Multilayer Obstacle-Avoiding X-Architecture Steiner Minimal Tree Construction Based on Particle Swarm Optimization

Genggeng Liu, Xing Huang, Wenzhong Guo, Yuzhen Niu, and Guolong Chen

***Abstract*—As the basic model for very large scale integration routing, the Steiner minimal tree (SMT) can be used in various practical problems, such as wire length optimization, congestion, and time delay estimation. In this paper, an effective algorithm based on particle swarm optimization is presented to construct a multilayer obstacle-avoiding X-architecture SMT (ML-OAXSMT). First, a pretreatment strategy is presented to reduce the total number of judgments for the routing conditions around obstacles and vias. Second, an edge transformation strategy is employed to make the particles have the ability to bypass the obstacles while the union-find partition is used to prevent invalid solutions. Third, according to the feature of ML-OAXSMT problem, we design an edge-vertex encoding strategy, which has the advantage of simple and effective. Moreover, a penalty mechanism is proposed to help the particle bypass the obstacles, and reduce the generation of via at the same time. Experimental results show that our algorithm from a global perspective of multilayer structure can achieve the best solution quality among the existing algorithms. Finally, to our best knowledge, we redefine the edge cost and then construct the obstacle-avoiding preferred direction X-architecture Steiner tree, which is the first work to address this problem and can offer the theory supports for chip design based on non-Manhattan architecture.**

***Index Terms*—Multilayer routing, particle swarm optimization (PSO), preferred direction, Steiner minimal tree (SMT), very large scale integration (VLSI), X-architecture.**

I. Introduction

THE multilayer obstacle-avoiding X-architecture Steiner minimal tree (ML-OAXSMT) problem is significant for the modern chip designs. On one hand, a Steiner minimal tree (SMT) is the best connection model for routing a multipin net of the chip [1]. On the other hand, due to more and more obstacles are appearing in the circuits, such as macrocells, IP blocks, and prerouted nets, the obstacle-avoiding SMT (OASMT) problem has attracted widespread attention. But most of the current research work mainly focused on Manhattan architecture and single-layer routing. With advance in very large scale integration (VLSI) fabrication technology, the interconnect effects have become the major challenge of chip performance. Manhattan architecture which restricts the routing directions to be only horizontal and vertical has limited the ability to optimize wire length and time delay. Consequently, more and more researchers begin to put their work into non-Manhattan architecture, which allows more routing directions and could further improve the routability. Especially, as one of the non-Manhattan architecture, X-architecture is fully supported by nearly all current manufacturing technologies [2]. The problem is that the related achievements are relatively few [3]–[5]. Moreover, with the integrated circuit design moving into the nanometer domain, the number of metal layers increases sharply, and both the wire width and the space between wires are greatly reduced. The circuit performance and density have been greatly improved. Thus the multilayer routing arises at the historic moment and causes a wide attention of many research institutions.

As a swarm-based evolutionary method, particle swarm optimization (PSO) which is a powerful optimization tool and has strong vitality, was presented by Eberhart and Kennedy [6]. In the past several years, PSO has been successfully applied in many research and application areas [7]–[14]. It has been demonstrated that PSO can get better results in a faster and cheaper way compared with other methods such as ant colony optimization [15] and genetic algorithm (GA) [16].

For the reasons mentioned above, in order to make more contributions in wire length optimization, we propose a ML-OASMT construction algorithm based on PSO called ML-OAXSMT-PSO. The main contributions of this paper can be summarized as follows.

1) We integrate the crossover operator and mutation operator of genetic algorithm into PSO to solve the discrete ML-OAXSMT problem. The two operators are blended with the union-find partition to make the particles have a greater ability to alter the topology of the routing tree and avoid the obstacles. Meanwhile, the edge-transformation strategy based on the new operators has a stronger ability of optimizing wire length.

Manuscript received January 26, 2014; revised May 22, 2014; accepted July 18, 2014. Date of publication August 6, 2014; date of current version April 13, 2015. This work was supported in part by the National Basic Research Program of China under Grant 2011CB808000, in part by the National Natural Science Foundation of China under Grant 11271002, Grant 61103175, and Grant 61300102, in part by the Fujian Province High School Science Fund for Distinguished Young Scholars under Grant JA12016, in part by the Fujian Natural Science Funds for Distinguished Young Scholar under Grant 2014J06017, and in part by the Program for New Century Excellent Talents in Fujian Province University under Grant JA13021. This paper was recommended by Associate Editor Y. Tan.

The authors are with the College of Mathematics and Computer Sciences, Fuzhou University, Fuzhou 350116, China and also with the Fujian Provincial Key Laboratory of Network Computing and Intelligent Information Processing, Fuzhou 350116, China (e-mail: liu_genggeng@126.com; fzu_hx@163.com; fzugwz@163.com; yuzhenniu@gmail.com; cgl@fzu.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2014.2342713

2) Some effective methods are proposed to further improve the performance of ML-OAXSMT-PSO algorithm, which including a simple encoding strategy, a pretreatment strategy, and a penalty function. Experimental results show that the proposed algorithm from a global perspective of multilayer structure is superior to the state-of-the-art multilayer routing algorithms on the total cost, which is the most important optimization target for constructing the multilayer Steiner tree.
3) This paper redefines the edge cost and then constructs the obstacle-avoiding preferred direction X-architecture Steiner tree (OAPD-XST) based on ML-OAXSMT-PSO algorithm. Furthermore, we discuss the four models of preferred direction constraints and analyze their advantages and disadvantages through the experimental simulation. To our best knowledge, this is the first work addressing the OAPD-XST problem.

The remainder of the paper is organized as follows. Section II introduces the related work of SMT and its deformations. We formulate the ML-OAXSMT problem, and introduce some definitions in Section III. The corresponding implementation details and the global convergence proof of ML-OAXSMT-PSO are described in Section IV. We describe four preferred direction models and the algorithm design of the OAPD-XST problem in Section V. In Section VI, several comparisons are presented to prove the good performance of ML-OAXSMT-PSO algorithm and the experimental results for OAPD-XST problem are also included. Finally, the whole work is concluded in Section VII.

II. Related Work

In a routing plane, the goal of the SMT problem is to connect all the given pins through some extra points (called Steiner points) to achieve a minimal total length. SMT and its deformations could be applied in congestion/wire length/time delay estimation during floorplanning, placement and routing. As a result, there are a lot of achievements on those topics, and in this section we introduce three categories of works related with our work, namely X-architecture SMT, single-layer OASMT, and multilayer SMT.

A. X-Architecture SMT

Scholars have done some tentative research in non-Manhattan architecture which can bring considerable optimization of physical design indexes such as wire length [12]. Especially, the special industry alliance arises to promote X-architecture and it provides the foundation of implementation and verification for the study of X-architecture. In this context, Hong *et al.* [17] pointed out that the studies of X-architecture would become a hot area of research, and the SMT problem is still one of the most critical problems under X-architecture. Meanwhile, some challenges and opportunities afforded by the X-architecture are presented in [2].

An exact algorithm with a variety of pruning techniques was introduced to construct the X-architecture SMT [18]. Since it has the high algorithm complexity, several heuristic methods were proposed for SMT problem. The spanning graph-based X-architecture Steiner tree construction algorithm was presented in [19]. Furthermore, consider that the SMT problem is an NP-hard problem [20], some evolutionary algorithms, which had been shown to have good application prospects in solving the NP-hard problems, were presented for solving the X-architecture SMT problem [15], [21]. In addition, some study for constructing the timing-driven non-Manhattan architecture SMT were carried out, including the timing-driven X-architecture SMT [22] and the timing-driven Y-architecture SMT [23].

B. Single-Layer OASMT

Although the related research of Section II-A is to construct the SMT based on non-Manhattan architecture and has made a series of achievements, it ignores the presence of obstacles. Here we introduce the study for constructing single-layer OASMT (SL-OASMT), including both rectilinear architecture and X-architecture. The algorithms for constructing SL-OASMT mainly contain four classes: construction-by-correction algorithm, nondeterministic algorithm, graph-based algorithm and exact algorithm.

Construction-by-correction algorithm first constructs a SMT regardless of the obstacles and then replaces the edges that overlap with obstacles [24]. The process of the algorithm is simple, but it is easy to get a routing tree with low quality. Nondeterministic algorithm is based on some meta-heuristics. An ant colony optimization-based local search strategy was presented to construct an obstacle-avoiding rectilinear SMT (OARSMT) [25]. Further considering the X-architecture, an efficient algorithm based on PSO was proposed for the obstacle-avoiding X-architecture SMT (OAXSMT) construction [3]. Most of the SL-OASMT construction algorithms are based on spanning graph. And it can reduce the complexity of the problem to some extent and has a trade-off between the wire length and running time. An efficient algorithm with a generalized connection graph was proposed to compute OARSMT, but only for a 3-terminal net [26]. An obstacle-avoiding constrained Delaunay triangulation method was designed for constructing the full connected tree and could handle the SL-OASMT problem under different architectures [5]. Based on a fast and accurate rectilinear SMT (RSMT) algorithm, Ajwani *et al.* [27] designed an effective approach for constructing OARSMT. A new concept of the Steiner point locations was introduced and an algorithm by choosing an appropriate Steiner point set was designed to obtain an effective solution [28]. Chow *et al.* [29] presented a maze routing based heuristics to solve OARSMT problem. The fourth class of algorithms is the exact algorithm which can get the accurate solution for OARSMT problem. A two-phase approach was developed for the construction of OARSMTs [30]. First, a set of full Steiner trees (FSTs) with blockages was generated. Then the branch-and-cut search was designed for choosing the appropriate FSTs to construct an OARSMT. Huang and Young [31] further constructed the OARSMT based on geometric approach which was able to handle complex-shape obstacles.

C. Multilayer SMT

With the advance of manufacturing process, the multilayer routing has become one of the research hotspots in VLSI automation design and then the multilayer routing algorithm becomes very important [32]. However, the related research achievements in Sections II-A and B are designed for single layer and have not yet involved the construction of multilayer SMT. The differences between SL-OASMT and the multilayer obstacle-avoiding SMT (ML-OASMT) can be summarized as follows [33]. On one hand, the shortest path between two pins for SL-OASMT can be constructed by considering only the pin vertexes and corners of obstacles while for ML-OASMT problem it can not be constructed in the same way for most cases. On the other hand, the total cost of SL-OASMT is wire length while ML-OASMT still needs to find the right position for vias and calculates the cost of vias as part of its total cost. In this subsection, we introduce the construction of the multilayer SMT, including the multilayer SMT with and without the presence of obstacles.

