IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received January 14, 2020, accepted January 31, 2020, date of publication February 4, 2020, date of current version February 11, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2971574

A Survey of Swarm Intelligence Techniques in VLSI Routing Problems

XIAOHUA CHEN[1], GENGGENG LIU[1], (Member, IEEE), NAIXUE XIONG[2], (Senior Member, IEEE), YARU SU[1], AND GUOLONG CHEN[1]

[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 Routing is a complex and critical stage in the physical design of Very Large Scale Integration (VLSI), minimizing interconnect length and delay to optimize overall chip performance. With the rapid development of modern technology, VLSI routing faces enormous challenges such as large delay, high congestion, and high-power consumption. As a rising optimization method, Swarm Intelligence (SI) inspired from collective intelligence behaviors through cooperation or interaction with the environment provides effectiveness and robustness for solving NP-hard problems. Many researchers have consequently used SI techniques to solve routing-related problems in VLSI. This paper reviews the application of several SI techniques to the VLSI routing filed. Firstly, five commonly used SI techniques and related models, and three classic routing problems are described: *Steiner tree construction*, *global routing* and *detailed routing*. Then an overview of the current state of this field is given according to the above categories, and the survey offers informative discussions from five aspects: 1) Steiner minimum tree construction; 2) wirelength-driven routing; 3) obstacle-avoiding routing; 4) timing-driven routing; 5) power-driven routing. Finally, under three new technology models: *X-architecture*, *multiple dynamic supply voltage* and *via-pillar*, the future development trends are pointed as follows: 1) suggesting suitable SI techniques to specific routing problems for advanced technology models; 2) exploring new and available SI techniques that have not yet been applied to VLSI routing.

INDEX TERMS Particle swarm optimization, swarm intelligence, routing, very large scale integration, Steiner tree construction.

LIST OF ABBREVIATION

ABC	Artificial Bee Colony
ACO	Ant Colony Optimization
BPSO	Binary Particle Swarm Optimization
CCPSO	Cooperatively Coevolving Particle Swarm Optimization
CLPSO	Comprehensive Learning Particle Swarm Optimization
CMOS	Complementary Metal-Oxide-Semiconductor
CPSO	Cooperative Particle Swarm Optimization
DDE	Discrete Differential Evolution
DE	Differential Evolution
DPSO	Discrete Particle Swarm Optimization
EDA	Electronics Design Automation
FA	Firefly Algorithm
GA	Genetic Algorithm
GRC	Global Routing Cell
GRG	Global Routing Graph
HST	Hexagonal Steiner Tree
HTS	Hybrid Transformation Strategy
IC	Integrated Circuit
ISPD	International Symposium on Physical Design
ILP	Integer Linear Programming
MDSV	Multiple Dynamic Supply Voltage
MST	Minimal Spanning Tree
MSV	Multiple Supply Voltage

The associate editor coordinating the review of this manuscript and approving it for publication was Sudipta Roy.

OAOSMT	Obstacle-Avoiding Octagonal Steiner Minimum Tree
OARSMT	Obstacle-Avoiding Rectilinear Steiner Minimum Tree
OST	Octagonal Steiner Tree
OSMT	Octagonal Steiner Minimum Tree
PSO	Particle Swarm Optimization
QPSO	Quantum Particle Swarm Optimization
RST	Rectilinear Steiner Tree
RSMT	Rectilinear Steiner Minimum Tree
SI	Swarm Intelligence
SMT	Steiner Minimum Tree
SPSO	Standard Particle Swarm Optimization
TSV	Through Silicon Vias
UFS	Union-Find Sets
VLSI	Very Large Scale Integration

I. INTRODUCTION

Very Large Scale Integration (VLSI) is a process of forming an Integrated Circuit (IC) by incorporating thousands 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. As the pillar of information industry, the design and manufacturing of VLSI are playing an increasingly important role in driving economic development, deepening industrial structure and lifestyle changes.

Physical design is the most time-consuming step in the design process of IC, and it is also one of the most important and active research field in VLSI computer-aided design technology. Due to its complexity, the whole process of physical design is often divided into *partition*, *floorplanning*, *placement* and *routing*. Routing plays a key role in VLSI physical design as it determines the specific shape and layout of interconnect, impacting performance, power and manufacturability, which is traditionally divided into the two steps of global and detailed routing. Global routing needs to connect all nets with all capacity constraints satisfied. While detailed routing takes a global routing solution with a small number of capacity violations (*overflows*), or none at all. And under the premise of ensuring spacing constraints and more sophisticated design rules, detailed routing completes the assignment of wires within the routing area.

With the rapid development of IC, these characteristics are presented: the feature size is getting smaller and smaller, the chip area is getting larger and larger, the power supply voltage is getting lower and lower, and the number of routing metal layers is also increasing. Traditional routing algorithms, which operate in isolated layout regions-channels or switch-boxes, are usually assumed to be routed on the premise of a small number of metal layers, making it difficult to accommodate more metal layers. Gradually, over-the-cell routing with six or multi-layer routing lead to the adoption of similar graph-theoretical techniques in global and detailed routing, perhaps with different layouts, resource and delay models [2]. The routing algorithms require a better balance between wirelength minimization and congestion. In general, congestion is optimized by considering constraints such as obstacles, number of vias, and capacitance. However, the continuous reduction of process size and the increase of integration complexity make traditional routing algorithms unable to perform such multi-objective tasks well, and the design of routing algorithms faces new challenges.

