Received February 28, 2020, accepted March 29, 2020, date of publication April 6, 2020, date of current version April 22, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2986138

A Survey on Steiner Tree Construction and Global Routing for VLSI Design

HAO TANG[1], GENGGENG LIU[1], (Member, IEEE), XIAOHUA CHEN[1], AND NAIXUE XIONG[2], (Senior Member, IEEE)
[1]College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350116, China
[2]College of Intelligence and Computing, Tianjin University, Tianjin 300350, China

Corresponding author: Genggeng Liu (liugenggeng@fzu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61877010 and Grant 11501114, and in part by the Natural Science Foundation of Fujian Province, China, under Grant 2019J01243.

ABSTRACT Global Routing (GR) is a crucial and complex stage in the Very Large-Scale Integration (VLSI) design, which minimizes interconnect wirelength and delay to optimize the overall chip performance. Steiner tree construction is one of the basic models of VLSI physical design, which is usually used in the initial topology creation for noncritical nets in physical design. In a GR process, a Steiner Minimum Tree (SMT) algorithm can be invoked millions of times, which means that SMT algorithm has great significance for the final quality of GR. Some of the research works are surveyed in this paper to understand GR and SMT problems and to learn the available solutions. Firstly, we systematically dissect three types of subproblems in Steiner tree construction and three types of GR methods. Then, we investigate the recent progress under two new technology models. Finally, the survey concludes with a summary of possible future research directions.

INDEX TERMS Steiner tree construction, particle swarm optimization, routing, very large scale integration, global routing.

LIST OF ABBREVIATION

ACO	Ant Colony Optimization
AMR	Adaptive Maze Routing
BFS	Breadth First Search
BGA	Batched Greedy Algorithm
BLMR	Bounded-Length Maze Routing
CAD	Computer-Aided Design
CMOS	Complementary Metal Oxide Semiconductor
DFS	Depth First Search
DR	Detailed Routing
DT	Delaunay Triangulation
EDA	Electronic Design Automation
FLUTE	Fast Lookup Table Estimation
FST	Full Steiner Tree
GR	Global Routing
HTS	Hybrid Transformation Strategy
IC	Integrated Circuit
ILP	Integer Linear Programming
LRSMT	Length-Restricted Steiner Minimum Tree
MAD	Modified Algorithm Dijkstra
MDSV	Multiple Dynamic Supply Voltage
MST	Minimum Spanning Tree
MSV	Multiple Supply Voltage
NDT	Non-Delaunay Triangulation
OAFST	Obstacle-Avoiding Full Steiner Tree
OARG	Obstacle Avoidance Routing Graph
OARSMT	Obstacle-Avoiding Rectilinear Steiner Minimum Tree
OARST	Obstacle-Avoiding Rectilinear Steiner Tree
OASG	Obstacle-Avoiding Spanning Graph
OASMT	Obstacle-Avoiding Steiner Minimum Tree
OAVG	Obstacle-Avoiding Voronoi Graph
OAXSMT	Obstacle-Avoiding X-architecture Steiner Minimum Tree
OFEMST	Obstacle-Free Euclidean Minimum Spanning Tree
PD	Physical Design
PDAR	Power Domain-Aware Routing
POWV	Potential Optimal Wirelength Vector
PSO	Particle Swarm Optimization
RSMT	Rectilinear Steiner Minimum Tree
SADP	Self Aligned Double Patterning

The associate editor coordinating the review of this manuscript and approving it for publication was Muhammad Omer Farooq.

SMMT	Steiner Min-Max Tree
SMT	Steiner Minimum Tree
VLSI	Very Large-Scale Integration
XSMT	X-architecture Steiner Minimum Tree

I. INTRODUCTION

Today, Very Large-Scale Integration (VLSI) design plays a fundamental role in the development of many high-tech electronic circuits. VLSI is the process of forming an Integrated Circuit (IC) by incorporating millions of transistors into a single chip [1]. As integration continues to increase, more and more features, even a complete system can be integrated into a single chip. Some IC foundries have even achieved large-scale mass production of 7nm chips, this market trend presents many challenges for Physical Design (PD) and verification. As the backbone of the information industry, VLSI design and manufacturing are playing an increasingly important role in promoting economic development, reforming industrial structure and changing lifestyle.

PD is the most time-consuming stage in the design process of IC, and it is also one of the most crucial and concerned research fields in VLSI Computer-Aided Design (CAD) technology. Due to its complexity, the whole process of PD is usually divided into several phases: partition [2], floorplanning [3], layout [4], [5], placement [6], routing and other stages [7]. Within the PD flow, one of the most critical steps is Global Routing (GR) [8], a stage where signal nets are connected coarsely under a given placement so that wire/via spaces are allocated to each signal net [9]. The feature size has been reduced due to scaling during the IC design process. As a result, interconnect wire delays now contribute to a great proportion of the total delay compared with gate delays. Due to this reason, routing plays a major role in current IC design process as it affects the overall delay in routed design. Besides, the quality of GR solution directly affects chip area, speed, manufacturability, power consumption and the number of iterations required to complete the design cycle, and hence this step plays an important role in determining circuit performance. On the other hand, GR is a notoriously difficult problem: Even the simplest version of the problem, where a set of two-pin nets is to be routed under congestion constraints, is an NP-complete problem [10].

As one of the basic models of VLSI PD, Steiner Minimum Tree (SMT) can be applied to the floorplanning, layout and routing stages. Given points on a plane, an SMT connects these points through some extra points (Steiner points) to achieve a minimum total length. An SMT is usually used in initial topology creation for noncritical nets in PD. For timing critical nets, wirelength minimization is not the only goal. However, most nets are noncritical in routing phase and an SMT gives the most desirable route of such a net. Thus, SMT is often used as accurate estimations for congestion and wirelength during floorplanning and placement. It indicates that an SMT algorithm can be invoked millions of times. On the other hand, there exist many large pre-routes in modern VLSI design. The pre-routes are usually modeled as large sets of points, which increase the input size of Steiner tree problem. Since the SMT is a problem that can be computed millions of times and many of them have very large input size, highly efficient algorithms with good performance are desired.

Nowadays, it is no longer enough to consider only the wirelength and congestion during GR and Steiner tree construction. With the rapid development of integrated circuits, chip process is more and more advanced. Similarly, the number of transistors in the chip can be more and more. With the same performance, the power consumption of the chip becomes lower and lower. Besides, the number of metal layers in chips is increasing dramatically, which leads the number of vias become more and more. Due to the above reasons, timing and power consumption issues have become increasingly prominent. More and more constraints have emerged in GR and Steiner tree construction, which leads to the fact that the traditional routing algorithms or SMT algorithms cannot adapt to such multi-objective tasks well. Thus, GR and SMT algorithms face new challenges.

Although GR and SMT construction are quite correlative and vibrant research fields, they are rarely mentioned together in a holistic manner. To the best of our knowledge, this is the first time that the GR and SMT algorithms to the VLSI realm are systematically dissected, categorized and put together in a review article. The survey has the following four main objectives. First, we introduce the principles, related concepts, and their advantages and disadvantages of some common basic strategies in GR. Second, we investigate three subproblems in Steiner tree construction and three different GR subproblems. Third, we investigate some of the current advanced processes and routing issues based on these processes. Finally, we point out the future research directions to guide the future work.

The rest of the paper is organized as follows: In Section II, we introduce some of the basic technologies in GR, as well as their advantages and disadvantages. In Section III, we describe the problem model for the two problems of Steiner tree construction and GR. In Section IV, we investigate three subproblems in the Steiner tree: SMT, Obstacle-Avoiding Steiner Minimum Tree (OASMT) and Length-Restricted Steiner Minimum Tree (LRSMT). In Section V, we investigate three types of GR algorithms: sequential methods, concurrent methods and performance-driven routing. In Section VI, a discussion of related issues is provided, including GR in a Multiple Dynamic Supply Voltage (MDSV) chip design and routing problems in a new via-pillar process. Finally, Section VII concludes this survey and suggests directions for future research.

II. BASIC ROUTING STRATEGIES

In this section, several basic strategies that are often used in GR are described.

A. MULTI-PIN NET DECOMPOSITION

Multi-pin net decomposition [11]–[15] is often used in the GR algorithms. A net with more than two pins is often decomposed into two-pin subnets, and then point-to-point routing for each subnet is performed in some order. This net decomposition is performed at the beginning of GR, which affects the quality of the final routing solution. Multi-pin net decomposition is adopted by many global routers. Two popular methods to decompose a multi-pin net are Rectilinear Steiner Minimum Tree (RSMT) construction and Minimum Spanning Tree (MST) construction. RSMT often provides tree topologies with shorter wirelength while MST provides greater flexibility due to more L-shaped two-pin nets produced.

B. MAZE ROUTING

A subproblem often encountered in GR is finding the shortest path connecting the two pins in the presence of blockages. The most widely known solution to this problem is the Lee's algorithm [16], that is, maze routing algorithm. Maze routing is a grid-based search algorithm that has long been considered a brute-force algorithm since it allows all possible paths. For an arbitrary cost function, maze routing uses the shortest possible path to connect the source and target points. Simple maze routing implementations typically use Breadth First Search (BFS) or Dijkstra algorithms. In addition, the A* algorithm usually performs better. Johann and Reis [17] used the A* algorithm to improve the Lee's algorithm to speed up the convergence.

C. MULTI-SOURCE AND MULTI-SINK MAZE ROUTING

Multi-source and multi-sink maze routing [18] evolved from traditional maze routing, which is an improvement over traditional maze routing, considering more and better routing paths. Since the maze routing can only obtain paths starting from a given source and ending at a given receiver, the router using the maze routing may lose some better solution for multi-pin net due to this limitation. Figure 1 shows a routing graph of a multi-pin net, which is divided into two subtrees. When using traditional maze routing, no matter how large the search space is, only the path starting from A and ending with B can be obtained. However, other possible paths connecting the two subtrees are also possible solutions to accomplish this multi-pin routing. To overcome the shortcomings of traditional maze routing, multi-source and multi-sink maze routing treats all grid points in the same subtree as source points, and all grid points in another subtree as sink points. Thus, the router can obtain a path like path CD, which not only avoids the blockage area, but also obtains a shorter wirelength.

D. STEINER TREE CONSTRUCTION

Both the maze routing and the line probe approach are designed for the connection of a two-pin net. However, in practical design, routing problems often encounter some

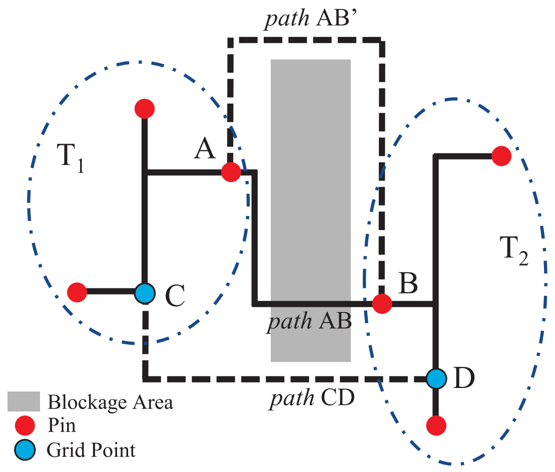


FIGURE 1. Alternative paths obtained by classical maze routing (path AB') and multi-source and multi-sink maze routing (path CD).

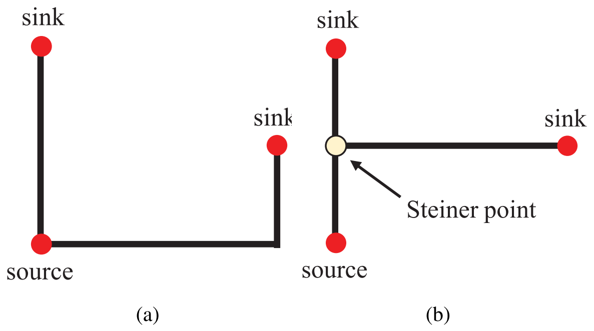
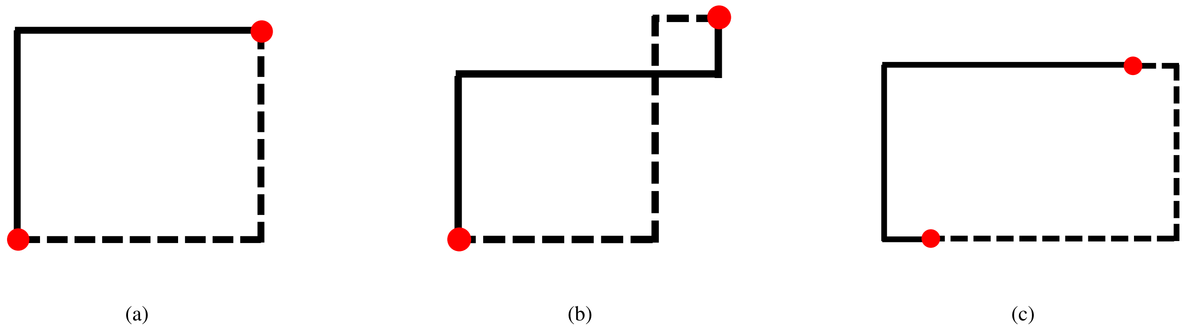


FIGURE 2. A three pin net connected by (a) a spanning tree and (b) a Steiner tree.

nets with more than two pins. A common approach to deal with multi-pin nets is to decompose them into a set of two-pin nets. One way to perform this decomposition is to first construct an MST on these pins and then perform maze routing on each pair of pins corresponding to each edge of the MST. As shown in Figure 2(a), a possible solution for routing a three-pin net uses this scheme, which is decomposed into two two-pin nets. It's easy to see that such a scheme probably won't find the best solution. The total length of the routing tree can usually be reduced by adding additional points outside of given pins and constructing an MST on all these nodes. The extra added node is the Steiner point, as shown in Figure 2(b). In most VLSI GR problems, since all wires are horizontal or vertical, only the RSMT [19]–[21] is considered. In recent years, due to the potential of non-Manhattan architecture, some work has focused on the X-architecture, so the X-architecture Steiner tree [22]–[24] has gradually become a research hotspot.

E. PATTERN ROUTING

Given a set of two-pin nets, a global router needs to find the path of each net under the capacity constraint. Most nets are routed with short paths to minimize wirelength. Maze routing



(a) (b) (c)

FIGURE 3. **Common patterns used to route two-pin nets such as (a) L-shapes, (b) Z-shapes and (c) U-shapes.**

methods such as Dijkstra's algorithm and A* search can be used to guarantee a shortest path between two pins. However, these techniques can lead to unnecessary slowdowns, especially when the generated topologies are composed of point-to-point connections, such as an L-shape. In practice, many nets' routes are not only short but also have few bends. Therefore, pattern routing [25], [26] is a better choice. Compared with the traditional maze routing, the pattern routing has the advantages of high efficiency and high speed. Pattern routing uses a predefined pattern for routing the two-pin nets, which limits point-to-point connections to a small number of fixed shapes. Given a $m \times n$ bounding box, where $n = k \times m$ and k is a constant, pattern routing only takes $O(n)$ time while maze routing requires $O(n^2 logn)$ time. As shown in Figure 3, topologies commonly used in pattern routing include L-shapes, Z-shapes, and U-shapes.

F. MONOTONIC ROUTING

Monotonic Routing [18] finds the best monotonic routing path. Assuming that the given source point is at the bottom left of the receiving point, monotonic routing can only move up or to the right from the grid point. Since the path of the monotonic routing does not exceed the bounding box of the two-pin nets, this method only needs to search for $\frac{(m+n-2)!}{(m-1)!(n-1)!}$ paths on the $m \times n$ grid. Therefore, monotonic routing increases the search space while maintaining the same time complexity as Z-shaped pattern routing. However, since monotonic routing is difficult to bypass some obstacles, Liu *et al.* proposed a hybrid method [27]. Its search solution space is between the monotonic routing and the maze routing, and has a good trade-off between the runtime and the quality of the solution.

G. INTEGER LINEAR PROGRAMMING (ILP) ROUTING

The mathematical model of the GR problem can always be easily modified to a 0-1 ILP problem, that is, ILP routing [28]–[30]. For each net and a routing graph, a set of Steiner trees is given. Thus, the goal of ILP is to select a Steiner tree from the set of Steiner trees in the net so that its route does not violate the capacity constrain and minimize the total wirelength. However, as the circuit scale increases, the time complexity of ILP is very large.

H. RIP-UP AND REROUTE

After determining the initial routing of a set of nets, it is common to find that some routing resources are overused. Then, existing routers are torn apart and reassigned in an iterative repair framework called rip-up and reroute [25], [31]–[34]. Modern ILP tools help ILP-based global routers to successfully complete hundreds of thousands of routes within hours. However, commercial Electronic Design Automation (EDA) tools require greater scalability and lower runtime. In the routing phase, if a net cannot be routed, it is usually due to physical obstacles or other routed nets occupying its path. The core idea is to allow temporary violations until all nets are routed, that is, iteratively rip some nets, and reroute them differently to decrease the number of violations. If only one net is processed at a time, this strategy is a sequential method, so the order of nets to be processed has a great influence on the quality of the final solution.

For each net in violation, Kuh and Ohtsuki [31] defined successful quantifiable probabilities (success rates) of rip-up and reroute, and only re-routed the most promising nets. However, whenever a net cannot be routed without violations, the success rates needs to be recalculated. This is very expensive, especially for large scale designs. Rip-up and reroute are often combined with other GR methods as a post-processing step to further improve routing quality.

I. NEGOTIATED CONGESTION ROUTING

The core idea of negotiated congestion routing [35]–[37] is to use the congestion history of each edge of routing graph as the basis for future routing. The original negotiated congestion routing was proposed by McMurchie [35] to balance the competing goals of eliminating congestion and minimizing the performance degrading due to timing critical paths. Nowadays, the idea of a negotiation mechanism has been widely used in the design of the current global router. Modern routers use negotiated congested routing to perform rip-up and reroute, where each edge is assigned a $cost(e)$ to reflect the needs of the edge e. A segment from net N that is routed through e pays a cost of $cost(e)$. The total cost of N is the sum of $cost(e)$ values taken over all edges used by net:

$$\cos t(N) = \sum_{e \in N} \cos t(e) \tag{1}$$

A higher $cost(e)$ value will resist the use of edge e to a greater extent and implicitly encourage the net to seek other less used edges. The iterative routing method uses methods such as A* search to find the least costly paths while respecting the edge capacity. That is, in the current iteration, all nets are routed according to the cost of the current edges. If any of the nets cause violations, such as congestion on some edges, the nets are ripped up and the edge costs of those nets are updated to reflect their congestion and rerouted in the next iteration. The process continues until all the nets are routed or certain termination conditions are met.

The edge cost $cost(e)$ is increased according to the edge congestion $\varphi(e)$, defined as the total number of nets using e divided by the capacity of e:

$$\varphi(e) = \frac{d(e)}{c(e)} \tag{2}$$

If e is uncongested, i.e., $\varphi(e) \leq 1$, then $cost(e)$ does not change. If e is congested, i.e., $\varphi(e) > 1$, then $cost(e)$ is increased to punish nets that use e in subsequent iterations. In negotiated congestion routing, $cost(e)$ can only grow or remain unchanged. Because if $cost(e)$ is reduced, the cost of the previous penalty for the net using edge e will no longer be valid. And it would be futile to try to route these nets in the iterations, because these nets will use the same edges as before when rerouted.

The range of the growth rate of $\Delta cost(e)$ must be controlled. If it is too high, the entire set of nets will be pushed from one edge to another. It can cause the routes of the net to jump back and forth between different edges, resulting in longer runtime and longer wirelength, and may affect the success of the routing. On the other hand, if $\Delta cost(e)$ is too low, then all nets without violations require too many iterations, causing an huge increase in runtime. Ideally, the cost growth rate should be gradual so that a small portion of the nets is routed differently in each iteration. In different routers, the growth rate is modeled as a linear function [38], a dynamically changing logic function [39], and an exponential function with a slowly increasing constant [14], [40]. In practice, a good negotiated congestion-based global router can effectively reduce congestion while maintaining small wirelength.

