

[14] Areibi, Shawki, Min Xie, and Anthony Vannelli, An efficient rectilinear Steiner tree algorithm for VLSI global routing., *Electrical and Computer Engineering, 2001. Canadian Conference on*. Vol. 2. IEEE, 2001.

[15] Kahng, Andrew B., Ion I. Mandoiu, and Alexander Z. Zelikovsky, Highly scalable algorithms for rectilinear and octilinear Steiner trees, *Design Automation Conference, 2003. Proceedings of the ASP-DAC 2003. Asia and South Pacific*. IEEE, 2003

[16] Zelikovsky, Alexander Z, An 11/6-approximation algorithm for the network Steiner problem, *Algorithmica* 9.5 (1993): 463-470.

[17] Zhou, Hai, Efficient Steiner tree construction based on spanning graphs, *Proceedings of the 2003 international symposium on Physical design*. ACM, 2003.

[18] Zhou, Hai, Narendra Shenoy, and William Nicholls, Efficient minimum spanning tree construction without Delaunay triangulation, *Proceedings of the 2001 Asia and South Pacific Design Automation Conference*. ACM, 2001.

[19] Cinel, Sertaç, and C. F. Bazlamacci, A Distributed Heuristic Algorithm for the Rectilinear Steiner Minimal Tree Problem, *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 27.11 (2008): 2083-2087.

[20] Lee, J., N. Bose, and F. Hwang, Use of Steiner's problem in suboptimal routing in rectilinear metric, *Circuits and Systems, IEEE Transactions on* 23.7 (1976): 470-476.

[21] Hwang, F, An O (n log n) algorithm for suboptimal rectilinear Steiner trees, *Circuits and Systems, IEEE Transactions on* 26.1 (1979): 75-77.

[22] Katsadas, Evagelos E., and Edwin Kinnen, A fast approximation to the rectilinear Steiner tree problem, *Circuits and Systems, 1990., Proceedings of the 33rd Midwest Symposium on*. IEEE, 1990.

[23] Hwang, F. K., An O (n log n) algorithm for rectilinear minimal spanning trees, *Journal of the ACM (JACM)* 26.2 (1979): 177-182.

[24] De Souza, Cid Carvalho, and Celso Carneiro Ribeiro, Heuristics for the minimum rectilinear Steiner tree problem: new algorithms and a computational study, *Discrete Applied Mathematics* 45.3 (1993): 205-220.

[25] Chu, Chris, and Yiu-Chung Wong, FLUTE: Fast lookup table based rectilinear steiner minimal tree algorithm for VLSI design, *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 27.1 (2008): 70-83.

[26] Wong, Yiu-Chung, and Chris Chu, A scalable and accurate rectilinear Steiner minimal tree algorithm, *VLSI Design, Automation and Test, 2008. VLSI-DAT 2008. IEEE International Symposium on*. IEEE, 2008.

[27] Julstrom, Bryant A, Encoding rectilinear Steiner trees as lists of edges, *Proceedings of the 2001 ACM symposium on Applied computing*. ACM, 2001.

[28] Julstrom, Bryant A, A scalable genetic algorithm for the rectilinear Steiner problem, *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*. Vol. 2. IEEE, 2002.

[29] Komlos, Janos, and M. T. Shing, Probabilistic partitioning algorithms for the rectilinear Steiner problem, *Networks* 15.4 (1985): 413-423.

[30] Dreyfus, Stuart E., and Robert A. Wagner, The Steiner problem in graphs, *Networks* 1.3 (1971): 195-207.

[31] Bern, Marshall W, Two probabilistic results on rectilinear Steiner trees, *Algorithmica* 3.1 (1988): 191-204.

[32] Hwang, Frank K., and Y. C. Yao, Comments on Bern's probabilistic results on rectilinear Steiner trees, *Algorithmica* 5.1 (1990): 591-598.0

[33] Chen, Chunhong, Jiang Zhao, and Majid Ahmadi, Probability-based approach to rectilinear Steiner tree problems, *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* 10.6 (2002): 836-843.

[34] Chung, F. R. K., and F. K. Hwang, The largest minimal rectilinear steiner trees for a set of n points enclosed in a rectangle with given perimeter, *Networks* 9.1 (1979): 19-36.

[35] Beasley, John E, A heuristic for Euclidean and rectilinear Steiner problems, *European Journal of Operational Research* 58.2 (1992): 284-292.

[36] Lim, Andrew, Siu-Wing Cheng, and Ching-Ting Wu, Performance oriented rectilinear Steiner trees, *Proceedings of the 30th international Design Automation Conference*. ACM, 1993.

[37] Chen, Hongyu, et al, Refined single trunk tree: a rectilinear steiner tree generator for interconnect prediction, *Proceedings of the 2002 international workshop on System-level interconnect prediction*. ACM, 2002.

[38] Soukup, Jiri, Circuit layout, *Proceedings of the IEEE* 69.10 (1981): 1281-1304.

[39] Bozorgzadeh, Elaheh, Ryan Kastner, and Majid Sarrafzadeh, Creating and exploiting flexibility in rectilinear steiner trees, *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 22.5 (2003): 605-615.

[40] Hu, Yu, et al, ACO-Steiner: Ant colony optimization based rectilinear Steiner minimal tree algorithm, *Journal of Computer Science and Technology* 21.1 (2006): 147-152.

[41] Wang, Yin, et al, The polygonal contraction heuristic for rectilinear Steiner tree construction, *Proceedings of the 2005 Asia and South Pacific Design Automation Conference*. ACM, 2005.

[42] Zachariasen, Martin, Rectilinear full Steiner tree generation, *Networks* 33.2 (1999): 125-143.

[43] Warme, David M., Pawel Winter, and Martin Zachariasen., Exact algorithms for plane Steiner tree problems: A computational study, *COMBINATORIAL OPTIMIZATION-DORDRECHT-* 6 (2000): 81-116.

[44] Warme, D. M., P. Winter, and M. Zacharisen. "GeoSteiner 3.1 package." *available at ftp. diku. dk/diku/users/martinz/geosteiner-3.1. tar. Gz.*

[45] Salowe, Jeffrey S., and David M. Warme, An exact rectilinear Steiner tree algorithm, Computer Design: VLSI in Computers and Processors, 1993. ICCD'93. Proceedings., 1993 IEEE International Conference on. IEEE, 1993.

[46] Ganley, Joseph L., and James P. Cohoon, A faster dynamic programming algorithm for exact rectilinear Steiner minimal trees, *VLSI, 1994. Design Automation of High Performance VLSI Systems. GLSV'94, Proceedings., Fourth Great Lakes Symposium on*. IEEE, 994.

[47] Ganley, Joseph L., and James P. Cohoon, Improved computation of optimal rectilinear Steiner minimal trees, *International Journal of Computational Geometry & Applications* 7.05 (1997): 457-472.

[48] Fößmeier, Ulrich, and Michael Kaufmann, On exact solutions for the rectilinear Steiner tree problem, *Proceedings of the thirteenth annual symposium on Computational geometry*. ACM, 1997.

[49] Warme, David M, A new exact algorithm for rectilinear steiner trees, *Network Design: Connectivity and Facilities Location* (1997): 357-395.