Swarm Intelligence (SI) is an important category of optimization technique, which is inspired from simple behaviors and self-organizing interaction among agents like ant colonies foraging, bird flocking, animal herding, bacterial growth, honey bees, fish schooling, and so on [3]. Each SI algorithm has its own unique advantages. The specific behaviors of these animals in solving problems have different help for searching optimal solutions, so the performance of each SI algorithm is different for different problems. But these SI algorithms have one thing in common, that is, each of them has several agents working simultaneously. And experience learning or information sharing or competition among individuals can enable swarm to grow rapidly, improving search efficiency and precision. Therefore, many researchers have started applying various SI algorithms with great promise because of their high computational efficiency, reliability, scalability, self-organizing, longevity and low-cost [4]. In recent years, SI techniques have been widely used on many practical real-world problems in the fields of scheduling [5], production [6], facility control [7], [8], transportation [9] and so on.

The survey has the following four main objectives. First, we give an introduction about key concepts in VLSI routing, such as Minimum Spanning Tree (MST), Steiner tree and related routing issues including global and detailed routing. Second, the applications of some SI techniques in routing problems are listed to provide an overview of the current state of this hot field. Thirdly, we investigate and research some advanced processes, and give a discussion about the construction of routing models and evaluation models that are suitable for such processes. The final important goal is to identify future trends and research directions to better guide subsequent research efforts. This study discussed the contributions which were published in the relevant journals, conference proceedings and theses.

The rest of the paper is organized as follows. The core definitions and related concepts of SI techniques and VLSI routing issues are respectively presented in Sections II and III, so that they are easily linked to the rest of the research in this paper. Then, Section IV focuses on the investigated routing problems using the five SI algorithms to present a current state of the art. Section V provides a discussion of related issues, including multi-layer routing based on X-architecture, global routing in a Multiple Dynamic Supply Voltage (MDSV) chip design environment, and routing problems under the via-pillar process. Section VI clarifies the future trends and opportunities under each type of problems, while Section VII is to make a conclusion for this paper.

TABLE 1. Overview of different types of SI technique.

Type of SI Technique	Proposal	Inspiration	Parameter Number	Parameters	Computational Complexity
ACO	Dorigo (1992) [11]	Ant colonies forage for food using pheromone as a mean of chemical messenger	5	Population size, M	High
				Number of iterations, t	
				Pheromone factor, α	
				Heuristic factor, β	
				Evaporation rate, ρ	
PSO	Kennedy and Eberhart (1995) [12]	Social behavior of bird flocking	7	Population size, M	Low
				Number of iterations, t	
				Inertia weight, ω	
				Acceleration coefficient, c_1, c_2	
				Random numbers with uniform distribution, r_1, r_2	
DE	Storn and Price (1997) [13]	Genetic evolution of biological populations	7	Population size, NP	Low
				Number of iterations, t	
				Scaling factor, F	
				Crossover rate, CR	
				Mutually exclusive integers randomly generated within the range [1, NP], r_1, r_2, r_3	
ABC	Karaboga (2005) [14]	Natural foraging behavior of real honey bees	5	Population size, NP	Medium
				Number of iterations, t	
				The number of employed foragers, SN	
				The threshold that a employed bee role is transformed into a scout bee, l	
				A defined number of iterations, c	
FA	Yang (2008) [15]	Behavior of fireflies and their flashing light	6	Population size, M	High
				Number of iterations, t	
				Attractiveness, β_0	
				Randomization parameter, α	
				Absorption coefficient, γ	
				Random numbers with Gaussian distribution or uniform distribution, ε	

II. SWARM INTELLIGENCE TECHNIQUES

A system in which unintelligent entities exhibit collective intelligence behavior through cooperation or interaction with the environment is called *Swarm Intelligence*, having the characteristics of natural distribution and self-organizing characteristics [10]. It can show obvious advantages without the premise of centralized control and providing a global model. Inspired by nature and biology, SI techniques obtain collective intelligence behaviors through two fundamental concepts, *self-organization* and *division of labor*, and are widely used in optimization problems. Many heuristic optimization algorithms are developed on the basis of simulating the behavior of different biological populations.

This section introduces the basic principles and mathematical models of five SI algorithms commonly used in VLSI routing problems: Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Differential Evolution (DE), Artificial Bee Colony (ABC), Firefly Algorithm (FA). Table 1 outlines several important characteristics of these techniques.

A. ACO

1) INSPIRATION AND BASIC IDEAS OF ACO

In the 1990's, ACO was introduced as a nature-inspired method for the solution of various combinatorial optimization problems [11]. The ACO algorithm is inspired by the feeding behavior of real ant colonies in nature. When looking for food, the ants will randomly explore the area near the nest. If an ant finds a source of food, it will evaluate the food and brings some food back to the nest, leaving a pheromone along the way to guide other ants to find the food source. And the concentration of pheromones may depend on the quantity and quality of the food. Then the pheromone will gradually evaporate. If two ants find the same food at the same time and take different routes to return to the nest, the pheromone smell on the more complicated path will be lighter, and the ant colony will tend to another route closer to find food. Thus, the probability of selecting the path of the ant from the nest is proportional to the concentration of the pheromone on each path, and each passing ant will leave a pheromone on the way to achieve communication between the individuals, forming a positive feedback phenomenon [16]. The diversity of ant colonies and the characteristics of positive feedback make the ACO algorithm both innovative and learning.

