ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

*International Journal of Advanced Research in Computer and Communication Engineering*
*Vol. 4, Issue 4, April 2015*

# Design approaches for Obstacle Avoiding Rectilinear Steiner Minimum Tree: A comparative study

**Mamatha G**

Information Science and Engineering, J.S.S Academy of Technical Education, Bangalore.

Visveswaraiah Technological University, India.

**Abstract:** With the advances in the field of VLSI design, routing a net the basic function of Global routing and Detailed routing stages of Physical design are required to be conducted at the earliest. Many efficient algorithms are proposed to construct the Rectilinear Steiner Minimum Tree (RSMT) which routes the net and can be used in the estimation of wire-length and timing for Floor-planning and Placement stages of IC-design. RSMT connects the terminals of a net rectilinearly without considering the presence of obstacles in the routing region, but blockages like pre-routed nets, macro-cells, IP-blocks and others cannot be ignored as they form the major components of the routing area. This leads for the construction of an extended RSMT called Obstacle Avoiding Rectilinear Steiner Minimum Tree (OARSMT), many exact and heuristic algorithms are proposed for the construction of OARSMT based on approaches like Geo-Steiner, Look-up table, Extended-Hanan grid, Maze routing and Spanning graph. This paper discusses OARSMT algorithms that belongs to different approaches and makes a comparative study of the features and performance of OARSMT designs. Making way for the selection of the key features of the existing algorithms and to improve them in the new designs of OARSMT, to match with the current VLSI technologies.
**Keywords:** Rectilinear Steiner Tree, Obstacle Avoiding RST, Global Routing, VLSI design.

## I. INTRODUCTION

In VLSI design, construction of Rectilinear Steiner Minimum Tree (RSMT) plays a very important role, as it is used in Global Routing and Detailed Routing stages of the Integrated-Circuit Physical design. RSMT is also used for estimating - wire-length, timing and congestion during Floor-planning and Placement stages. Many efficient algorithms and approaches are designed for the construction of RSMT, but they have not considered blockages present in the routing region. In the modern IC designs rectilinear blockages like Macro cells, IP blocks, pre-routed nets, etc will be always present in the routing area. Therefore construction of RSMT in the presence of blockages becomes a more fundamental and practical problem and it is called as Obstacle Avoiding Rectilinear Steiner Minimum Tree (OARSMT). OARSMT problem connects a set of terminal pins using rectilinear edges avoiding rectilinear obstacles using a set of additional points called Steiner points on a 2-D plane. Fig.1 shows an ORSMT. Distance between two points is measured in Manhattan distance, given by d(p1, p2) = |x1- x2| + |y1-y2|. In [3] RSMT problem has been proved to be NP-complete, this implies that OARSMT

Problem cannot be solved in polynomial time and is expected to have exponential time in worst cases. Many Exact and Heuristic algorithms are designed to solve OARSMT; they are discussed under different design approaches in this paper.

Rest of the paper is organized as follows; Section II discusses various OARSMT designs under two major headings like Rectilinear Complete Graph (RCG) and Non-Rectilinear Complete Graph (NRCG).Section III, presents a table that summarizes the features and performance of OARSMT designs. Section IV, concludes the comparative study.

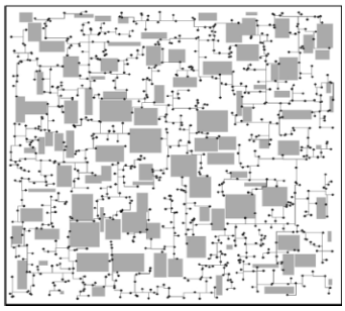

Fig.1 OARSMT connecting terminals avoiding Rectangular Obstacles. [19]

## II. OARSMT Design approaches

While constructing an OARSMT, the very initial step is to interconnect terminal pins and corner points of the obstacles to form a Complete Graph. OARSMT design approaches differ in the construction of the Complete Graph. Complete graph can be of two types, Rectilinear Complete Graph (RCG) and Non-Rectilinear Complete Graph (NRCG).

### 1. OARSMT design approaches based on Rectilinear Complete Graph (RCG)

Hanan grid [1] forms the basis for all types of Rectilinear Complete Graph whose edges are rectilinear in nature. Some of the RCGs are Escape graph, Track graph, Virtual graph, Grid-graph of Maze routing, etc.

*1.1 Escape Graph/ Extended Hanan-Grid*

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

*International Journal of Advanced Research in Computer and Communication Engineering*
*Vol. 4, Issue 4, April 2015*

It is a strong connection graph constructed by extending horizontal and vertical lines from terminal pins and obstacle boundaries as shown in fig.2 (a). Escape graph has $O(n^2)$ vertices in worst case and most of the edges and vertices are redundant. Escape graph guarantees a good OARSMT for a multi-terminal net. OARSMT designs based on Escape graph are discussed below.

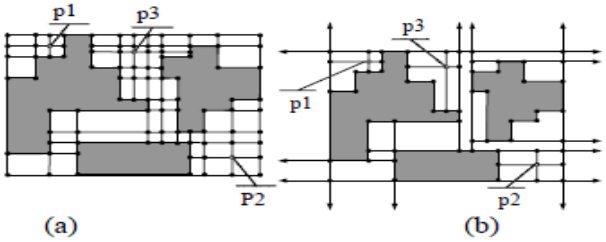


Fig.2 Rectilinear Connection Graphs (a) Escape graph (b) Track graph. [20]

[10] Presents a model that provides both exact and heuristic solution to route multi-terminal nets. Model transforms OARST, a Geometric problem into a Graph problem, whose size is a function of input size rather than the routing area, unlike Maze routing which optimally routes two terminal nets , but time and space usage corresponds to the size of the routing area. Exact algorithm based on Escape graph routes multi-terminal nets with three or four terminals, time required for generating Escape graph is O(max{n,mlogm}) where n is the number of escape segment intersections, m is the number of obstacle boundary segments. For nets with five or more terminals, K-Steinerization algorithm heuristically computes Obstacle Avoiding Steiner trees by replacing MST with K adjacent terminals with an optimal Steiner sub-tree. OARST problem is solved by G3S and G4S. Batching technique speeds up Steinerization heuristics. Worst case Steiner ratio (ratio of length of MST to length of optimal Steiner tree) of OARST is 2. Analysis and experiments show that algorithms work well in both theory and practice.