J. LAYER ASSIGNMENT

Layer Assignment plays a crucial role in routability, timing, crosstalk and manufacturability. If an excessive number of wires are assigned to a specific layer, congestion and crosstalk will be exacerbated [41], [42]. Besides, if global timing critical nets are assigned to the lower layer, the timing is deteriorated due to narrower wire width/spacing. A large number of vias due to poor layer assignment may result in a routability/pin access problem because vias require a larger area and a wider pitch than the wires. For nano-designs, minimizing the number of vias is especially important because via malfunction is one of the key manufacturability issues [43]–[45]. Layer assignment is a critical step in multi-layer GR because it maps the results of 2D GR back to the original multi-layer solution space. Modern integrated circuits or chips are often multi-layered structures. When routing, most routers usually do not directly route in three-dimensional space. Instead, they are routed on a two-dimensional plane, and then the 2D routing result is restored to the 3D space by the layer assignment technique. Some work is implemented based on dynamic programming [11], [46], [47]. In addition, BoxRouter 2.0 [13] implements a complex linear programming technique that restores 3D routes from 2D projections and optimizes 2D routers from 3D routing results. The goal of layer assignment is usually to maintain the total overflow of the 2D GR, and then minimize the number of vias [48]–[50]. Lee and Wang [51] considered antenna effect in the layer assignment, by properly assigning wires to higher metal layers (not necessarily the top layer), antenna violations can be effectively reduced.

III. PROBLEM MODEL

A. PROBLEM MODEL OF STEINER TREE CONSTRUCTION

The SMT problem is to connect all pins through some extra points (called Steiner points) to achieve a minimum total wirelength in VLSI routing.

One of the most basic problems in VLSI routing is the shortest path problem of two-pin nets, looking for the shortest routing path given the position of the two pins while considering the obstacles. Commonly used strategies are maze routing, line probe methods, pattern routing and so on. However, in the actual routing problems, there are often more than two pins in a net. A common approach to dealing with multi-pin nets is to decompose the multi-terminal net into a set of two-terminal nets, that is, construct an MST with pins as nodes. In order to reduce the wirelength of the routing tree, in addition to original nodes formed by given pins, the final MST can be constructed by introducing additional nodes. As shown in Figure 2, a three-pin net connected by spanning tree is given. By adding an extra point to form the Steiner tree, the total wirelength is greatly reduced. Therefore, the Steiner tree model has gradually become the best connection model for multi-pin net which is a key issue in VLSI routing.

In most routing problems, the segments can only route horizontally and vertically. We call this kind of routing tree RSMT. The RSMT construction is an NP-hard problem, which is as follows: Given a set of points in the plane, the RSMT problem seeks to connect the points by added some extra points: A tree made up of horizontal and vertical line segments that has the minimum cost. The cost of any edge in the tree is the rectilinear or manhattan distance between its endpoints, and the cost of a tree is the sum of its edge costs. However, since only the horizontal and vertical routing directions are included, the Manhattan architecture-based routing cannot fully utilize the routing area, which makes it difficult to achieve the desired wirelength. In order to break this limitation, it is necessary to change the traditional Manhattan architecture [52]. Therefore, researchers began to try to use the X-architecture as the base model for routing to achieve the overall performance optimization of the chip. Diversification of interconnection structures not only reduces the wirelength, but also improves routing quality and improves chip performance.

B. PROBLEM MODEL OF GR

The basic goal of routing is to successfully connect to each net and resolve resource contention issues. In modern VLSI design, completing the above tasks in one step is a very complicated problem because there may be millions of components and nets on a single chip. Therefore, the routing stage is often divided into two sub-phases: GR and Detailed Routing (DR). In GR, chip areas are divided into a set of areas, and wires that cross the boundaries of these areas are roughly assigned certain paths indicating which areas they must pass through. After performing this phase, the routing within each such area will be performed in the detailed routing phase. For GR, modern designs typically have multiple metal layers and two adjacent layers are connected by vias [26].

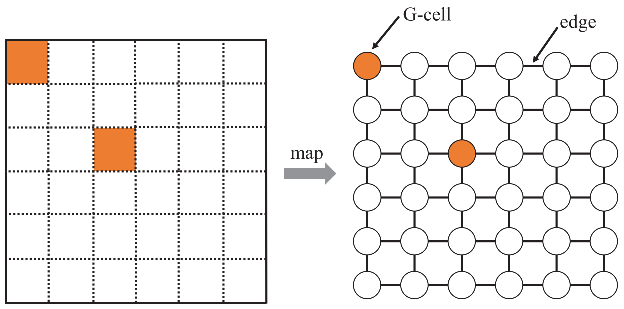


FIGURE 4. Routing area and its corresponding grid graph.

The GR problem can be described as follows: In the GR phase, the routing area is divided into a set of rectangular grid cells called G-Cells, as shown in Figure 4. Therefore, the GR problem typically models the routing area as a grid graph, and the available routing resources are represented by the edges with capacity in the grid graph. The grid graph (routing area) is $G(V, E)$, where the node $v_i \in V$ represents a rectangular grid unit (G-Cell). The two G-cells with adjacency are mapped to one edge $e_{ij} \in E$ of the grid graph. The capacity of the grid edge, $c(e_{ij})$, represents the number of routing tracks available between two adjacent grid cells G-Cell$_i$ and G-Cell$_j$. And $d(e_{ij})$ indicates the number of routing tracks taken up. According to $d(e_{ij})$ and $c(e_{ij})$, the overflow of the edge can be obtained as follows:

$$overflow = \begin{cases} d_e - c_e, & if\ d_e > c_e \\ 0, & otherwise \end{cases} \tag{3}$$

When evaluating the quality of a GR solution, three metrics are usually considered, that is, overflow, wirelength and runtime. The goal of the algorithm is to minimize these three metrics [53]. Overflow refers to the total amount of demand that exceeds capacity over all edges. In the ideal case, we want the overflow to be zero. Similarly, wirelength should be as small as possible. It is for this reason that various SMT construction algorithms need to be designed to optimize the wirelength of each net in a set of nets to minimize the total wirelength.

IV. STEINER TREE CONSTRUCTION

Given a set of input points, the Steiner tree construction is a tree that looks for the minimum length of the connected input points, where new points can be added to minimize the length of the tree. Solving the Steiner tree construction is of great importance since it is one of the fundamental problems in network, VLSI routing, multicast routing, wirelength estimation, computational biology, and many other areas. However, Steiner tree construction is an NP-hard problem, which smashes any hope of finding a polynomial time algorithm to solve the problem precisely. This is the reason that most studies focus on finding effective heuristic algorithms. In addition, the SMT algorithm may be invoked millions of times in a complete GR process. This means that an efficient SMT algorithm plays a key role in the GR.

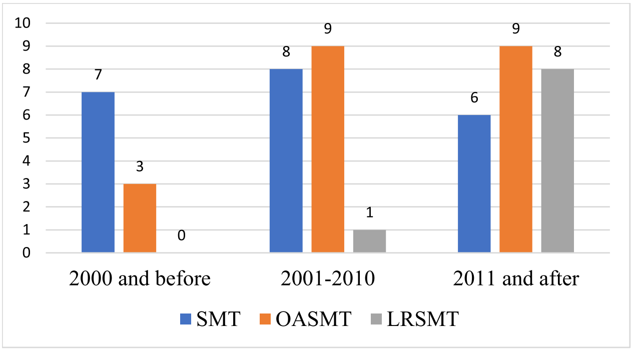


FIGURE 5. Distribution of three types of SMT in different periods.

Figure 5 shows the distribution of three types of work in different periods. It is easy to see that most of the early work are focused on SMT. After 2000, more and more researchers began to consider obstacles. During this period, [80] proposed the original LRSMT model. In the last decade, more and more work has focused on the LRSMT problem. This is in line with the development trend of VLSI design today. As shown in Figure 6, early work requires little consideration of obstacles. Later, researchers have begun to consider a variety of constraints. It is known that the basic optimization goal of SMT construction is to minimize the interconnected wirelength while considering as many other optimization objectives as possible like obstacles, timing and buffering. Therefore, every surveyed paper usually involves more than one constraint mentioned in Table 1. And the

TABLE 1. Constraints considered in SMT construction over the last decade.

Three Types of SMT Construction	Constraint of Consideration	References
SMT	considering wirelength	[52], [54-60]
	considering X-architecture	[52], [59]
	considering multi-layer	[52]
	considering timing/delay	[58], [59]
	considering memory	[55], [56]
OASMT	considering wirelength	[61-70]
	considering obstacles	[61-70]
	considering X-architecture	[67-70]
	considering multi-layer	[62], [65], [69]
LRSMT	considering wirelength	[71-76]
	considering obstacles	[71-76]
	considering timing/delay	[74], [76]
	considering buffering	[72-76]

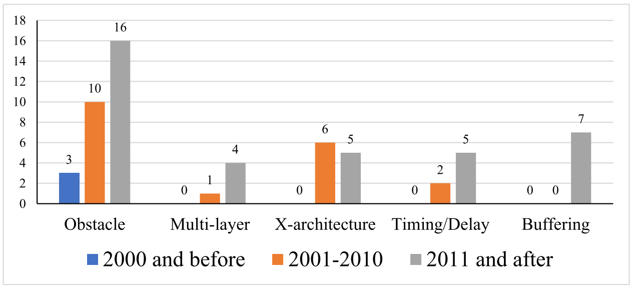


FIGURE 6. Constraints considered at different periods in SMT.

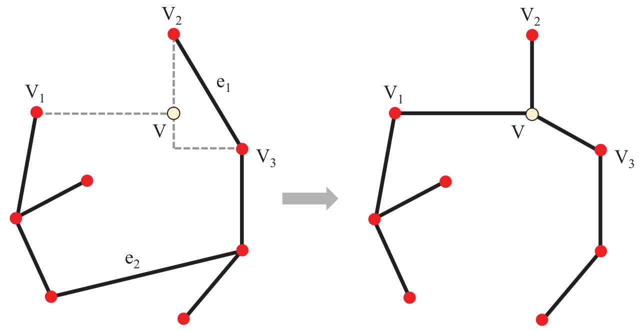


FIGURE 7. Example of edge-based update.

constraints considered in SMT construction over the last decade in Table 1 is given according to the most critical optimization objectives of surveyed papers.

A. SMT

As one of the basic models of VLSI PD, the SMT can be applied to the planning, layout and routing stages. Given points on a plane, an SMT connects these points through some extra points (called Steiner points) to achieve a minimum total length. An SMT is usually used in initial topology creation for noncritical nets in PD. For timing critical nets, minimization of wirelength is not the only goal. However, since most nets are noncritical in routing stage and an SMT gives the most desirable route of such a net, it is often used as an accurate estimation of congestion and wirelength during floorplanning and placement. It indicates that an SMT algorithm can be invoked millions of times. On the other hand, there exists many large pre-routes in modern VLSI design. The pre-routes are usually modeled as large sets of points, which increase the input sizes of the Steiner tree problem. Since SMT is a problem that can be computed millions of times and many of them have very large input sizes, the highly efficient algorithms with good performance are desired. There are many exact algorithms and heuristic algorithms to solve the SMT problem due to its importance. However, SMT is an NP-hard problem, which means that any exact algorithm [20], [23] is expected to have an exponential worst-case runtime. Thus, much more previous efforts are put on heuristic algorithms. Since an MST is proved a 3/2 approximation of an SMT [77], some works construct an SMT by improving on an MST topology [78]. However, since the backbones are restricted to the MST topology in these approaches, there is a limit on the improvement ratios over the MSTs. The iterated 1-Steiner algorithm [79] is an early approach to deviate from this restriction, which is the first heuristic which has been shown to have a performance ratio less than 3/2.

A much more efficient approach called edge substitute was later proposed by Borah *et al.* [19], which is an edge-based method that starts with an MST, and iteratively joins a point to a nearby edge and then deletes the longest edge in the formed ring. They first used the Prim algorithm to calculate the minimum spanning tree of a set of vertices in $O(n^2)$ time. As shown in Figure 7, assuming e_2 is the longest edge of the path between v_1 and v_2 in the tree, they made the following modifications to the tree: (1) Add node v. (2) Delete edge e_1.

(3) Delete edge e_2. (4) Add a new edge from node p to node v_1. (5) Add a new edge from node p to node v_2. (6) Add a new edge from node p to node v_3. In the above process, a new node, the Steiner point, is added to the whole tree, and a pair of existing edges are replaced with three new edges. The sum of the costs of the newly added two edges (v, v_2) and (v, v_3) is equal to the cost of the original edge e_1. Therefore, the above steps reduce the cost (gain) of the tree to:

$$gain = length(e_2) - length(v, v_1) \tag{4}$$

However, this operation is also subject to certain restriction. Considering the tree in Figure 8, the nodes v_1 and v_3 may be connected by e_2, so that the operation based on the above steps can be performed. However, node v_2 cannot be connected to e_2 in the same way. In a sense, e_1 blocks v_2 from connecting to e_2 in the tree, while nodes v_1 and v_3 are visible to edge e_2. Therefore, a node can be connected to an edge for edge-pair replacement only if the node is visible to the edge.

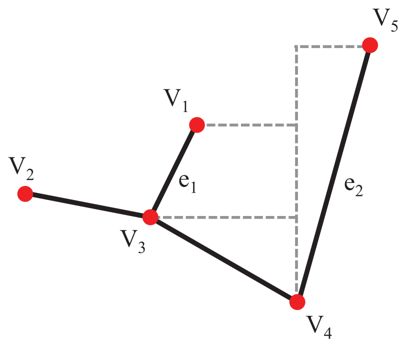


FIGURE 8. Visible nodes and blocking.

Zhou et al. [80] established a general framework of spanning graphs which includes both Delaunay Triangulation (DT) and Non-Delaunay Triangulation (NDT) approaches. Based on the framework and using the property that each point needs to be connected to only a few other points, they designed a sweep-line algorithm to construct a spanning graph for rectilinear distance. Then, MST can be easily calculated on the spanning graph. Their spanning graph is a set of edges that connect all points and does not form a loop. In fact, Zhou's spanning graph is a sparse graph, which contains at least one MST. The number of edges in the graph is called the cardinality of the graph, and they propose an efficient algorithm to construct the spanning graph of the cardinality. Starting from each point, the plane is divided into eight areas (Figure 9). If use rectilinear distance, the distance between any two points in one region is always less than the maximal distance from them to p. Based on the cyclic property of the MST, the longest edge on any circuit should not be included in any MST, which means that the point closest to p in each region only needs to be connected to p. Considering all the given points, these connections will form a spanning graph of the cardinality $O(n)$.

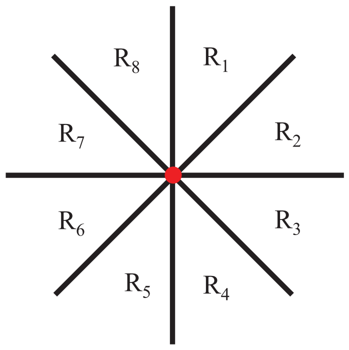


FIGURE 9. Eight partition.

A heuristic SMT algorithm was proposed in [81], which combines Borah's edge substitution and Zhou's spanning graph. It is a heuristic algorithm that starts from RMST and updates it to solve the RSMT problem. It is much faster than the iterated 1-Steiner algorithm and has similar performance. However, they didn't use a sweep line algorithm to find visibility relations between points and edges. In order to skip this complex step, they used the geometrical information, which is embedded in the spanning graph. For each edge in RMST, the end points of the edge are considered. Then, all neighbors of both end points spanning graph are considered as point components of point-edge pairs. Since the number of possible point edge pairs is $O(n)$, then the complexity of this scheme is also $O(n)$.

In addition, Cinel and Bazlamacci [82] used the Zhou's algorithm [81] and Kahng's Batched Greedy Algorithm (BGA) algorithm [83] as the basis for proposing an RSMT algorithm to produce outstanding effects in a shorter time. They made some modifications to the RSMT algorithm [81] that the longest edge computation part of the RSMT algorithm is replaced by the BGA approach in which parent-edge arrays are calculated and used for this purpose. Thus, they got a non-recursive and much faster version. Besides, they proposed the partial parallelization and distributed form of the modified algorithm.

Kahng's BGA algorithm [83] finds RMST and the optimal full Steiner triple on the sparse graph, rather than at the fully connected graph. The full Steiner triple is the optimal three-terminal Steiner tree, with the terminal located on the leaf. The algorithm considers combining each complete Steiner triple into RMST. The most expensive edges in each cycle that will be formed with each merger should be removed from RMST. Thus, the BGA algorithm gets an approximate RSMT.

A very fast and accurate RSMT algorithm called FLUTE was proposed by Chu and Wong [21]. Based on the pre-computed lookup table, the FLUTE algorithm can solve the optimal RSMT for low-degree nets (up to 9 degrees). For high-degree nets, they also implemented a net-breaking technique to reduce the size of the net until the method can be used. Chu et al. showed that according to the relative positions of pins, all nets with degree n can be divided into

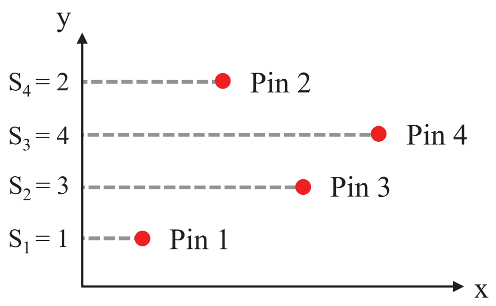


FIGURE 10. An illustration of the vertical sequence of a net.

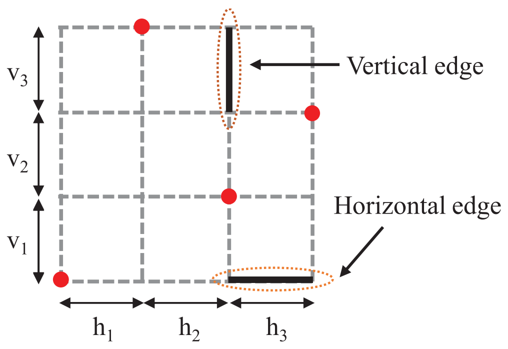


FIGURE 11. An illustration of horizontal and vertical edge lengths.

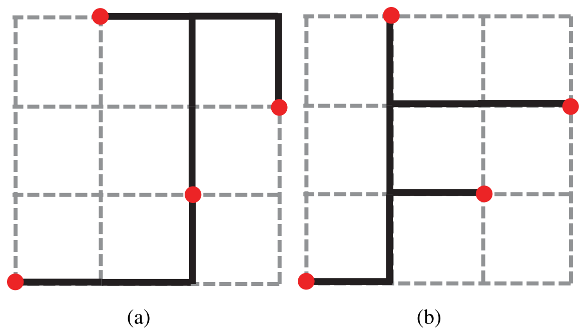


FIGURE 12. Two possible routings for the net in Figure 10. (a) Vector (1,2,1,1,1,2). (b) Vector (1,2,1,1,1,1).

n! groups. For each group, the wirelength of all possible routing topologies can be written as few linear combinations of distances between adjacent pins. Each linear combination is called a Potential Optimal Wirelength Vector (POWV). They stored few POWVs for each group into a lookup table. To find the optimal wirelength of a specific net, they simply calculated the wirelength corresponding to the POWV of the group to which the net belongs, and then got the minimum wirelength. The FLUTE method only considers routing along the Hanna grid, as Hanan points out that the optimal RSMT can always be constructed based on the Hanan grid [84]. When generating the lookup table, they make the vertical sequence $s_1s_2 \ldots s_n$ the index list of all the pins sorted in ascending order according to the y coordinate. As shown in Figure 10, the vertical sequence is 1342. The length of the horizontal edge in the Hanan grid is equal to the distance between two adjacent vertical Hanan grid lines. The length of the horizontal edge is represented as $h_i = x_i + 1 - x_i$, and the length of the vertical edge is expressed as $v_i = y(s_{i+1}) - y(s_i)$, as shown in Figure 11. For a given net with four pins, the wirelength of the three possible routing solutions can be written as $h_1 + 2h_2 + h_3 + v_1 + v_2 + 2v_3$ (Figure 12(a)) and $h_1 + 2h_2 + h_3 + v_1 + v_2 + v_3$ (Figure 12(a)). The wirelength is expressed as a vector coefficient and is called a wirelength vector. The corresponding wirelength vector of Figure 12(a) and Figure 12(b) is $(1, 2, 1, 1, 1, 2)$ and $(1, 2, 1, 1, 1, 1)$, respectively. More importantly, they observed that it is only necessary to consider few wirelength vectors that have the potential to produce the optimal wirelength. Most vectors are redundant because they are larger or equal in all coefficients than the other vectors. For example, of the three vectors above, $(1, 2, 1, 1, 1, 2)$ can be ignored because the vector $(1, 2, 1, 1, 1, 1)$ produces smaller v_3. They groupd the nets with the same vertical sequence together to share a set of POWVs. Because if two nets have the same vertical sequence, each routing solution for one net is topologically equivalent to the solution of another net. Next, for each vertical, the algorithm generates all possible routing topologies, finds the corresponding wirelength vectors and prunes away the redundant wirelength vectors. The remaining set of wirelength vectors is the POWV of the group. An easy way to generate all possible routing topologies is to enumerate all possible combinations of using and not using each edge in the Hanan grid graph, and check if the generated subgraph is a Steiner tree containing all the pins. However, this method is very expensive. Therefore, they implement an algorithm based on boundary compaction technology. For a given group, they reduced the mesh size by compressing any of the four boundaries, that is, moving all the pins on the boundary to the grid lines adjacent to the boundary. The routing topology set of the original problem can be generated by extending the routing topology of the reduced grid back to the original grid.