2) MATHEMATICAL REPRESENTATION OF ACO

The feasible solution of the problem to be optimized is represented by the ant's walking path, and all the paths of the whole ant colony constitute the solution space of the problem. Artificial ants construct solutions from sequences of solution components taken from a finite set of m available solution components $C = \{c_{ij}\}$. And a solution construction starts with an empty partial solution $s^P = \emptyset$. Then, at each step, the current partial solution s^P is extended by adding a feasible solution component from the set $N(s^P) \in C \backslash s^P$, which is done probabilistically as represented in Eq. (1) [16].

$$p(c_{ij} \left| s^P \right.) = \frac{\tau_{ij}^{\alpha} \cdot \eta(c_{ij})^{\beta}}{\sum_{c_{il} \in N(s^P)} \tau_{il}^{\alpha} \cdot \eta(c_{il})^{\beta}}, \quad \forall c_{ij} \in N(s^P) \tag{1}$$

where τ_{ij}^{α} is the pheromone value associated with component c_{ij}, and $\eta(\cdot)$ is a weighting function that assigns a heuristic value to each feasible solution component $c_{ij} \in N(s^P)$ at each step. α and β are positive parameters determining the relation between pheromone information and heuristic information.

The pheromone in Eq. (1) through the following pheromone evaporation process to increase the pheromone values associated with good or promising solutions, and decrease those that are associated with bad ones [17].

$$\tau_{ij} \leftarrow \begin{cases} (1-\rho)\tau_{ij} + \rho\Delta\tau, & if\ \tau_{ij} \in s_{ch} \\ (1-\rho)\tau_{ij}, & otherwise \end{cases} \tag{2}$$

where $\rho \in (0, 1]$ is the evaporation rate. Different versions of ACO algorithms update pheromones in different ways.

B. PSO

1) INSPIRATION AND BASIC IDEAS OF PSO

PSO was first proposed by Kennedy and Eberhart in 1995 [12]. Its main idea is derived from the study of birds clustering behavior, using the characteristics of bird population attracted by habitat to guide human decision-making process. At the beginning, the flock flies in the air in no specific direction until one bird finds its habitat. Trapped by the habitat, other companions will be affected by neighboring partners and habitats, and gradually fly to the habitat. In this process, two kinds of crucial information will be fully utilized, namely the experience from the bird itself and its neighboring partners. In the PSO algorithm, each solution of the optimization problem, called ''particle'', is regarded as a bird in the search space. All particles correspond to all possible solutions in the solution space. Each particle has no weight and volume, and its fitness value is determined by the objective function. The speed of a particle determines the direction and distance of its flight. And the particle completes the search in the solution space by learning form its own experience and the optimal particle in the group. In recent years, many scholars have carried out related research on PSO due to its simple calculation, easy implementation and low control parameters.

2) MATHEMATICAL REPRESENTATION OF PSO

In the PSO algorithm, particles dynamically adjust their position information by learning their best individual position and the global best position in the swarm. Consider a minimization problem with D-dimensional search space, assuming that the population size of is M, and the speed and position update formulas of the Standard Particle Swarm Optimization (SPSO) [18] are as follows:

$$V_{ij}^{t+1} = \omega \cdot V_{ij}^{t} + c_1 \cdot r_1 \cdot (P_{ij}^{t} - X_{ij}^{t}) + c_2 \cdot r_2 \cdot (G_j^{t} - X_{ij}^{t}) \tag{3}$$

$$X_{ij}^{t+1} = V_{ij}^{t+1} + X_{ij}^{t} \tag{4}$$

where $1 \leq i \leq M, 1 \leq j \leq D$. In Eq. (3), ω is the inertia weight to better balance the exploration and exploitation of the algorithm. c_1 and c_2 are acceleration coefficients, which respectively adjust the step size of the particle flying to its personal best position and the global best position. r_1 and r_2 are mutually independent random numbers uniformly distributed in the interval (0,1). P_{ij}^{t} and G_j^{t} are the best position of the particle i and the global optimal position of the population respectively, satisfying the following formula:

$$P_i^t = \begin{cases} X_i^t, & if\ f(X_i^t) < f(P_i^t) \\ P_i^{t-1}, & if\ f(X_i^t) \geq f(P_i^t) \end{cases} \tag{5}$$

$$G^t = P_g^t, \quad g = \arg \min_{1 \leq i \leq M} [f(P_i^t)] \tag{6}$$

In 1999, Clerc and Kennedy introduced constriction coefficient [19] in the evolution equation, relaxed the speed limit, and thus improved the convergence speed of the algorithm. The speed update formula is as follows:

$$V_{ij}^{t+1} = \chi \cdot (V_{ij}^{t} + \varphi_1 r_1 (P_{ij}^{t} - X_{ij}^{t}) + \varphi_2 r_2 (G_j^{t} - X_{ij}^{t})) \tag{7}$$

where φ_1 and φ_2 are acceleration coefficients, and χ is constriction coefficient satisfying:

$$\chi = \frac{2}{[2 - \varphi - \sqrt{\varphi^2 - 4\varphi}]}, \quad \varphi = \varphi_1 + \varphi_2, \ \varphi > 4$$

However, some dimensions that are close to the optimal solution are likely to be far from the optimal solution although the particle position is improved after updating. Therefore, a kind of Cooperative Particle Swarm Optimization (CPSO) [20] is proposed to solve this problem. CPSO divides the dimensions of particles into several groups and each group is optimized with a single particle swarm. Then calculate the fitness value after combining the dimensions. Finally, the corresponding update is performed according to the update rules of the SPSO algorithm.

In order to solve the problem that PSO is easily trapped into local optimal when dealing with complex multi-peak problems, Liang *et al.* [21] proposed Comprehensive Learning PSO (CLPSO), in which particles have more learning objects and thus have more potential flight space.