[17] Propose a two step Escape graph based O(mn) heuristic for multi-terminal tree construction. Step1 constructs a RSMT without considering the obstacles. Step2 with four processes transforms the primary tree into RSMTO. Edges of the primary tree overlapping with obstacles is removed, rectilinear short paths reconnect the edge vertices by going around the obstacles. Post processing improves the solution quality by removing cycles and U-shape paths. On an average 5.31%, redundant paths are generated. Algorithm has good wire-length and runtime is within 1sec for all cases.

*1.2 Track Graph*

Track graph is a variant of Escape graph, where rectilinear lines extend from terminals and Extreme edges of obstacles as shown in fig. 2(b). The number of vertices and edges in a track graph is $O(e^2)$, where e is the number of extreme edges of all obstacles. Track graph is not a strong connection graph, as optimal solution cannot be found in some cases. OARSMT design based on Track graph and ACO is discussed below.

An OARSMan[20] is a Track graph based non deterministic heuristic. Search space of Track graph is reduced by T-reduction method. ACO is used to interconnect the terminals on the track graph optimally, by placing ant on each terminal, a greedy metric Obstacle-Penalty distance estimate the distance between two terminals in the presence of obstacle. An OARSMan is better than FORst and gives good length performance for small scale cases.

*1.3 Geo-Steiner approach*

GeoSteiner package finds optimal solution for RSMT problem and is used as measuring Standard against which proposed methods are compared to determine how far-off they are from optimum. RSMTs are unions of Full Steiner Trees (FSTs) in which every pin-terminal is a leaf. FST forms the basis for Geo-Steiner framework; it considers subsets of pins at a time. FST generation and FST concatenation are two major steps of framework. FST generation step uses algorithm proposed in [14] to grow FSTs, which applies several optimality conditions to eliminate some FSTs that cannot be part of any RSMTs. Running time of this step, is quadratic and 4n FSTs are generated on average for random instances. [12] proposed that FST concatenation step is equivalent to MST problem on a Hyper-graph with the vertex set V of pins and subsets spanned by FSTs as hyper-edges; this can be formulated as an Integer Linear Program (ILP) and solved using Branch-and-cut search.

OARSMT designs based on Geo-Steiner framework, with suitable modifications done on two phases is discussed below.

[13] Presents an Obstacle-Avoiding Euclidean Steiner tree (OAEST) problem, the first exact algorithm. It uses a two phase framework, Generation and Concatenation of FSTs. FST Generation uses Equilateral point generation strategy, all possible FSTs are generated for Subsets of terminals, and shortest one is retained. Several tests are applied to prune away FSTs that cannot be in any optimal solution. Surviving FSTs are concatenated to obtain trees spanning all terminals. FST Concatenation is formulated as a Steiner Minimum Tree Problem of a Hyper-Graph (SPHG) with terminals and obstacle corners as vertices and Hyper-edges are equal to the length of corresponding FSTs. ILP solves the MST problem in Hyper-graphs. Results show that, moderate instances with up to 150 terminals are solved optimally within few hours of CPU-time.

[33] Extended Geo-Steiner approach to allow rectilinear blockages in routing area. FSTs are generated after four Virtual pins (at corners of blockage) are added to terminal set. Proofs on the structure and topologies of FST in [2] are extended to allow blockages as shown in fig.3. Potentially useful FSTs are generated by following methods of [14], FSTs to be part of the OARSMT pass the necessary conditions like, bottleneck Steiner distance, empty-diamond property ,empty corner rectangle property defined in[14]. In FST concatenation step, ILP select and concatenate subset of FSTs to construct an optimal OARSMT. Algorithm handles hundreds of pins with multiple blockages, generating an optimal solution in a reasonable amount of time. Results show that running time of the algorithm is combination of the FSTs

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

*International Journal of Advanced Research in Computer and Communication Engineering*
*Vol. 4, Issue 4, April 2015*

generation time and the time for solving ILP, which dominates the running time.

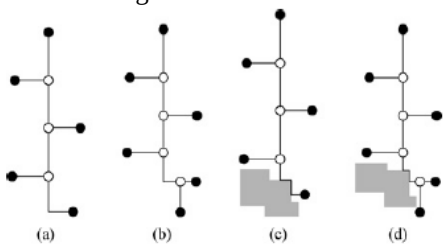


Fig.3 Structures of FSTs among obstacles. (a) Type I (b) Type II (c) Type III (d) Type IV structure. [41]

[37] Developed a two phase approach based on GeoSteiner. Four virtual terminals were added to each rectangular obstacle this makes structures of FSTs with obstacles same as FSTs without obstacles. This provides theoretical support for using GeoSteiner approach to generate OARSMTs. In FST generation phase, Pruning process reduces the number of virtual terminals by constructing a Reduced-Escape graph and reduces the number of FSTs. FSTs with exactly two terminals are constructed according to lemmas proposed by [16] and [4]. New IP formulation for the concatenation of FSTs with blockages is proposed and solved by branch-and-cut search, which includes Separation algorithm to adapt to the presence of virtual terminals, framework's lower bounds are provided by LP relaxation. Approach handles 100s of terminals and obstacles, generating optimal solution.

ObSteiner[41] an exact two phase algorithm, solves OARSMT problem among complex obstacles using Geometric approach. Optimal solutions are constructed by the concatenation of FSTs by the extended Geo-Steiner approach in the presence of obstacles. To simplify the structure of FSTs, Single Virtual terminal is added on Essential edge of each obstacle. Virtual graph is constructed from terminals, virtual points and boundaries of obstacles. It is a strong and smaller connection graph compared to Escape graph. OARSMT can be partitioned into a set of FSTs by splitting at terminals or virtual points with degree more than one. In the FST generation phase Pruning procedure with four tests are performed to eliminate FSTs that cannot be part of an optimal OARSMT. Survived FSTs are concatenated through ILP. ObSteiner adopts the Incremental construction approach to check for obstacles that overlap with the solution, the procedure repeats until no more obstacles overlap with final OARSMT. Instances with 100s of terminals and obstacles are solved optimally.

*1.4 Maze Routing/Lee's algorithm*

Maze-running one of the Shortest Path (SP) algorithms is characterized by target-directed grid propagation. Lee's algorithm an instance of Maze routing applies Dijkstra's breadth first shortest path search on uniform grid-graph. Its drawback is that it requires O(mn) memory and time for a grid of m x n and each node requires O(log L) bits, where L is the length of shortest path from source to target. Node labeling is the main operation of Maze-routing. Nodes of grid G are labeled from source until target node is labeled and then a path is extracted between them with node labels, as shown in fig.4. Maze routing optimally routes two-pin nets, its run time and space is proportional to the size of routing area rather than the actual problem size.