Totsukawa *et al.* [34] proposed a parallel genetic algorithm for constructing the multilayer rectilinear Steiner minimal tree (ML-RSMT). Based on the refinement of minimum routing region and the concept of the hidden Steiner-point assignment, a merging based approach was proposed to construct a timing-driven multilayer rectilinear Steiner tree with obstacle avoidance [38]. Lin *et al.* [33] considered the presence of obstacles and first formulated the multilayer obstacle-avoiding rectilinear SMT (ML-OARSMT) problem. However, Lin *et al.* did not take into account an important routing constraint of actual chip design, which is called preferred direction constraint. In view of preferred direction constraint, Yildiz and Madden [36] presented the preferred direction Steiner tree model, but it did not consider the obstacles. Thus Liu *et al.* [37] constructed the obstacle-avoiding preferred direction rectilinear Steiner tree (OAPD-RST) and it can better handle actual chip routing with the acceptable completion rate. Moreover, a Steiner point location selection strategy for SL-OASMT was extended to construct ML-OASMT [38]. Chuang and Lin [39] first determined a connection ordering for all pins, and then iteratively connected every two neighboring pins by a greedy heuristic algorithm. Then an effective algorithm for constructing ML-OARSMT was proposed [39].

All of the above-mentioned work focused on ML-OARSMT problem. However, for non-Manhattan architecture, Lin *et al.* [40] designed a method for solving the ML-OASMT in X-architecture. On one hand, considering the optimizations of wire length and vias, the method of [40] separately constructed OAXSMT for each routing layer and then found the shortest connection path between every two abutting layers. However, the method of [40] transformed the ML-OAXSMT problem into multiple OAXSMT problems, and greatly deteriorated the quality of routing solution since it cannot search the solution from a global perspective of multilayer structure. As for how to influence and reduce the routing quality, the related analysis and experimental simulation results were given in Sections IV-F and VI-D, respectively. On the other hand, the OAPD-XST problem is a more restricted version of the ML-OAXSMT problem. The construction of OAPD-XST requires each edge to satisfy the preferred direction constraint and is more in line with the industrial production. Thus, the OAPD-XST is also one of the important models in routing problem. However, the study of this more practical significant problem has not been carried out in [40].

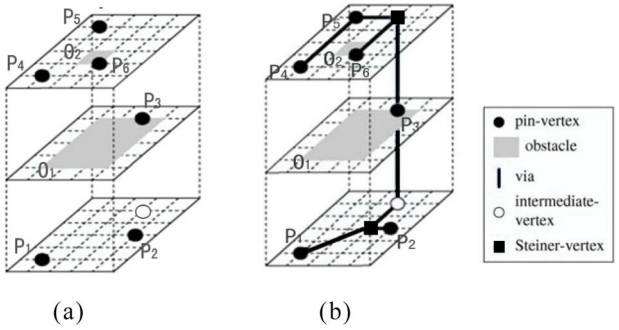


Fig. 1. Problem description of ML-OAXSMT. (a) Distribution map of pins and obstacles [33]. (b) Constructed ML-OAXSMT.

With the introduction of rectilinear architecture, multiple routing layers, and preferred direction constraint, the OASMT problem raised further to OARSMT, ML-OARSMT, and OAPD-RST. For the X-architecture routing, it raised to OAXSMT and ML-OAXSMT. However, the OAPD-XST problem has not been studied.

In view of the shortages of the existing work and the requirements of VLSI physical design, this paper presents an effective algorithm for ML-OAXSMT problem to optimize the total cost from a global perspective of multilayer structure. Moreover, the study of OAPD-XST problem has been carried out in this paper and to our best knowledge, this is the first time to analyze and construct the OAPD-XST.

III. Problem Formulation and Method Model

A. ML-OAXSMT Problem

Given the cost of a via C_v, the number of layers N_l, a set of n pins $P = \{P_1, P_2, P_3, \ldots, P_n\}$, and a set of m obstacles $O = \{O_1, O_2, O_3, \ldots, O_m\}$, construct a multilayer X-architecture Steiner tree to connect the pins in P through some Steiner points such that: 1) each edge of routing tree satisfies the X-architecture constraint and cannot intersect with any obstacle in O and 2) via is used to connect different routing layers and cannot intersect any obstacle in O. And the total cost of the routing tree is minimized.

The ML-OAXSMT problem, as shown in Fig. 1(a), contains three routing layers ($N_l = 3$), six pins ($n = 6$), and two rectangular obstacles ($m = 2$). One solution of ML-OAXSMT is given in Fig. 1(b) and has two vias. We select the bottom-left endpoint, as shown in Fig. 1, as the coordinate origin and set up the 3-D coordinate system, and then the input information for pins and obstacles are listed in Tables I and II, respectively. In Table I, n represents the serial number of pins and x, y, z represent the x-coordinate, y-coordinate and z-coordinate (the z-coordinate is the serial number of the routing layer which the pin located in), respectively. In Table II, m represents the serial number of obstacles, and x_b, y_b, z_b, x_t, y_t, z_t represent

TABLE I
PIN INFORMATION FOR ML-OAXSMT

n	P_1	P_2	P_3	P_4	P_5	P_6
x	1	4	3	1	1	3
y	1	3	5	1	5	2
z	1	1	2	3	3	3

TABLE II
OBSTACLE INFORMATION FOR ML-OAXSMT

m	x_b	y_b	z_b	x_t	y_t	z_t
O_1	1	1	2	4	5	2
O_2	2	2	3	3	3	3

the x-coordinate, y-coordinate and z-coordinate of bottom-left and top-right corners of obstacles, respectively.

B. Definitions

Definition 1 (Pseudo-Steiner): For convenience, we assume that the endpoints except for the pin vertexes can be referred to as the pseudo-Steiner points. Both S_1 and S_2 in Fig. 2 are the pseudo-Steiner points which contain the Steiner points in our paper.

Definition 2 (0 Choice): In Fig. 2(a), let $P_1 = (x_1, y_1, z_1)$ and $P_2 = (x_2, y_2, z_2)$ be the two endpoints of a line segment L, where $x_1 < x_2$. The 0 Choice of L is given in Fig. 2(b), which first from P_1 leads a rectilinear side toa pseudo-Steiner point S_1 and then connects to S_2 in the other layer by the via and finally from S_2 leads an X-architecture direction side to P_2. If both P_1 and P_2 locate in the same layer, it does not cause the via since S_1 overlaps with S_2.

Definition 3 (1 Choice): The 1 Choice of L is given in Fig. 2(c), which first from P_1 leads an X-architecture direction side to a pseudo-Steiner point S_1 and then connects to S_2 in the other layer by the via and finally from S_2 leads a rectilinear side to P_2. If both P_1 and P_2 locate in the same layer, it does not cause the via.

Definition 4 (2 Choice): The 2 Choice of L is given in Fig. 2(d), which first from P_1 leads a vertical side to a pseudo-Steiner point S_1 and then connects to S_2 in the other layer by the via and finally from S_2 leads a horizontal side to P_2. If both P_1 and P_2 locate in the same layer, it does not cause the via.

Definition 5 (3 Choice): The 3 Choice of L is given in Fig. 2(e), which first from P_1 leads a horizontal side to a pseudo-Steiner point S_1 and then connects to S_2 in the other layer by the via and finally from S_2 leads a vertical side to P_2. If both P_1 and P_2 locate in the same layer, it does not cause the via.

IV. DETAILS OF ML-OAXSMT-PSO ALGORITHM

The basic PSO can not obtain a discrete solution for the ML-OAXSMT problem as a result of its continuous nature, and thus some available methods must be designed in the process of PSO to solve discrete problem. Three typical discrete PSO algorithms have been proposed.

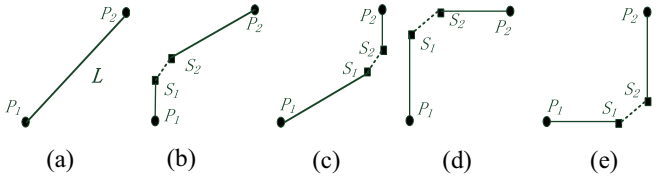


Fig. 2. Four options of the steiner point for line segment L. (a) Line segment L. (b) 0 Choice. (c) 1 Choice. (d) 2 Choice. (e) 3 Choice.

1) The speed of particle is viewed as the probability of position change [4], [42].
2) The continuous PSO is directly discredited and used to solve the discrete problem [43], [44].
3) The PSO operators are redefined according to the discrete problem [45].

In our previous works, the importance of VLSI physical design and the advantages of PSO have led to many research results in partitioning, floorplanning, and routing [8], [17], [46]. According to the ML-OAXSMT problem, this paper presents an effective discrete PSO algorithm on the basis of the previous work.

A. Encoding Scheme

Due to its robustness, PSO algorithm does not demand the rigorous coding requirements. However, a good encoding scheme can not only improve the search efficiency, but also reduce the redundancy of the search space, and thereby enhances the performance of the algorithm. Generally, the selection of the encoding scheme should consider three main principles: completeness, soundness, and nonredundancy [46].

In this paper, we adopt an edge-vertex encoding strategy which is suitable for X-architecture and multilayer structure. This encoding strategy can satisfy the completeness and nonredundancy principle. One candidate Steiner tree is represented as lists of spanning tree edges, and each edge augments with a pseudo-Steiner point choice which specifies the conversion from the spanning tree edge to the X-architecture edge. The pseudo-Steiner point choice, namely *pspc*, includes four types as shown in Definitions 2–5 and the value of *pspc* is 0, 1, 2 or 3 which denotes 0 Choice, 1 Choice, 2 Choice, or 3 Choice, respectively. Each routing edge encoding has four digits and the first three digits (the serial number of two pins and the pseudo-Steiner point choice) is encoded one routing edge while the last digit (namely label) represents the route situation of each edge which can be used to judge whether or not the edge avoids obstacles and at the same time record the number of vias caused by the edge. The absolute value of label represents the number of vias. If label is positive, the edge avoids obstacle; if not, the edge intersects obstacle. When the number of vias caused by the edge e is zero, i.e., the two pins of the edge e are in the same layer, if e does not violate the obstacle-avoiding constraint, the label value is set to be 0; otherwise, it is set to be -0.9.