C. DE

1) INSPIRATION AND BASIC IDEAS OF DE

The DE was proposed by Storn and Price on the basis of evolutionary ideas such as Genetic Algorithm (GA) in 1997 [13]. The essence is a multi-objective (continuous variables) evolutionary algorithm. And it is used for solving the overall optimal solution in multidimensional space. In community of DE, the individual trial solutions which constitute a population are called parameter vectors or genomes [22]. DE generates a population of individuals by encoding with a floating vector, and the optimization process includes mutation, hybridization, and selection operations. The basic idea is described as follows: Starting from a randomly generated initial population, a new individual is generated by summing the vector difference of any two individuals in the population with a third individual, and the fitness value is used to determine whether to retain the old individual. Through continuous evolution, retaining good individuals, and eliminating inferior individuals, DE guides search to the optimal solution.

2) MATHEMATICAL REPRESENTATION OF DE

The DE algorithm involves three control parameters, namely population size NP, scaling factor F, and crossover rate CR. The optimization process includes the following three operations.

a: MUTATION OPERATION

After initializing the population, DE employs the mutation operation to produce a mutant vector. Reference [23] gives five commonly used variation strategies as follows:

$$V_i^t = X_{r_1}^t + F \cdot (X_{r_2}^t - X_{r_3}^t) \tag{8}$$

$$V_i^t = X_{best}^t + F \cdot (X_{r_1}^t - X_{r_2}^t) \tag{9}$$

$$V_i^t = X_i^t + F \cdot (X_{best}^t - X_i^t) + F \cdot (X_{r_1}^t - X_{r_2}^t) \tag{10}$$

$$V_i^t = X_{best}^t + F \cdot (X_{r_1}^t - X_{r_2}^t) + F \cdot (X_{r_3}^t - X_{r_4}^t) \tag{11}$$

$$V_i^t = X_{r_1}^t + F \cdot (X_{r_2}^t - X_{r_3}^t) + F \cdot (X_{r_4}^t - X_{r_5}^t) \tag{12}$$

where $V_i^t = \{v_{i,1}^t, v_{i,2}^t, \ldots, v_{i,D}^t\}$ is mutant vector. The indices r_1, r_2, r_3, r_4, r_5 are mutually exclusive integers randomly generated within the range $[1, NP]$, and these indices are randomly generated once for each mutant vector. The scaling factor F is a positive control parameter for scaling the difference vector. X_{best}^t is the best individual vector with the best fitness value in the population at generation t.

b: CROSSOVER OPERATION

Crossover operation is applied to each pair of the target vector X_i^t and mutant vector V_i^t to generate a trial vector: $U_i^t = \{u_{i1}^t, u_{i2}^t, \ldots, u_{iD}^t\}$. Crossover operation is defined as follows:

$$u_{ij}^t = \begin{cases} v_{ij}^t, & \text{if } rand_j[0,1) \le CR \text{ or } j = j_{rand} \\ x_{ij}^t, & \text{otherwise} \end{cases} \tag{13}$$

where the crossover rate CR is a user-specified constant within the range [0,1), and controls the fraction of parameter values copied from the mutant vector.

c: SELECTION OPERATION

Selection operation typically screens individuals based on fitness values. The selection operation can be expressed as follows:

$$X_i^{t+1} = \begin{cases} U_i^t, & \text{if } f(U_i^t) \le f(X_i^t) \\ X_i^t, & \text{otherwise} \end{cases} \tag{14}$$

D. ABC

1) INSPIRATION AND BASIC IDEAS OF ABC

The ABC algorithm was proposed by Karaboga in 2005 [14]. The basic idea is that the bee colony cooperates with each other to complete the process of collecting honey through individual division of labor and information exchange. The minimal search model for bee colony to achieve collective intelligence includes three essential components: *food sources*, *employed bees* and *unemployed bees*, and two behavior patterns: the recruitment to a rich nectar source and the abandonment of a poor source [24]. The employed bees correspond to the collected food sources, store information about a certain food source, and share the information with other bees with a certain probability. The unemployed bees are mainly responsible for finding and mining food sources, and there are two types of unemployed bees, respectively: *scouts*, searching the environment surrounding the nest for new food sources, and *onlookers*, waiting in the nest and establishing a food source through the information shared by employed bees. In the beginning, all food sources are discovered by scout bees. Thereafter, the mining of nectar is completed by employed bees and onlooker bees. The sharing of food sources information is realized in the form of a swing dance in the dance area. After finding the food sources and collect their nectar, employed bees return to the nest and express the yield rate of food source through the duration of swing dance. That is, the yield rate is proportional to the likelihood that the food source is selected. Onlooker bees are employed after watching the swing dance and begin searching for the corresponding food source neighborhood and collecting honey. While the employed bee whose food source has been exhausted becomes a scout bee. ABC works through the transformation of three different types of roles to find high-quality honey sources.

2) MATHEMATICAL REPRESENTATION OF ABC

In ABC, the location of food sources represents the possible solutions of the problem, and the amount of nectar in a food source corresponds to the quality (fitness) of the relevant solution. In the basic form, the number of employed bees is equal to the number of food sources (solutions).

Assume that the dimension of the problem is D, the number of food sources is SN, and the position of honey source i is expressed as $X_i^t = [x_{i1}^t, x_{i2}^t, ..., x_{iD}^t]$. The initial position of honey source i is randomly generated in the search space according to Eq. (15):

$$x_{ij} = L_j + rand(0, 1) \cdot (U_j - L_j) \tag{15}$$

where L and U represent the lower limit and upper limit of the search space, respectively.