A line segment-based RSMT construction algorithm was proposed by Vani and Prasad [54], which constructs a RSMT by incrementally increasing the length of four line segments drawn at each point. When two line segments are intersected, add an edge to the tree. The intersection of the line segments of the two points forms a rectangular layout, generating two Steiner points. Steiner point that reduces the total length must be selected. At the same time, a new strategy is used in the decision to select one of the two edges. The wirelength error problem caused by selecting one of the two edges is eliminated, and an approximate optimal RSMT is generated.

As shown in Figure 13, when the line segments of vertex 3 and vertex 4 intersect, one edge between the two vertices must be added to the RSMT. However, since the lengths of the edges of the upper L-shaped layout and the lower L-shaped layout are equal, the decision to select a specific layout is postponed, and both layouts are temporarily selected and not added to the RSMT. Next, when the line segments

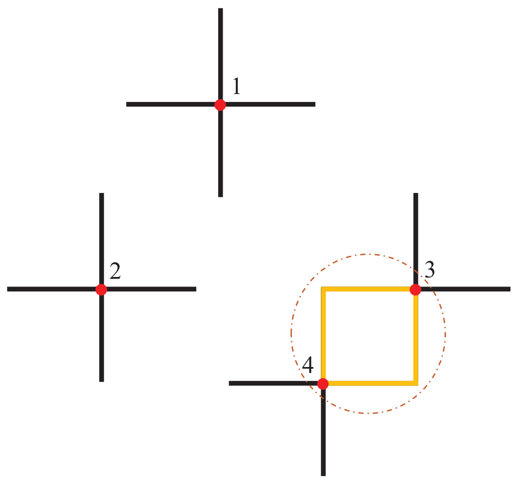


FIGURE 13. Line segments of points 3 and 4 intersect and both the L-shaped layouts (edges) are temporarily selected.

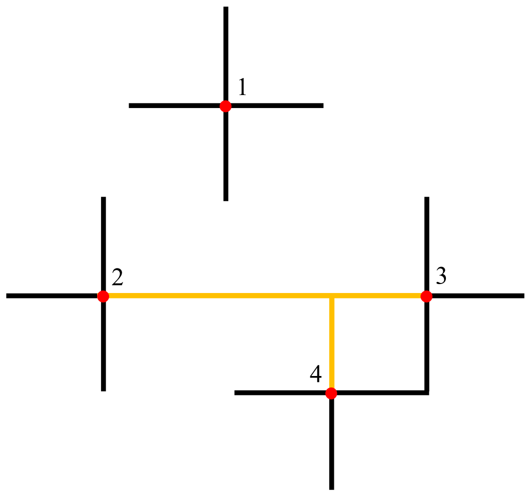


FIGURE 14. The upper L shaped layout of points 3 and 4 are made permanent when 2 and 3 are connected.

of vertex 3 and vertex 4 intersect, the only edge between them is selected and added to the RSMT. Once the vertices 2 and 3 are connected, due to the presence of overlapping edges (Figure 14), the edges of the upper L-shaped layout between the previous temporary selection of vertex 3 and vertex 4 will be made permanent and added to the RSMT. Whenever a line segment of point p_i touches a line segment of another point p_j, those two points have to be connected in a rectilinear fashion if and only if connecting those two points does not form a loop in already constructed tree. Those two points can be connected using either the upper L-shaped layout or the lower L-shaped layout. The time complexity of the proposed algorithm depends on the length between the two points in the RSMT that have the largest difference in x-axis or the difference in y-axis, since all the line segments are incremented till this last edge is added to the RSMT.

In [55], [56], a memory efficient RSMT construction was proposed. Its model is an improvement of original FLUTE. The memory optimization is not considered in RSMT construction of FLUTE and it adopts BFS to find MST as a result it induces memory overhead. To address this issue, it adopts divide and conquer and Depth First Search (DFS) to find the MST. The computation time is reduced due to less I/O disk access. The model also supports computation for larger degree net with limited memory utilization.

One of the applications using the RSMT construction is GR in which RSMTs are used for constructing topologies. For example, MaizeRouter [15], DpRouter [85], FastRoute [12], [46], BoxRouter [34], GRIP [40], and NTHU-Route 2.0 [26] use FLUTE for routing topology generation. However, FLUTE constructs only one RSMT for a net. Recently, Lin and Kim [57] built a database to rapidly construct all RSMTs on the Hanan grid for a given set of pins. They applied the database to timing-driven RSMT construction and congestion-aware GR. Compared with FLUTE, this algorithm can significantly reduce the wirelength and delay of the critical path.

For the X-architecture, due to its advantages in wirelength optimization, in recent years there have been some special alliances in the industry to promote the X-architecture. In this context, Xianlong *et al.* [86] pointed out that the X-architecture will become a hot spot for VLSI PD. They emphasized that X-architecture Steiner Minimum Tree (XSMT) is one of the most critical issues under the X-architecture. In addition, [87] presented some challenges and opportunities for the routing tree and routing algorithms under the X-architecture and gave the good prospects. Therefore, as a new basic problem, some algorithms have been proposed to solve the XSMT construction problem.

Two XSMT construction algorithms, called OST-E and OST-T, were proposed by Zhou [81]. They both generate graphs based on the X-architecture, and combine edge replacement method and triangular contraction, respectively. And both they can be extended to any geometry of the routing architecture. Kruskal's algorithm first sorts all the edges of a spanning graph by nondecreasing length and then considers them in the order. If the ends of the current edge have not been connected, the edge will be included in the MST; otherwise, it will be excluded. It can represent these connection operations by a binary tree, where the leaves represent the points, and the internal nodes represent the edges. For each internal node, its two children represent the two components connected by the corresponding edge in the MST. To illustrate this, a spanning tree and its merging binary tree are shown in Figure 15. As we can see, the longest edge between any two points is the least common ancestor of the two points in the binary tree. For example, the longest edge between p and b in Figure 15 is (b, c). To find the longest edge in the cycle formed by connecting a point to an edge, they need to find which end of the edge is in the same component with that point before connecting the edge. The least common ancestor of these two points is the longest edge that needs to be deleted. For example, after connecting p with edge (a, b), because p and b are in the same component before connecting (a, b), the longest edge to be deleted in the cycle is (b, c). It is crucial to find $(point, edge)$ pairs in the OST-E algorithm and to find triples in the OST-T algorithm. For edge substitution, in order to reduce the number of point-edge candidates from $O(n^2)$ to $O(n)$, they just considered the neighbors of either end of the edge to form $(point, edge)$ pairs. Since the cardinality of the spanning graph is $O(n)$, the number of possible $(point, edge)$ pairs generated by this way is also $O(n)$. Then, the longest edge can be removed as shown in Figure 16.

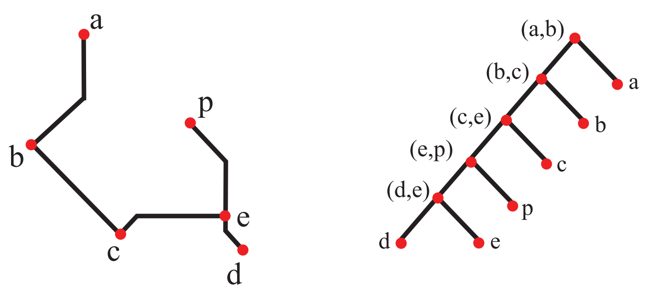


FIGURE 15. MST and its merging binary tree.

Coulston [23] proposed an accurate XSMT construction algorithm and pruning technique. This exact algorithm forms XSMT in two stages: generating all possible full components for a given point set, and then merging the full components to form XSMT. However, because of its exponential time complexity, it is less practical.

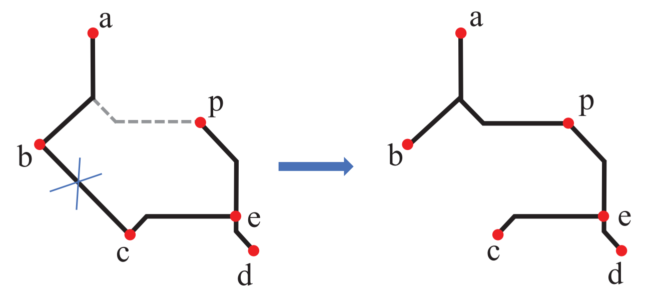


FIGURE 16. Illustration of remove the longest edge.

The first multi-layer architecture using X-architecture full-chip routing was proposed by Ho *et al.* [24]. In order to make full use of the advantages of X-architecture, they studied the optimal routing of the three-terminal nets on X-architecture, and developed a general XSMT algorithm based on DT method.

Tu *et al.* [58] proposed an algorithm to construct a timing-driven routing tree to balance the wirelength and timing. The construction of the tree is based on the properties of the shortest path tree and the MST. They proposed a graph with significantly fewer edges, but there is still an optimal solution to the problem. Based on the RC delay model, they modeled the timing using the geometric relationships of the circuits. The optimization goal is to minimize the total length if the length of each path is within a certain range. They iteratively added an edge to the MST to optimize the solution, and in order to further improve performance, the batch algorithm is also introduced. In addition, [59], [88] have proposed two algorithms for constructing timing-driven XSMT, and can significantly improve the timing performance of the chip compared with RSMT.

Recently, Liu *et al.* [60] presented an algorithm SMT construction based on Hybrid Transformation Strategy (HTS) and self-adapting Particle Swarm Optimization (PSO). They used HTS to enlarge the search space and improve the convergence speed. However, the proposed HTS in the evolutionary process may produce an ineffective solution. To this end, the crossover and mutation operators of Genetic Algorithm based on Union-Find Sets is introduced. The experimental results show that the proposed algorithm can efficiently provide a better solution for SMT problem both in X-architecture and rectilinear architectures than others. Moreover, the algorithm can obtain several topologies of SMT, which is beneficial for optimizing congestion in VLSI GR stage.

B. OASMT

The OASMT problem has been widely studied in recent years. Figure 17 shows the distribution of OASMT in different periods, early Steiner tree construction rarely considered obstacles. Since the RSMT problem is an NP-complete problem, the existence of obstacles further increases the difficulty of finding the optimal solution. Since macro-cells, IP blocks, and pre-routed nets are often seen as obstacles to the routing stage, the Obstacle-Avoiding Rectilinear Steiner Minimum Tree (OARSMT) algorithm is very useful for practical routing applications. There are many heuristic algorithms that can achieve good results in a short period of time. In addition, some precise algorithms have been proposed to generate optimal OARSMT. For example, [89] defined a grid-like trajectory graph. Then, a hybrid method for searching for one-to-all shortest paths while constructing MST was presented. The resulting routing tree can be regarded as an approximation of the Steiner tree.

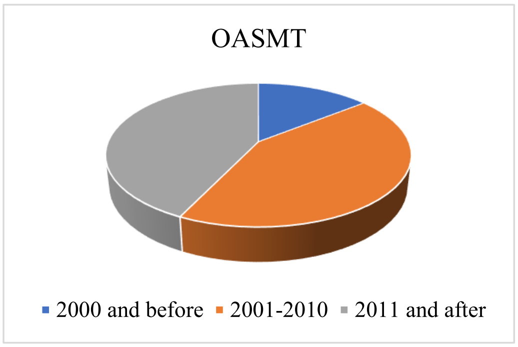


FIGURE 17. Distribution of OASMT in different periods.

Li *et al.* [90] used an efficient method to solve the obstacle-avoiding rectilinear Steiner tree problem. This work is developed based on the GeoSteiner method [20]. Full Steiner Tree (FST) is a basic concept used in GeoSteiner.

In FST, all the pins in consideration are leaf nodes. By splitting at a node with a degree greater than 1, any Steiner tree can be decomposed into a set of FST with disjoint edges. Since FST is easier to construct than SMT, most exact SMT algorithms will first generate its FST component. The GeoSteiner framework consists of two main phases. The first is FST generation, which involves pruning away unnecessary nodes based on some preprocessing information. As can be seen, the runtime of this stage is quadratic. The second is the concatenation of FSTs which are generated at the first phase. It can be foreseen as an ILP and can be solved using branch and cut search method. Initially, consider a set of actual pins. Every corner of each obstacle is treated as a virtual pin. For the FST generation, these virtual pins are also included, regardless of whether they are considered in the final optimal tree generation. In the incremental method generated by FST, the FST of two points is firstly generated, and these two points can be implemented by either of the two methods. Firstly, both points are real points, and secondly, at least one of them is virtual point. Techniques are used to ensure that redundant edges due to multiple FSTs are eliminated. The method is extended to generate FST with three or more points recursively. Once all the useful FST is generated, ILP is set to select the appropriate subset of FST and concatenate them to obtain the OARSMT.

A method based on Steiner point selection was proposed by Liu *et al.* [91]. This Steiner point-based framework can also be extended to multiple layers of OARSMT. The algorithm is mainly divided into four steps: graph construction, Steiner point selection, MST construction and final refinement strategy. Hanan grid and escape graphs are used to find the position of the Steiner point. Both methods generate a vertex from the intersection of the terminal or the extension line of the obstacle corner point. However, it produces many such vertices, making the vertices selection take too long. Therefore, the algorithm uses a different approach. In graph construction step, Obstacle Avoiding Voronoi Graph (OAVG) is constructed with a given set of terminals, obstacle vertices and desirable Steiner point candidates in $O(nlogn)$ time. Then, using Prim's algorithm and shortest path region, a Steiner point is selected from the constructed OAVG graph. In the third step, Obstacle-Avoiding Rectilinear Steiner Tree (OARST) is constructed in $O(nlogn)$ time using MST algorithm proposed by Long [92] and Liu [93]. MST on OAVG is constructed by the given vertices and the selected Steiner points. Long's MST algorithm includes the terminal path generation and MST generation using these paths, and Liu's algorithm includes critical path generation. Refinement is done in $O(nlogn)$ time to reduce redundant segments created in the above steps.

A top-down divide and conquer approach called FOARS was proposed by Ajwani *et al.* [61]. FOARS first divides a group of pins into several subsets, and then generates an obstacle-avoiding Steiner tree for each subset of pins by using FLUTE algorithm that considers obstacles. The initial solution is decomposed into multiple subproblems and OARSTs are generated and reconstructed. FOARS uses Obstacle-Avoiding Spanning Graph (OASG) to generate initial connected graphs. Similarly, Long *et al.* [92] used OASG to construct OARSMTs. FLUTE is a very powerful and fast tool but is not designed for the situation with obstacles. In order to create OARSMT, Steiner tree generated by FLUTE is applied with obstacles to perform local optimization, and then all locally optimized trees are combined to get the final solution. However, since there is no global view taken into consideration, this scheme fails to support large number of points or complexly placed obstacles. In order to overcome this problem, a partitioning algorithm is used to provides a global view for the problem at higher level and partition problem into smaller ones. This algorithm is divided into five stages. In the first stage, the connectivity information between the pins and obstacle corner vertices is obtained by using a novel octant OASG generation algorithm. In the second stage, FOARS constructs a MST based on the OASG and then obtained an obstacle penalized MST. In the third stage, FOARS partitions the pin vertices based on the obstacle penalized MST. After partitioning, they constructed an obstacle-aware Steiner tree by calling obstacle-avoiding FLUTE. Then they rectilinearized the pin-to-pin connections avoiding obstacles to construct an OARSMT. In the last stage, they performed V-shape refinement on the OARSMT to further reduce the wirelength. The runtime complexity of FOARS algorithm is $O(nlogn)$.

A routing graph called Obstacle Avoidance Routing Graph (OARG) was proposed by Liu *et al.* [93]. A three-step algorithm with refinement scheme is utilized to find a sub-optimal solution for the OARSMT problem. The three-step algorithm can be summarized as the first step is to construct OARG with $O(nlogn)$ time and $O(n)$ space, and the second step is constructing MST-OARG in $O(nlogn)$ time, the third step is OARST transformation from MST-OARG using a scheme called as MST-OARG reduction. During the local optimization process, the total wirelength of the OARST is reduced. General cases are used in this refinement scheme and sorting method is used to make the scheme greedier. Wirelength estimated by the algorithm can be reduced significantly in $O(nlogn)$ time. As mentioned OARG in this work maintains $O(n)$ space, hence contributing in the better initial solution to $O(nlogn)$ time. Besides, it significantly contributes to the performance with its linear space and obstacle avoidance properties.

Lin *et al.* [94] proposed an efficient algorithm based on OASG, which provides a certain theoretical optimal guarantee for the construction of OARSMT. They built an OASG with "essential" edges and proved that there is a rectilinear shortest path between any two pins, which is not guaranteed in the OASG in [95]. With this feature, the item works to ensure optimal OARSMT for any two-pin nets and many multi-pin nets. After constructing the initial OARSMT, they developed an effective improvement for U-shaped connections in OARSMT to further reduce wirelength.

Hu *et al.* [96] proposed a non-deterministic local search heuristic algorithm to deal with small-scale OARSMT problems with complex concave and convex polygon obstacles. The method is based on Ant Colony Optimization (ACO). Although this non-deterministic approach is flexible in dealing with complex obstacles, it can lead to very expensive runtimes for large-scale designs. The method first constructs a Steiner tree or a spanning tree for a multi-pin net, and then the edge that overlaps with obstacle is replaced by the edge around the obstacle. This method is very popular in the industry because of its simplicity and efficiency. However, the tree constructed in the first step may not have a global view of the obstacle, so the second part may only eliminate local overlap around the obstacle. Therefore, the quality of the solution for this work may be limited.

Compared to non-maze routing-based algorithms, the benefits of maze routing-based algorithms include the feasibility of applying various additional constraints on routing graph. However, multi-layer routing graphs are much more complex than single-layer routing graphs. This greatly increases the runtime required to solve the multi-layer OARST problem using maze routing. Lin *et al.* [62] proposed an algorithm based on maze routing, and proposed a Steiner point pre-selection strategy to guide the construction of multi-layered barrier Steiner trees. The algorithm provides a good trade-off between routing quality and runtime. The quality of the routing depends on the total cost, which is the sum of the wirelength and the vias cost. The algorithm first constructs a routing graph through a modified Hanan grid graph and evaluates the fitness values of each vertex to select the appropriate vertices to form the set Q of Steiner points. The initial multi-layer OARST is formed by connecting the pins and the points in the set Q. Then the Steiner point and tree topology are changed through the rip-up and re-build of Q, the final multi-layer OARST is generated.

In terms of exact algorithms, Ganley and Cohoon [97] proposed a strong connection graph called escape graph for the OARSMT problem and proved that there is an optimal solution composed only of escape segments in the graph. Based on the escape graph, they proposed an algorithm to construct optimal three-terminal and four-terminal OARSMTs.

In addition, there are several exact algorithms that give the optimal routing tree for the obstacle. However, for large-scale problems, this is lack of practicality since any precise algorithm requires exponential worst case time complexity. An exact algorithm was proposed by Huang and Young [98], which is a major improvement on [90]. They first proposed a new kind of full Steiner trees, namely the Obstacle-Avoiding Full Steiner Tree (OAFST). They proved that there is an optimal tree consisting of only OAFSTs for any OARSMT problem. Then they extended the possible topology of the FSTs in [77] to find possible topologies for OAFSTs, which suggests that OAFSTs can be easily constructed. Then a two-stage algorithm is proposed, in which the first phase is the generation of OAFST and the second phase is the construction of OARSMT. In this approach, each obstacle uses only two virtual points. The OAFSTs for each terminal are recursively generated by using bottleneck Steiner distance, empty diamond, empty corner rectangle and empty inner rectangle properties. In the second phase, OAFSTs are generated by formulating these OAFSTs into ILPs and using branch and cut searches. This approach combines the incremental method so that the OAFST explosion can be avoided. Later, Huang and Young [63] first implemented a geometric method to solve the OARSMT problem. Virtual terminals are added to simplify the structure of the FST. In this method, at least one virtual terminal is added to each edge of each obstacle. The algorithm can solve complex concave obstacles and convex obstacles. The best solution is the concatenation of FST. In [64], Huang *et al.* introduced virtual terminals at the four corners of each obstacle. In the FST generation phase, a virtual terminal pruning process is implemented to effectively reduce the number of FSTs, and an efficient method is proposed to construct a two-terminal FST in the presence of obstacles.