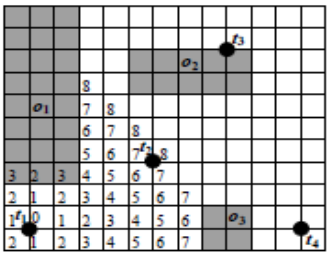


Fig.4 Node labeling/Wave propagation in Maze routing. [40]

OARSMT designs based on Maze Routing with suitable modifications are discussed below.

[11] Provides a framework for a class of algorithms that solve Shortest Path related problems like one-to-one, one-to-many and MST problem in the presence of Obstacles. A sparse strong connection graph forms the search space and its searched portion is constructed incrementally on-the fly using A* heuristic search. Algorithms time and space depends on actual search behavior and as it does not construct the entire connection graph explicitly, this fact reduces the cost of VLSI design. One-to-many SPs and MST problem takes O((e+n+N)log(e+n)) time and O(e+n+N) space where e, is number of sides of obstacles; n is number of points in S and N the total number of visited grid nodes.

[27] Propose a heuristic based on Maze routing that can handle- multi-pin nets, large scale cases and complex obstacles producing quality solution for run time and memory. Unlike the sequential traditional Maze routing, this method stores multiple paths between pins and then a MST method constructs final route globally. Propagation is made on a Simplified Hanan grid, heap data structure is used to implement maze routing step, post-processing further reduces wire-length. Results show that, on average an OARSMT with 2.01% less wirelength is generated and there is an improvement of 27.04% in wirelength compared with lower bound of optimal solution and shorter run time is achieved than [26].

[34] Presented a Modified Lee algorithm called as Weighted Lee algorithm that uses Extended Hanan grid to compute sub-optimal RSMTs and A new approach for reduction of the Routing area. Original Lee algorithm was applied to routing area of Unit-grids, Paper applies Lee algorithm to grids of varying lengths called as Weighted Hanan grid which solves situations of Concave boundary. Extended Hanan grid transforms the routing problem into a Graph problem, and the Weighted Hanan grid transforms the Computing scale from Routing area into the Input Size of terminals and obstacles. Minimum Convex Polygon MCP(V) of terminal set is designed to restrict Routing areas and to get rid of some parts of Hanan grid. Weighted Lee algorithm has three phases: labeling of vertices called as Filling or Wave propagation phase, Retrace phase and Updating phase. Algorithm runtime is $O(n^2(n+m)^2\log(n+m))$ , where n is number of terminals, m is number of vertices of obstacles and boundary.


DOI 10.17148/IJARCCE.2015.4473

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

*International Journal of Advanced Research in Computer and Communication Engineering*
*Vol. 4, Issue 4, April 2015*

*1.5 Path-based framework*

OARSMT problem solutions can be found by generating Critical paths without constructing the initial Routing graph which consumes large time and space. Path-based framework has global view of obstacles and provides potential ways to increase the overlapping between different paths.

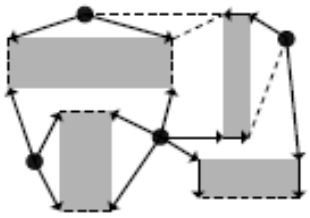

Fig.5 Critical paths connecting terminals and corners of obstacles. [31]

[31] Proposed a O(nlogn) four phase Path-based algorithm, which generates O(n) Critical paths, that guarantees the existence of desirable solution. First two phases generate an Obstacle Avoiding Steiner Tree (OAST) using critical paths without constructing a routing graph or generating invalid solution as shown in fig.5. Critical paths are generated by bridging edges between adjacent Shortest Path Map regions which takes the information from Multi-source Shortest Path trees constructed based on wave-front method. A greedy method constructs OAST out of critical paths and attempts to increase the overlap between paths to improve solution quality. Slant edge transformations and dynamic local refinements construct the final OARSMT.

*1.6 Steiner point based framework*

If OARSMT designs does not focus on the generation and usage of good Steiner point candidates OARSMT problem will become an Obstacle Avoiding Rectilinear Minimum Spanning Tree (OARMST) problem, which can be solved in polynomial time. Steiner point -based framework gives more importance for the usage of Steiner points, which causes the NP-completeness of the OARSMT problem. This framework devices method to generate Steiner point candidates efficiently. Framework can be extended to the practical generalizations of Multilayer OARSMT problem and Obstacle-Avoiding Preferred Direction Steiner tree (OAPD-ST) problem.

[32] Presents a four step algorithm based on Steiner-point based framework, which focuses more on the usage of Steiner points while handling obstacles. In step1, Routing graph- Obstacle-avoiding Voronoi graph (OAVG) is constructed out of Steiner point candidates, pin-vertices and obstacle corners, with only O(n) vertices and edges. Desirable Steiner point candidates are generated by the new concept of Steiner point Locations instead of line segment intersection methods (viz. Hanan grid, Escape graph). In step2, Based on Prim's concept, Steiner point selection method first sets the farthest pin-vertex as source and selects good Steiner points, Shortest Path Region (SPR), a subdivisions of plane forms the basis of Steiner point selection and it extends Dijkstra's algorithm to propagate vertices to find the current closest pin-vertex. In step3, Initial OARST is constructed from OAVG in O(nlogn) time by integrating MTST algorithm of [30] and path-overlapping scheme of [31]. In step4, Refinement scheme is applied on OARST to reduce the redundant segments in O(nlogn) time. Algorithm achieves best practical performance in both wire-length and run time, on average a speedup of 26.67 times is got and for the largest benchmark it takes only 1.565sec.

[39] Extended the Steiner-point based framework of [32]. Step1 constructed A routing graph called Geodesic Voronoi graph in O(nlogn) time and O(n) space, routing graph had multi-source Short Path Maps, the L-shaped Shortest Path Region of vertices located the Steiner point candidates. In step2, MTST constructed the OARSMT in O(nlogn) time. Final step used Liu's refinement method [31] to reduce wirelength. Time complexity of algorithm is O(mnlogn) in worst case and O(nlogmlogn) in average case, where m is the number of pin-vertices and n is the input size.

*1.7 Plane Sweep technique*

An xy-path monotone in both x and y direction with y-direction–preferred or x-direction–preferred can be used to find the Shortest Path (SP) between two points avoiding horizontal and vertical blockages in L1 metric.

RSP-RB [5] uses Plane-Sweep technique instead of Graph theoretic approach. Algorithm finds a SP between two points (source, target) avoiding rectangular barriers and the path is monotone in either x or y-direction. Time complexity is O (nlogn) where n, is number of barriers. A query form of the problem is also solved, given a source point and n- number of barriers, the shortest distance to query point avoiding barriers is found in O(t + log n) time, where t is the number of rectilinear turns in the path.