If a net has n pins, a spanning tree would have $n-1$ edges, $n-1$ pseudo-Steiner points, $n-1$ label, and one extra digit which is the particle's fitness. Besides, two digits represent two vertices of each edge, so the length of one particle is $4(n-1)+1$. For example, one routing tree ($n = 7$) can be

expressed as one particle whose code can be expressed as the following numeric string:

1210 3431 5620 6720 1310 4611 19.8284

where the number "19.8284" is the fitness of the particle. Taken the second substring (3, 4, 3, 1) for example, it represents one edge of the spanning tree which composed of pin vertex 3 and pin vertex 4 and the pseudo-Steiner point choice is 3 Choice in Definition 5. The last digit "1" shows that the edge avoids obstacles but crosses one layer.

B. Pretreatment Strategy

In the ML-OAXSMT-PSO algorithm, the introduction of label (the route situation of each edge) is used to judge whether or not the edge avoids obstacles and at the same time record the number of vias caused by the edge. The computational complexity of label is one of the main factors influencing the speed of the algorithm. While calculating the fitness function value of each particle, the algorithm always needs to calculate the value of label. It will increase the computation of the algorithm since many judgments about obstacle-avoiding and much calculation of the number of vias are repeated. For this reason, the pretreatment strategy for calculating the value of label is designed for our algorithm. Before the algorithm enters the iterative process of PSO, we calculate the label value of every possible routing edge under the proposed encoding strategy and store them in the array. When the algorithm enters the iterative process, it can directly query the required value of label. Thereby the computation of label has been significantly reduced and some redundant computation can be avoided. The specific reduction of the computation with the pretreatment strategy is reported in detail in Section VI-A.

C. Fitness Function and Penalty Mechanism

The total cost of ML-OAXSMT does not only contain the wire length while it still needs to find the right position for vias and calculate the cost of vias as part of the total cost. By using the edge-vertex encoding strategy, the distance between two pins i and j is calculated as follows:

$$Odis(i,j) = \begin{cases} (\sqrt{2}-1)\min(|x_i-x_j|, |y_i-y_j|)+ \\ \max(|x_i-x_j|, |y_i-y_j|), \text{if } pspc = 0 \text{ or } 1 \\ |x_i-x_j| + |y_i-y_j|, \text{else} \end{cases} \quad (1)$$

where x_i (x_j) and y_i (y_j) represent horizontal and vertical coordinates of point i (j), respectively. Clearly, $|x_i - x_j|$ ($|y_i - y_j|$) denotes the absolute value, and the symbol min (max) denotes the minimum (maximum) between $|x_i - x_j|$ and $|y_i - y_j|$.

The cost $C_{\text{via}}(i,j)$ of via between pins i and j can be computed by $|z_i - z_j| \times C_v$, where $z_i(z_j)$ represents the serial number of the layer which i (j) is located in (the serial number of the layer is from 1 to 10) and the value of C_v is 3 or 5 [33]. And the total cost of the ML-OAXSMT T can be computed as follows:

$$\begin{aligned} \cos t(T) &= \sum_{(i,j)\in T} \cos t(i,j) \\ &= \sum_{(i,j)\in T} (Odis(i,j) + C_{\text{via}}(i,j)). \end{aligned} \quad (2)$$

The fitness function value of our proposed algorithm can not contain the total cost of ML-OAXSMT only since the solution under the edge-vertex encoding strategy maybe violates the obstacle-avoiding constraint. For this reason, we incorporate the penalty mechanism into the constructed fitness function to punish the invalid solutions. Futhermore, the number of vias is an important optimization index for routability and manufacturability of VLSI chip design and hence this paper optimizes the number of vias. The generation of vias tends to be inevitable in the multilayer routing but could be partly reduced by an effective method. Thus we consider punishing the generation of vias through the design of penalty mechanism. In conclusion, the penalty mechanism which inflicts the different degrees of punishment on both the invalid solutions and the generation of vias is incorporated into the fitness function.

Definition 6 (Penalty Function): $P(X, Q) = N(X) + Q \times S(X)$, where $P(X, Q)$ is a penalty function and $Q \times S(X)$ is a penalty term. Q is a penalty factor and a sequence whose limit is ∞.

Penalty function method is a widely used and effective optimization method. The basic idea is to give the iteration points, which attempt to violate the constraint, a lot of punishment on the objective function in the process of solving constrained optimization problems and then the constrained problems are converted into the unconstrained ones. As for Definition 6, $N(X)$ usually represents the cost of solution without considering the constrained condition, while the $S(X)$ may represents the additional cost with the constrained problem. $N(X)$ and $S(X)$ maybe have the different meaning. However, they may be the same value.

The fitness function based on the penalty mechanism for ML-OAXSMT-PSO algorithm is computed as follows:

$$F = P(X, Q) = \sum_{e\in T} \text{penalty}(e) \times \cos t(T) \quad (3)$$

where $Q = \sum_{(i,j)\in T} \text{penalty}(i,j) - 1$, (i,j) represents the edge of the routing tree T, $N(X) = \text{cost}(T)$, $S(X) = \text{cost}(T)$.

In the experimental tests, this paper selects three functions f_1, f_2, f_3 as the calculation formula of penalty(e) and these three functions are defined as follows:

$$f_1(x) = \begin{cases} lb, x > 0 \\ ub, x < 0 \\ 1, x = 0 \end{cases} \quad (4)$$

$$f_2(x) = \begin{cases} \frac{lb-1}{l_{\max}}x + 1, x > 0 \\ \frac{ub-1}{-l_{\max}}x + 1, x \le 0 \end{cases} \quad (5)$$

$$f_3(x) = \begin{cases} \frac{1}{\frac{1}{lb}+\frac{1}{1+e^x}}, x > 0 \\ \frac{1}{\frac{1}{ub}+\frac{1}{1+e^{-x}}}, x \le 0 \end{cases} \quad (6)$$

where the parameters lb, ub are user-defined and greater than 1. The two parameters of the penalty function are given as follows: lb = 10/9 and ub = 2. x represents the label value of each edge and $l_{\max}$ represents the maximum serial number of the routing layer. We can find that the three extreme points of the penalty functions f_1, f_2, and f_3 are the same and their function values are (0, 1), (−10, 2), and (10, 10/9), respectively. However, the major difference is their slopes.

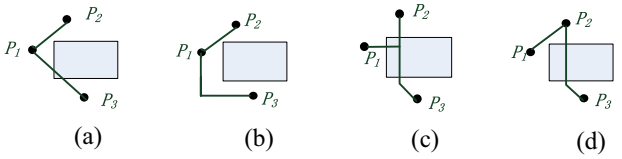


Fig. 3. Steiner point transformation and the need of the edge transformation. (a) One situation violating the obstacle-avoiding constraint. (b) Introduction of Steiner point transformation. (c) Another situation violating the obstacle-avoiding constraint. (d) Edge transformation is required to avoid the obstacle.

The fitness function based on the penalty mechanism is proposed, and the algorithm does not directly delete the particles which violate the constraints before the next iteration. And thus the partial optimal information carried by those particles is retained and might be useful in the later iterations. As a result, the merits of the particle could be evaluated reasonably on one hand. On the other hand, the reasonable preservation of the partial optimal information can maintain the high population diversity of the ML-OAXSMT-PSO algorithm.

D. Update Formula of the Particle

The ML-OAXSMT problem is a discrete optimization problem. Therefore, the particle update formula of the basic PSO can no longer suit for solving the ML-OAXSMT problem. For this reason, the crossover and mutation operators combined with union-find partition are designed to construct the discrete particle update formula for ML-OAXSMT-PSO algorithm. The improved particle update formula is defined as follows:

$$X_i^t = N_3\left(N_2(N_1(X_i^{t-1}, w), c_1), c_2\right) \tag{7}$$

where N_1 denotes the mutation operation and N_2, N_3 denote the crossover operations. We assume that r_1, r_2, r_3, which show up in (8–10), are random numbers on the interval (0, 1).

For constructing XSMT [21], the update operations only considered the transformation of the Steiner point and caused the ability of obstacle-avoiding to be limited. For example, as shown in Fig. 3(a), the edge $\overline{P_1P_3}$ with two endpoints P_1 and P_3 violates the obstacle-avoiding constraint. With the update operations in [21], the pseudo Steiner point choice of $\overline{P_1P_3}$ transforms from 0 Choice to 2 Choice and the routing tree without the constraint violations is obtained in Fig. 3(b), that is, $\overline{P_1P_3}$ avoids the obstacle. However, as shown in Fig. 3(c), if we only consider the transformation of the Steiner point in the iterative process, the routing tree can not avoid the obstacle, as shown in Fig. 3(d). Therefore the topology of the routing tree needs to be changed, that is, the edge transformation is necessary to be introduced in the update operation and then $\overline{P_2P_3}$ can transform to $\overline{P_1P_3}$ and finally the obstacle-avoiding Steiner tree can be obtained. For this reason, the edge transformation is proposed in the ML-OAXSMT-PSO algorithm.

The introduction of the edge transformation not only enhances the obstacle-avoiding ability of the Steiner tree, but also further optimizes the wire length of the Steiner tree (the experimental results are shown in Section VI-A). However, the introduction of the edge transformation may lead to the appearance of the loop and generate the invalid solution in the iterative process, which destroys the soundness principle of the particle encoding. How to overcome the shortcoming caused by the edge transformation in the evolutionary process of ML-OAXSMT-PSO? The union-find partition which keeps track of the components connected so far is integrated into the following update operators.

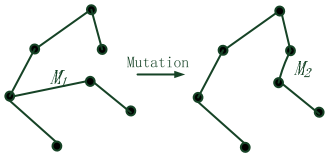


Fig. 4. Mutation operation with the union-find partition.

The velocity of particles can be written as

$$W_i^t = N_1\left(X_i^{t-1}, w\right) = \begin{cases} M\left(X_i^{t-1}\right), r_1 < w \\ X_i^{t-1}, \text{others} \end{cases} \tag{8}$$

where w denotes the mutation probability.

Mutation operator randomly deletes an edge from the spanning tree, and replaces it with another randomly generated edge in order to make sure that the spanning tree is connected. We use the union-find partition to record all the rest of points as two sets after one edge is deleted, and randomly select a point from the two point sets respectively, and then connect the two points to form a new spanning tree. The principle is shown in Fig. 4, where the edge M_1 represents the edge to be removed and the edge M_2 represents the new edge.