The existing multi-layer OASMT algorithm considers pin-to-pin connections rather than area-to-area connections, which limits the quality of the solution due to lack of regional information. Wang *et al.* [65] proposed a method based on region-to-area generation of graphs to construct a multi-layered barrier-shaped Steiner tree. The method connects all net shapes through the edges on one layer or vias between the layers. They used a binary search tree to handle obstacle avoidance constraints by capturing obstacles and inter-area connections with less time complexity. Their algorithm can guarantee an optimal solution for a net connecting any two net shapes on a single layer. The algorithm can be roughly divided into four stages: In the first phase, the overlapping net shapes are concentrated into one cluster to reduce the scale of the problem. In the second phase, a multi-layer spanning graph that connects all net shapes and corner vertices is constructed. In the third phase, a multi-layer obstacle-avoiding region-to-region MST that connecting all net shapes is generated, which may use some corner vertices and eliminate some extra edges along the obstacle. In the final step, all slant edges are converted to rectilinear ones, and the multi-layer OARSMT is generated.

Recently, Guo and Huang [66] proposed a first Physarum-inspired obstacle-avoiding routing algorithm for the PD of integrated circuits. They simulated the foraging behaviors of Physarum polycephalum using a novel nutrition absorption/consumption mathematical model, thereby presenting an efficient routing tool called Physarum router. With the proposed routing approach, for a given set of pin vertices and a given set of on-chip functional modules, a RSMT connecting all the pin vertices while avoiding the blockage of functional modules can be constructed automatically. Furthermore, several heuristics including a divide-and-conquer strategy, a non-pin leaf node pruning strategy, a dynamic parameter strategy, etc., are integrated into the proposed algorithm to fundamentally improve the performance of the Physarum router. Simulation results on multiple benchmarks confirm that the proposed algorithm leads to shorter wirelength compared with several state-of-the-art methods.

For the non-Manhattan architecture, there are also some existing work focused on the Obstacle-Avoiding X-architecture Steiner Minimum Tree (OAXSMT) problem [67]–[70], [99]. Jing *et al.* [99] first constructed a fully connected tree based on DT of the obstacle-avoiding constraint, and then embedded the tree into the obstacle-avoiding Steiner minimum tree by region combination. It is effective because the time complexity is only $O(nlogn)$. However, since it is designed to solve λ-geometry problems, the routing results for the X-architecture are not satisfactory. Especially when the scale of the problem is large, the results obtained are even worse than the rectilinear architecture.

Huang *et al.* [67] proposed an OAXSMT construction algorithm based on PSO, which is the first to specifically solve the problem of single-layer X-architecture around the Steiner tree problem. PSO has the characteristics of simple execution, few parameters and fast convergence [100]–[102]. Two genetic operators based on union-find are added into Huang's PSO process, namely crossover and mutation operations, which update the velocity and position of particles and further enhance the search ability. In addition, they also combined some heuristic strategies to further optimize the quality of the global optimal particle. Later, Huang *et al.* proposed a fast four-step heuristic algorithm [68], which first constructed a DT from given pins, and the Obstacle-Free Euclidean Minimum Spanning Tree (OFEMST) is constructed based on DT. Then, two lookup tables are generated that record the connection information about the edge of OFEMST. And the corner points on some obstacles are selected as the intermediate nodes by capturing the global view of obstacles, leading edges in OFEMST to avoid all obstacles accurately.

An algorithm called MLXR was proposed by Huang *et al.* [69], which is based on a lookup table and builds a 3D obstacle-free minimum tree as the basic architecture. An effective projection-based obstacle avoidance strategy is utilized to accurately capture the appropriate Steiner point position in a multi-layer environment. MLXR first constructs a 3D obstacle-free minimum tree to connect all pins and builds two lookup tables based on this obstacle-free minimum tree to provide fast information inquiry for the post steps. The 3D obstacle-free minimum tree is then converted into a multi-layer XSMT by means of a lookup table. Finally, the obstacles are handled by two refinement strategies. The experimental results show that compared with the most advanced algorithms, MLXR is excellent in both the total wirelength and the running speed.

C. LRSMT

In the practical design, routing areas typically contain multiple routing layers. Equipment often only occupies the equipment layer and some lower metal layers, and thus does not completely block the wires. When routing on a higher layer, wires may pass through larger obstacles. However, the repeater cannot be placed inside obstacles. It is indeed possible to avoid obstacles by increasing the wirelength, but it can lead to timing violations. Therefore, it is necessary to relax the routing resources in the routing process, so that wires can pass through obstacles to a certain extent, thus ensuring timing closure and avoiding signal distortion. Figure 18 shows the distribution of LRSMT in different periods, it is easy to see that no one paid any attention to the LRSMT issue in the early stages. However, in the last decade researchers have focused on the LRSMT problem because it is more in line with practical industrial designs.

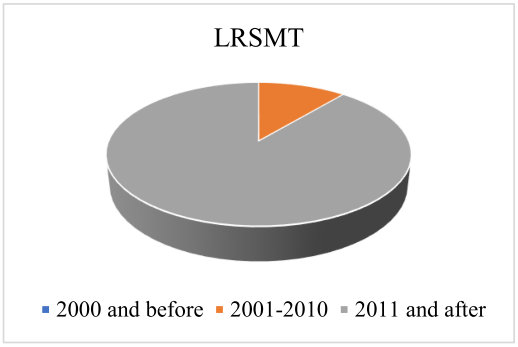


FIGURE 18. **Distribution of LRSMT in different periods.**

Hannemann and Peyer [103] defined an LRSMT model, which modified the distance network heuristic algorithm [104] to construct a feasible solution to the LRSMT problem. The algorithm is a 2-approximation algorithm. Held and Spirkl [71] followed the LRSMT model, and they built a range viewable view based on the view [105]. The shortest path that satisfies the LRSMT problem constraint between two points in the pin and obstacle inflection point must be included in the range. In the post-processing, FLUTE algorithm and Prim algorithm are used to reconnect the pins in the greatly unobstructed area to improve the solution quality. To ensure that the solution contained in the range view can satisfy the constraints, the LRSMT model is further simplified. Thus the range view not contain the Steiner point inside obstacles. Therefore, the Steiner tree obtained by this algorithm does not contain any steiner points inside obstacle, and this simplification will increase the total wirelength. A more accurate model, called the OARSMT with slew rate constraint (OARSMT_SC), is introduced in [72], [73]. This model uses the PERI model [106] to calculate the specific slew rate and ensures that the feasible solution part of the sub-tree inside obstacles does not violate slew rate constraint. OARSMT_RC can accurately meet the constraints, but in the design process, it is necessary to frequently calculate more complex voltage conversion rate values.

Zhang *et al.* [72] designed a heuristic algorithm. They first constructed an initial RSMT structure using Flute. In the repair process, three basic operations to reduce slew rate is introduced, and an integer linearity is established. The ILP model is used to correct the voltage conversion rate of the violation. The incremental method is used to repair the violation constraint one by one. The correction process divides the initial RSMT into multiple connected components. Finally, the length-restricted maze routing algorithm is used to interconnect these components to obtain the final feasible solution. The number of specific constraints in the ILP model depends on the shape of the obstacle, the position of the pin, and the accuracy of the solution. It is necessary to calculate the slew rate of each possible candidate internal tree in the ILP model. The maze routing algorithm takes a long time in this method.

A deterministic algorithm was proposed by Huang *et al.* [73], which can be used to obtain the optimal solution embedded in the extended Hanna grid. The LRSMT simplifies the constraint model and improves the efficiency of the solution, but the calculation of the constraint is not accurate enough. It is easy to cause detour or violate the actual constraints, which increases the difficulty of subsequent work.

Zhang and Pan [74] used the over-the-blocking routing to reduce delay in key paths and wirelength. It can meet the slew constraint and place the buffers in nonblocked places. Pre-buffering is used to provide a timestamp leading to be useful in finding good topology with respect to time and optimization is done on the embedded tree to improve the Delay to critical sinks. The approach proposed in this paper is time-driven, over-the-block RSMT with slew constraints. It consists of mainly five steps. First is the construction of timing-driven initial RSMT T with pre-buffering. Then change the topology of T to in accordance with the slew constraints. Later perform buffering on T and refine the topology of T based on buffering information. Again perform buffering on T.

Zhang *et al.* [75] proposed a heuristic algorithm that constructs an extended rectilinear Steiner tree grid as a routing graph, ensuring that the routing graph contains at least one optimal solution for RSMT problem and one near optimal solution OARSMT problem. And by modifying the shortest path heuristic algorithm, a step-growth heuristic algorithm is designed to solve the constrained SMT problem. The precomputation strategy is used to avoid frequent calculation of slew rate.

Shyamala and Prasad [76] further refined the LRSMT problem by considering both slew constraint and delay. They first presented an accurate delay optimization information. Then RSMT is iteratively computed until a convergence is met and delay optimization is done to find the delay resisted path. Secondly, it solves the slew and delay optimizer failure to improve the slack performance. Lastly, in the delay optimizer, position of Steiner points is optimized to reduce optimizing cost and wirelength.

V. GR

Within the PD flow, one of the most critical steps is GR, a stage where signal nets are connected coarsely under a given placement so that wire/via spaces are allocated to each signal net. The quality of the GR solution directly affects chip area, speed, power consumption and the number of iterations required to complete the design cycle, and hence this step plays an important role in determining circuit performance. On the other hand, GR is a notoriously difficult problem: even the most simple version of the problem, where a set of two-pin nets is to be routed under congestion constraints, is an NP-complete problem. Due to the increasing dominance of interconnect issues in VLSI design, it is desirable to incorporate GR into early design stages to get accurate interconnect information. Generally, routing is divided into GR and DR. After GR is completed, the result of GR is used as a guide for detailed routing to determine the precise routing trajectory of the wire mesh. Therefore, GR has always been the focus of researchers, because the quality of its routing has a great impact on the timing, power consumption and manufacturability of the chip.

GR techniques can be broadly divided into sequential methods and parallel methods. The sequential method uses maze routing [16], pattern routing [25], or rip-up and reroute, and only one net is routed at a time. The parallel method routes all the nets simultaneously. For example, Albrecht [39] defined the GR problem as a multicommodity flow problem and then solved it using ILP. Since the ILP problem has a large time complexity, it is usually simplified to a linear programming problem and solved by an approximation method. In terms of the implementation process, the GR method used by the global router can be divided into two categories: One is complete 3D routing, and the other is to perform 2D routing first, and then layer the results of 2D routing. For the complete 3D routing method, FGR [14] and GRIP [40] apply maze routing and ILP on the 3D routing graph, respectively. Due to the high complexity of modern designs, full 3D routing typically takes more time than other methods. For other routers that use 2D routing methods [11]–[13], [15], [38], [46], [107], [108], they first project the routing instance onto a plane, and then map the solution from the projection plane to the original multiple routing layers through the method of layer assignment.

A. SEQUENTIAL METHODS

The most primitive and straightforward strategy for routing multiple nets is to select a specific order and then route the nets in that order. The main advantage of this method is that when routing the current net, it is possible to know and consider the congestion information of the previously routed net. For example, in early algorithms that decompose multi-pin nets into two-pin nets, techniques such as obstacle-avoiding maze routings or line probes are used to route each net. In these approaches, a cell boundary is said to be open to path searching until all of the tracks have been occupied by previously considered nets. After that point, the boundary is treated as an obstacle. The disadvantage of sequential method is that the quality of solution depends largely on the order in which nets are processed, and it is difficult to find a good order. In any particular order, it is more difficult to route later considered nets because they are subject to more blockages.

In addition, there is no feedback mechanism that allows these nets to feed back information to earlier routed nets to leave some areas for later routed nets. Abel [109] concluded in his early work that no single net-sorting technique consistently performed well. Despite the controversy over the sequencing problem of the net, there are some good research results on sequential routing, mainly through the iterative loop to feed the congestion information from the later routed nets to the earlier routed nets.

The maze routing proposed by Lee [16] can optimize the routing of two-pin nets. The advantage of Lee's algorithm is that if the shortest path between two points exists, it will be found. However, as the routing area increases, its time complexity and memory usage become very large. For a $n \times n$ grid, the Lee's algorithm requires $O(n^2)$ time complexity. In addition, the multi-pin version [110], [111] of this work were originally designed for two-pin nets, and their solution quality is not satisfactory.

FastRoute 1.0 was proposed by Pan and Chu [112], it is a very fast router that is orders of magnitude faster than traditional routing algorithms. FastRoute 1.0 avoids the use of maze routing algorithms in its implementation, as it is well known that maze routing consumes a large amount of runtime of the global router. FastRoute 1.0 uses the Steiner tree construction to avoid the maze path. Due to the good quality of the Steiner tree obtained during the GR process, FastRoute 1.0 performs only one maze routing throughout the GR process.

The implementation process of FastRoute 1.0 can be roughly divided into three phases:

1) GENERATION OF CONGESTION GRAPH

At this stage, FLUTE is used to create a Steiner tree for all nets. All generated Steiner trees are decomposed into two-pin nets and routed using the L-shaped pattern routing. Thereafter, a congestion graph is obtained from the rough routing process.

2) CONSTRUCTION OF CONGESTION-DRIVEN STEINER TREE

In order to reduce congestion and build the optimal Steiner tree, two important techniques are applied during the stage. First, based on the congestion graph, the Steiner tree topology is constructed to reduce routing congestion. This algorithm extends the process of achieving congestion minimization in the FLUTE method, using fewer edges in a crowded area. In the second phase, in order to further reduce congestion, the edge shifting technique is adopted after the Steiner tree topology is modified. Move the edges to different locations to improve congestion without changing the rectilinear wirelength of the tree. As the edge shifts, routing requirements also shift from crowded areas to uncongested areas, reducing local congestion. The method is suitable for every net with more than 4 pins, and the congestion graph will change with each nets.

3) USE PATTERN ROUTING AND MAZE ROUTING FOR TWO-PIN NETS

This phase breaks down the Steiner tree into smaller two-pin nets. Then use Z-shaped pattern routing for 2-pin nets and perform rip-up and re-route. When performing rip-up and re-route, FastRoute 1.0 utilizes a logic function-based cost function that guides the maze routing to find less congested paths. Due to this high speed feature of FastRoute 1.0, it can be integrated into the early stages of design. Use in the early stages of design and execute in a fast manner can improve the quality of the design.

Pan *et al.* introduced two main techniques in FastRoute 2.0 [18] to further improve the quality of the solution of FastRoute 1.0. First, replace the pattern routing in FastRoute 1.0 by using monotonic routing technology. Second, a multi-source and multi-sink maze routing strategy is used. FastRoute 2.0 achieved better solution quality than FastRoute 1.0, reducing total overflow by more than an order of magnitude and running at 73% slower than the FastRoute 1.0. In FastRoute 1.0, each Steiner tree is decomposed into two-pin nets and routed using Z-shaped pattern routing. Since the pattern routing limits the shape of the routing path, the GR process can be greatly accelerated. However, the quality of the pattern routing may be much worse than the maze routing. Maze routing ensures that the lowest-cost routing scheme is found, while pattern routing considers only a very small fraction of all possible routing schemes. For two-pin nets spanning *mn* grids, L-shaped pattern routing considers only two different paths, and Z-shaped pattern routing only considers $m + n$ different paths. Therefore, in many complicated situations, the pattern routing will fail. FastRoute 2.0 hopes to find a balance between the pattern routing and the maze routing, so that the quality of the solution can be better than the pattern routing, but the runtime is not much slower than the pattern routing. Therefore, FastRoute 2.0 uses monotonic routing instead of pattern routing in FastRoute 1.0. The basic idea is to find the best monotonic routing path for a two-pin net. Let one pin be source (S) and the other pin be sink (T). The monotonic routing path from S to T always points to T. Figure 19 shows two different monotonic routing paths from S to T. Note that all monotonic routing paths do not deviate from the bounding boxes of S and T. In addition, the maze routing can only find the shortest path between two specific pins. For multi-pin nets, it is usually decomposed into two-pin nets and then maze routing is used for each two-pin net. The disadvantage of such a solution is that it ignores the information of the tree. Each edge is independently routed and often misses the optimal path. With this in mind, FastRoute 2.0 uses a multi-source and multi-sink maze routing strategy. Fastroute 2.0 is the same as the first two phases of FastRoute 1.0. In the final stage, Fastroute 2.0 first applies monotonic routing to each edge of each routing tree, and then performs a round of multi-source and multi-sink maze routing. However, in this round of maze routing, maze routing is not performed on all sides. Rather, only the lengths that

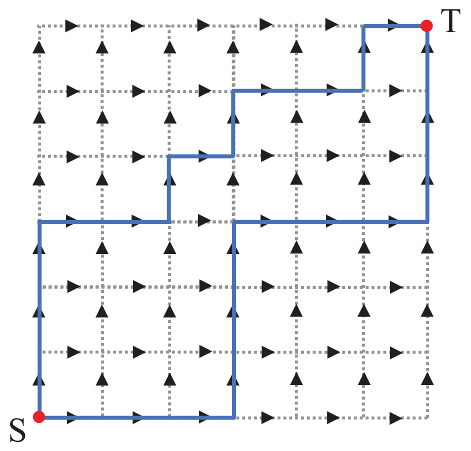


FIGURE 19. **Two different paths of monotonic routing.**

exceed a threshold and the edges that span the crowded area will be routed by the maze routing, while the other edges still use monotonic routing. The algorithm avoids short nets and long nets not passing congested area by maze routing. Otherwise, unnecessary detours may be created to use more wirelength and lead to crowded routing. Of course, another important reason is to reduce the runtime of the maze routing. If there is still a lot of overflow, Fastroute 2.0 will carry more rounds of maze routing.

Zhang *et al.* proposed the concept of virtual capacity in Fastroute 3.0 [107], which guides the rip-up and reroute of the maze routing stage through ‘‘virtual’’ capacity rather than real capacity. The process of Fastroute 3.0 has a total of six main steps. Steps 1, 3 and 4 are inherited from Fastroute 2.0. Step 1 is still the decomposition of multi-pin nets, generating congestion-driven RSMT by using FLUTE. Then, the RSMT of all multi-pin nets is broken down into a set of two-pin nets. Step 2 is virtual capacity initialization, which initializes the virtual capacity by subtracting the estimated overflow from the real edge capacity. Step 3 is to perform L-shaped pattern routing and Z-shaped pattern routing. Step 4 is still the multi-source and multi-sink maze routing in Fastroute 2.0. The step 5 is the virtual capacity update, which is performed every iteration of the maze routing. The varies of the virtual capacity value depends on the comparison between the capacity used by the current edge and the original edge capacity. Steps 4 and 5 will be applied iteratively until the entire overflow gets stuck. The last step is to perform layer assignment.

A Steiner Min-Max Tree (SMMT) approach was proposed in [113]. An SMMT is a Steiner tree whose maximum-weight edge is minimized over all Steiner trees. Chiang *et al.* solved the GR problem by constructing Steiner min-max trees for each net sequentially, defining each edge weight by estimating the routing congestion. The algorithm sorts the perimeter of bounding box of the net and routes only one net at a time, where the net with the smaller bounding box perimeter is routed earlier.

In the method of Chiang *et al.* [114], the goal is to minimize congestion and wirelength simultaneously. The method uses the minimum weighted rectilinear Steiner tree approximation algorithm to sequentially route the nets. The method divides the routing area into a set of region and assigns a corresponding weight to each region. The weight is defined according to the complexity of the region and the congestion, and the weight of the blocked region is set to infinity. For wires routed along the boundary of two different weighted regions, it is assumed that a weight with a smaller value is selected. Then the weight of an edge is the product of the length of the edge and the weight of the region to which the edge belongs, and the goal of the process is to minimize the total weight of the rectilinear Steiner tree. On the routing graph, there is always the smallest weight path connecting two nodes. As with the SMMT-based method, the nets are first sorted in increasing order of the bounding box, and each net is then routed in this order. The weights for all areas are updated after each net is routed.

Up to now, several best-performing routers are: NTHU-route 2.0 [26], FastRoute 4.0 [46], NCTU-GR 2.0 [49] and MGR [115], respectively. NCTU-GR 2.0 [49], has the best performance among all academic routers, and it uses two Bounded-Length Maze Routing (BLMR) algorithms (optimal-BLMR and heuristic-BLMR) that perform much faster routing than traditional maze routing algorithms. In addition, they develop a parallel multithreaded global router on a multicore platform based on the proposed sequential GR algorithm. MGR [115] is a multilevel 3D router that runs much faster than traditional 3D router [14]. Besides, for the last two years, researchers have tried to solve the GR problem by using machine learning-based approaches [116]–[118].