**2. OARSMT design approaches based on Non-Rectilinear Complete Graph (NRCG)**

Non-linear data structure, Graph G (V, E) forms the basis of NRCG. Here G is an undirected graph, the vertex set V is a Unoin of terminal pins and corner-points of obstacles and edge set E has edges formed by connecting terminal to terminal or terminal to obstacle corner points. Spanning Graph based approach has two variations – Sequential approach and Connected Graph approach

*Sequential approach* is a construction by correction method; it consists of two steps, step1, constructs MST without considering Obstacles. Step2, Transforms MST into a RSMT by substituting overlapping edges with edges around the obstacles. Industry uses this approach because of its simplicity. Drawbacks are, step1 does not have a global view of obstacles and paths so step2 removes overlaps locally, quality of solution and wire length is limited.

*Connected Graph based approach* first constructs a Connection graph from terminals and corner points of blockages as shown in fig. 6 which guarantees the presence of at least one OARSMT embedded in the graph. A Graph searching technique is then applied to find an optimal OARSMT. This approach has global view of both pins and obstacles and describes their geometrical relationship with few edges. Efficiency of the approach depends on size of the graph. OARSMT designs based on Spanning Graph and efficient Search techniques namely, Sweep-line algorithm, Ant Colony Optimization, Delaunay Triangulation and Look-up table; for improving

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

*International Journal of Advanced Research in Computer and Communication Engineering*
*Vol. 4, Issue 4, April 2015*

the performance of the designs at the early stages are discussed below.

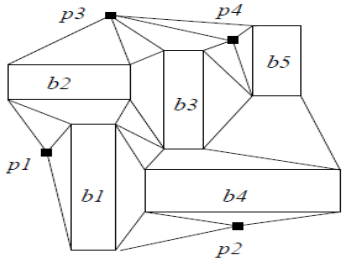


Fig.6 Complete Spanning graph interconnecting terminal pins and corner points of obstacles. [21]

*2.1 Sweep-line Algorithm*

Sweep-line algorithm [21] constructs a Spanning graph for vertex set V union of terminals and corner-points of blockages in O(nlogn) time. Algorithm makes only four passes over the search regions, shown in fig.7, each pass make edge connection in two regions simultaneously. Basic operation of algorithm is to maintain an active set of vertices in each pass.

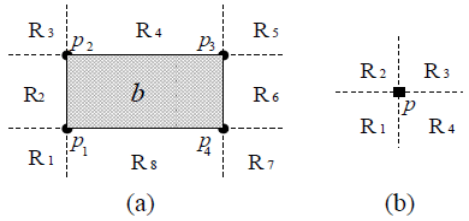


Fig.7 Search regions of (a) Rectangular blockage (b) Terminal pin. [21]

RSMTRB [21] is a six step heuristic, it proposes an efficient and effective Connection graph called Spanning graph that contains only O(n) vertices and edges, Spanning graph construction is done by a O(nlogn) Sweep-line algorithm followed by the rectilinearization of the MST to get the final RSMT with Rectilinear Blockages. Results show that RSMTRB has reduction of 12.081% wire-length compared to sequential approach with an increased runtime by only 48.44% on average.

[28] Proposed a four step heuristic based on Obstacle Avoiding Spanning graph (OASG). First, a OASG with O(nlgn) edges is constructed using Sweep-line algorithm of [21] but with more "essential edges". Second, step to constructs OAST is as follows, Dijkstra's algorithm is used for shortest path computation between edges of OASG, then Prim's algorithm constructs an initial OAST, then a local refinement is applied to remove unwanted cycles of OASG and to improve performance. Third, an OARST is constructed by transforming slant edges to rectilinear edges by applying some rules. Finally, OARSMT is constructed by U-shaped pattern refinement and the removal of overlapping edges and redundant vertices. Algorithm has Empirical run time of $O(n^{1.46})$, theoretical time complexity at worst case as $O(n^3)$ and random case as $O(n^2 lgn)$. The average wire-length improvement over [21] in OASG construction is about 3.69% and the overall improvement is about 5.79%. For large cases algorithm takes only 0.83s and achieves 4.46% improvements over [21].

EBOARST [30] is a four step algorithm. firstly, a sparse OASG is constructed in O(nlogn) time by a Sweep line algorithm which makes only a 45 degree sweep on the Quadrant partition of the plane, a balanced binary search tree data structure stores the active edges. Secondly, MTST is constructed by selecting edges from OASG and a shortest path terminal forest is obtained, then an extended Dijkstra–Kruskal algorithm solves MTST problem in O(nlogn) time. Thirdly, an OARST is constructed from MTST by an Edge-based heuristic in batched-mode and it also handles the global and local Steiner tree refinement. Finally, on the rectilinearized OARST further optimization is done by a local refinement technique called Segment translation. Time Complexity of EBOARST is O(nlogn) and it achieves 16.56 times speedup on average.

*2.2 Ant Colony Optimization (ACO)*

ACO method takes its idea from Ant colonies, which exhibits cooperative and social behavior. Ants communicate by secretion of Pheromone on the path. ACO achieves complex computations through multiple iterations; in each iteration one or more ants move leaving behind pheromone trail which evaporates at a constant rate. Tabu-list of ant has the record of visited vertices, when ant *A* meets ant *B*, *A* dies off giving its list to *B*. Designs for OARSMT which uses ACO, places ants on terminals [20] or on roots of sub-trees [25], which need to be interconnected. Ant *m* will choose to move on the Edge (vi, vj) which has a higher Pheromone Trail intensity and Desirability given by,

$$v_j = \max_{v_k}([\tau_{v_i,v_k}]^{\alpha} \cdot [\eta^m_{v_i,v_k}]^{\beta})$$

FORst [19] is a three step heuristic, Partitioning of terminals, Hyper graph construction, OARSMT construction for each sub-graphs by ACO-RSMT or Greedy FST-RSMT then detouring to connect all sub-graphs. FORst can handle both small and large scale cases with good performance.

Fast and Stable algorithm [25] is a four step heuristic, it constructs MSTs for the partitioned terminals, discards the edges overlapping with obstacles thus forming a Set of Sub-Trees (SST), Merges SSTs optimally using ACO by placing ants on roots of each sub-trees, a greedy method Rectilinearizes the tree edges and Refinement removes redundant edges to improve wire length. Results show that runtime is small for large cases and better than FORst, An OARSman, CDCtree.