At the beginning of the search, ABC algorithm utilizes the following expression to generate a new honey source location $V_i^t = [v_{i1}^t, v_{i2}^t, ..., v_{iD}^t]$ around the honey source i:

$$v_{ij} = x_{ij} + \phi(x_{ij} - x_{kj}) \tag{16}$$

where $k \in \{1, 2, \ldots, SN\}$, and not equal to i. This parameter indicates that the employed bee randomly selects a nectar source other than i from SN nectar sources. ϕ is a random number between [-1,1], which decides the magnitude of the perturbation. When the fitness of the new honey source V_i is better than X_i, the greedy selection is adopted to replace X_i with V_i, otherwise keep X_i. Eq. (16) indicates that when the search is close to the optimal solution, the step size will also be adaptively reduced.

After all the employed bees complete the operation of Eq. (16), they fly back to the information exchange area to share the honey source. And an onlooker bee chooses a food source depending on the probability value p_i, which is associated with food source. And it is calculated by the following expression:

$$p_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n} \tag{17}$$

During the search process, if the honey source X_i has reached a threshold l by a defined number of iterations c and does not find a better food source, the honey source will be abandoned, and the corresponding employed bee will be converted into a scout bee. The scout bee will randomly generate a new food source in the search space to instead X_i. The above process is described as Eq. (18):

$$X_i^{t+1} = \begin{cases} L + rand(0, 1) \cdot (U - L), & c_i \geq l \\ X_i^t, & c_i < l \end{cases} \tag{18}$$

E. FA

1) INSPIRATION AND BASIC IDEAS OF FA

FA is a simulation of the biological characteristics of firefly luminescence in nature. It was proposed by Yang to effectively deal with multi-modal and global optimization problems [15]. The algorithm simulates the search and optimization processes into the attraction and movement of the firefly individuals, and measures the objective function of solving the problem as the position of the individual. In this algorithm, the reason why fireflies attract each other depends on two factors: their own *brightness* and *attractiveness*. The fluorescence brightness emitted by fireflies depends on the target value of their location. The higher the brightness, the better the position, that is the better the target value. Attractiveness is proportional to their brightness and they both decrease as their distance increases. For any two flashing fireflies, the less bright one will move towards the brighter one. If no one is brighter than a particular firefly, it will move randomly.

2) MATHEMATICAL REPRESENTATION OF FA

The FA achieves target optimization through continuous updating of brightness and attractiveness. And the brightness of fireflies with distance r is expressed as follows:

$$I = I_0 e^{-\gamma r} \tag{19}$$

where I_0 is the original light intensity and γ is the light absorption coefficient.

The attractiveness of a firefly is expressed as below:

$$\beta = \beta_0 e^{-\gamma r^2} \tag{20}$$

where β_0 is the attractiveness at $r = 0$.

Firefly i is attracted by firefly i' and the position update formula is as follows:

$$X_i^{t+1} = X_i^t + \beta(X_{i'}^t - X_i^t) + \alpha^t \varepsilon_i^t \tag{21}$$

where the third term is randomization with α^t being the randomization parameter, and ε_i^t is a vector of random numbers drawn from a Gaussian distribution or uniform distribution at time t. Furthermore, the randomization ε_i^t can easily be extended to other distributions such as Lévy flights [15].

III. ROUTING PROBLEMS IN VLSI

As the current IC industry continues to advance toward ultra-deep sub-micron processes, the integration of chips is further improved. Since more and more circuit components can be integrated on a single chip with the limitation of storage space and packaging process, the VLSI design approach faces new challenges. Among them, routing is one of the top ten problems that the current physical design needs to solve urgently [25]. In terms of chip size and capacity, the routing problem requires a circuit chip scale of tens of thousands of large modules and millions of small modules, and requires this work to be completed within a reasonable time. In addition, the quality of the routing severely affects other requirements in the design process, including timing and interconnection analysis. In order to reduce the high complexity of the routing process in chip design, the routing is usually divided into two steps: *global routing* and *detailed routing*. The global routing is a rough routing process, its function

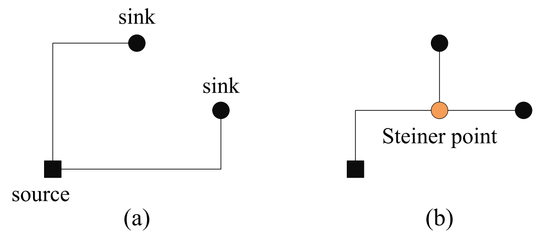


FIGURE 1. A three-pin net connected by (a) a spanning tree and (b) a Steiner tree, respectively.

is to reasonably distribute each part of the nets to be routed to each routing area in the chip, thereby clearly defining the routing problem of each routing area. The detailed routing is to complete the specific routing of each routing area under the guidance of the global routing results. This section gives a list of common subproblems in VLSI routing.

A. STEINER TREE

The shortest path problem of two-pin nets is one of the most basic problems in VLSI routing, which looks for the shortest routing path by given the position of the two pins while considering the obstacles. Commonly found strategies are *maze routing* [26], *line probe methods* [27], *pattern routing* [28] and so on. However, in 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 length of the routing tree, in addition to original nodes formed by given pins, the final MST can be constructed by introducing additional nodes called Steiner points. As shown in Figure 1(a), a three-pin net connected by spanning tree is given. The Steiner tree connection model corresponding to the net is shown in Figure 1(b). It can be seen that the length of the routing tree is greatly reduced after the introduction of the Steiner point. Therefore, the Steiner tree model has gradually become the best connection model for multi-pin net which is a key link in VLSI routing.

1) PROBLEM DESCRIPTION

The Steiner Minimum Tree (SMT) problem is to connect all pins through some extra points (called Steiner points) to achieve a minimal total length in VLSI routing.