B. CONCURRENT METHODS

1) ILP-BASED ROUTING

In the sequential techniques, routes are generated one at a time based on a predetermined ordering. Sequential methods are very fast, but because of their sequential nature can result in sub-optimal solutions. The concurrent techniques attempt to solve the problem using global optimization techniques. These methods can provide a global view of the circuit's routing, but take a considerable amount of time. Concurrent routing of all nets is a complex issue from the point of view of computational complexity. The use of ILP [28]–[30] is one of the ways to achieve this. In fact, the mathematical model of the GR problem can always be modified into a 0-1 integer programming problem. Linear programming consists of a set of constraints and optional objective functions. This function judges whether it is the largest or the smallest according to the constraint. Both the constraint and the objective function must be linear. These constraints form a system of linear equations and linear inequalities. ILP is a special linear programming in which each variable can only take integer values. In ILP, all variables are binary, called 0-1 ILP.

The ILP has three inputs: (1) A routing graph of size $W \times H$, which is a grid of G-cells, (2) The capacity of the routing edge, (3) *Netlist*, which stores the information of nets: including which pins each net needs to connect, the coordinates of each pin and so on. Suppose the horizontal edges extend from left to right: $G(i, j)$ $G(i + 1, j)$, while the vertical edges extend from the bottom to top: $G(i, j)$ $G(i, j+1)$.

ILP uses two sets of variables. The first set contains k Boolean variables x_{net_1}, $x_{net_2}, \ldots, x_{net_k}$ and each variable acts as an indicator for one of k specific paths or routing options, each net $net \in Netlist$. If $x_{net_k} = 1$, the routing option x_{net_k} is available, and if $x_{net_k} = 0$, the routing option x_{net_k} is not available. The second set consists of k variables $w_{net_1}, w_{net_2}, \ldots, w_{net_k}$ each of which represents the net weight of the specific routing option, each $net \in Netlist$. The net weights reflect net satisfaction with each routing option (larger weights mean that net has a higher level of satisfaction with routing options, for example, with fewer routing turns). It is known that each net $net \in Netlist$ consists of k available routes, and the total number of variables in each set is $k \times Netlist$. Building an ILP depends on two types of constraints. First, each network must choose a separate routing (mutually exclusive). Second, to prevent overflow, the routes (total usage) assigned to each edge cannot exceed its capacity. ILP maximizes the total number of routing nets but may leaves some unrouted nets. That is, if the optional routing causes a solution overflow, then this routing will not be selected. If all routes for a particular net cause overflow, then no routes will be chosen and thus the net will not be routed.

Sidewinder [119] and BoxRouter 1.0 [34] both decompose multi-pin nets into two-pin nets by using FLUTE, and the routing of each net is selected or not selected from two alternative schemes. If neither of the two possible routing options for a net is selected, Sidewinder performs maze routing to find an alternate route and replaces the unavailable routing in ILP. Besides, a net that is successfully routed and does not conflict with an unrouted net can be removed from the ILP. Thus, Sidewinder only solves the multiple ILP problem until it cannot be further optimized. In contrast, BoxRouter 1.0 post-processes the results of its ILP using maze-routing techniques.

BoxRouter 1.0 first performs fast and effective prerouting to identify the most crowded areas or boxes, then incrementally expands the routing box and performs different routing strategies inside and outside each expanded box, repeating the process until it covers the entire circuit. In addition, BoxRouter 1.0 uses the key box routing idea, which is to expand from prerouting estimates of the most crowded areas, where routing takes precedence over other areas that are more crowded. Adaptive Maze Routing (AMR) is used to implement ILP. ILP and AMR only route the area inside the box. For the area inside the box, BoxRouter 1.0 first routes with ILP to fully utilize the routing capacity inside the box. Then use AMR to detour the parts of the box where ILP cannot be routed. However, the limitation of BoxRouter 1.0 is that its gradual ILP routing only considers L-shaped mode, which is not ideal for most nets that need to be detour in complex mode.

BoxRouter 2.0 was proposed by Cho *et al.* [13]. It uses A* search based on congestion negotiation for routing, and achieves layer assignment through efficient progressive via/blockage-aware integer ILP. At the same time, because the sudden change of the congestion graph will mislead the router, resulting in inaccurate congestion estimation, BoxRouter 2.0 uses a rip-up that considers the topology. When a path is selected to be re-routed, exploring more flexibility by disassembling some adjacent paths in its same net. Boxrouter 2.0 is mainly divided into 2D GR and layer assignment, but its routing algorithm can also be directly applied to multi-layer (3D) situations. The advantage of 2D GR over 3D GR is that it requires less computing power and memory because the routing graph shrinks significantly.

A global router based on the concurrent techniques was presented by Behjat and Chiang [120]. The proposed technique includes an estimate of congestion in the routing area to produce high quality routes. Besides, the routes that are likely to be in the congested areas are discarded to reduce the input sizes of the ILP. To increase the speed of the router while keeping the quality of results optimal, an algorithm that examines the characteristics of the possible routes in the ILP formulation is used. This algorithm reduces the time to solve the ILP problem by performing ILP in two stages. In the first stage, the trees that have low cost in the objective function are chosen. This set of chosen trees taken out of the ILP problem and the size of the ILP problems are reduced. In the second stage, the modified ILP problem is solved. By eliminating trees in the first stage, significant improvements in solving time are obtained. The router is able to gain global view of whole circuit with solving times that are comparable to global routers based on sequential techniques.

A scalable GR technique via ILP called GRIP was proposed by Wu *et al.* [40]. GRIP works by decomposing the chip into rectangular subregions together with their net assignments to form smaller-sized subproblems. It then applies an ILP-based procedure to solve the subproblems in a systematic order. By applying ILP in a systematic manner, GRIP obtained a significant improvement in solution quality for benchmark instances but with prohibitively long execution runtimes. Later a parallel global router that is based on an extended ILP procedure of GRIP was proposed by Wu *et al.* [121], which can remove the synchronization barriers between subproblems, effectively utilizing many more processors and reducing runtimes to an acceptable practical level.

The high volume and complexity of cells and interconnect structures in modern designs are causing serious challenges to routability. Rapid congestion analysis is becoming crucial to fix the routability problem at the early stages of the design, for example during placement [122] and in conjunction with GR [123], [124]. In modern designs, several new factors contribute to routing congestion including significantly-different wire size and spacing among the metal layers, sizes of inter-layer vias, various forms of routing blockages (e.g., reserved for power-grid, clock nets, or IP blocks in an SoC), local congestion due to pin density and routing inside a global-cell, and virtual pins located at the higher metal layers. However, none of the early estimation techniques such as [125]–[128] capture these new sources of congestion comprehensively. In [129], a fast framework was presented, which aims to accurately predict the congestion hotspots. The framework utilizes the flexible model of GR that captures many necessary modern design features. The framework relies on an ILP formulation. Since solving the ILP formulation to optimality is impractical for realistically-sized problem instances, this work also proposed several new ideas for a practical implementation of the framework to obtain a high quality approximate solution to the formulation. An approximation method shrinks the original ILP formulation without seriously degrading its solution quality, as well as a way to integrate the solution of the formulation with the traditional rip-up and reroute procedures used by many GR frameworks. Furthermore, this work speeds up the conventional rip-up and reroute GR procedures by performing multiple rip-ups and a single simultaneous rerouting of equivalent nets. It also provides a congestion-aware layer assignment procedure to account for varying wire size, spacing, blockages, and pins at different metal layers.

In addition, there are some ILP-based algorithms [8], [130]–[133]. In [130], a top-down hierarchical method is combined with the flat ILP-based GR approach, which can optimize several GR objectives simultaneously and speedup the computation time of the proposed ILP-based global router. Sen [131] considered the GR problem of selecting a maximum set of nets, such that every net can be routed entirely in one of the given layers without violating the physical capacity constraints. The algorithm gives results which are guaranteed to be within a certain range of the global optimal solution and runs in polynomial-time. And it demonstrates that the complexity can be significantly reduced in the worst case prediction. Lu *et al.* [132] proposed a two-stage transistor routing approach for CMOS standard cells. This two-stage approach synergizes the merits of channel routing and ILP approach. The segment-based ILP makes their router more scalable than a grid-based ILP. Liu *et al.* [8], [133] combined PSO and ILP to solve the GR problem under X-architecture. They adopted the partitioning strategy to reduce the size of the GR problem, thus contributing to the efficient solution of the ILP model by PSO.

2) MULTICOMMODITY FLOW-BASED ROUTING

The objective of the GR problem is to allocate a limited set of resources optimally to a given set of demands. In a way, the nature of this problem is quite coherent with the problem of finding optimal flows in a network [134]. A network is a connected graph consisting of vertices and edges. In a basic network flow model, there are two types of vertices: One called the source and the other one called the sink. For one commodity, a certain amount of flow must be shipped from the source to the target vertex. Each edge has a flow capacity, which represents the upper limit of the flow allowed through the edge. One form of the problem requires that as much flow as possible be transmitted through the network without exceeding any edge capacity, which is referred to as the max-flow problem. The other form assigns a cost per unit flow to each edge, and sets the problem objective to be the minimization of the total transportation cost through the network for a given flow from the source to the sink. This formulation is called the min-cost flow problem. An advantage of the network flow problem is that it can be solved in polynomial time to obtain an optimal integer solution when edge capacities are integers. A good description of network flow methods can be found in the book by Ahuja *et al.* [135]. While it is not possible to model the entire GR problem as a single commodity network flow problem, this technique can be employed to solve some subproblems in GR [136], [137] and can lead to high-quality solutions.

There is a special type of network flow model, the multicommodity flow problem, which can be applied directly to the GR problem. In multicommodity flow problem, many commodities must be shipped on a common network, and each commodity has its own unique sources and targets. When it is used to solve the GR problem, each net can be regarded as a commodity. An advantage of this approach is that the multicommodity flow problem can be formulated as a linear programming problem. Since linear program solvers are slow for the size of problems encountered in GR, research on the multicommodity flow problem has mostly focuses on heuristics and combinatorial approximation algorithms. While sequential methods, rip-up and reroute and other heuristics may be effective in practice, they do not provide a specific answer as to whether a feasible solution exists. In other words, if these methods fail to find a viable solution, it is not clear whether this is because the feasible solution does not exist or because of the shortcomings of the heuristic method. In addition, when the heuristic method finds a feasible solution, it is not known whether the solution is optimal or not, and how far it is from the optimal solution. However, if we consider the GR problem as a multicommodity flow problem, we can answer these questions. Multicommodity flow-based technology can handle simultaneous routing of multi-nets as a multicommodity flow problem. The main idea is to model the nets as different commodities that flow in the network of a routing resource graph. The flow problem is typically solved by linear programming which results in fractional flow. Therefore, the solution is discretized by the randomization rounding method.

Given a graph $G = (V, E)$, and $V = \{v_1, v_2, v_3, \ldots, v_n\}$ is a set of n vertices and $E = \{e_1, e_2, e_3, \ldots, e_m\}$ is a set of m edges. Some work defines this graph as a directed graph, while others define it as an undirected graph. The difference only affects the formalization of the problem. The form of undirected graph is often used to solve the GR problem of multi-pin net. In this graph, k commodities must be transported from some vertices to other vertices. In GR, each net is regarded as a commodity and there is a set of $N = \{N_1, N_2, N_3, \ldots, N_k\}$ commodities that are to be shipped. For each commodity N_i, a certain amount d_i, namely the demand, is required to be shipped. And each N_i corresponds to a net with more than one pin and each d_i is always one. Each edge has a flow capacity $u(e)$ and cost $c(e)$. In general, two constraints must be met in a multicommodity flow problem. The first is the demand constraint, which requires that the the amount of flow shipped for each commodity should be equal to its demand. And the second is the bundle constraint, which means that the amount of flow passing through each edge must not beyond its capacity $u(e)$.

Reference [138] was the first to use the multicommodity flow model for GR. The algorithm is based on directed networks and is limited to two-pin nets. On the surface, the algorithm is similar to the rip-up and reroute algorithm, and gives an analysis of the convergence and convergence speed of the algorithm, but there is no relevant analysis and statement on the optimality. This router can be used for the routing of chips and boards with rectilinear or nonrectilinear channel topology in a multi-layer environment. This makes it more universal than other known global routers. The router is based on a multicommodity flow model in the graph form with hierarchical cost function. This model is proved to be NP-complete. By maximizing on each iteration the decrease in the number of channels at the highest level of overflow and the number of cells with overflown via count, it moves from the optimal solution to an initial cost function in the direction of the constraints. If a solution exists for each iteration, then algorithm will converge in polynomially bounded number of steps to the solution of the multicommodity flow problem. If a solution does not exist for some iteration, then an escape procedure is applied and the process continues. The objective of escape procedure is find nets which have wrong topology and are blocking improvement of the route distribution, then change their topology by changing pairs of connected nodes.

A formulation for multi-terminal nets routing to multicommodity flow model was conducted by Raghavan and Thompson [139]. The GR problem is formulated as a multicommodity flow problem: For each net, units of flow are to be shipped from each pin to Steiner point. Their objective is to minimize the maximum flow among all edges. The runtime of the algorithm is exponential in the number of terminals in a net but polynomial in the number of nets and the size of the array. The algorithm is thus best suited to cases where the number of terminals on each net is small. Carden *et al.* [140] established the first reported global router with a theoretical bound from the optimal solution, and they extended Shahrokhi and Matula's two-terminal multicommodity fractional flow algorithm [141] to obtain a multi-terminal multicommodity integer flow solution. The router uses a randomized rounding technique to derive a discrete net connection with an error bound on the derivation from the optimal fractional solution. And it improves the final results by employing an iterative procedure. In the iteration, the cost of each edge is an exponential function of the congestion.

Albrecht [39] modified the Garg and Konemann's multicommodity flow approximation algorithm [142] to solve the linear programming relaxation of GR problem. They used Newton's method as an additional optimization step. In this way, not only the maximum relative congestion is minimized, but also the edge congestion is evenly distributed. Besides congestion, wirelength is also considered in this method. Albrecht's method is simpler and faster than Shahrokhi and Matula's algorithm [141].

C. PERFORMANCE-DRIVEN ROUTING

As device dimensions shrink rapidly, interconnect latency plays a critical role in chip performance. Therefore, it is becoming more and more important to minimize wirelength and delay. Since interconnect latency has become a dominant factor in determining system performance, it is no longer sufficient to consider only congestion. Daboul *et al.* [143] balanced a multitude of different objectives via the resource sharing framework,such as timing, power and wirelength. In the routing process, the addition of timing constraints and power constraints is more in line with actual industrial manufacturing and is of great significance in both theoretical research and actual production.

1) POWER-DRIVEN ROUTING

Interconnect power consumption is a major challenging research issue in deep submicron solutions that affect overall circuit performance. Due to high packaging density of components, power is increasingly becoming the bottleneck for the design of high performance VLSI circuits. It is essential to analyze how the various components of power are likely to scale in the future, thereby identifying the key problematic areas. While most analysis focus on the timing aspects of interconnects, power consumption is a very important indicator.

Global interconnect delay has dominated over gate delay in recent technologies. Though optimal buffer insertion minimizes interconnect delay, it poses large power overhead [144]. In deep submicron design, many thousands of repeaters have to be used. In general, the repeaters are optimally sized and separated to minimize the interconnect delay. However, since these optimally sized repeaters are quite large and also dissipate a significant amount of power, the total power dissipation by such repeaters in large high-performance designs can be prohibitively high and they can cause severe routing congestion. Designers must seek a trade-off between performance and power. The optimal repeater insertion technique is used to minimize the wirelength (shortest path between source to sinks), power dissipation and wire size. This wire size is inversely proportional to the power dissipation and buffer insertion is directly proportional to the power dissipation. Nallathambi and Rajaram [145] used PSO to consider buffer insertion, buffer size adjustment, and wire sizing to minimize interconnect power consumption. The shortest path constraints, buffer insertion constraints, and wire size constraints are considered for analysis and optimal power distribution.

Yang *et al.* [146] proposed an on-chip routing scheme based on a new power model and a dynamic XY routing algorithm that provides adaptive routing based on adjacent congestion conditions, and ensures deadlock-free and lovelock-free routing at the same time. The solution adjusts the routing strategy based on power conditions, optimizes power allocation, and avoids hot spots in the network. Effectively adjust the power allocation of the network to meet the power balance requirements, and the loss of network performance is negligible.

Wu *et al.* [147] proposed a method of GR to minimize interconnect power. They used a Multiple Supply Voltage (MSV) design in which a level shifter was added to the network to connect the drive unit to the receiver unit with a higher supply voltage. The level converter is modeled as an additional terminal in the GR process, and an initial solution is given. With the minimum wirelength as the target, a winding method is proposed, which further saves the interconnection power. When the path is bypassed by this process, the overflow is not increased, and the increase in the wirelength is limited. Opportunities for energy savings include: 1) reducing the area of the capacitor by bypassing the lower the metal layer from the higher metal layer, 2) reducing the coupling capacitance between adjacent lines, and 3) considering the routing of each segment of different power-weights network level converter (capturing the corresponding supply voltage and activity factor). A mathematical formula is presented in this paper to capture these energy savings opportunities and solve it using integer programming techniques.

As Swarm Intelligence algorithms have been shown to have good application prospects in solving NP-hard problems, Arora *et al.* proposed an algorithm based on ACO in [148]. The combination of ant colony learning strategies provides a multi-agent framework for combinatorial optimization. It applies the ant colony algorithm to the NP-hard problem of VLSI chip interconnect routing, which is to find the optimal routing with the smallest capacity. The goal of this algorithm is to minimize power consumption by simultaneously minimizing wirelength, capacitance, and vias.

The latest developments in nanotechnology in different industries and the growing demand in the microelectronics market for high performance, high complexity and low power on chip systems are driving EDA vendors to explore and innovate at all stages and aspects of the design development cycle. Nowadays, some IC factories have achieved mass production of 7 nm chips and attracted many industries to use the technology in their future devices. This market trend presents many challenges for EDA vendors as well as PD and validation experts who need to consider many new physical constraints and design rules to meet foundry requirements. On the other hand, 7 nm node has new opportunities and advantages that the early design node did not have. With regard to 7 nm designs, Chentouf *et al.* [149] had achieved better clock power reduction without affecting circuit timing or area. They take advantages of the large resistivity difference between Self Aligned Double Patterning (SADP) and non-SADP layers to reduce the total clock parasitic load that the clock driver balances, reduce the number of inverters and buffers needed to drive all the clock leaf units, and, most importantly, reduce clock power.

2) TIMING/DELAY-DRIVEN ROUTING

With the rapid development of technology, timing has become more and more important in modern design. A number of techniques have been used to reduce circuit latency during routing. The GR greatly impacts the timing behavior of a chip. If the routing performs detours to avoid congestion, timing constraints might get violated. In fact, critical nets are competing for using congested routing resources. A good timing-driven routing tree structure can significantly affect layout and routing. As circuits become more complex, previous algorithms may not be effective in modern design. In modern integrated circuits, interconnections can cause considerable signal delays. Therefore, interconnect delays should be taken into account in routing. There are two main optimization objectives of timing-driven routing. One is: maximum sink delay, that is, the maximum interconnection delay from source to any sink in a given net. The other one is the wirelength, which affects the load-dependent delay of the net's driving gate.

Given a signal net, let s_0 be the source node, and $sinks = \{s_1, s_2, \ldots s_n\}$ be the sink. Let $G = (V, E)$ be the corresponding weight map, where $V = \{v_0, v_1, \ldots, v_n\}$ represents the source and sink points in net, and the edge $e(v_i, v_j) \in E$ represents the routing costs between endpoints v_i and v_j. For any spanning tree T on G, let $radius(T)$ denote the length of the longest source-sink path in spanning tree T, and $cost(T)$ denote the sum of the weights of all edges of T, that is, the total routing cost.

Since the source-sink length reflects the signal delay between source and sink, the length is linear with the delay and is closely related to the Elmore model [150], [151]. Therefore, in timing-driven routing, it is an ideal choice to use the routing tree to simultaneously obtain the minimum of the radius and cost. However, for most signal nets, the radius and cost cannot reach the minimum at the same time. How to achieve a good compromise between the two is the key to timing-driven routing. Figure 20 shows a compromise between radius and cost, with the sign on it representing the cost of the edge. Figure 20(a) has the smallest radius, the shortest possible path length from source to individual sinks. Therefore, it is a shortest path tree that can be constructed with Dijkstra. Figure 20(b) has the least cost, so it is a MST that can be constructed with the prim algorithm. Figure 20(a) and Figure 20(b) are not practical due to their large cost and large radius, respectively. Figure 20(c) has achieved a good balance between the two.