*2.3 Delaunay Triangulation*

A Delaunay triangulation DT (P) for a set P of vertices in a plane is a triangulation such that no vertex in P is inside the Circum-circle of any triangle in DT (P) as shown in fig.8. In λ- Geometry, λ = 2, 3, 4, and ∞ it corresponds to Manhattan architecture, Y-architecture, X-architecture and Euclidean geometry. Academia and Industry are giving importance to λ- Geometry Routing (λ = 3, 4), as the total wire-length and crosstalk can be reduced drastically compared to Manhattan architecture (λ=2) thus improving the IC performance.


DOI 10.17148/IJARCCE.2015.4473

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

*International Journal of Advanced Research in Computer and Communication Engineering*
*Vol. 4, Issue 4, April 2015*

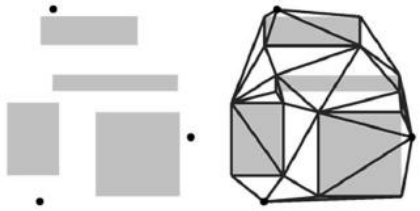

Fig.8 DT of terminals and corner points of rectangular Obstacles. [24]

λ- OAT [24] presents an O(nlogn) heuristic for λ-OARSMT construction in λ-geometry plane (λ=2), based on Obstacle-Avoiding Constrained Delaunay triangulation a fully connected tree is constructed which is then embedded into λ-OASMT by Zonal combination method (which unifies geometries). Compared with An-OARSMan [20] and FORst [19], λ-OAT speedsup to 30-Kx with quality solution.

*2.4 Look-up table based approach*

Lookup tables with predefined or pre-computed values have always made job easier and faster. FLUTE [29] constructs RSMT very quickly and accurately based on pre-computed lookup table. Runtime of FLUTE is O(nlogn) for a net of degree n. Degree-n nets can be partitioned into n! groups according to the relative positions of their pins. To find optimal RSMT for a low degree net, compute the wire-lengths corresponding to the POWVs for the group the net belongs to and then return a minimum wire-length POST associated with that POWV. POWV (Potentially Optimal Wire-length Vectors) corresponds to a linear combination of distances between adjacent pins. Few pre-computed POWVs of each group are stored in a table. Associated with each POWV, one corresponding Steiner tree is stored called as Potentially Optimal Steiner Tree (POST).

FOARS [36] is a Look-up table based five step heuristic, follows top-down approach and partitions the pins into subsets. OASG construction based on Octant partition of vertices guides the partitioning and captures the proximity of pins and corners of obstacles. Obstacle Penalized Minimum Spanning Tree (OPMT) is constructed by applying extended Dijkstra's and Kruskal's algorithm on OASG, in the case of major detour between vertices, an edge is penalized for the obstacle in its path in the form of weight on that edge. On the partitioned OPMT, Obstacle-Aware FLUTE constructs Obstacle Aware Steiner Tree (OAST) on the small degree nets satisfying Threshold value. Rectilinearization and Refinement generates OARSMT with an improved wire-length. OA-FLUTE uses OBTree data structure to accelerate the overlap checking with obstacles, this reduces runtime of FOARS by 59%. FOARS has better performance than [28] by 2.3% and [30] by 2.7%. FOARS runtime is slower compared with [31], [32].

## 3. Special cases of OARSMT design

Efforts were made to design OARSMT with Electrical and Electronic features and Genetic Algorithmic simulation. Designs showed good performance in comparison with the peer algorithmic designs, but their design complexity increases with increasing number of terminals and obstacles in the routing areas of ICs.

*3.1 Circuit based approach*

CDCTree [22] is a four step heuristic based on Current Driven Circuit model, circuit takes the topology of Escape graph, edges being replaced with resistors and terminals by current source. Algorithm makes use of Coulomb's law (repulsion of like charges), DC analysis is performed for current distribution on the circuit and edges with minimum current are selected to construct OARSMT with shorter wirelength. For test cases with 50 terminals CDCTree achieves shorter wirelength than An-OARSMan.

*3.2 Circuit simulation based approach*

cktSteiner[23] uses RC-network to model a Routing graph, here terminals are input-ports with unit impulse current source and hanan nodes are output-ports. Global Routing Graph (GRG) formally a unit tile hanan grid is mapped into RC mesh; edges are modeled as unit resistors, vertex of GRG is connected to ground via a unit capacitor and unit resistor in parallel. Impulse current are applied at terminals, a hanan node becomes a Steiner point if its voltage response reaches peak value at the earliest. In an iterative process, one or block of Steiner points can be added to build RSMT this give rise to two algorithms 1-cktsteiner, B-cktsteiner respectively. cktSteiner applies to both RSMT and OARSMT with a slight runtime difference. Circuit simulation is done on MATLAB; compared with An-OARSMan, 1-cktSteiner reduces 6.12% of wire-length for large test cases; B-cktSteiner gets average speedup of 352X with similar wire-length.

*3.3 Genetic algorithm (GA)*

Genetic algorithms give optimized solutions to problems based on natural concept of birth, inheritance, genetic-codes, chromosomes, mutation, etc.

[18] presents a Genetic Algorithm (GA) and compared its performance with a Greedy Heuristic.GA encodes candidate RSTs in terms of their underlying Spanning tree edges, each augmented with a Steiner point. A spanning tree of n points has n-1 edges which are taken as the length of a Chromosome. A Chromosome is a list of triples: two vertices describe a spanning tree edge and a Steiner point for the edge. Chromosome's fitness is the length of the RST it represents. GA operators like – Crossover and Mutation are constrained to generate only valid RSTs whose vertical and horizontal segments do not intersect obstacle. GA operator Crossover generates one offspring from two parents. GA operator Mutation modifies a single parent chromosome; it removes a random edge pair from the parent, and then selects a new edge pair and Steiner point at random to reconnect the tree. A variation of Kruskal's algorithm generates random RST for the GA's initial population. Greedy heuristic for RST with obstacles imitates Kruskal's algorithm. Results shows that GA consistently generated shorter RST than the Greedy heuristic for tests on 45 instances of up to 469 pins and 325 obstacles.

*3.4 Parallel techniques*

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

*International Journal of Advanced Research in Computer and Communication Engineering*
*Vol. 4, Issue 4, April 2015*

Many algorithms proposed to solve the OARSMT problem are Sequential, rather than Parallel. Multi-core processors and Computer systems are widely available and inexpensive. Developing parallel algorithms allows the exploitation of the computational power of Shared-memory multi-core systems or Array processors (group of processing elements).Considerable speed-up can be achieved while executing the parallel program which uses threads and OpenMP functions.