The cognitive personal experience of particles can be written as

$$S_i^t = N_2\left(W_i^t, c_1\right) = \begin{cases} C_p\left(W_i^t\right), r_2 < c_1 \\ W_i^t, \text{others} \end{cases} \tag{9}$$

where c_1 denotes the crossover probability of the particles and the personal optimal solution.

The cooperative global experience of particles can be written as

$$X_i^t = N_3\left(S_i^t, c_2\right) = \begin{cases} C_g\left(S_i^t\right), r_3 < c_2 \\ S_i^t, \text{others} \end{cases} \tag{10}$$

where c_2 denotes the crossover probability of the particles and the global optimal solution.

When implementing the crossover operator in (9) and (10), we sort the edges according to the serial number of pins from small to large order in both of the two spanning tree, then by using the union-find partition to choose the same edges from the two sorted tree as a set, the rest of the different edges as another set, and the first set of edges directly as the new spanning tree's edges. Finally, we iteratively select one edge randomly from the second set and add it to the new spanning tree, while we prevent generating circle by using the union-find partition. The principle is shown in Fig. 5 where $C_1, C_2, C_3, C_4, C_5, C_6$ represents the different edges between the two spanning tree, and the set C_1, C_3, C_5 represents the new edges of the new spanning tree.

E. Procedure of ML-OAXSMT-PSO

The detail procedure of ML-OAXSMT-PSO can be summarized as follows.

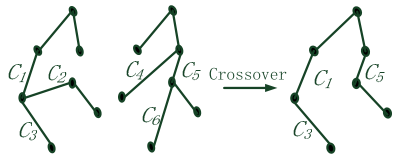


Fig. 5. Crossover operation with the union-find partition.

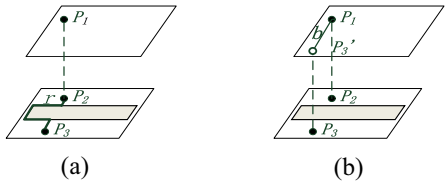


Fig. 6. Optimization ability on the total cost between GM-SLP and ML-OAXSMT-PSO (situation 1). (a) GM-SLP. (b) ML-OAXSMT-PSO.

Step 1: Load circuit net-list data and sort them in ascending order according to the serial number of layer and the size of x-coordinate and y-coordinate.

Step 2: Initialize the population size, maximum iterations, inertia weight and acceleration factors, and randomly initialize the population.

Step 3: Preprocess the label value of every possible routing edge under our proposed encoding strategy and store them in the array. When the algorithm enters the iterative process, it can directly query the required value of label.

Step 4: Calculate the fitness of each particle among the initial population according to (3). And initialize the personal optimal solution of each particle and select the particle with minimal fitness as the global optimal solution of the population.

Step 5: Adjust the position and velocity of each particle according to (7)–(10) and then calculate the fitness value of each particle. If its fitness value is less than the personal optimal value, it is set as the personal optimal solution. If its fitness value is less than the global optimal value of the population, it is set as the global optimal solution.

Step 6: Check the termination condition. If fulfilled, the run is terminated and the ML-OAXSMT solution is obtained. Otherwise, go to Step 5.

F. Properties of ML-OAXSMT-PSO

Current work for solving ML-OAXSMT problem is to transform the ML-OAXSMT problem into multiple SL-OAXSMT problems [40], which is called the greedy method based on single-layer point (GM-SLP) in this paper. However, our proposed algorithm considers the ML-OAXSMT problem from a global perspective of multilayer structure, and can effectively use the appropriate vias to search more excellent solution space. In this subsection, we analyze the difference between ML-OAXSMT-PSO algorithm and GM-SLP in the view of the optimization ability of the total cost and the problem model.

Theorem 1: For the ML-OASMT problem, our proposed ML-OAXSMT-PSO algorithm can achieve a great improvement, which ranges from 0% to 100%, over the GM-SLP algorithm on the total cost.

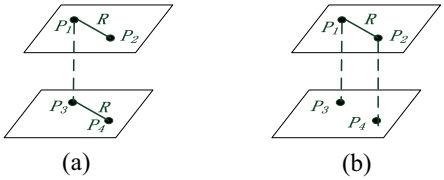


Fig. 7. Optimization ability on the total cost between GM-SLP and ML-OAXSMT-PSO (situation 2). (a) GM-SLP. (b) ML-OAXSMT-PSO.

Proof: As shown in Fig. 6(a), considering the shortest path among P_1, P_2 and P_3, if the GM-SLP algorithm is adopted, the total cost is $2\times r+C_v+b$, where b is the distance between P_2 and P_3. The total cost is $b+2\times C_v$ by ML-OAXSMT-PSO algorithm as shown in Fig. 6(b). And the improvement of the ML-OAXSMT-PSO algorithm over GM-SLP on the total cost is $(2\times r-C_v)/(2\times r+C_v+b)$. When the length of $2\times r$ is far greater than C_v and b, the improvement is close to 100%. This situation exists in the benchmark circuits and the proposed ML-OAXSMT-PSO algorithm can has a stronger ability on optimizing the total cost than the GM-SLP algorithm. ■

Theorem 2: For the multilayer SMT (ML-SMT) problem, our proposed ML-OAXSMT-PSO algorithm can achieve a great improvement, which ranges from 0% to 50%, over the GM-SLP algorithm on the total cost.

Proof: As shown in Fig. 7(a), considering the shortest path among P_1, P_2, P_3 and P_4, if the GM-SLP algorithm is adopted, the total cost is $2\times R+C_v$. The total cost is $R+2\times C_v$ by our proposed algorithm as shown in Fig. 7(b). And the improvement over GM-SLP on the total cost is $(R-C_v)/(2\times R+C_v)$. When the length of R is far greater than C_v, the improvement is close to 50%. In most of the benchmark circuits, the length of R is far greater than C_v and it further explains that our proposed algorithm can search more excellent solution space of the ML-OAXSMT problem than the GM-SLP algorithm. ■

In order to better analyze the model difference between GM-SLP (which mainly solves multiple SL-OAXSMT and finally constructs the ML-OAXSMT) and ML-OAXSMT-PSO (which designs the encoding strategy according to the feature of ML-OAXSMT problem and directly constructs the ML-OAXSMT), we define the connected model and unconnected model as follows.

Definition 7 (Connected Model): In a routing model, the existence of the obstacles may inhibit some points from be selected as the Steiner point, but each pin can link with the other through no more than one time of 0, or 1, or 2, or 3 Choice, as shown in Fig. 8(a). We call it the connected model.

Definition 8 (Unconnected Model): In a routing model, there is at least one pin can be connected to the other pin by two or more times of 0, or 1, or 2, or 3 Choice, as shown in Fig. 8(b). We call it the unconnected model.

Property 1: Some of benchmark circuits which are unconnected models for the GM-SLP algorithm maybe belong to the connected model for our proposed ML-OAXSMT-PSO algorithm. We fortunately find that using our proposed ML-OAXSMT-PSO algorithm, all benchmark circuits [33] can belong to the connected model by introducing the proper vias.

Fig. 8. Two situations of the obstacle-avoiding Steiner tree problem. (a) Connected model. (b) Unconnected model.

This kind of circumstance can further enhance the soundness of our proposed encoding scheme.

As for SL-OAXSMT of GM-SLP, the edge-vertex encoding scheme under the unconnected model can not satisfy the soundness principle. However, as for ML-OAXSMT-PSO, we construct the routing tree in global view of multilayer structure and can avoid obstacles by the introduction of the proper vias. As a result, the unconnected model for SL-OAXSMT of GM-SLP can become the connected model for the problem under our proposed encoding scheme. The specific process of this situation can consult the analysis of Fig. 6. As shown in Fig. 6(a), the routing problem in the layer which P_2 and P_3 are located is an unconnected model and the effective solution can not be found with the edge-vertex encoding strategy. But with the ML-OAXSMT-PSO algorithm, it can find the feasible solution, as shown in Fig. 6(b), by introducing two vias and the route of the upper layer. We implement ML-OAXSMT-PSO algorithm in all standard benchmarks of ML-OASMT problem and fortunately find that those standard benchmarks belong to the connected model by introducing the proper vias.

Lemma 1: Assume the population size is *pop*, the number of iterations is *iters*, the number of pins is n, and the number of obstacles is m. The time complexity of the proposed algorithm is $O(m \times n^2 + pop \times iters \times n \log n)$.

Proof: Before the algorithm enters the iterative process of PSO, we calculate the status value (label) of every possible routing edge under our proposed encoding strategy and store them in the array. It needs to estimate $n \times (n-1)$ edges to avoiding the m obstacles, and then it costs the time complexity $O(m \times n^2)$. In the inner loop of the proposed algorithm, as shown in Step 5 during the detail procedure of ML-OAXSMT-PSO, it includes the mutation operation, crossover operation and the calculation of fitness function. In the mutation and crossover operations, the sorting steps determine those operations time complexity: $O(n \log n)$, because the time of the union-find partition is just more than linear [47]. Besides, in the process of calculating the fitness function, it can directly query the required value of label in the array and its time complexity is $O(1)$. Moreover, the outer loop is related to the number of particles *pop* and the number of iterations *iters*. Consequently, the time complexity of the proposed algorithm is $O(m \times n^2 + pop \times iters \times n \log n)$. ■

G. Convergence Analysis of ML-OAXSMT-PSO

Theorem 3: The Markov chain of the proposed PSO is finite and homogeneous.

Proof: Follow the similar method of Lv *et al.* [48] and the definition of Finite Markov Chain has been given by Rudolph [49]. ■

Theorem 4: Transition probability matrix of the Markov chain made up of the proposed PSO is positive definite.

Proof: Follow the similar method of Rudolph [49]. ■

Theorem 5 (Limit theorem for Markov chain): Assume that P is a positive stochastic transition matrix of the definite homogeneous Markov chain, then the following apply.

1) There exists a unique probability vector $\overline{P}^T > 0$ which satisfies $\overline{P}^T P = \overline{P}^T$.
2) For any initial state i (e_i^T as its corresponding initial probability), we get $\lim_{k\to\infty} e_i^T P^k = \overline{P}^T$.
3) Limit probability matrix $\lim_{k\to\infty} P^k = \overline{P}$, where $\overline{P}$ is a $n \times n$ stochastic matrix, and all its rows equal to $\overline{P}^T$.

The limit theorem explains that the long-term probability of Markov chain does not depend on its initial states. This theorem is the basis for the convergence of a algorithm.