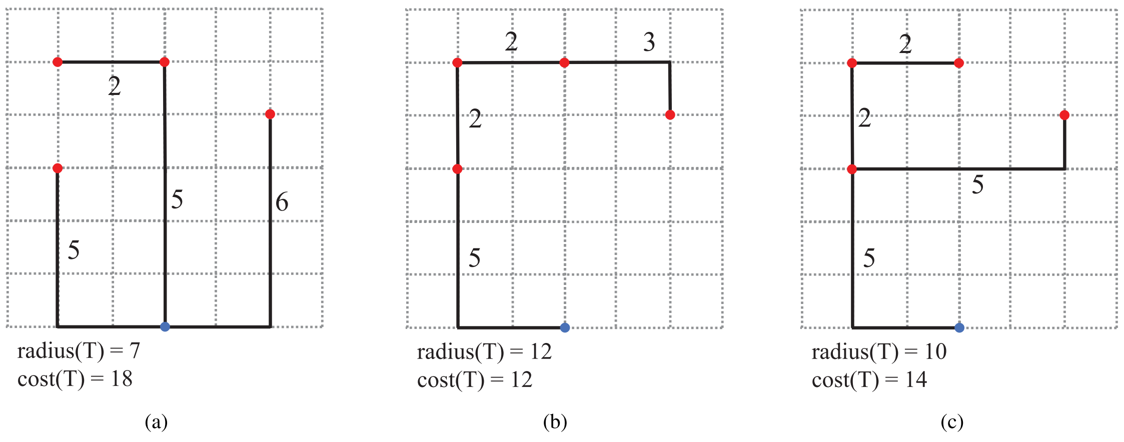


FIGURE 20. Tradeoff in tree construction. (a) A shortest-paths tree. (b) A minimum-cost tree. (c) A compromise with respect to radius and cost.

Qiao and Hong *et al.* [152] proposed a loop routing performance optimization approach. This method creates loops in the existing routing trees to reduce the delay of the selected critical path or the maximum delay of the net. The interconnect tree is formulated as a tree of distributed transmission lines, and the Elmore delay is used for delay calculation. This method introduces new links with appropriate R, C values by reference nodes and key nodes in the existing tree topology, significantly reducing the delay of the selected critical path or the maximum delay of the network. And it gives the wirelength selection on the basis of pre-calculated time delay of node and the resistance and capacitance array.

In [153], Hong *et al.* developed a timing-driven global router called TIGER. This global router is based on a multicommodity flow model, similar to [140]. This approach requires the construction of a Steiner tree for each net to minimize congestion. In addition to congestion, this approach also includes timing issues to always satisfy path-based timing issues. The delay model used in the router TIGER uses the upper bound of the signal propagation delay in the RC tree. The definition is as follows: Let R_d denote the driver resistance and C_L denote the load capacitance of each sink. The resistance and capacitance of the wire per unit length are r and c respectively. Consider a net $N = \{v_0, v_1, v_2, \ldots\}$, where v_0 is the source node and others are the sink nodes. A routing tree T of length W is constructed for N: For a sink point v_i in the net N, if the length of routing path from the source point v_0 to the sink point v_i is $p_r(v_0, v_i)$, the upper limit of the delay is defined as follows:

$$t_{up}(v_i) = \beta(R_d + rp_r(v_0, v_i))(cW + C_L) \tag{5}$$

where $\beta = 2.21$, indicating that the signal reaches 90% of the final value, which represents the point at which the signal reaches 90% of its final value.

One of the ultimate goals of timing-driven design is to achieve a specific clock cycle time so that each register-to-register delay along each path is less than a certain constraint. This is called a path-based constraint. There are usually multiple wire meshes along the path between two adjacent registers, as shown in Figure 21, where the critical path is (v_1, v_2, v_3, v_4) and the nets N_1 and N_2 are along this path. However, in net-based approaches, the path delay constraint is decomposed into a set of timing budgets assigned to the nets. If these budgets are allocated poorly, some nets located in congested areas may not have adequate timing slack to achieve a feasible routing avoiding the congestion. In TIGER, the routing tree is chosen so that the timing constraint along the critical path can be satisfied, which provides greater flexibility than the net-based constraints. Similarly, the method of Wang *et al.* [154] is also based on the path. It preferentially routing each critical synchronization path and minimizes its delay.

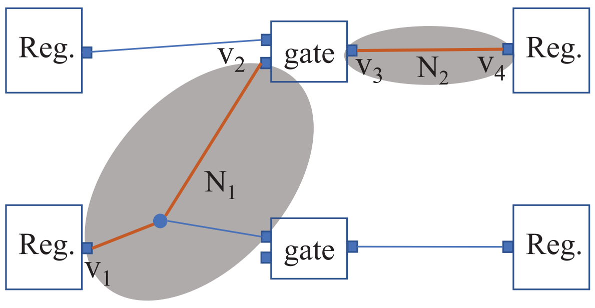


FIGURE 21. An example of timing graph.

Many modern routers use the FLUTE algorithm as the basic method for constructing the Steiner tree, while in [155] another method called Modified Algorithm Dijkstra (MAD) is proposed to construct timing-driven Steiner tree for each net. The Elmore delay of the tree constructed by MAD and FLUTE is almost the same. Next, they used a gradient algorithm to reduce congestion. Reference [155] utilizes the MAD is iteratively to generate a set of perspective trees for each net, and then applies a gradient method to select a unique tree for each net, so that the capacity of the routing path is minimal.

Cong and Madden reported a timing-driven GR method for standard cell design in [156]. The method first constructs a minimum spanning tree for each net. The experimental results show that in the standard cell design, the wirelength reduction from MST to the approximate Steiner minimum tree is limited. If an MST cannot satisfy its time constraints, a timing-driven tree topology [157] and wire sizing techniques [158], [159] are used to ensure that timing constraints are met. On this basis, the routing topology of the critical net is determined, and a simplified connection diagram is constructed for non-critical nets. Finally, congestion is minimized by an iterative deletion process [160].

Hu and Sapatnekar [137] optimized congestion and delay simultaneously, and these two indicators are usually competing with each other. Their method provides a general framework that can use any single-net routing algorithm and any delay model in GR. It is based on the observation that there are several routing topology flexibilities under timing constraints. These flexibilities are exploited for congestion reduction through a network flow based hierarchical bisection and assignment process. The algorithm can be roughly divided into three stages: In the first stage, each network is routed to meet its timing constraints, but there is no need to consider congestion. At this stage, any single-net performance driven routing method can be applied here. In the second stage, a routing region is recursively bisected into subregions in a top-down manner. Then assigning soft edges to boundaries along the bisector, where the length of the soft edge is fixed but the precise edge route is not determined. The last step is a timing-constrained rip up and reroute process. It rips up the edges on a set of most congested boundaries and reroutes them through maze routing. The difference from traditional rip-up and reroute is that it imposes restrictions on the length of the edges to ensure that timing is not violated.

Huang *et al.* [161] formulated the timing-driven multi-layer OARSMT problem and proposed an algorithm to solve the problem. Their optimization goal is to minimize the maximum source-to-sink length. Their approach includes multi-layer timing-driven partitioning, the construction of a multi-layer routing tree with barriers, and the construction of a timing-driven multi-layer balanced tree.

Net-based delay bounds were proposed by Huang *et al.* [162], rejecting Steiner trees that violate the delay bound. Path-based delay bounds were proposed by Hong *et al.* [153], discarding Steiner trees for a net that result in a delay violation of a path through that net. Differently, net-based and path-based delay bounds were considered by Vygen [163]. Here, delay bounds on paths are treated in the same way as routing space constraints. That is, delay violations are not banned but minimized simultaneously with congestion.

Samanta *et al.* [164] precomputed a set of alternative timing-driven Steiner trees for each net. Then, they found near-optimal fractional solutions minimizing the total quadratic over-congestion, choosing convex combinations of the pre-processed alternatives. Besides, they proposed a Gradient algorithm for minimizing the total overflow. This is a concurrent approach considering all the nets simultaneously contrary to the existing approaches of sequential rip up and reroute.

Recently, Held *et al.* [165] proposed a new model based on the min-max resource sharing framework. This framework integrates all static timing constraints as a linear number of additional resources and customers into the resource sharing model for GR, and works for RC-delay, linear delay, and other delay models in which a net determines the delay through its driving gate and the net itself. The algorithm can dynamically adjust the delay budget to balance the congestion and delay. An implicit delay budgeting is done as part of the algorithm. In addition to timing, net length, power consumption, or manufacturing yield can simultaneously be bounded or optimized. Their implementation contains speed-up tricks and runs fast also for nets with many terminals. Besides, Yan and Yen [166], [167] proposed two delay-driven routing tree for graphene nanoribbon based devices and interconnects.

VI. RELATED DISCUSSIONS

Early GR only considered overflow and wirelength. With the development of VLSI design, modern GR faces different constraints and various optimization goals. Figure 22 shows the distribution of different optimization objectives for GR over the past decade. It is easy to see that more and more factors are considered in the routing process, such as: vias, delay, timing, power consumption etc. to further optimize chip performance. With the continuous development of modern technology, the industry urgently needs an efficient automated routing process. The researchers summarized the characteristics of routing problems in different design environments, and built a series of routing models and evaluation models to adapt to various new types of processes, such as: MDSV GR in IC design, new problem models under the advanced via pillar process.

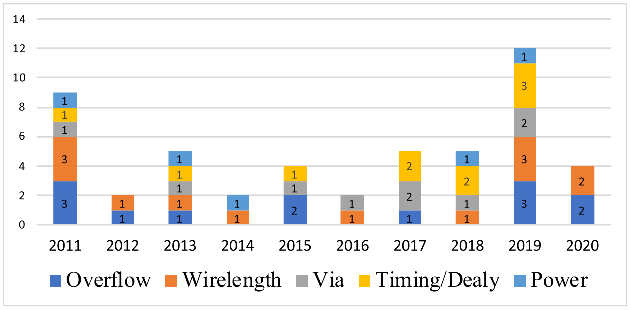


FIGURE 22. Distribution of different optimization objectives for GR over the past decade.

A. GR FOR MDSV DESIGN

In today's large-scale integrated circuit design, with the development of process technology, the transistor density of nanoscale Complementary Metal-Oxide Semiconductor (CMOS) circuit has increased dramatically in the past 30 years, resulting in the increase of power consumption density in the circuit. According to research, the power density on the microprocessor increases twice as fast every three years. Such high power consumption will cause the chip's heat dissipation performance to receive great pressure, so that the temperature will overheat, making the circuit reliability and safety decreased. At the same time, when the chip overheats, the chip performance will also drop sharply. Therefore, the problem of power consumption has to be taken seriously. However, at present, most of the GR algorithm is to reduce the overflow, wirelength and calculation time as the optimization goal, and all the functional components are working in the same voltage mode. As the density of chips increases, power consumption increases. High power consumption shortens the battery life of portable devices, causing heat dissipation and reliability problems. In the traditional voltage supply mode, it is easy to cause excessive unnecessary power consumption. This is because all the functional components of the chip work in the same high voltage mode, and some other devices that could work in the lower voltage mode also work at high voltage, increasing the power consumption of the chip and thus reducing the battery life.

In order to change this voltage supply model, industry chip companies and researchers have proposed a multi-voltage design model, which can effectively reduce power consumption by controlling the voltages of different functional components through complex control strategies. Therefore, multi-voltage design is widely used in high-end applications or low-power applications. In recent years, compared with MSV design, MDSV technology can further reduce power consumption. In MDSV design, the voltage of each power local region can be changed dynamically according to the corresponding power mode. In some power modes, such as energy-saving mode, waiting mode, sleep mode and so on, some power locales can even be set to completely off state to save power.

Total power consumption typically includes static power and dynamic power. Static power consumption is caused by leakage of current, while dynamic power consumption is caused by switching of the active state of the device. In modern circuit design, most of the total power consumption is dynamic power, which is proportional to the square of the supply voltage. Although lowering the supply voltage can greatly reduce dynamic power consumption, it also reduces the speed and performance of the device. MDSV is typically used in high-end or low-power applications because they effectively reduce dynamic power consumption through complex control of voltage across different functional units.

The proposed MDSV brings new opportunities and challenges to the PD of VLSI circuits. The proposed MDSV brings new opportunities and challenges to the PD of VLSI circuits. It puts forward new requirements for the PD process with MDSV as the basic model including floorplanning, layou, placement and routing, etc. And it also poses a great challenge to the study of EDA tools, especially for the global routers. Reference [168] was the first time to construct the corresponding GR for multi-dynamic voltage design mode. As the first study on MDSV-GR problem, they defined the Power Domain-Aware Routing (PDAR) problem, and proposed point-to-point PDAR algorithm, forward-looking path selection method and lookup table acceleration method. For multi-pin net routing, a new constant schedule lookup mechanism is used that quickly calculates the lowest cost monotonic path from each node to the target subtree by calling four enhanced monotonic routing. However, it had oversimplified the mathematical problem of the MDSV design pattern, and failed to conduct corresponding experimental research from the goal of reducing power. In MDSV design mode [169], the net may pass through more than one power localization, some of which may be off while others are still in active mode. For a live net with long detour length in the closed power domain, a functional conflict may occur if the repeater is placed in the closed power domain. Therefore, limiting the detour length of the live net in the closed power domain is a very important general routing problem in MDSV design. A maze routing algorithm with length restriction was proposed in [170] to control the wirelength of the routing path. Meanwhile, [71] also proposed an SMT construction algorithm which take into account the limitation of the wirelength within the obstacle. However, neither of them took into account the different length constraints under different power supply localization. Therefore, the GR in reference [71], [170] may still produce some illegal routing results under MDSV design.

The existing research on MDSV design mode is mainly focused on some local stages such as layout and clock tree construction, lacking an effective and complete solution to the GR problem under MDSV design. If the novel MDSV is introduced into the general routing stage, new general routing problems will be generated, including new constraints and mathematical models, and effective algorithms need to be proposed. According to the investigation and research conducted by the research group and chip design related companies including Cadence Design Systems in the United States, we found that the research on the GR algorithm under MDSV design can effectively reduce a large amount of power consumption, while the current academic circle lacks the research on the GR algorithm under MDSV Design. In fact, in order to solve the problem of dynamic power consumption, scholars have developed some new tools based on MSV, MDSV and other related technologies. However, in these tools, the relevant routing algorithm needs to make some changes to solve the routing problem in the new design mode. Therefore, looking for an effective MDSV design under the GR algorithm, the construction of an efficient low-power GR, not only has important theoretical value, but also has practical manufacturing significance.

B. GR FOR VIA-PILLAR TECHNOLOGY

In routing process, in addition to considering the wirelength optimization goal, it is more necessary to consider the performance optimization goal such as time delay. With the development of process technology, the resistance of metal lines and vias increases exponentially, which brings more stringent constraints to traditional routing-related algorithms, leads to the existing methods being prone to excessive delay/timing problems and seriously affects the performance of the chip. To this end, Synopsys and Taiwan Semiconductor Manufacturing Company jointly launched via-pillar, a key process in 2017, as a representative technology for chip performance in designs of 7 nm and below [171]. The via pillar approach using metal layer promotion and multiplewidth configurable wires. This fully automated via pillar design flow mitigates the high resistance impact, thus greatly providing the delay optimization ability and realizing the overall performance optimization of the chip. Besides, due to multiple vias can be placed side by side, the manufacturability of the chip is improved. The proposal of via-pillar technology brings new opportunities and challenges to the physical design of VLSI circuit, that is, it puts forward new requirements in planning, layout, routing and parameter extraction, etc., as well as great challenges to EDA design tool research, among which, routing is most closely connected [171].

After the introduction of the via-pillar process, many problems in the traditional routing stage, including layer assignment, track assignment, detailed routing, etc., need to be updated. It is necessary to construct a new problem model under the via-pillar process, and then design the corresponding effective algorithm. Related issues under the via-pillar process have become more complex, requiring careful consideration of via location, via size issues, and latency optimization issues. However, the research work on the traditional routing problem mostly regards the via as having no shape and size. The size and position of the via hole are not considered in the processing of the related routing subproblem, which leads to the inconsistency between the final routing result and the actual chip design requirement, and the failure rate of chip production is intensified. At the same time, the industry still implements some manual implementation of the related routing problems under the via-pillar process. At present, there is still lack of an effective automated design process. Therefore, it is of great theoretical and practical significance to seek an effective automated routing algorithm under the via-pillar process.

In recent years, the work of layer assignment have not only targeted the minimization of the number of vias, but further attempt to reduce the delay of the layer assignment scheme [172]–[175]. Reference [176] uses a negotiation mechanism-based algorithm to jump out of local optimum, thus obtaining a better layer assignment scheme. Reference [172] extends the algorithm of [176], and considers the delay and congestion problem, and proposes a time-driven dynamic programming algorithm to deal with the layer assignment problem of multi-layer nets. In order to accurately estimate the delay, [158], [174] further consider the delay of the vias. Reference [175] further considers the influence of capacitive coupling on delay, and proposes a delay-driven layer assignment algorithm based on negotiation mechanism. However, these considerations for delay optimization do not take into account the size of vias. It is assumed that vias do not occupy routing resources and do not affect the routing of the metal wire, which seriously affect the accuracy of GR's final results.

Existing research work on layer assignment problems either does not consider the delay problem, or does not consider shape of the via, which is far from the actual chip design requirements. After the introduction of the via-pillar process, the relevant routing model needs to consider the existence of via size, and the key performance index of delay optimization. Therefore, the existing layer assignment algorithm no longer applicable to the layer under the via-pillar process. For layer assignment problems, researchers need to design corresponding effective algorithms. The introduction of the via-pillar process can effectively optimize the delay and improve manufacturability. Therefore, the effective solution algorithm for global routing under the via-pillar process is also a crucial problem worth exploring.

VII. CONCLUSION AND FUTURE WORK

SMT construction in VLSI has been the focus of researchers. This is one of the most important steps in the process of VLSI physical design and it is important for GR. For the problem of SMT construction, this paper studies SMT, OASMT and LRSMT respectively. Early SMT construction had little regard for obstacles, which were clearly not practical for industrial design. With the development of integrated circuit technology, the modern VLSI design have shifted toward the system-on-chip paradigm, more and more obstacles appear in the physical design process, such as some pre-routed nets and macro cells and intellectual property blocks, which makes the OASMT a hot spot for researchers to study. In practical design, routing areas typically contain multiple routing layers. Equipment often only occupies the equipment layer and some lower metal layers, does not completely block the wires. When routing on a higher layer, wires may pass through larger obstacles. However, the repeater cannot be placed inside obstacles. It is indeed possible to avoid obstacles by increasing the wirelength, but it can lead to capacitance, conversion and timing violations. Therefore, it is necessary to relax the routing resources in the routing process, so that wires can pass through obstacles to a certain extent, thus ensuring timing closure. In other words, LRSMT is closer to actual industrial design. In the future, work on SMT construction should focus more on issues based on LRSMT, and further consider constraints such as multi-layer, timing, and buffer-insertion.

For the GR aspect, there is no perfect solution, and each method has its strengths and weaknesses. If one needs a simple yet reasonably effective approach, perhaps sequential method is the best choice. When computation speed is a crucial issue, hierarchical methods can be applied. Besides, the method based on ILP can find the optimal solution in theory, but only approximate solution can be obtained in practice due to the huge amount of computation. By far the best global routers are sequential methods based on heuristic algorithms. However, some existing work has used distributed algorithms to greatly reduce the time to solve ILP. We have reason to believe that in the future, with the development of hardware, distributed algorithms and parallel algorithms will play a very important role in GR. In addition, as portable and wearable devices become more and more popular, longer battery life has become an indispensable feature of mobile devices. Today's chip designs often require high performance while maintaining low power consumption, which makes power-driven GR becomes a research hot spot. The proposal of MDSV is in line with this trend, which can significantly reduce power consumption. Therefore, it is of great theoretical value and practical significance to find an effective GR algorithm under the MDSV design and construct an efficient low-power global router.

And for interconnect delay, the resistance of metal wires and vias increases exponentially, which places stricter constraints on traditional routing-related algorithms. This results in the existing methods being prone to higher delays and timing violations during the routing process. Via-pillar technology can greatly reduce resistance due to the ability to place vias side by side. However, the existing GR algorithms hardly consider the actual size and shape of the vias. After introducing the via-pillar technology, many problems in the traditional routing stage need to be changed, including layer assignment, track assignment, etc. In the future, it is necessary to define a uniform problem model under the via-pillar process, and then design the corresponding effective algorithm.

Finally, we want to point out that some machine learning-based methods have been applied to the substages of GR, such as congestion prediction, and achieve good results. Compared with traditional routers, these methods are ten times faster than traditional global routers at predicting congestion. In other words, the machine learning-based methods have great prospects in the application of GR. Today, 7 nm chips have achieved mass production, which brings more powerful performance to mobile devices. VLSI technology will continue to improve people's quality of life, while SMT construction and GR will continue to play crucial roles in VLSI physical design.