In most routing problems, the segments can only route horizontally or vertically. This kind of routing tree is called Rectilinear Steiner Tree (RST). Rectilinear Steiner Minimum Tree (RSMT) construction is a NP-hard problem, which is as follows [29]: Given a set of points in the plane, the RSMT problem seeks to connect the points with a RST, which is a tree made up of horizontal and vertical line segments with 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, this way of routing only horizontally and vertically limits the optimization of wirelength, so more

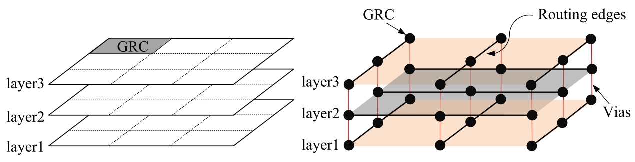


FIGURE 2. (a) Multilayer design. (b) Grid graph for routing.

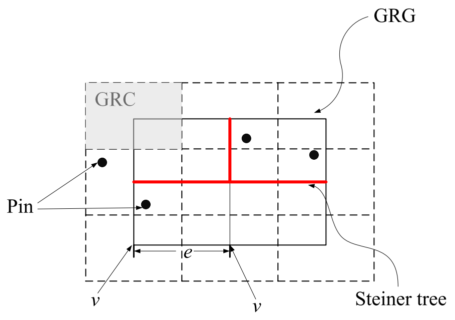


FIGURE 3. A GRG based on grid-graph.

research-oriented non-Manhattan architectures were proposed [30], [31]. Non-manhattan architecture routing trees mainly include Hexagonal Steiner Tree (HST) [30] and Octagonal Steiner Tree (OST) [31]. The routing directions of HST are considered at 0°, 60° and 120° to the horizontal direction called Y-routing, while OST with X-routing allows 45° and 135° directions in additon to traditional horizontal and vertical orientations. The non-Manhattan architecture related issues involved in this paper refers to X-architecture. Diversification of interconnection structures improves routing quality and improves chip performance.

B. GLOBAL ROUTING

The global routing problem is a typical graph theory problem. The Global Routing Graph (GRG) abstracts the routing area, routing capacity of each area, pin information of routing areas, and the relationship between different routing areas into a grid graph, and then uses the corresponding graph algorithm to solve this problem.

1) PROBLEM FORMULATION

For global routing, modern designs usually have several metal layers and two adjacent layers are linked by *vias* as shown in Figure 2(b). We use a grid diagram to describe the global routing model, that is, the entire routing area is divided into a set of rectangles with rows and columns interlaced. Each Global Routing Cell (GRC) is represented by one vertex, and the adjacency relationship between GRCs is represented by horizontal and vertical edges. For a given set of nets, their pin sets are mapped to the vertices corresponding to the GRC according to the placement of this GRC. Figure 3 gives a GRG which is a set of dotted grids of interlaced rows and columns. Each dotted grid box represents a GRC, and a pin set is placed in the corresponding GRC according to the result of the layout. Each GRC is mapped into GRG as a vertex (v) of GRG, and two GRCs having an adjacency relation are connected by an edge (e) of GRG. So the relationship between GRC and GRG is one-to-one.

The global routing problem of VLSI can be described as follows: There is a grid-graph G specifying a set of vertices V and a set of edges E. As shown in Figure 2, each vertex $v_i \in V$ corresponds to a rectangular region (or GRC) of the chip, and each edge $e_{ij} \in E$ corresponds to a boundary between adjacent vertices. And there is a set of nets N, each net $n_i \in N$ includes a set of pins P_i and each pin corresponds to a vertex v_i. The routing solution for a net is to find a tree that connects all pins of the net by using vias and routing edges. A solution of global routing is a set of SMTs in a grid graph, and a net corresponds to a Steiner tree that satisfies constraints. The goal of this phase is to minimize the length of the Steiner trees.

When evaluating a routing solution, one is typically concerned with three metrics, that is *overflow*, *wirelength*, *runtime* [32].

Overflow refers to the total amount of demand that exceeds capacity over all edges, here the demand corresponds to the number of routes that pass though the vertex [28]. In the actual design, we want this index to be as small as possible, ideally zero. The overflow of a routing edge e is defined as shown below:

$$overflow_e = \begin{cases} d_e - c_e, & if\ d_e > c_e \\ 0, & otherwise \end{cases} \tag{22}$$

where d_e represents the amount of nets that pass through e. The capacity c_e of routing edge e represents the number of available routing tracks it contains.

Wirelength is the total length of the segments that need to be connected for all nets, and it is desirable to be as small as possible. When using the routing tree models, the wirelength is generally equal to the length of the routing trees. Therefore, in many cases, various minimum trees are needed to be designed to solve the global routing problem. The Steiner tree mentioned above is one of the most common and effective models for solving global routing. In multi-dimensional routing, this calculation can also include vias.

Time is an indicator that we are particularly concerned about especially when reusing global routing to guide the layout algorithms [33].