Lemma 2: If mutation probability $m \succ 0$, the algorithm is an ergodic irreducible Markov chain which has only one limited distribution and nothing to do with the initial distribution, moreover the probability at a random time and random state is greater than zero.

Proof: At the tth time, the jth state probability distribution of population $X(t)$ is

$$P_j(t) = \sum_{j\in S} P_i(1) P_{ij}^{(t)}, t = 1, 2, \ldots \tag{11}$$

According to Theorem 4, we can get the formulation as follows:

$$P_j(\infty) = \lim_{t\to\infty} \left(\sum_{i\in S} P_i(1) P_{ij}^{(t)} \right) = \sum_{i\in S} P_i(1) P_{ij}^{(\infty)} > 0, \forall j \in S. \tag{12}$$

■

Definition 9: Suppose a stochastic variant $Z_t = \max\{f(x_k^{(t)}(i))|k = 1, 2, \ldots, N\}$ which represents the individual best fitness at the t-th step and i-th state of the population. Then the algorithm converges to the global optimum, if and only if

$$\lim_{t\to\infty} P\{Z_t = Z^*\} = 1 \tag{13}$$

where $Z^* = \max\{f(x)|x \in S\}$ represents the global optimum.

Theorem 6: For any i and j, the time transiting of an ergodic Markov chain from the ith state to the jth state is limited.

Theorem 7: The proposed PSO algorithm can converge to the global optimum.

Proof: Suppose that $i \in S$, $Z_t < Z^*$ and $P_i(t)$ is the probability of the proposed PSO algorithm at ith state and the tth step. Obviously $P\{Z_t \neq Z^*\} \geq P_i(t)$, hence we can know that $P\{Z_t = Z^*\} \leq 1 - P_i(t)$. ■

According to Lemma 2, the ith state probability of the operator in the proposed PSO algorithm is $P_i(\infty) > 0$, then

$$\lim_{t\to\infty} P\{Z_t = Z^*\} \leq 1 - P_i(\infty) < 1. \tag{14}$$

Observe a new population such as $X_t^+ = \{Z_t, X_t\}, t \geq 1$, $x_{ti} \in S$ denoting the search space (which is a finite set or a countable set), where Z_t, the same to that in Definition 9,

represents the individual best fitness in current population, X_t denotes the population during the search. As it is easy to prove that the group shift process $\{X_t^+, t \geq 1\}$ is still a homogeneous and ergodic Markov chain, we can know that

$$\begin{aligned} P_j^+(t) &= \sum_{i \in S} P_i^+(1) P_{ij}^+(t) \\ P_{ij}^+ &> 0 (\forall i \in S, \forall j \in S_0) \\ P_{ij}^+ &= 0 (\forall i \in S, \forall j \notin S_0). \end{aligned} \tag{15}$$

So

$$\begin{aligned} &(P_{ij}^+)^t \to 0 (t \to \infty) \\ &P_j^+(\infty) \to 0 (j \notin S_0) \\ &\lim_{t \to \infty} P\{Z_t = Z^*\} = 1. \end{aligned} \tag{16}$$

V. OAPD-XST

A. Preferred Direction Constraint Model of OAPD-XST

The research about the obstacle-avoiding preferred direction Steiner tree (OAPD-ST) problem in Manhattan architecture has been carried out and the preferred direction model of each layer is constant which only vertical edges are allowed in odd number layers, and only horizontal edges are allowed in even number layers. However, the study of preferred direction constraint model of OAPD-XST is open and the research work for the OAPD-XST problem has not yet been found to our best knowledge. This paper explores the superiority and inferiority of some preferred direction constraint models and design an algorithm for the OAPD-XST problem.

The preferred direction constraint models for the OAPD-XST problem mainly include four types.

1) Every five layers alternate model, which the routing direction is 0^0, 90^0, 0^0, 45^0, and 135^0, respectively (namely Model *a*), as shown in Fig. 9(a).
2) Only 0^0 and 135^0 are allowed in odd number layers, and only 45^0 and 90^0 are allowed in even number layers (namely Model *b*), as shown in Fig. 9(b).
3) Only 0^0 and 90^0 are allowed in odd number layers, and only 45^0 and 135^0 are allowed in even number layers (namely Model *c*), as shown in Fig. 9(c).
4) Only 0^0 and 45^0 are allowed in odd number layers, and only 90^0 and 135^0 are allowed in even number layers (namely Model *d*), as shown in Fig. 9(d).

The merits of the above-mentioned four models will be summarized and analyzed from the number of vias, the violation (Definition 11) and the total cost of OAPD-XST. And thus the best preferred direction constraint model for OAPD-XST problem can be obtained and this paper can offer the theory supports for chip design based on non-Manhattan architecture.

B. OAPD-XST Algorithm

The overview of OAPD-XST algorithm can be described as follows.

Step 1: Redefine the cost of each edge in the different layers according to Definition 10.

Definition 10 (The Routing Cost of Each Edge Among OAPD-XST): Given an edge e in the layer *LE*, if the direction of edge e satisfies the preferred direction constraints of *LE*, the cost of edge e (namely $costpd(e)$) is defined as the original distance (namely $olength(e)$) under the proposed edge-vertex encoding strategy. Otherwise, $costpd(e)$ is defined as $olength(e) \times \alpha$. The parameter α is user-defined and greater than 1. In our experiments, the value of α is set to be 3.

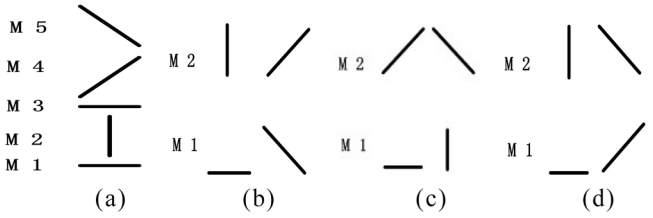


Fig. 9. Four types of the preferred direction constraint models for the OAPD-XST problem. (a) Model *a*. (b) Model *b*. (c) Model *c*. (d) Model *d*.

Step 2: Call the proposed ML-OAXSMT-PSO algorithm and the fitness function is calculated as follows:

$$C(T_x) = \sum_{e_i \in T_x} (costpd(e_i) + \Delta Z \times C_v) \tag{17}$$

where $costpd(e_i)$ is the routing cost of the edge e_i which can be computed according to Definition 10. C_v is the cost of one via and ΔZ represents the number of the vias caused by e_i.

Step 3: Calculate the total violations and the number of vias of OAPD-XST. The violation is described in Definition 11. At the end, the number of vias, the total violations and the total cost of OAPD-XST can be obtained.

Definition 11 (Violations): Each routing edge has the different preferred direction constraints for each model of OAPD-XST. If the routing direction of edge e is inconsistent with the preferred direction constraints of the corresponding layer, one violation is caused.

VI. Experiment Results

All algorithms have been implemented and executed in C/C++. The benchmark circuits in [33] are used to study the performance of the ML-OAXSMT-PSO algorithm. There are totally ten benchmark circuits, five test cases (ind1-ind5) from Synopsys and five random test cases (rt1-rt5) generated by Lin *et al.* [33]. Their parameters are listed in Columns 1 and 2 of Table IV, where n represents the number of pins, m represents the number of obstacles, and N_l is the total number of routing layers. From Table IV, we can find that the number of pins is from 25 to 1000, and the number of obstacles is from 6 to 100, the number of layers is from 5 to 10. To study the performance of the ML-OAXSMT-PSO algorithm, four groups of experiments are carried out from Sections VI-A–D, while the experiment in Section VI-E is to verify the metrics of different models for the OAPD-XST problem. The results of our algorithm are based on average score over 20 runs.

A. Experiment Validation of Edge Transformation and Pretreatment Strategy

The introduction of the edge transformation not only enhances the obstacle-avoiding ability of the Steiner tree (as mentioned in Section IV-E), but also further optimizes the wire length of the Steiner tree. In order to study the effectiveness of the edge transformation strategy for optimizing the wire

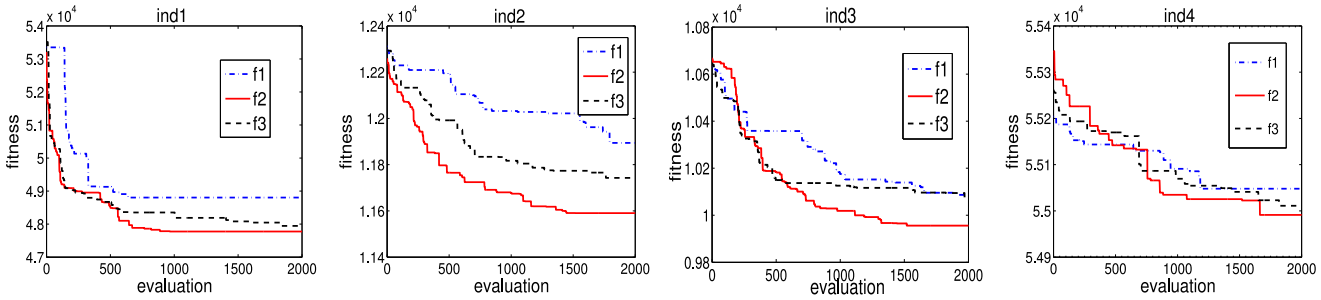


Fig. 10. Convergence curves of the ML-OAXSMT-PSO algorithm based on three different penalty functions. (a) Convergence curves based on f_1, f_2, and f_3 for ind1. (b) Convergence curves based on f_1, f_2, and f_3 for ind2. (c) Convergence curves based on f_1, f_2, and f_3 for ind3. (d) Convergence curves based on f_1, f_2, and f_3 for ind4.

TABLE III
COMPARISON ON THE WIRE LENGTH BETWEEN EXSMT AND XSMT

instance	XSMT	EXSMT	Imp%
1	33.0711	33.0711	0.00
2	27.4853	26.5563	3.38
3	35.4853	35.3848	0.28
4	27.0711	26.8995	0.63
5	26.2426	26.2426	0.00
6	34.7279	34.7279	0.00
7	36.4853	35.3137	3.21
8	34.8995	34.3137	1.68
9	31.8284	31.2426	1.84
10	33.1421	32.8995	0.73
11	28.7279	28.3137	1.44
12	27.0711	26.8995	0.63
13	29.0711	27.6569	4.86
14	32.0711	30.7279	4.19
15	27.2426	27.0711	0.63
average	30.9748	30.4881	1.57

length, the experiment is carried out in the test cases of OR-Library [50]. As shown in Table III, the introduction of the edge transformation (this algorithm is named EXSMT) increases the ability to optimize the wire length which is 1.57% smaller than the wire length of XSMT [21].