REFERENCES

[1] S. B. V. Kumar, P. V. Rao, H. A. Sharath, B. M. Sachin, U. S. Ravi, and B. V. Monica, "Review on VLSI design using optimization and self-adaptive particle swarm optimization," *J. King Saud Univ.-Comput. Inf. Sci.*, early access, Jan. 31, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1319157817302793, doi: 10.1016/j.jksuci.2018.01.001.

[2] W. Guo, G. Liu, G. Chen, and S. Peng, "A hybrid multi-objective PSO algorithm with local search strategy for VLSI partitioning," *Frontiers Comput. Sci.*, vol. 8, no. 2, pp. 203–216, Apr. 2014.

[3] J. Chen, Y. Liu, Z. Zhu, and W. Zhu, "An adaptive hybrid memetic algorithm for thermal-aware non-slicing VLSI floorplanning," *Integration*, vol. 58, pp. 245–252, Jun. 2017.

[4] X. Li and W. Zhu, "Two-stage layout decomposition for hybrid E-beam and triple patterning lithography," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 23, no. 1, pp. 1–23, Oct. 2017.

[5] X. Li, Z. Zhu, and W. Zhu, "Discrete relaxation method for triple patterning lithography layout decomposition," *IEEE Trans. Comput.*, vol. 66, no. 2, pp. 285–298, Feb. 2017.

[6] W. Zhu, J. Chen, and W. Li, "An augmented lagrangian method for VLSI global placement," *J. Supercomput.*, vol. 69, no. 2, pp. 714–738, Aug. 2014.

[7] X. Li, B. Yu, J. Ou, J. Chen, D. Z. Pan, and W. Zhu, "Graph-based redundant via insertion and guiding template assignment for DSA-MP," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 11, pp. 2504–2517, Nov. 2018.

[8] G. Liu, W. Guo, R. Li, Y. Niu, and G. Chen, "XGRouter: High-quality global router in X-architecture with particle swarm optimization," *Frontiers Comput. Sci.*, vol. 9, no. 4, pp. 576–594, Aug. 2015.

[9] J. Hu and S. S. Sapatnekar, "A survey on multi-net global routing for integrated circuits," *Integration*, vol. 31, no. 1, pp. 1–49, Nov. 2001.

[10] M. R. Kramer, "The complexity of wirerouting and finding minimum area layouts for arbitrary vlsi circuits," *Adv. Comput. Res.*, vol. 2, no. 1, pp. 129–146, 1984.

[11] J.-R. Gao, P.-C. Wu, and T.-C. Wang, "A new global router for modern designs," in *Proc. Asia South Pacific Design Autom. Conf. (ASPDAC)*, Jan. 2008, pp. 232–237.

[12] M. M. Ozdal and M. D. F. Wong, "Archer: A history-based global routing algorithm," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 28, no. 4, pp. 528–540, Apr. 2009.

[13] M. Cho, K. Lu, K. Yuan, and D. Z. Pan, "BoxRouter 2.0: A hybrid and robust global router with layer assignment for routability," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 14, no. 2, p. 32, Mar. 2009.

[14] J. A. Roy and I. L. Markov, "High-performance routing at the nanometer scale," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 6, pp. 1066–1077, Jun. 2008.

[15] M. D. Moffitt, "MaizeRouter: Engineering an effective global router," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 11, pp. 2017–2026, Nov. 2008.

[16] C. Y. Lee, "An algorithm for path connections and its applications," *IEEE Trans. Electron. Comput.*, vol. EC-10, no. 3, pp. 346–365, Sep. 1961.

[17] M. Johann and R. Reis, "Net by net routing with a new path search algorithm," in *Proc. 13th Symp. Integr. Circuits Syst. Design*, 2000, pp. 144–149.

[18] M. Pan and C. Chu, "FastRoute 2.0: A high-quality and efficient global router," in *Proc. Asia South Pacific Design Autom. Conf.*, Jan. 2007, pp. 250–255.

[19] M. Borah, R. M. Owens, and M. J. Irwin, "An edge-based heuristic for Steiner routing," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 13, no. 12, pp. 1563–1568, Dec. 1994.

[20] D. M. Warme, P. Winter, and M. Zachariasen, "Exact algorithms for plane Steiner tree problems: A computational study," in *Advances in Steiner Trees*. Boston, MA, USA: Springer, 2000, pp. 81–116.

[21] C. Chu and Y.-C. Wong, "FLUTE: Fast lookup table based rectilinear Steiner minimal tree algorithm for VLSI design," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 1, pp. 70–83, Jan. 2008.

[22] Q. Zhu, H. Zhou, T. Jing, X.-L. Hong, and Y. Yang, "Spanning graph-based nonrectilinear Steiner tree algorithms," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 24, no. 7, pp. 1066–1075, Jul. 2005.

[23] C. S. Coulston, "Constructing exact octagonal Steiner minimal trees," in *Proc. 13th ACM Great Lakes Symp. VLSI*, 2003, pp. 1–6.

[24] T.-Y. Ho, C.-F. Chang, Y.-W. Chang, and S.-J. Chen, "Multilevel full-chip routing for the X-based architecture," in *Proc. 42nd Design Autom. Conf.*, Jun. 2005, pp. 597–602.

[25] R. Kastner, E. Bozorgzadeh, and M. Sarrafzadeh, "Pattern routing: Use and theory for increasing predictability and avoiding coupling," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 21, no. 7, pp. 777–790, Jul. 2002.

[26] Y.-J. Chang, Y.-T. Lee, J.-R. Gao, P.-C. Wu, and T.-C. Wang, "NTHU-route 2.0: A robust global router for modern designs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 29, no. 12, pp. 1931–1944, Dec. 2010.

[27] W.-H. Liu, Y.-L. Li, and C.-K. Koh, "A fast maze-free routing congestion estimator with hybrid unilateral monotonic routing," in *Proc. Int. Conf. Comput.-Aided Design (ICCAD)*, 2012, pp. 713–719.

[28] A. Youssef, Z. Yang, M. Anis, S. Areibi, A. Vannelli, and M. Elmasry, "A power-efficient multipin ILP-based routing technique," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 1, pp. 225–235, Jan. 2010.

[29] C.-J. Chang, P.-J. Huang, T.-C. Chen, and C.-N.-J. Liu, "ILP-based inter-die routing for 3D ICs," in *Proc. 16th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2011, pp. 330–335.

[30] H. Kong, T. Yan, and M. D. F. Wong, "Automatic bus planner for dense PCBs," in *Proc. ACM/IEE 46th Annu. Design Autom. Conf. (DAC)*, Jul. 2009, pp. 326–331.

[31] E. S. Kuh and T. Ohtsuki, "Recent advances in VLSI layout," *Proc. IEEE*, vol. 78, no. 2, pp. 237–263, Feb. 1990.

[32] R. T. Hadsell and P. H. Madden, "Improved global routing through congestion estimation," in *Proc. 40th Conf. Design Autom. (DAC)*, 2003, pp. 28–31.

[33] Z. Cao, T. T. Jing, J. Xiong, Y. Hu, Z. Feng, L. He, X.-L. Hong, "Fashion: A fast and accurate solution to global routing problem," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 4, pp. 726–737, Apr. 2008.

[34] M. Cho and D. Z. Pan, "BoxRouter: A new global router based on box expansion and progressive ILP," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, no. 12, pp. 2130–2143, Dec. 2007.

[35] L. McMurchie and C. Ebeling, "Pathfinder: A negotiation-based performance-driven router for FPGAs," in *Reconfigurable Computing*. Amsterdam, The Netherlands: Elsevier, 2008, pp. 365–381.

[36] X. Wei, W.-C. Tang, Y. Diao, and Y.-L. Wu, "ECO timing optimization with negotiation-based re-routing and logic re-structuring using spare cells," in *Proc. 17th Asia South Pacific Design Autom. Conf.*, Jan. 2012, pp. 511–516.

[37] J. He, M. Burtscher, R. Manohar, and K. Pingali, "SPRoute: A scalable parallel negotiation-based global router," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2019, pp. 1–8.

[38] H.-Y. Chen, C.-H. Hsu, and Y.-W. Chang, "High-performance global routing with fast overflow reduction," in *Proc. Asia South Pacific Design Autom. Conf.*, Jan. 2009, pp. 582–587.

[39] C. Albrecht, "Global routing by new approximation algorithms for multicommodity flow," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 20, no. 5, pp. 622–632, May 2001.

[40] T.-H. Wu, A. Davoodi, and J. T. Linderoth, "GRIP: Scalable 3D global routing using integer programming," in *Proc. 46th Annu. Design Autom. Conf. (DAC)*, 2009, pp. 320–325.

[41] R. Kay and R. A. Rutenbar, "Wire packing-a strong formulation of crosstalk-aware chip-level track/layer assignment with an efficient integer programming solution," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 20, no. 5, pp. 672–679, May 2001.

[42] D. Wu, J. Hu, R. Mahapatra, and M. Zhao, "Layer assignment for crosstalk risk minimization," in *Proc. Asia South Pacific Design Autom. Conf. (ASP-DAC)*, 2004, pp. 159–162.

[43] G. Xu, L.-D. Huang, D. Z. Pan, and M. D. F. Wong, "Redundant-via enhanced maze routing for yield improvement," in *Proc. Conf. Asia South Pacific Design Autom. (ASP-DAC)*, 2005, pp. 1148–1151.

[44] K.-Y. Lee and T.-C. Wang, "Post-routing redundant via insertion for yield/reliability improvement," in *Proc. Asia South Pacific Conf. Design Autom.*, 2006, p. 6.

[45] T.-R. Lin, T. Edwards, and M. Pedram, "QGDR: A via-minimization-oriented routing tool for large-scale superconductive single-flux-quantum circuits," *IEEE Trans. Appl. Supercond.*, vol. 29, no. 7, pp. 1–12, Oct. 2019.

[46] Y. Xu, Y. Zhang, and C. Chu, "FastRoute 4.0: Global router with efficient via minimization," in *Proc. Asia South Pacific Design Autom. Conf.*, Jan. 2009, pp. 576–581.

[47] T.-H. Lee and T.-C. Wang, "Congestion-constrained layer assignment for via minimization in global routing," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 9, pp. 1643–1656, Sep. 2008.

[48] T.-H. Wu, A. Davoodi, and J. T. Linderoth, "GRIP: Global routing via integer programming," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 30, no. 1, pp. 72–84, Jan. 2011.

[49] W.-H. Liu, W.-C. Kao, Y.-L. Li, and K.-Y. Chao, "NCTU-GR 2.0: Multithreaded collision-aware global routing with bounded-length maze routing," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 5, pp. 709–722, May 2013.

[50] N. J. Naclerio, S. Masuda, and K. Nakajima, "The via minimization problem is NP-complete," *IEEE Trans. Comput.*, vol. 38, no. 11, pp. 1604–1608, Nov. 1989.

[51] T.-H. Lee and T.-C. Wang, "Simultaneous antenna avoidance and via optimization in layer assignment of multi-layer global routing," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2010, pp. 312–318.

[52] G. Liu, X. Huang, W. Guo, Y. Niu, and G. Chen, "Multilayer obstacle-avoiding X-Architecture Steiner minimal tree construction based on particle swarm optimization," *IEEE Trans. Cybern.*, vol. 45, no. 5, pp. 1003–1016, May 2015.

[53] M. D. Moffitt, J. A. Roy, and I. L. Markov, "The coming of age of (academic) global routing," in *Proc. Int. Symp. Phys. Design (ISPD)*, 2008, pp. 148–155.

[54] V. Vani and G. R. Prasad, "Augmented line segment based algorithm for constructing rectilinear Steiner minimum tree," in *Proc. Int. Conf. Commun. Electron. Syst. (ICCES)*, Oct. 2016, pp. 1–5.

[55] N. R. Latha and G. R. Prasad, "Wirelength and memory optimized rectilinear Steiner minimum tree routing," in *Proc. 2nd IEEE Int. Conf. Recent Trends Electron., Inf. Commun. Technol. (RTEICT)*, May 2017, pp. 1493–1497.

[56] L. N R and G. R. Prasad, "Memory and I/O optimized rectilinear Steiner minimum tree routing for VLSI," *Int. J. Electron., Commun., Meas. Eng.*, vol. 9, no. 1, pp. 46–59, Jan. 2020.

[57] S.-E.-D. Lin and D. H. Kim, "Construction of all rectilinear Steiner minimum trees on the Hanan grid and its applications to VLSI design," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, early access, May 20, 2019, doi: 10.1109/TCAD.2019.2917896.

[58] P. Tu, W.-K. Chow, and E. F. Y. Young, "Timing driven routing tree construction," in *Proc. ACM/IEEE Int. Workshop Syst. Level Interconnect Predict. (SLIP)*, Jun. 2017, pp. 1–8.

[59] G. Liu, W. Guo, Y. Niu, G. Chen, and X. Huang, "A PSO-based timing-driven octilinear Steiner tree algorithm for VLSI routing considering bend reduction," *Soft Comput.*, vol. 19, no. 5, pp. 1153–1169, May 2015.

[60] G. Liu, Z. Chen, Z. Zhuang, W. Guo, and G. Chen, "A unified algorithm based on HTS and self-adapting PSO for the construction of octagonal and rectilinear SMT," *Soft Comput.*, vol. 24, no. 6, pp. 3943–3961, Mar. 2020.

[61] G. Ajwani, C. Chu, and W.-K. Mak, "FOARS: FLUTE based obstacle-avoiding rectilinear Steiner tree construction," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 30, no. 2, pp. 194–204, Feb. 2011.

[62] K.-W. Lin, Y.-S. Lin, Y.-L. Li, and R.-B. Lin, "A maze routing-based algorithm for ML-OARST with pre-selecting and re-building Steiner points," in *Proc. Great Lakes Symp. VLSI*, 2017, pp. 399–402.

[63] T. Huang and E. F. Y. Young, "An exact algorithm for the construction of rectilinear Steiner minimum trees among complex obstacles," in *Proc. 48th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2011, pp. 164–169.

[64] T. Huang, L. Li, and E. F. Y. Young, "On the construction of optimal obstacle-avoiding rectilinear Steiner minimum trees," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 30, no. 5, pp. 718–731, May 2011.

[65] R.-Y. Wang, C.-C. Pai, J.-J. Wang, H.-T. Wen, Y.-C. Pai, Y.-W. Chang, J. C. M. Li, and J.-H. Jiang, "Efficient multi-layer obstacle-avoiding region-to-region rectilinear Steiner tree construction," in *Proc. 55th Annu. Design Autom. Conf. (DAC)*, 2018, p. 45.

[66] W. Guo and X. Huang, "PORA: A physarum-inspired obstacle-avoiding routing algorithm for integrated circuit design," *Appl. Math. Model.*, vol. 78, pp. 268–286, Feb. 2020.

[67] X. Huang, G. Liu, W. Guo, Y. Niu, and G. Chen, "Obstacle-avoiding algorithm in X-Architecture based on discrete particle swarm optimization for VLSI design," *ACM Trans. Design Autom. Electron. Syst.*, vol. 20, no. 2, p. 24, Mar. 2015.

[68] X. Huang, W. Guo, G. Liu, and G. Chen, "FH-OAOS: A fast four-step heuristic for obstacle-avoiding octilinear Steiner tree construction," *ACM Trans. Design Autom. Electron. Syst.*, vol. 21, no. 3, p. 48, Jul. 2016.

[69] X. Huang, W. Guo, G. Liu, and G. Chen, "MLXR: Multi-layer obstacle-avoiding X-architecture Steiner tree construction for VLSI routing," *Sci. China Inf. Sci.*, vol. 60, no. 1, p. 19102, Jan. 2017.

[70] X. Huang, W. Guo, and G. Chen, "Fast obstacle-avoiding octilinear Steiner minimal tree construction algorithm for VLSI design," in *Proc. 16th Int. Symp. Qual. Electron. Design*, Mar. 2015, pp. 46–50.

[71] S. Held and S. T. Spirkl, "A fast algorithm for rectilinear Steiner trees with length restrictions on obstacles," in *Proc. Int. Symp. Phys. Design (ISPD)*, 2014, pp. 37–44.

[72] Y. Zhang, A. Chakraborty, S. Chowdhury, and D. Z. Pan, "Reclaiming over-the-IP-block routing resources with buffering-aware rectilinear Steiner minimum tree construction," in *Proc. Int. Conf. Comput.-Aided Design*, 2012, pp. 137–143.

[73] T. Huang and E. F. Y. Young, "Construction of rectilinear Steiner minimum trees with slew constraints over obstacles," in *Proc. Int. Conf. Comput.-Aided Design*, 2012, pp. 144–151.

[74] Y. Zhang and D. Z. Pan, "Timing-driven, over-the-block rectilinear Steiner tree construction with pre-buffering and slew constraints," in *Proc. Int. Symp. Phys. Design*, 2014, pp. 29–36.

[75] H. Zhang, D.-Y. Ye, and W.-Z. Guo, "A heuristic for constructing a rectilinear Steiner tree by reusing routing resources over obstacles," *Integration*, vol. 55, pp. 162–175, Sep. 2016.

[76] G. Shyamala and G. Prasad, "Obstacle aware delay optimized rectilinear Steiner minimum tree routing," in *Proc. 2nd IEEE Int. Conf. Recent Trends Electron., Inf. Commun. Technol. (RTEICT)*,May 2017, pp. 2194–2197.

[77] F. K. Hwang, "On Steiner minimal trees with rectilinear distance," *SIAM J. Appl. Math.*, vol. 30, no. 1, pp. 104–114, Jan. 1976.

[78] J.-M. Ho, G. Vijayan, and C. K. Wong, "New algorithms for the rectilinear Steiner tree problem," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 9, no. 2, pp. 185–193, Feb. 1990.

[79] A. B. Kahng and G. Robins, "A new class of iterative Steiner tree heuristics with good performance," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 11, no. 7, pp. 893–902, Jul. 1992.

[80] H. Zhou, N. Shenoy, and W. Nicholls, "Efficient minimum spanning tree construction without delaunay triangulation," in *Proc. Asia South Pacific Design Autom. Conf.*, 2001, pp. 192–197.

[81] H. Zhou, "Efficient Steiner tree construction based on spanning graphs," in *Proc. Int. Symp. Phys. Design*, 2003, pp. 152–157.

[82] S. Cinel and C. F. Bazlamacci, "A distributed heuristic algorithm for the rectilinear Steiner minimal tree problem," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 11, pp. 2083–2087, Nov. 2008.

[83] A. B. Kahng, I. I. Mandoiu, and A. Z. Zelikovsky, "Highly scalable algorithms for rectilinear and octilinear Steiner trees," in *Proc. Asia South Pacific Design Autom. Conf.*, 2003, pp. 827–833.

[84] M. Hanan, "On Steiner's problem with rectilinear distance," *SIAM J. Appl. Math.*, vol. 14, no. 2, pp. 255–265, 1966.

[85] Z. Cao, T. Jing, J. Xiong, Y. Hu, L. He, and X. Hong, "DpRouter: A fast and accurate dynamic-pattern-based global routing algorithm," in *Proc. Asia South Pacific Design Autom. Conf.*, Jan. 2007, pp. 256–261.

[86] H. Xianlong, Z. Qi, J. Tong, W. Yin, Y. Yang, and C. Yici, "Non-rectilinear on-chip interconnect–an efficient routing solution with high performance," *Chin. J. Semicond.*, vol. 3, no. 24, 2003, pp. 225–233.

[87] S. L. Teig, "The x architecture: Not your father's diagonal wiring," in *Proc. Int. Workshop Syst.-Level Interconnect Predict.*, 2002, pp. 33–37.

[88] J.-T. Yan, "Timing-driven octilinear Steiner tree construction based on Steiner-point reassignment and path reconstruction," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 13, no. 2, p. 26, Apr. 2008.

[89] 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," *IEEE Trans. Comput.*, vol. C-36, no. 3, pp. 321–331, Mar. 1987.

[90] L. Li, Z. Qian, and E. F. Y. Young, "Generation of optimal obstacle-avoiding rectilinear Steiner minimum tree," in *IEEE/ACM Int. Conf. Comput.-Aided Design-Dig. Tech. Papers*, Nov. 2009, pp. 21–25.

[91] 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 *IEEE/ACM Int. Conf. Comput.-Aided Design-Dig. Tech. Papers*, Nov. 2009, pp. 26–32.

[92] J. 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.

[93] C.-H. Liu, S.-Y. Yuan, S.-Y. Kuo, and S.-C. Wang, "High-performance obstacle-avoiding rectilinear Steiner tree construction," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 14, no. 3, p. 45, May 2009.