[9] Proposed, parallel techniques for computing rectilinear shortest paths avoiding obstacles. CREWPRAM model processors computed shortest paths in $O(\log^2 n)$ time for all the three cases. Case1, when source and destination pins are on the boundary of convex obstacle polygon, then the model requires $O(n^2/\log^2 n)$ processors. Case2, when source is an obstacle vertex and destination is a vertex pin, then the model requires $O(n^2/\log n)$ processors, and case3, if both source and destination are obstacle vertices model takes $O(n^2)$ processors. Noticed that Single processor obtains the path length between query points in a constant time, while O (n/log n) processors retrieve the shortest path in logarithmic time. If the query points are arbitrary, then single processor takes O(logn) time. Parallel technique use Staircase separators or convex paths for fast computations.

[40] Proposed a parallel algorithm for constructing OARSMT on a gridded xy-plane based on Watanabe's Steiner tree construction algorithm [7] suitable for use on a shared-memory multi-core system. Algorithm is based on Maze routing and a two phase repetitive double front-wave expansion. Algorithm uses two parallelized procedures, which efficiently reduces the program execution time, namely PARALLEL-CONNECT() used to connect a Steiner point with other points within a region and PARALLEL-CLEANUP() a process of resetting distance numbers of grid points in maze routing. Algorithm is implemented in C++ with the use of OpenMP functions. Program achieves 23% speed-up on average while running on a multi-core workstation and generates very short wires.

## III. Comparison of OARSMT design approaches

Table gives the comparison of key features and performance parameters like runtime and wire-length estimations of different OARSMT designs.

| **Algorithm and its Underlying concepts** | **Performance and Time complexity** | **Salient features of OARSMT designs** |
|---|---|---|
| [10] Exact algorithm. Multi-terminal net, Escape graph, Greedy k-Steinerization | Escape graph generation in O(max {n, m log m }), G3S heuristic time complexity is $O(k^2n)$, G4S is $O(k^3n^2)$. Worst case Steiner ratio is 2 | Transforms geometric problem into graph problem of input size. Optimally routes multi-terminal net by exact and heuristically by GkS method. Batching technique speeds up Steinerization heuristics. |
| FORst [19] -Partitioning of terminals, Hyper graph of terminals and FSTs, ACO and Greedy FST for routing subtrees, Detour technique | max ($O(n^3)$,$O(n^2$ e log (e))) n is number of terminals; e is number of edges of obstacles. | Handles large scale cases and concave polygonal obstacles with a short running time.<br>Two routing algorithms ACO-RSMT and GFST-RSMT works well on small and large scale cases. Their cooperation is a trade-off between CPU-time and solution quality. |
| An OARSMan [20]<br>-Track Graph, ACO search heuristics, greedy OP-distance metric | Optimal solution for instances less than 7 terminals. cases with 100 terminals take 30sec. | Handles complex obstacles. Track graph has reduced search space. Good length performance for small cases. |
| CDCTree[22] -Current Driven Circuit, Escape graph, Coulomb's law, Kirchoff's law | Major time is spent in solving linear equations. | Efficient for routing nets with less than 50 terminals and Not suitable for Large cases. Practical for Routing applications. |
| λ-OAT [24] -Obstacle-Avoiding Constrained Delaunay triangulation; Zonal Combination | O(nlogn), n is sum of terminals and corner points of obstacles. | Global search by λ-OAT provide better solution for Large cases than FORst and Speeds up to 30Kx.<br>Longer wire-length for Small cases with few obstacles as edge sharing is limited. |
| Fast and Stable algorithm [25] -Partitioning of terminals, Set of Sub-Trees, ACO, Rectilinearization and Refinement. | runtime is small for large cases, compared to λ-OAT algorithm takes 4.2 sec and 54.37% less wire-length | Greedy Rectilinearization shares edges; Refinement improves wire-length by eliminating U-shape paths. Good solution quality makes it suitable for routing process. But for large cases Spanning graph loses information. |
| [27] -Multi-pin variant of Maze routing, maze propagation uses simplified Hanan grid, maze routing step uses Heap data structure. | Improved wirelength and running time than basic Maze routing | Handles multi-pin net, complex obstacles, large scale cases taking less runtime and storage disproves the drawbacks of traditional Maze routing. Post-processing improves wirelength but Large cases increases Maze searching space. |
| [28] -Obstacle Avoiding Spanning graph with "essential edges", Dijkstra's & Prim's algorithm, local & U-shape pattern refinement. | OASG has O(nlgn) edges. Empirical run time is $O(n^{1.46})$. Theoretical time complexity is $O(n^3)$ in worst case, $O(n^2$lgn) in random case. | "essential" edges in OASG guarantees to find an optimal OARSMT for two-pin and higher-pin nets;<br>OASG with less number of edges ensures efficient searching and processing; local and U-shaped pattern Refinement schemes- reduce total wire-length. |
| EBOARST[30] -Edge-based heuristics, Sweep line algorithm, Quadrant partition of plane, terminal forest, extended Dijkstra–Kruskal, | O(nlogn) Algorithm achieves 16.56 time speedup and only 0.46% longer on average. OASG and MTST are O(nlogn) | 45degree sweep by Sweep line algorithm.<br>Edge-based heuristic enables both local and global refinements so small length Steiner trees is generated.<br>Segment translation refinement enhances the quality of |

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

*International Journal of Advanced Research in Computer and Communication Engineering*
*Vol. 4, Issue 4, April 2015*