2) COMMON GLOBAL ROUTING TECHNIQUES

Here is a brief list of some common global routing techniques.

a: MAZE ROUTING

Maze routing is one of the most classic ways to solve the global routing problem, which considers all possible routes between a given source and a given sink on a routing graph. This technique finds the routing path with the least cost from source to sink by applying various shortest path algorithms. The earliest maze routing algorithm comes from Lee's algorithm [26], which is the most widely used minimum path algorithm for finding two-pin nets, but it has the problems of large search space and high complexity. Subsequently, the A* search technique [34] is used to improve Lee's algorithm to speed up convergence. For the problem of multi-pin nets, multi-source and multi-sink maze routing [35] developed from maze routing is applied to the tree edges, which considers more potentially better routes.

b: PATTERN ROUTING

Pattern routing routes a two-pin net with predefined patterns like L-shaped and Z-shaped patterns, which is more efficient than maze routing. However, the solution quality may become worse because not all possible routes in the bounding box of a two-pin net are considered [36].

c: DECOMPOSITION TECHNOLOGY FOR MULTI-PIN NETS

This technology decomposes the task of global routing, that is, a multi-pin net is decomposed into several two-pin nets. The commonly used methods are SMT construction and MST construction. SMT often provides tree topologies with shorter wirelength, while MST shows greater flexibility because it can produce more L-shaped two-pin nets.

d: LAYER ASSIGNMENT

For multi-layer global routing, layer assignment maps a 2-D global routing result back to the original multi-layer solution space. Dynamic programming [37], Integer Linear Programming (ILP) [38], etc. are often used on this problem. Its main research focus is on minimizing the number of vias.

e: OPTIMIZATION STRATEGIES

In order to improve the quality of solutions, researchers have put forward various optimization strategies. *Rip-up and reroute* [39] allows the nets passing through congested regions be ripped up and rerouted for finding alternative routes. And *negotiated congestion routing* can balance the competing goals of eliminating congestion and minimizing the performance degrading due to timing critical paths [40]. This idea based on negotiation mechanism has been widely used in the design of routing system.

C. DETAILED ROUTING

Detailed routing realizes exact routing paths considering geometrical constraints based on the global routing solution, which is an important stage because it is directly related to the routing completion and design rule satisfaction.

1) PROBLEM FORMULATION

For detailed routing based on grid models, the goal is to find a legal routing path on a given routing grid and detour the crowded area as much as possible. The relevant definitions and problem formula for detailed routing are given below [41].

Definition 1: Channel (Routing Area). A rectilinear polygonal region between circuit blocks that can be used for interconnections (shown in Figure 4).

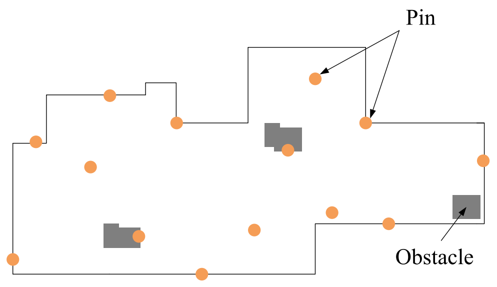


FIGURE 4. **Routing area with obstacles.**

Definition 2: Component. A set of pins and wire segments of a net which have been interconnected. Each unconnected pin is a trivial component.

The routing area is a right angle polygon. The location where the pins appear may be in the area or on the boundary. At the same time, we would like to take into account obstacles of various shapes and sizes. Detailed routing requires that all components of each net be connected within the routing area. The primary goal is to make the routing area as small as possible, and second, to minimize the number of vias and the wirelength of each net. At this stage, many other factors are also considered like power consumption, delay, coupling of interconnects, and so on.

2) COMMON DETAILED ROUTING TECHNIQUES

Here is a brief list of some common detailed routing techniques.

a: BASIC PATH SEARCH ALGORITHM

Like global routing, detailed routing also uses the path search algorithm to find the wire routes such as Lee's algorithm, A*, Soukup's algorithm [42], LCS* [34], etc.

b: RIP-UP AND REROUTE

Most of detailed routing researches are based on serial routing. As with the global routing, serial routing is often optimized by rip-up and reroute. However, such a sequential net-by-net approach is ineffective in handling congested designs and it usually creates unnecessary detour [43].

c: PARALLEL ALGORITHMS

In order to reduce the dependence of routing results on the order of nets, a parallel detailed routing algorithm based on multi-commodity flow model is proposed [44], which can simultaneously route for multiple nets. Ozdal [45] presented an insightful technique to perform escape routing for dense pin clusters and a Lagrangian relaxation based heuristic was proposed.

d: TRACK ASSIGNMENT

Track assignment is an ideal bridge between the global and detailed routing, which can effectively solve the mismatch between the two, thus better guiding detailed routing. In track assignment, segments extracted from global routing solution are assigned to routing tracks. This technology provides a better initial routing scheme for detailed routing by considering constraints such as local nets, congestion, and the location of vias.

IV. ROUTING PROBLEMS USING SWARM INTELLIGENCE

The purpose of this section is to introduce the application of SI techniques in VLSI routing. By analyzing the roles of SI techniques in routing problem, readers can more quickly and clearly realize the key points and difficulties to be solved in VLSI routing and advantages of SI in these aspects.

It is known that the basic optimization goal of routing is to minimize the interconnected wirelength while considering as many other optimization objectives as possible like obstacles, power consumption, timing and congestion. Therefore, every surveyed paper usually involves more than one routing problem mentioned in Table 2. And the application of SI techniques on global routing problems in Table 2 is given according to the most critical optimization objectives of surveyed papers.

A. APPLICATION OF ACO

1) APPLYING ACO TO SMT CONSTRUCTION

ACO algorithm is widely used to construct SMTs [46]–[48]. In general, we generate the Hanan grid of the terminal set T. Then place the ants in each terminal that needs to be connected. An ant will determine a new vertex by some rules and move to that vertex via an edge in Hanan grid. Each ant maintains its own tabu-list, which records the visited vertices to avoid revisiting it again [46]. Every time the ants move, the ants will leave a footprint called pheromone on the edge that has just passed, which will evaporate at a constant rate.