The design of pretreatment strategy which calculates the label value of route situation before entering the major loop of PSO can avoid the redundant computation. Among all the benchmarks of [33], the computation of label with pretreatment strategy (WP) is 95.14% smaller than the computation without pretreatment strategy (WOP), as shown in Table IV. The experimental results show the effectiveness of our proposed pretreatment strategy.

B. Comparison Among Three Kinds of Penalty Mechanism for ML-OAXSMT-PSO Algorithm

For the ML-OAXSMT problem, we design the fitness function based on the penalty mechanism, which considers both the presence of obstacles and the generation of vias, to effectively evaluate the merits of the feasible solution. Three kinds of

TABLE IV
REDUCTION IN THE COMPUTATION OF LABEL WITH AND WITHOUT THE PRETREATMENT STRATEGY

Test	$n/m/N_l$	WOP	WP	Imp%
ind1	50/6/5	2299890	1854	99.98
ind2	200/85/6	3806905	37481	99.75
ind3	250/13/10	502468	46287	97.70
ind4	500/100/5	9447739	748815	98.02
ind5	1000/20/5	487215	260445	86.64
rt1	25/10/10	89628	183	99.95
rt2	100/20/10	403012	6190	99.62
rt3	250/50/10	1551069	74281	98.80
rt4	500/50/10	1075891	284737	93.38
rt5	1000/100/5	2418451	2169159	77.58
average		2208227	362943	95.14

penalty function f_1, f_2, and f_3 are integrated into the fitness function of the ML-OAXSMT-PSO algorithm to test their effectiveness for the ML-OAXSMT problem and then the most suitable one is selected as our penalty mechanism.

The experimental result of the ML-OAXSMT-PSO algorithm based on the different penalty functions is shown in Fig. 10, which tests in four benchmark circuits (ind1, ind2, ind3, ind4). From Fig. 10, we can find that the ML-OAXSMT-PSO algorithm based on f_2 is superior to the ones based on f_1 and f_3. Moreover, from the volatility of the four groups of convergence curves as shown in Fig. 10, we can find that the ML-OAXSMT-PSO algorithm based on f_2 has better diversity than others. The convergence curves of the proposed algorithm based on f_1, f_2, and f_3 for other benchmark circuits (ind5, rt1-rt5) are similar to the ones for these four ones (ind1-ind4). Therefore, this paper selects a relatively optimal f_2 as the penalty function to be applied in the ML-OAXSMT-PSO algorithm.

C. Comparison With Three ML-OARSMT Algorithms

In order to validate the effectiveness of the ML-OAXSMT-PSO algorithm, we compare the ML-OAXSMT-PSO algorithm (namely OURS) with three ML-OARSMT algorithms in the above-mentioned benchmark circuits. These ML-OARSMT algorithms are the construction-by-correction method for the ML-OARSMT problem [33] (namely CC), the ML-OARSMT algorithm based on the spanning graphs [33] (namely SG),

TABLE V
COMPARISON BETWEEN OUR ALGORITHM AND THREE ML-OARSMT ALGORITHMS ON THE TOTAL COST UNDER $C_v = 3$ ($C_v = 5$)

Test	CC	SG	CG	OURS	Imp(CC)%	Imp(SG)%	Imp(CG)%
ind1	82556(82674)	55537(55635)	54207(54305)	48996(49082)	40.65(40.63)	11.78(11.78)	9.61(9.62)
ind2	17568(17389)	12512(12899)	12008(12420)	11652(12028)	33.67(30.83)	6.87(6.75)	2.96(3.15)
ind3	17837(17599))	10973(11698)	10555(11251)	9654(10306)	45.88(41.44)	12.02(11.90)	8.54(8.40)
ind4	273235(273235)	77033(77033)	77292(77292)	54965(54965)	79.88(79.88)	28.65(28.65)	28.89(28.89)
ind5	2331494(2331494)	14515511(14515511)	14599961(14599961)	14356523(14356523)	38.42(38.42)	1.10(1.10)	1.67(1.67)
rt1	5095(5277)	4334(4486)	4169	3855(3987)	24.33(24.44)	11.05(11.12)	7.53(7.47)
rt2	12885(13874)	9434(9812)	9132	8774(9148)	31.90(34.06)	6.99(6.76)	3.92(4.21)
rt3	23233(22727)	15569(16384)	14750	14084(14974)	39.38(34.12)	9.54(8.61)	4.52(4.66)
rt4	29464(30542)	22034(23883)	21013(22885)	21070(22836)	28.49(25.23)	4.38(4.39)	-0.27(0.22)
rt5	38702(43775)	27890(29598)	26970(28598)	27410(28898)	29.18(33.99)	1.72(2.37)	-1.63(-1.05)
average					39.18(38.31)	9.41(9.36)	6.57(6.74)

TABLE VI
COMPARISON BETWEEN OUR ALGORITHM AND THREE ML-OARSMT ALGORITHMS ON THE NUMBER OF VIAS UNDER $C_v = 3$ ($C_v = 5$) AND CPU TIME (SECONDS)

Test	CC	SG	CG	OURS	Imp(CC)%	Imp(SG)%	Imp(CG)%	Time(CC)	Time(SG)	Time(OURS)
ind1	59(59)	49(49)	49(49)	43(43)	27.12(27.12)	12.24(12.24)	12.24(12.24)	0.02	0.07	1.11
ind2	293(283)	224(208)	206(206)	188(188)	35.84(33.57)	16.07(9.62)	8.74(8.74)	0.21	3.01	99.64
ind3	529(480)	359(343)	348(348)	326(326)	38.37(32.08)	9.19(4.96)	6.32(6.32)	0.32	3.35	50.62
ind4	0(0)	0(0)	0(0)	0(0)	0.00(0.00)	0.00(0.00)	0.00(0.00)	0.97	8.21	186.72
ind5	0(0)	0(0)	0(0)	0(0)	0.00(0.00)	0.00(0.00)	0.00(0.00)	3.28	45.71	212.01
rt1	91(91)	76(76)	70(70)	66(66)	27.47(27.47)	13.16(13.16)	5.71(5.71)	0.01	0.06	0.41
rt2	290(263)	215(200)	209(209)	187(187)	35.52(28.90)	13.02(6.50)	10.53(10.53)	0.06	0.90	8.68
rt3	705(633)	490(429)	478(478)	445(445)	36.88(29.70)	9.18(-3.73)	6.90(6.90)	0.41	6.84	212.84
rt4	1282(1144)	918(879)	936(936)	883(883)	31.12(22.81)	3.81(-0.46)	5.66(5.66)	1.45	17.56	560.92
rt5	1102(1025)	869(814)	814(814)	744(744)	32.49(27.41)	14.38(8.60)	8.60(8.60)	5.25	56.45	865.41
average					26.48(22.91)	9.11(5.09)	6.47(6.47)	1.20	14.22	219.84

and the ML-OARSMT algorithm based on the computational geometry techniques [38] (namely CG). The experimental results are shown in Tables V and VI.

Table V lists the comparison between our algorithm and three ML-OARSMT algorithms on the total costs under the condition $C_v = 3$ and the condition $C_v = 5$. The total costs under $C_v = 5$ is displayed in parentheses next to the ones under $C_v = 3$. Columns 2–5 list the total costs of these algorithms, and the last three columns list the improvement over three ML-OARSMT algorithms on the total cost. From Table V, considering the total cost, the average improvements over CC, SG, and CG on the total cost are 39.18%, 9.41%, and 6.57% under $C_v = 3$, respectively. Furthermore, the improvement over CC can be up to 79.88% (for ind4). Since our proposed algorithm introduces the X-architecture and has a global view of multilayer structure, the proposed ML-OAXSMT-PSO algorithm has a stronger optimization ability on the total cost than these ML-OARSMT algorithms.

Table VI lists the comparison between our algorithm and three ML-OARSMT algorithms on the number of vias under $C_v = 3$, $C_v = 5$, and CPU time. The number of vias under $C_v = 5$ is displayed in parentheses next to the ones under $C_v = 3$. From Columns 6–8 of Table VI, the average improvements over CC, SG, and CG on the number of vias are 26.48%, 9.11%, and 6.47% under $C_v = 3$, respectively.

TABLE VII
COMPARISON BETWEEN OUR ALGORITHM AND TWO ML-OAXSMT ALGORITHMS ON THE TOTAL COST AND THE NUMBER OF VIAS UNDER $C_v = 3$

Test	X1	X2	OURS	Imp(TC)%	
	TC(VIA)	TC(VIA)	TC(VIA)	X1	X2
ind1	96780(4)	61392(4)	48996(43)	49.37	20.19
ind2	*	17569(5)	11652(188)	*	33.68
ind3	25569(9)	17700(9)	9654(326)	62.24	45.46
ind4	73588(0)	55154(0)	54965(0)	25.31	0.34
ind5	14625950(0)	14368566(0)	14356523(0)	1.84	0.08
rt1	8206(9)	6488(9)	3855(66)	53.03	40.58
rt2	*	17783(9)	8774(187)	*	50.66
rt3	*	32876(9)	14084(445)	*	57.16
rt4	54872(9)	34142(9)	21070(883)	61.60	38.29
rt5	59238(4)	52326(4)	27410(744)	42.15	47.62
average				42.22	33.41

Since the obstacles of ind4 and ind5 distribute in multiple layers but the pins are all in the same layer, these algorithms can optimize the number of vias as low as 0. Our algorithm achieves such significant improvements on the number of vias, since the proposed penalty mechanism has the certain optimization ability of the number of vias. From the last three columns of Table VI, the average times of CC, SG, OURS

TABLE VIII
COMPARISON AMONG THE FOUR PREFERRED DIRECTION CONSTRAINT MODELS FOR OAPD-XST (1)

test	$m/k/N_l$	a			b			c			d		
		VIA	VO	TC	VIA	VO	TC	VIA	VO	TC	VIA	VO	TC
ind3	250/13/10	262	315	22636	248	253	20209	243	196	18118	248	233	19568
rt1	25/10/10	22	33	10195	22	19	8694	23	11	6057	22	22	8865
rt2	100/20/10	131	140	21915	142	93	18369	130	76	16844	123	89	18527
rt3	250/50/10	309	351	37639	332	240	31736	315	216	29015	309	239	31892
rt4	500/50/10	733	712	52503	701	480	43000	713	428	38606	689	503	43198
average		291	310	28978	289	217	24402	285	185	21728	278	217	24410