| | | |
|---|---|---|
| Edge substitution. | | OARST |
| [31] Path-based algorithm. -Critical paths, Shortest Path Maps, Multi-Source SPs, Wave-front method | O(n) critical paths in O(n log n) O(nlogn) algorithm achieves best speedup and 1.1% longer on average | Critical paths guarantee the existence of optimal solutions. Overlapping of paths and refinement schemes improves solution quality. |
| [32] [39] Steiner point based framework. -Voronoi graph, Shortest Path Region (SPR), Steiner point location method. | OAVG of O(n) vertices and edges in O(nlogn) time. Best solution quality in Θ(nlogn) empirical time. | Steiner point location method generates desirable Steiner point candidates that cause NP-completeness of OARSMT problem. OAVG integrates the effectiveness and the efficiency of routing graphs. |
| [34] Weighted Lee algorithm. -Lee algorithm, extended Hanan grid, Minimum Convex Polygon (MCP). | MCP(V) in O(nlogn), algorithm runtime is O ($n^2$(n+m)$^2$ log (n+m)), n terminals, m is number of vertices of obstacles & boundary. | Routing area is a Weighted Hanan grid. MCP(V) restricts routing area. Maze/Lee algorithm considers boundary components. Large cases increase complexities. |
| FOARS [36] Fast Look-up table, Octant partition, Sweep-line algorithm, Dijkstra's, Kruskal's algorithms, Obstacle aware-FLUTE, OBTree data structure. | OASG has O(n) edges; FOARS has similar wirelength as [27][31] [32]. FOARS runtime is 84% faster than [30] on average and 46 times and 123 times faster than [27], [28]. | OARSMT follow obstacle boundary on necessity and avoids congestion while routing large nets. At top-level OASG guide partitioning. FLUTE is recursively called for local optimization of OAST. Refinement reduces wire-length by 1 to 2%. Obstacle Aware-FLUTE is less effective for high degree nets and dense obstacle region. |
| [37] Geo-Steiner approach. FST generation and FST Concatenation, four Virtual points, Reduced Escape graph, ILP formulation. | For large cases, FST concatenation phase dominates the total run time. On average, 29.3% reduction of FSTs achieved by pruning process in 1.7s | for small cases -running time of FST generation is Comparable to FST concatenation phase. Pruning process is effective for small nets. Lower bound by LP relaxation is very tight and Separation procedure efficiently finds ILP violated constraints. |
| [40] Parallel algorithm. Maze routing, Double front-wave expansion, Shared-memory multi-core system, OpenMP | On average 23% speed-up achieved when parallel program executed using four threads on Multi-core | Parallel algorithm exploits the computational power of shared-memory multi-core systems. Minimizes total wire length. Does not use Refinement techniques |
| ObSteiner [41] Exact algorithm. -Geosteiner approach, Essential edge with one virtual point, Virtual graph, Pruning procedure, Incremental approach, ILP formulation. | Total run time is dependent on the number of obstacles, more obstacles lead to more iterations of algorithm. Instances with less than 500 Obstacles were solved in minutes. | Handles complex obstacles. Pruning procedure eliminates useless FSTs leading to significant improvement in total run time. ILP constraints on FST and virtual point minimize the total wire length. For small benchmarks, the benefit of using Pruning procedure and Incremental approach is limited |

## IV. CONCLUSION

Paper discusses OARSMT design approaches and makes a comparative study of the features, wire-length and runtime performance of OARSMT algorithms. Some of the interesting findings of the study are, OARSMan[20],[25]based on ACO show good time and wire-length performance for small scale cases, similarly FORst[19],λ-OAT[24] for large scale cases. ObSteiner[41] and FOARS[36] based on the extensions of popular RSMT methods Geo-Steiner and FLUTE gives quality solutions and handles large cases. [27] gave a multi-terminal variant of Maze routing which has good space and time responses. [31] Path-based algorithm, a unique method that avoids construction of routing graph by directly generating critical paths. Steiner point selection based algorithm [32] gives more importance for the generation of the Steiner point candidates while handling obstacles. Exact algorithms [38] [41] based on Geo-Steiner handles complex obstacles without dissecting them and provide quality solutions. Parallel algorithms [40] are proposed to solve OARSMT problems at higher speed on Multi-core parallel systems. As there is continuous advancement in the VLSI technology, there is a need for the best performing OARSMT design; this can be achieved by suitably integrating or improving the key features of the existing algorithms or designs.

## REFERENCES

[1] M. Hanan, "On Steiner's problem with rectilinear distance," *SIAM J. Appl.Math.*, vol. 14, no. 2, pp. 255–265, Mar. 1966.
[2] F. K. Hwang, "On Steiner minimal trees with rectilinear distance," *SIAM J. Appl. Math*., vol. 30, no. 1, pp. 104–114, Jan. 1976.
[3] M. R. Garey and D. S. Johnson, "The rectilinear Steiner tree problem is NP-complete," *SIAM J. Appl. Math*., vol. 32, no. 4, Jun. 1977.
[4] A. C. C. Yao, "On constructing minimum spanning trees in k dimensional spaces and related problems," *SIAM J. Comput.*, vol. 11, no. 4, pp. 721–736, 1982.
[5] P. J. Rezende, D. T. Lee, and Y.F. Wu, "Rectilinear shortest paths with rectangular barriers," in *Proc. Second Annual Conf Computat. Geom.*, pp. 204-13, ACM, 1985.
[6] Y.F. Wu, P. Widmayer, M. D. F. Schlag, and C. K. Wong, "Rectilinear shortest paths and minimum spanning trees in the presence of rectilinear obstacles," *ZEEE Trans. Comput.*, vol. '2-36, pp. 321-31, 1987.
[7] Takumi Watanabe, Hitoshi Kitazawa, and Yoshi Sugiyama, "A Parallel Adaptable Routing Algorithm and Its Implementation on a Two-Dimensional Array Processor," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 6, no. 2, pp. 241-250, 1987.
[8] J. S. B. Mitchell, "An optimal algorithm for shortest rectilinear paths among obstacles in the plane," in *Abstracts First Canadian Conf. Computational Geometry*, 1989, p. 22.
[9] Mikhail J.Atallah,D. Z.Chen,"Parallel Rectilinear Shortest Paths with Rectangular Obstacles", ACM 1990
[10] 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.
[11] S.Q.Zheng, J.S.Lim, and S.S.Iyengar, "Finding obstacle-avoiding shortest paths using implicit connection graphs", *IEEE Trans on CAD*, 1996, 15(1): pp. 103-110.
[12] D.M. Warme, "A New Exact Algorithm for Rectilinear Steiner Minimal Trees", Technical Report, System Simulation Solutions, Inc., Alexandria, VA 22314, USA, 1997.
[13] M.Zachariasen and P.Winter, "Obstacle-avoiding Euclidean Steiner trees in the plane: an exact algorithm", Extended abstract presented at Workshop, ALENEX, 1999.