[94] C.-W. Lin, S.-Y. Chen, C.-F. Li, Y.-W. Chang, and C.-L. Yang, "Efficient obstacle-avoiding rectilinear Steiner tree construction," in *Proc. Int. Symp. Phys. Design*, 2007, pp. 127–134.

[95] Z. Shen, C. C. N. Chu, and Y.-M. Li, "Efficient rectilinear Steiner tree construction with rectilinear blockages," in *Proc. Int. Conf. Comput. Design*, 2005, pp. 38–44.

[96] Y. Hu, T. Jing, X. Hong, Z. Feng, X. Hu, and G. Yan, "An-OARSMan: Obstacle-avoiding routing tree construction with good length performance," in *Proc. Asia South Pacific Design Autom. Conf.*, 2005, pp. 7–12.

[97] J. L. Ganley and J. P. Cohoon, "Routing a multi-terminal critical net: Steiner tree construction in the presence of obstacles," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, vol. 1, May 1994, pp. 113–116.

[98] T. Huang and E. F. Y. Young, "Obstacle-avoiding rectilinear Steiner minimum tree construction: An optimal approach," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2010, pp. 610–613.

[99] T. T. Jing, Z. Feng, Y. Hu, X. L. Hong, X. D. Hu, and G. Y. Yan, "λ-OAT: λ-geometry obstacle-avoiding tree construction with $O(n \log n)$ complexity," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, no. 11, pp. 2073–2079, Nov. 2007.

[100] W. Guo, J. Li, G. Chen, Y. Niu, and C. Chen, "A PSO-optimized real-time fault-tolerant task allocation algorithm in wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 12, pp. 3236–3249, Dec. 2015.

[101] B. Lin, W. Guo, N. Xiong, G. Chen, A. V. Vasilakos, and H. Zhang, "A pretreatment workflow scheduling approach for big data applications in multicloud environments," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 3, pp. 581–594, Sep. 2016.

[102] X. Chen, G. Liu, N. Xiong, Y. Su, and G. Chen, "A survey of swarm intelligence techniques in VLSI routing problems," *IEEE Access*, vol. 8, pp. 26266–26292, 2020.

[103] M. Müller-Hannemann and S. Peyer, "Approximation of rectilinear Steiner trees with length restrictions on obstacles," in *Proc. Workshop Algorithms Data Struct.* Berlin, Germany: Springer, 2003, pp. 207–218.

[104] K. Mehlhorn, "A faster approximation algorithm for the Steiner problem in graphs," *Inf. Process. Lett.*, vol. 27, no. 3, pp. 125–128, Mar. 1988.

[105] K. Clarkson, S. Kapoor, and P. Vaidya, "Rectilinear shortest paths through polygonal obstacles in $O(n(logn)^2)$ time," in *Proc. 3rd Annu. Symp. Comput. Geometry*, 1987, pp. 251–257.

[106] C. V. Kashyap, C. J. Alpert, F. Liu, and A. Devgan, "PERI: A technique for extending delay and slew metrics to ramp inputs," in *Proc. 8th ACM/IEEE Int. Workshop Timing Issues Specification Synth. Digit. Syst.*, Dec. 2002, pp. 57–62.

[107] Y. Zhang, Y. Xu, and C. Chu, "FastRoute3.0: A fast and high quality global router based on virtual capacity," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2008, pp. 344–349.

[108] K.-R. Dai, W.-H. Liu, and Y.-L. Li, "Efficient simulated evolution based rerouting and congestion-relaxed layer assignment on 3-D global routing," in *Proc. Asia South Pacific Design Autom. Conf.*, Jan. 2009, pp. 570–575.

[109] L. C. Abel, "On the ordering of connections for automatic wire routing," *IEEE Trans. Comput.*, vol. C-21, no. 11, pp. 1227–1233, Nov. 1972.

[110] D. W. Hightower, "A solution to line-routing problems on the continuous plane," in *Proc. 6th Annu. Conf. Design Autom.*, 1969, pp. 1–24.

[111] K. Mikami, "A computer program for optimal routing of printed circuit connectors," in *Proc. IFIPS*, 1968, pp. 1475–1478.

[112] M. Pan and C. Chu, "FastRoute: A step to integrate global routing into placement," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design*, Nov. 2006, pp. 464–471.

[113] C. Chiang, M. Sarrafzadeh, and C. K. Wong, "Global routing based on Steiner min-max trees," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 9, no. 12, pp. 1318–1325, Dec. 1990.

[114] C. Chiang, C. K. Wong, and M. Sarrafzadeh, "A weighted Steiner tree-based global router with simultaneous length and density minimization," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 13, no. 12, pp. 1461–1469, Dec. 1994.

[115] Y. Xu and C. Chu, "MGR: Multi-level global router," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2011, pp. 250–255.

[116] H. Liao, W. Zhang, X. Dong, B. Poczos, K. Shimada, and L. Burak Kara, "A deep reinforcement learning approach for global routing," *J. Mech. Des.*, vol. 142, no. 6, pp. 1–12, Jun. 2020.

[117] Z. Zhou, S. Chahal, T.-Y. Ho, and A. Ivanov, "Supervised-learning congestion predictor for routability-driven global routing," in *Proc. Int. Symp. VLSI Design, Autom. Test (VLSI-DAT)*, Apr. 2019, pp. 1–4.

[118] T. Zhang, X. Liu, W. Tang, J. Chen, Z. Xiao, F. Zhang, W. Hu, Z. Zhou, and Y. Cheng, "Predicted congestion using a density-based fast neural network algorithm in global routing," in *Proc. IEEE Int. Conf. Electron Devices Solid-State Circuits (EDSSC)*, Jun. 2019, pp. 1–3.

[119] J. Hu, J. A. Roy, and I. L. Markov, "Sidewinder: A scalable ILP-based router," in *Proc. 10th Int. Workshop Syst. Level Interconnect Predict.*, 2008, pp. 73–80.

[120] L. Behjat and A. Chiang, "Fast integer linear programming based models for VLSI global routing," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2005, pp. 6238–6243.

[121] T.-H. Wu, A. Davoodi, and J. T. Linderoth, "A parallel integer programming approach to global routing," in *Proc. 47th Design Autom. Conf.*, 2010, pp. 194–199.

[122] C. Li, M. Xie, C. Koh, J. Cong, and P. H. Madden, "Routability-driven placement and white space allocation," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Feb. 2004, pp. 394–401.

[123] M. Pan and C. Chu, "IPR: An integrated placement and routing algorithm," in *Proc. 44th ACM/IEEE Design Autom. Conf.*, Jun. 2007, pp. 59–62.

[124] J. A. Roy, N. Viswanathan, G.-J. Nam, C. J. Alpert, and I. L. Markov, "CRISP: Congestion reduction by iterated spreading during placement," in *Proc. Int. Conf. Comput.-Aided Design*, 2009, pp. 357–362.

[125] U. Brenner and A. Rohe, "An effective congestion-driven placement framework," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 4, pp. 387–394, Apr. 2003.

[126] P. Spindler and F. M. Johannes, "Fast and accurate routing demand estimation for efficient routability-driven placement," in *Proc. Design, Autom. Test Eur. Conf. Exhib.* San Jose, CA, USA: EDA Consortium, Apr. 2007, pp. 1226–1231.

[127] M. Wang, X. Yang, K. Eguro, and M. Sarrafzadeh, "Multi-center congestion estimation and minimization during placement," in *Proc. ISPD*, 2000, pp. 147–152.

[128] J. Westra, C. Bartels, and P. Groeneveld, "Probabilistic congestion prediction," in *Proc. Int. Symp. Phys. Design*, 2004, pp. 204–209.

[129] H. Shojaei, A. Davoodi, and J. T. Linderoth, "Congestion analysis for global routing via integer programming," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2011, pp. 256–262.

[130] Z. Yang, A. Vannelli, and S. Areibi, "An ILP based hierarchical global routing approach for VLSI ASIC design," *Optim. Lett.*, vol. 1, no. 3, pp. 281–297, May 2007.

[131] S. Sen, "VLSI routing in multiple layers using grid based routing algorithms," *Int. J. Comput. Appl.*, vol. 93, no. 16, pp. 41–45, 2014.

[132] H.-J. Lu, E.-J. Jang, A. Lu, Y. T. Zhang, Y.-H. Chang, C.-H. Lin, and R.-B. Lin, "Practical ILP-based routing of standard cells," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*. San Jose, CA, USA: EDA Consortium, 2016, pp. 245–248.

[133] G. Liu, Z. Zhuang, W. Guo, and G. Chen, "A high performance X-architecture multilayer global router for vlsi," (in Chinese), *Acta Automatica Sinica*, vol. 46, no. 1, pp. 79–93, 2020.

[134] L. R. Ford, Jr., and D. R. Fulkerson, *Flows Network*. Princeton, NJ, USA: Princeton Univ. Press, 2015.

[135] R. K. Ahuja, T. L. Magnanti, J. B. Orlin, and K. Weihe, "Network flows: Theory, algorithms and applications," *ZOR-Methods Models Oper. Res.*, vol. 41, no. 3, pp. 252–254, 1995.

[136] J. D. Cho and M. Sarrafzadeh, "Four-bend top-down global routing," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 17, no. 9, pp. 793–802, Sep. 1998.

[137] J. Hu and S. S. Sapatnekar, "A timing-constrained algorithm for simultaneous global routing of multiple nets," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design*, Nov. 2000, pp. 99–103.

[138] E. Shragowitz and S. Keel, "A global router based on a multicommodity flow model," *Integration*, vol. 5, no. 1, pp. 3–16, Mar. 1987.

[139] P. Raghavan and C. D. Thompson, "Multiterminal global routing: A deterministic approximation scheme," *Algorithmica*, vol. 6, nos. 1–6, pp. 73–82, Jun. 1991.

[140] R. C. Carden, J. Li, and C.-K. Cheng, "A global router with a theoretical bound on the optimal solution," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 15, no. 2, pp. 208–216, Feb. 1996.

[141] F. Shahrokhi and D. W. Matula, "The maximum concurrent flow problem," *J. ACM*, vol. 37, no. 2, pp. 318–334, Apr. 1990.

[142] N. Garg and J. Könemann, "Faster and simpler algorithms for multicommodity flow and other fractional packing problems," *SIAM J. Comput.*, vol. 37, no. 2, pp. 630–652, Jan. 2007.

[143] S. Daboul, S. Held, B. Natura, and D. Rotter, "Global interconnect optimization," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2019, pp. 1–8.

[144] G. S. Garcea, N. P. van der Meijs, and R. H. J. M. Otten, "Simultaneous analytic area and power optimization for repeater insertion," in *Proc. Int. Conf. Comput. Aided Design (ICCAD)*, Nov. 2003, pp. 568–573.

[145] G. Nallathambi and S. Rajaram, "Power aware VLSI routing using particle swarm optimization for green electronics," in *Proc. Int. Conf. Green Comput. Commun. Electr. Eng. (ICGCCEE)*, Mar. 2014, pp. 1–6.

[146] S.-G. Yang, L. Li, Y. Xu, Y.-A. Zhang, and B. Zhang, "A power-aware adaptive routing scheme for network on a chip," in *Proc. 7th Int. Conf. ASIC*, Oct. 2007, pp. 1301–1304.

[147] T.-H. Wu, A. Davoodi, and J. T. Linderoth, "Power-driven global routing for multisupply voltage domains," *VLSI Des.*, vol. 2013, pp. 1–12, Jul. 2013.

[148] T. Arora and M. E. Moses, "Ant colony optimization for power efficient routing in manhattan and non-manhattan VLSI architectures," in *Proc. IEEE Swarm Intell. Symp.*, Mar. 2009, pp. 137–144.

[149] M. Chentouf, L. Cherif, and Z. El Abidine Alaoui Ismaili, "Power-aware clock routing in 7 nm designs," in *Proc. 4th Int. Conf. Optim. Appl. (ICOA)*, Apr. 2018, pp. 1–6.

[150] W. C. Elmore, "The transient response of damped linear networks with particular regard to wideband amplifiers," *J. Appl. Phys.*, vol. 19, no. 1, pp. 55–63, 1948.

[151] T. F. Chan, J. Cong, and E. Radke, "A rigorous framework for convergent net weighting schemes in timing-driven placement," in *IEEE/ACM Int. Conf. Comput.-Aided Design-Dig. Tech. Papers*, Nov. 2009, pp. 288–294.

[152] C. Qiao and X. Hong, "A loop routing approach for decreasing critical path delay," in *Proc. 4th Int. Conf. Solid-State IC Technol.*, 1995, pp. 355–357.

[153] X. Hong, T. Xue, J. Huang, C.-K. Cheng, and E. S. Kuh, "TIGER: An efficient timing-driven global router for gate array and standard cell layout design," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 16, no. 11, pp. 1323–1331, Nov. 1997.

[154] L. Y. Wang, B. D. Liu, Y. T. Lai, and M. Y. Yeh, "Performance-driven global routing based on simulated evolution," in *Proc. IEEE Region Int. Conf. Comput., Commun. Autom. (TENCON)*, vol. 1, Oct. 1993, pp. 511–514.

[155] A. I. Erzin, V. V. Zalyubovskiy, Y. V. Shamardin, I. I. Takhonov, R. Samanta, and S. Raha, "A timing driven congestion aware global router," in *Proc. 2nd Int. Conf. Emerg. Appl. Inf. Technol.*, Feb. 2011, pp. 375–378.

[156] J. Cong and P. H. Madden, "Performance driven global routing for standard cell design," in *Proc. Int. Symp. Phys. Design*, 1997, vol. 14, no. 16, pp. 73–80.

[157] J. Cong, K.-S. Leung, and D. Zhou, "Performance-driven interconnect design based on distributed RC delay model," in *Proc. 30th Int. Design Autom. Conf.*, 1993, pp. 606–611.

[158] J. Cong and K.-S. Leung, "Optimal wiresizing under the distributed elmore delay model," in *Proc. Int. Conf. Comput. Aided Design (ICCAD)*, 1993, pp. 634–639.

[159] J. Cong and C.-K. Koh, "Simultaneous driver and wire sizing for performance and power optimization," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 2, no. 4, pp. 408–425, Dec. 1994.

[160] J. Cong and B. Preas, "A new algorithm for standard cell global routing," in *Proc. ICCAD*, 1988, pp. 176–179.

[161] H.-H. Huang, H.-Y. Huang, Y.-C. Lin, and T.-M. Hsieh, "Timing-driven obstacles-avoiding routing tree construction for a multiple-layer system," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2008, pp. 1200–1203.

[162] J. Huang, X.-L. Hong, C.-K. Cheng, and E. S. Kuh, "An efficient timing-driven global routing algorithm," in *Proc. 30th Int. Design Autom. Conf. (DAC)*, 1993, pp. 596–600.

[163] J. Vygen, "Near-optimum global routing with coupling, delay bounds, and power consumption," in *Proc. Int. Conf. Integer Program. Combinat. Optim.* Berlin, Germany: Springer, 2004, pp. 308–324.

[164] R. Samanta, A. I. Erzin, S. Raha, Y. V. Shamardin, I. I. Takhonov, and V. V. Zalyubovskiy, "A provably tight delay-driven concurrently congestion mitigating global routing algorithm," *Appl. Math. Comput.*, vol. 255, pp. 92–104, Mar. 2015.

[165] S. Held, D. Müller, D. Rotter, R. Scheifele, V. Traub, and J. Vygen, "Global routing with timing constraints," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 2, pp. 406–419, Feb. 2018.

[166] J.-T. Yan and C.-H. Yen, "Construction of delay-driven GNR routing tree," in *Proc. 17th IEEE Int. New Circuits Syst. Conf. (NEWCAS)*, Jun. 2019, pp. 1–4.

[167] J.-T. Yan, "Single-layer delay-driven GNR nontree routing under resource constraint for yield improvement," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 3, pp. 736–749, Mar. 2020.

[168] W.-H. Liu, Y.-L. Li, and K.-Y. Chao, "High-quality global routing for multiple dynamic supply voltage designs," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2011, pp. 263–269.

[169] M. Terres, C. Meinhardt, G. Bontorin, and R. Reis, "Exploring more efficient architectures for multiple dynamic supply voltage designs," in *Proc. IEEE 5th Latin Amer. Symp. Circuits Syst.*, Feb. 2014, pp. 1–4.

[170] W.-H. Liu, W.-C. Kao, Y.-L. Li, and K.-Y. Chao, "Multi-threaded collision-aware global routing with bounded-length maze routing," in *Proc. 47th Design Autom. Conf. (DAC)*, 2010, pp. 200–205.

[171] L.-C. Lu, "Physical design challenges and innovations to meet power, speed, and area scaling trend," in *Proc. ACM Int. Symp. Phys. Design (ISPD)*, 2017, p. 63.

[172] J. Ao, S. Dong, S. Chen, and S. Goto, "Delay-driven layer assignment in global routing under multi-tier interconnect structure," in *Proc. ACM Int. Symp. Int. Symp. Phys. Design (ISPD)*, 2013, pp. 101–107.

[173] B. Yu, D. Liu, S. Chowdhury, and D. Z. Pan, "TILA: Timing-driven incremental layer assignment," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2015, pp. 110–117.

[174] D. Liu, B. Yu, S. Chowdhury, and D. Z. Pan, "TILA-S: Timing-driven incremental layer assignment avoiding slew violations," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 1, pp. 231–244, Jan. 2018.

[175] S.-Y. Han, W.-H. Liu, R. Ewetz, C.-K. Koh, K.-Y. Chao, and T.-C. Wang, "Delay-driven layer assignment for advanced technology nodes," in *Proc. 22nd Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2017, pp. 456–462.

[176] W.-H. Liu and Y.-L. Li, "Negotiation-based layer assignment for via count and via overflow minimization," in *Proc. 16th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2011, pp. 539–544.

HAO TANG received the B.E. degree in computer science from Fuzhou University, Fuzhou, China, in 2018, where he is currently pursuing the master's degree in computer technology with the College of Mathematics and Computer Science. His research interests include VLSI physical design and its application.

GENGGENG LIU (Member, IEEE) received the B.S. degree in computer science and the Ph.D. degree in applied mathematics from Fuzhou University, Fuzhou, China, in 2009 and 2015, respectively. He is currently an Associate Professor with the College of Mathematics and Computer Science, Fuzhou University. His research interests include computational intelligence and very large scale integration physical design.

XIAOHUA CHEN received the B.E. degree in computer science and technology from Fujian Normal University, Fuzhou, China, in 2018. She is currently pursuing the master's degree in computer system architecture with the College of Mathematics and Computer Science, Fuzhou University. Her research interests include swarm intelligence algorithms and EDA design.

NAIXUE XIONG (Senior Member, IEEE) received the Ph.D. degree in sensor system engineering from Wuhan University and the Ph.D. degree in dependable sensor networks from the Japan Advanced Institute of Science and Technology.

He was with Northeastern State University, Georgia State University, the Wentworth Institute of Technology, and Colorado Technical University (as a Full Professor for about five years) for about ten years. He is currently a Professor with the College of Intelligence and Computing, Tianjin University, China. His research interests include cloud computing, security and dependability, parallel and distributed computing, networks, and optimization theory. He has published over 300 international journal articles and over 100 international conference papers. Some of his works were published in the IEEE Journal on Selected Areas in Communications, the IEEE/ACM Transactions, the ACM Sigcomm Workshop, IEEE INFOCOM, ICDCS, and IPDPS. He is a Senior Member of the IEEE Computer Society. He has received the Best Paper Award from the 10th IEEE International Conference on High Performance Computing and Communications, in 2008, and the Best Student Paper Award from the 28th North American Fuzzy Information Processing Society Annual Conference in 2009. He is the Chair of Trusted Cloud Computing Task Force, IEEE Computational Intelligence Society, and the Industry System Applications Technical Committee. He has been the General Chair, the Program Chair, the Publicity Chair, a PC member, and a OC Member of over 100 international conferences and a Reviewer of over 100 international journals, including the IEEE Journal on Selected Areas in Communications, the IEEE Transactions on Systems, Man, and Cybernetics (Part: A/B/C), the IEEE Transactions on Communications, the IEEE Transactions on Mobile Computing, and the IEEE Transactions on Parallel and Distributed Systems. He is serving as the Editor-in-Chief, an Associate Editor, or an Editorial Member of over ten international journals (including as an Associate Editor of the the IEEE Transactions on Systems, Man and Cybernetics: Systems and the Information Science and the Editor-in-Chief of *Journal of Internet Technology* and the *Journal of Parallel and Cloud Computing* and a Guest Editor for over ten international journals, including the *Sensors* journal, WINET, and MONET.