TABLE IX
COMPARISON AMONG THE FOUR PREFERRED DIRECTION CONSTRAINT MODELS FOR OAPD-XST (2)

test	$m/k/N_l$	Imp:$(a-c)/a\%$			Imp:$(d-c)/d\%$			Imp:$(d-c)/d\%$		
		VIA	VO	TC	VIA	VO	TC	VIA	VO	TC
ind3	250/13/10	7.25	37.78	19.96	2.02	22.53	10.35	2.02	15.88	7.41
rt1	25/10/10	-4.55	66.67	40.59	-4.55	42.11	30.33	-4.55	50.00	31.68
rt2	100/20/10	0.76	45.71	23.14	8.45	18.28	8.30	-5.69	14.61	9.08
rt3	250/50/10	-1.94	38.46	22.91	5.12	10.00	8.57	-1.94	9.62	9.02
rt4	500/50/10	2.73	39.89	26.47	-1.71	10.83	10.22	-3.48	14.91	10.63
average		0.85	45.70	26.61	1.87	20.75	13.55	-2.73	21.00	13.56

are 1.20s, 14.22s, and 219.84s (the code file of CG has not been received). Although the CC algorithm is the fastest, its solution quality is relatively poor. The proposed algorithm has the high quality solution and is efficient. For ind4, our algorithm achieves 79.88% improvement over CC algorithm with about 3 min. Our algorithm takes some time to achieve the high quality routing solution. In the future work, we will seek and design an appropriate local search strategy to speed up our algorithm.

In Tables V and VI, under $C_v = 5$, the average improvements over CC, SG, and CG on the total cost are 38.31%, 9.36%, and 6.74%, respectively, while the average improvements over CC, SG, and CG on the number of vias are 22.91%, 5.09%, and 6.47%, respectively. Therefore, it can further verify that the proposed algorithm has the stronger optimization ability on the total cost than these ML-OARSMT algorithms and the effectiveness of our proposed penalty mechanism.

D. Comparison With Two ML-OAXSMT Algorithms

Table VII lists the comparison between our algorithm and two ML-OAXSMT algorithms on the total costs under the condition $C_v = 3$, where "*" represents that the effective solution can not be obtained using the corresponding algorithm. "TC" represents the total cost and 'VIA' represents the number of vias (from Tables VII–IX). The first ML-OAXSMT algorithm (namely X1) proposed by Lin *et al.* [40] separately constructed OAXSMT for each routing layer and then found the shortest connection path between every two abutting layers. The second ML-OAXSMT algorithm (namely X2) is designed according to the main idea of Lin *et al.* [40], but the OAXSMT construction method comes from our previous work [3] for SL-OAXSMT problem. From Table VII, the average improvements over X1 and X2 on the total costs are 42.22% and 33.41% under $C_v = 3$, respectively. The main reasons that our algorithm can achieve such significant improvements on the total cost can be summarized as follows.

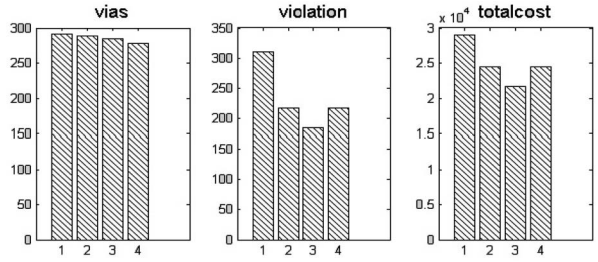


Fig. 11. Comparison among the four models for OAPD-XST on the number of vias, the violations, and the total cost.

Firstly, both X1 and X2 are designed to optimize the total costs of the ML-OAXSMT problem based on the GM-SLP method (Section IV-F). As a result, base on this idea for constructing ML-OAXSMT, it makes the two algorithms lack the ability to search the optimal solution and the two algorithms even might not search the effective solution. The shortest path between two pins for SL-OASMT can be constructed by considering only the pin vertexes and corners of obstacles while for the ML-OASMT problem it can not be constructed by considering only the pin vertexes and corners of obstacles in most cases. From Theorem 1, the improvement of our algorithm over the GM-SLP method (X1 and X2) on the total cost is close to 100% in some cases. This is one of the reasons that our algorithm can achieve such significant improvements on the total cost.

Secondly, the two algorithms based on the GM-SLP method focus on the optimization of the number of vias while the algorithm of [40] also optimizes the maximum path, and thus their optimization ability of the total cost is limited.

From Theorem 2, the improvement of our algorithm over the GM-SLP method (X1 and X2) on the total cost is close to 50% in most cases. This is another reason that our algorithm can achieve such significant improvements on the total cost.

In conclusion, both X1 and X2 are not really designed from a global perspective of multilayer structure to construct the ML-OAXSMT, but they are the ML-OAXSMT algorithm based on the GM-SLP method. Since our proposed algorithm constructs the ML-OAXSMT from a more global view of multilayer and obstacles and can search the better solutions with the better use of vias, the ML-OAXSMT-PSO algorithm can achieve such significant improvements on the total cost.

E. Comparison Among the Four Preferred Direction Constraint Models for OAPD-XST Problem

The number of vias, the violations and the total costs of the four preferred direction constraint models for the OAPD-XST problem are listed in Tables VIII and IX, respectively. Since the repeating cycle of Model *a* is 5 and others is 2, we choose the benchmark circuits with $N_l = 10$ as the test cases of the experiment in this subsection. As shown in Fig. 11, can find that the Model *c* is superior to the Model *a*, *b*, and *d* on both the violations (namely VO) and the total cost, while the number of vias obtained by all models is little different. In Fig. 11, the horizontal axis of 1, 2, 3, and 4 represents the Model *a*, *b*, *c*, and *d*, respectively. In Table IX, we select the superior Model *c* to compare with the Model *a*, *b*, and *d*. The number of vias of Model *c* is slightly less than both the Model *a* and *b* (0.85% and 1.87%), while is slightly more than Model *d* (2.73%). However, for the Model *c*, the improvement over the Model *a*, *b*, and *d* on the VO are 45.70%, 20.75%, and 21.00%, respectively. Moreover, the improvement over the Model *a*, *b* and *d* on the total costs are 26.61%, 13.55%, and 13.56%, respectively. We can find that the priority order of these models applied in the OAPD-XST problem is *c*, *b*, *d*, and *a*. Especially, the model *c* is superior to others while the routing quality between the Model *b* and *d* is slightly different.

VII. Conclusion

In this paper, we proposed an effective algorithm, which integrates the mutation operator and crossover operator of genetic algorithm into the ML-OAXSMT-PSO algorithm, and successfully constructed the ML-OAXSMT. Moreover, a useful pretreatment method, an effective penalty mechanism, and a simple edge-vertex encoding strategy are designed according to the feature of ML-OAXSMT problem. The main target is to optimize the total cost which is the most important index of ML-OAXSMT in the modern VLSI chip designs. Experiments on various benchmarks show that our algorithm achieves better total cost than the existing algorithms.

Due to OAPD-XST is more in line with the industrial routing process in X-architecture, on the basis of the ML-OAXSMT-PSO algorithm, we design an algorithm to construct the OAPD-XST to analyze the superior of different models. To our best knowledge, this is the first work addressing the OAPD-XST problem. Furthermore, this research can offer the theory supports for the chip design based on non-Manhattan architecture.

Acknowledgment

The authors would like to thank C.-W. Lin *et al.* of National Taiwan University, Taiwan and Y.-C. Lin *et al.* of Academia Sinica, Taiwan, for their valuable help with this paper. Wenzhong Guo is the corresponding author.

References

[1] A. Hashimotao and J. Stevens, "Wire routing by optimizing channel assignment with in large apertures," in *Proc. 8th ACM/IEEE Design Autom. Conf.*, 1988, pp. 35–49.

[2] S. L. Teig, "The X architecture: Not your father's diagonal wiring," in *Proc. ACM Int. Conf. Syst. Level Interconnect Prediction*, 2002, pp. 33–37.

[3] X. Huang, G. G. Liu, W. Z. Guo, and G. L. Chen, "Obstacle-avoiding octagonal Steiner tree construction based on particle swarm optimization," in *Proc. 9th Int. Conf. Natural Comput.*, Shenyang, China, Jul. 2013, pp. 539–543.

[4] Z. Feng *et al.*, "An O(nlogn) algorithm for obstacle-avoiding routing tree construction in the λ-geometry plane," in *Proc. Int. Symp. Phys. Des.*, San Jose, CA, USA, 2006, pp. 48–55.

[5] T. Jing *et al.*, "λ-OAT: λ-geometry obstacle-avoiding tree construction with O(nlogn) complexity," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, no. 11, pp. 2073–2079, Nov. 2007.

[6] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micro Mach. Human Sci.*, Nagoya, Japan, Oct. 1995, pp. 39–43.

[7] L. L. Liu, S. X. Yang, and Q. Wang, "Particle swarm optimization with composite particles in dynamic environments," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 40, no. 6, pp. 1373–1392, Dec. 2010.

[8] G. L. Chen, W. Z. Guo, and Y. Z. Chen, "A PSO-based intelligent decision algorithm for VLSI floor planning," *Soft Comput.*, vol. 14, no. 12, pp. 1329–1337, 2009.

[9] H. H. Chou, L. Y. Hsu, and H. T. Hu, "Turbulent-PSO-based fuzzy image filter with no-reference measures for high-density impulse noise," *IEEE Trans. Cybern.*, vol. 43, no. 1, pp. 296–307, Feb. 2013.

[10] B. Y. Qu, J. J. Liang, and P. N. Suganthan, "Niching particle swarm optimization with local search for multi-modal optimization," *Inf. Sci.*, vol. 197, pp. 131–143, Aug. 2012.

[11] P. P. San, S. H. Ling, S. L. Nuryani, and H. T. Nguyen, "Evolvable rough-block-based neural network and its biomedical application to hypoglycemia detection system," *IEEE Trans. Cybern.*, vol. 44, no. 8, pp. 1338–1349, Aug. 2014.

[12] M. Nasir *et al.*, "A dynamic neighborhood learning based particle swarm optimizer for global numerical optimization," *Inf. Sci.*, vol. 209, pp. 16–36, Nov. 2012.