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

*International Journal of Advanced Research in Computer and Communication Engineering*
*Vol. 4, Issue 4, April 2015*

[14] M. Zachariasen, "Rectilinear Full Steiner Tree Generation", *Networks*, Vol.33, pp.125-143, 1999.
[15] D W.Warme, P.Winter, Zachariasen,"Geosteiner 3.1 package." 2000
[16] U. F¨oßmeier and M. Kaufmann, "On exact solutions for the rectilinear Steiner tree problem part I: Theoretical results," *Algorithmica*, vol. 26, no. 1, pp. 68–99, 2000.
[17] Y.Yang, Q.Zhu, T.Jing, X.L.Hong, and Y.Wang, "Rectilinear Steiner Minimal Tree among Obstacles", in *Proc. of IEEE ASICON*, Beijing, China, 2003: pp. 348-351.
[18] Rita M. Hare, Bryant A. Julstrom, "A Spanning-Tree-Based Genetic Algorithm for some Instances of the Rectilinear Steiner problem with Obstacles", ACM 2003
[19] Y. Hu, Z. Feng, T. Jing, X. L. Hong, Y. Yang, G. Yu, X. D. Hu, and G. Y. Yan, "FORst: A 3-Step heuristic for obstacle-avoiding rectilinear Steiner minimal tree construction," *J. Inf. Comput. Sci*., vol. 1, no. 3, pp. 107–116, Dec. 2004.
[20] Y. Hu, T. Jing, X. L. Hong, Z. Feng, X. D. Hu, and G. Y. Yan, "An-OARSMan: Obstacle-avoiding routing tree construction with good length performance," in *Proc. ASP-DAC*, 2005, pp. 7–12.
[21] Z. Shen, C. Chu, and Y. Li, "Efficient rectilinear Steiner tree construction with rectilinear blockages," in Proc. *Int. Conf. Comput. Des*., 2005, pp. 38–44.
[22] Y. Shi, T. Jing, L. He, and Z. Feng, "CDCTree: novel obstacle-avoiding routing tree construction based on current driven circuit model", *Proc. ASP-DAC*, pp. 630-635, 2006.
[23] Yiyu Shi, Paul Mesa, Hao Yu and Lei He, "Circuit Simulation Based Obstacle-Aware Steiner Routing", DAC 2006.
[24] Tom Tong Jing, Zhe Feng, Yu Hu, Xianlong L. Hong, Xiaodong D. Hu, and Guiying Y. Yan, "λ-OAT: λ-Geometry Obstacle-Avoiding Tree Construction With O(n log n) Complexity", *IEEE Transactions On Computer-Aided Design Of Integrated Circuits and Systems*, Vol. 26, No. 11, November 2007.
[25] P.C. Wu and J.R. Gao and T.C. Wang, "A Fast and Stable Algorithm for Obstacle-avoiding Rectilinear Steiner Minimal Tree Construction", *Proceedings ASP-DAC*, pp.262-267, 2007.
[26] C.W. Lin and S.Y. Chen and C.F. Li and Y.W. Chang and C.L. Yang, "Efficient Obstacle-avoiding Rectilinear Steiner Tree Construction", *Proceedings ISPD*, 2007.
[27] L. Li and E. F. Y. Young, "Obstacle-avoiding rectilinear Steiner tree construction," in *Proc. Int. Conf. Comput.-Aided Design*., 2008.
[28] C.W. Lin, S.Y.Chen, C.F. Li, Y.W. Chang and C.L.Yang,"Obstacle-avoiding rectilinear Steiner Tree construction based on spanning graphs,"*IEEE Trans.CAD*,Vol.27,No.4, pp.643–653, 2008
[29] C. Chu and Y.C. Wong, "FLUTE: Fast lookup table based rectilinear Steiner minimal tree algorithm for VLSI design," *IEEE Trans. CADesign Integr. Circuits Syst.*, vol. 27, no.1, pp.70–83, Jan. 2008.
[30] J. Y. Long, H. Zhou, and S. O. Memik, "EBOARST: An efficient edge based obstacle-avoiding rectilinear Steiner tree construction algorithm", *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 12, pp. 2169–2182, Dec. 2008.
[31] C.H. Liu, S.Y. Yuan, S.Y. Kuo, and Y.H. Chou, "An O (n log n) path based obstacle-avoiding algorithm for rectilinear Steiner tree construction," in *Proc. DAC*, 2009, pp. 314–319.
[32] C.H. Liu, S.Y. Yuan, S.Y. Kuo, and J.H. Weng, "Obstacle-avoiding rectilinear Steiner tree construction based on Steiner point selection," in *Proc. ICCAD*, 2009, pp. 26–32.
[33] L. Li, Z. Qian, and E. F. Y. Young, "Generation of optimal obstacle avoiding rectilinear Steiner minimum tree," in *Proc. ICCAD*, 2009
[34] Xueliang Li and Yang Luo,"Weighted Lee Algorithm on Rectilinear Steiner Tree with Obstacles and Boundary", IEEE 2009.
[35] T. Huang and Evangeline F. Y. Young, "Obstacle-avoiding Rectilinear Steiner Minimum Tree Construction: An Optimal Approach," in *Proc. Int. Conf. CAD*., pp. 610–613, 2010.
[36] Gaurav Ajwani, Chris Chu, and Mak Wai-Kei, "FOARS: FLUTE Based Obstacle-Avoiding Rectilinear Steiner Tree Construction," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 2, pp. 194-204, Feb 2011.
[37] T. Huang, L. Li, and Evangeline F. Y. Young, "On the construction of optimal obstacle-avoiding rectilinear Steiner minimum trees," *IEEE Trans. On Comput.-Aided Des. Integr. Circuits Syst.*, vol. 30, no. 5, pp. 718–731, 2011.
[38] T. Huang and Evangeline F. Y. Young, "An Exact Algorithm for the Construction of Rectilinear Steiner Minimum Trees Among Complex Obstacles," in *Proc. Design Automation Conf.*, pp. 164–169, 2011.
[39] Chih-Hung Liu, Sy-Yen Kuo, D. T. Lee, Chun-Syun Lin, Jung-Hung Weng, and Shih-Yi Yuan ,"Obstacle-Avoiding Rectilinear Steiner Tree Construction: A Steiner-Point-Based Algorithm", *IEEE Transactions On CAD Of Integrated Circuits and Systems*, July 2012.
[40] Cheng-Yuan Chang and I-Lun Tseng, "A Parallel Algorithm for Constructing Obstacle-Avoiding Rectilinear Steiner Minimal Trees on Multi-Core Systems", International Conference PDPTA, 2012.
[41] Tao Huang and Evangeline F. Y. Young, "ObSteiner: An Exact Algorithm for the Construction of Rectilinear Steiner Minimum Trees in the Presence of Complex Rectilinear Obstacles", *IEEE Transactions On Computer-Aided Design Of Integrated Circuits and Systems*. Vol. 32, No. 6. June 2013.

## BIOGRAPHY

**Mamatha.G** is presently working as Asst. Prof. in the Dept. of ISE, JSSATE, Bangalore. She obtained her B.E (CSE) from Bangalore University, M.Tech (CNE) from VTU, Belgaum and pursuing her PhD (CSE) under VTU. She has 15yrs of academic experience. She is a member of CSI, LM-ISTE. Her interests include Designing algorithms for routing nets in VLSI physical design, Algorithm implementation using Reconfigurable Logic, UNIX system programming and Automata Theory.