[13] B. Xue, M. J. Zhang, and M. N. Browne, "Particle swarm optimization for feature selection in classification: A multi-objective approach," *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 1656–1671, Dec. 2013.

[14] R. Ruiz-Cruz, E. N. Sanchez, F. Ornelas-Tellez, A. G. Loukianov, and R. G. Harley, "Particle swarm optimization for discrete-time inverse optimal control of a doubly fed induction generator," *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 1698–1709, Dec. 2013.

[15] T. Arora and M. E. Mose, "Ant colony optimization for power efficient routing in manhattan and non-manhattan VLSI architectures," in *Proc. Swarm Intell. Symp.*, Nashville, TN, USA, Mar./Apr. 2009, pp. 137–144.

[16] Z. Bingul, "Adaptive genetic algorithms applied to dynamic multiobjective problems," *Appl. Soft Comput.*, vol. 7, no. 3, pp. 791–799, 2007.

[17] X. L. Hong *et al.*, "Non-rectilinear on-chip interconnects: An efficient routing solution with high performance," *Chin. J. Semicond.*, vol. 24, no. 3, pp. 225–233, 2003.

[18] C. S. Coulston, "Constructing exact octagonal Steiner minimal tree," in *Proc. Great Lakes Symp. VLSI*, Washington, DC, USA, 2003, pp. 28–29.

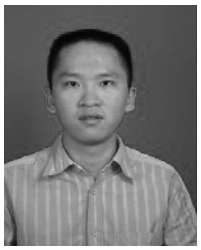
[19] Q. Zhu, H. Zhou, T. Jing, X. L. Hong, and Y. Yang, "Spanning graph-based nonrectilinear Steiner tree algorithms," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 24, no. 7, pp. 1066–1075, Jul. 2005.

[20] M. R. Garey and D. S. Johnson, "The rectilinear Steiner tree problem is NP-complete," *SIAM J. Appl. Math.*, vol. 32, no. 4, pp. 826–834, Jun. 1977.

[21] G. G. Liu, G. L. Chen, and W. Z. Guo, "DPSO based octagonal Steiner tree algorithm for VLSI routing," in *Proc. 5th Int. Conf. Adv. Comput. Intell.*, Nanjing, China, 2012, pp. 383–387.

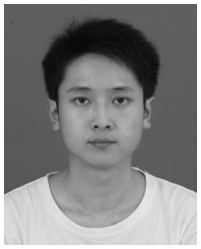
[22] J. T. Yan, "Timing-driven octilinear Steiner tree construction based on Steiner-point reassignment and path reconstruction," *ACM Trans. Design Autom. Electron. Syst.*, vol. 13, no. 2, p. 26, Apr. 2008.

[23] T. Samanta, H. Rahaman, and P. Dasgupta, "Near-optimal Y-routed delay trees in nanometric interconnect design," *IET Comput. Digit. Techn.*, vol. 5, no. 1, pp. 36–48, Jan. 2011.
[24] Y. Yang, Q. Zhu, T. Jing, X. Hong, and Y. Wang, "Rectilinear Steiner minimal tree among obstacles," in *Proc. IEEE ASICCON*, 2003, pp. 348–351.
[25] Y. Hu *et al.*, "An-OARSMan: Obstacle-avoiding routing tree construction with good length performance," in *Proc. Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2005, pp. 7–12.
[26] Z. Zhou, C. D. Jiang, L. S. Huang, and J. Gu, "On optimal rectilinear shortest paths and 3-Steiner tree routing in presence of obstacles," *J. Softw.*, vol. 14, no. 9, pp. 1503–1514, 2003.
[27] G. Ajwani, C. Chu, and W. K. Mak, "FOARS: FLUTE based obstacle-avoiding rectilinear Steiner tree construction," in *Proc. Int. Symp. Phys. Des.*, 2010, pp. 27–34.
[28] C. H. Liu *et al.*, "Obstacle-avoiding rectilinear Steiner tree construction: A Steiner-point-based algorithm," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, no. 7, pp. 1050–1060, Jul. 2012.
[29] W. K. Chow, L. Li, E. F. Young, and C. W. Sham, "Obstacle-avoiding rectilinear Steiner tree construction in sequential and parallel approach," *Integr. VLSI J.*, vol. 47, no. 1, pp. 105–114, 2013.
[30] T. Huang, L. Li, and E. F. Young, "On the construction of optimal obstacle-avoiding rectilinear Steiner minimum trees," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 30, no. 5, pp. 718–731, May 2011.
[31] T. Huang and E. F. Young, "ObSteiner: An exact algorithm for the construction of rectilinear Steiner minimum trees in the presence of complex rectilinear obstacles," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 6, pp. 882–893, Jun. 2013.
[32] C. C. Tong and C. L. Wu, "Routing in a three-dimensional chip," *IEEE Trans. Comput.*, vol. 44, no. 1, pp. 106–117, Jan. 1995.
[33] C. W. Lin, S. L. Huang, K. C. Hsu, M. X. Lee, and Y. W. Chang, "Multilayer obstacle-avoiding rectilinear Steiner tree construction based on spanning graphs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 11, pp. 2007–2016, Nov. 2008.
[34] H. Totsukawa, H. Senou, and M. Ohmura, "A parallel genetic algorithm for 3-D rectilinear Steiner tree with bounded number of bends," in *Proc. 51st Midwest Symp. Circuits Syst. (MWSCAS)*, Knoxville, TN, USA, 2005, pp. 7–12.
[35] T. Yan and Z. W. Chen, "Timing-driven multi-layer Steiner tree construction with obstacle avoidance," in *Proc. Asia Pacific Conf. Circuits Syst. (APCCAS)*, 2008, pp. 1684–1687.
[36] M. C. Yildiz and P. H. Madden, "Preferred direction Steiner trees," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 21, no. 11, pp. 1368–1372, Nov. 2002.
[37] C. H. Liu, Y. H. Chou, S. Y. Yuan, and S. Y. Kuo, "Efficient multilayer routing based on obstacle-avoiding preferred direction Steiner tree," in *Proc. Int. Symp. Phys. Design (ISPD)*, Yokohama, Japan, 2008, pp. 118–125.
[38] C. H. Liu, I. C. Chen, and D. T. Lee, "An efficient algorithm for multilayer obstacle-avoiding rectilinear Steiner tree construction," in *Proc. Design Automation Conf.*, San Francisco, CA, USA, 2012, pp. 613–622.
[39] J. R. Chuang, and J. M. Lin, "Efficient multi-layer obstacle-avoiding preferred direction rectilinear Steiner tree construction," in *Proc. 16th Asia South Pacific Design Autom. Conf.*, Yokohama, Japan, 2011, pp. 527–532.
[40] Y. C. Lin, H. A. Chien, C. C. Shih, and H. M. Chen, "A multi-layer obstacles-avoiding router using X-architecture," *WSEAS Trans. Circuits Syst.*, vol. 7, no. 8, pp. 879–888, 2008.
[41] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Proc. Int. Conf. Syst. Man, Cybern.*, Orlando, FL, USA, 1997, pp. 4104–4108.
[42] X. Hu, R. C. Eberhart, and Y. Shi, "Swarm intelligence for permutation optimization: A case study of n-queens problem," in *Proc. Swarm Intelligence Symp.*, 2003, pp. 243–246.
[43] K. E. Parsopoulos and M. N. Halgamuge, "Recent approaches to global optimization problems through particle swarm optimization," *Natural Comput.*, vol. 1, nos. 2–3, pp. 235–306, 2002.
[44] A. Salman, I. Ahmad, and S. Al-Madani, "Particle swarm optimization for task assignment problem," *Microprocess. Microsyst.*, vol. 26, no. 8, pp. 363–371, 2002.
[45] Q. K. Pan, M. F. Tasgetiren, and Y. C. Liang, "A discrete particle swarm optimization algorithm for the permutation flowshop sequencing problem with makespan criteria," in *Research and Development in Intelligent Systems XXIII*. London, U.K.: Springer, 2006, pp. 19–31.
[46] W. Z. Guo, G. G. Liu, G. L. Chen, and S. J. Peng, "A hybrid multi-objective PSO algorithm with local search strategy for VLSI partitioning," *Front. Comput. Sci.*, vol. 8, no. 2, pp. 203–216, 2014.
[47] B. A. Julstrom, "Encoding rectilinear Steiner trees as lists of edges," in *Proc. ACM Symp. Appl. Comput.*, 2001, pp. 356–360.
[48] H. Lv, J. Zheng, J. Wu, C. Zhou, and K. Li, "The convergence analysis of genetic algorithm based on space mating," in *Proc. 5th Int. Conf. Natural Comput.*, 2009, pp. 557–562.
[49] G. Rudolph, "Convergence analysis of canonical genetic algorithms," *IEEE Trans. Neural Netw.*, vol. 5, no. 1, pp. 96–101, Jan. 1995.
[50] J. E. Beasley, "OR-Library: Distributing test problems by electronic mail," *J. Oper. Res. Soc.*, vol. 41, no. 11, pp. 1069–1072, 1990.



Genggeng Liu is currently pursuing the Doctorate degree from the College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China.

His current research interests include computational intelligence and very large scale integration physical design.



Xing Huang is currently pursuing the Doctorate degree from the College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China.

His current research interests include very large scale integration routing.

Wenzhong Guo received the B.S. and M.S. degrees in computer science and the Ph.D. degree in communication and information system from Fuzhou University, Fuzhou, China, in 2000, 2003, and 2010, respectively.

He is currently a Full Professor with the College of Mathematics and Computer Science, Fuzhou University. His current research interests include mobile computing and evolutionary computation. He currently leads the Network Computing and Intelligent Information Processing Laboratory, which is a key laboratory of Fujian Province, China.

Yuzhen Niu received the B.S. and Ph.D. degrees in computer science from Shandong University, Jinan, China, in 2005 and 2010, respectively.

She was a Post-Doctoral Researcher with the Department of Computer Science, Portland State University, Portland, OR, USA. She is currently a Professor with the College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China. Her current research interests include computer graphics, vision, and multimedia.

Guolong Chen received the B.S. and M.S. degrees in computational mathematics from Fuzhou University, Fuzhou, China, in 1987 and 1992, respectively, and the Ph.D. degree in computer science from Xi'an Jiaotong University, Xi'an, China, in 2002.

He is a Full Professor with the College of Mathematics and Computer Science at Fuzhou University. His current research interests include computation intelligence, computer networks, and information security.