The process of an ant moving to the next vertex depends on a higher value $p_{i,j}$, which is a trade-off between the desirability and the trail intensity. Given ant m in vertex i, the desirability of vertex j is defined as follows:

$$\eta_j^m = \frac{1}{c(i,j) + \gamma \cdot \psi_j^m} \tag{23}$$

where γ is a constant, ψ_j^m is the shortest distance from vertex i to all the vertices in the tabu-list of others as quickly as possible. We use Eq. (2) to update the trail intensity in Hanan edge (i,j), where the increment of updating is given by the following formula:

$$\Delta\tau_{i,j} = \begin{cases} \dfrac{Q}{c(S_t)}, & if\,(i,j) \in E_t \\ 0, & otherwise \end{cases} \tag{24}$$

where $c(S_t)$ is the total cost of the current result tree S_t, E_t is the edge set of it, and Q is a constant which matches the quantity of the tree cost. Based on Eq. (1), the probability of an ant using edge (i,j) to move is defined as follows:

$$p_{ij} = \begin{cases} \dfrac{[\tau_{ij}]^\alpha \cdot [\eta_j^m]^\beta}{\sum_{k \neq tabu-list(m)} [\tau_{ik}]^\alpha \cdot [\eta_k^m]^\beta}, & if\ j \in A \\ 0, & otherwise \end{cases} \tag{25}$$

TABLE 2. **Application of SI techniques to the VLSI routing problems.**

SI Techniques	Routing problems		References
ACO	SMT construction		[46]–[48]
	Global routing	considering obstacles	[48]
		considering power consumption	[49], [50]
		considering interconnected wirelength	[46]–[50]
PSO	SMT construction		[29], [51]–[62]
	Global routing	considering obstacles	[54], [56], [57], [59]
		considering timing	[58], [63]–[66]
		considering congestion	[67], [68]
		considering interconnected wirelength	[29], [51]–[68]
DE	SMT construction		[69], [70]
	Global routing considering interconnected wirelength		[69]–[71]
	Restrictive channel routing		[72]
ABC	SMT construction		[73], [74]
	Global routing considering interconnected wirelength		[73]–[77]
FA	Global routing	considering timing	[78]
		considering interconnected wirelength	[62], [77], [78]

Algorithm 1 Construct Steiner Tree by ACO

```
   Input: terminal set $T$, connection graph
   Output: a rectilinear Steiner tree $T$
1  Place an ant on each vertex in the terminal set $T$ and put
   the vertex into its tabu-list;
2  Set the current sub-tree $t$ empty;
3  while ant number $> 1$ do
4      Select an ant $m$ randomly;
5      AntMove($m$);
6      Add the edge $m$ passing into $t$;
7      if $m$ meets $m_1$ then
8          Add vertices in tabu-list of $m$ to that of $m_1$;
9          $m$ dies;
10         Relocate($m_1$);
11 Prune($t$);
12 Return;
```

where A is the set made up of all vertices, which are connected with i and are not in the tabu-list of ant m [46].

The process of constructing an RST using ACO is shown in Algorithm 1 [47]. In Step 5, function *AntMove*(m) decides the next vertex that the current ant m will move to. And the encounter between two ants is described in Step 7 to 10. When ant m encounters ant m_1, ant m dies, and the vertices in the m's tabu-list are added into m_1's. If m_1 is still in the original position, *Relocate*(m_1) will re-create its position. When there is only one ant left, the tree is built. At the same time, *Prune*(t) is used to delete all 1-degree non-terminal vertices in the tree.

However, using ACO to find SMT is still time consuming because the ant's movement is based on the Hanan grid,

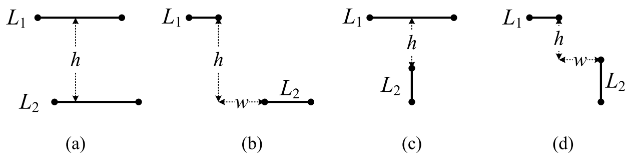


FIGURE 5. **Definition of distance between two edges.**

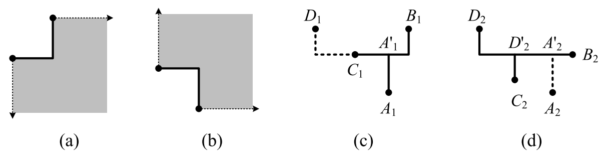


FIGURE 6. **(a) BOTTOM_ORIENT. (b) TOP_ORIENT. (c) BOTTOM_ORIENT segment. (d) TOP_ORIENT segment.**

and only a small segment can be moved per iteration. So [47] extends the tabu-list of each ant to record the edges instead of the vertices that this ant has visited so that every movement is not constrained by Hanan grid. They defined the distance between the two sides as follows: The distances between edge L_1 and L_2 in Figure 5(a), 5(b), 5(c), and 5(d) are h, $h+w$, h and $h+w$, respectively. And there are two possible ways to move when the shortest path is L-shape, which are TOP_ORIENT and BOTTOM_ORIENT. One orientation is chosen to move based on both the trail intensity and the topology. For a given edge orientation, the rules we find the closest vertex to the edge are shown in Figure 6. In Figure 6(c) and 6(d), vertex B represents the current location of an ant, and the closest vertex out of its tabu-list is vertex C. Next, the ant needs to decide the orientation of edge (B, C). Figure 6(c) shows that the closest vertex to edge (B_1, C_1) is A_1 with distance $|A_1A'_1|$ for BOTTOM_ORIENT. While for TOP_ORIENT, the distance between A_2 and edge (B_2, C_2) is $|A_2A'_2|$, which is shown in Figure 6(d). So the gain in BOTTOM_ORIENT is $|A_2A_2'| - |A_1A_1'|$ and the gain in TOP_ORIENT is $|D_1C_1| - |D_2D_2'|$. The authors rewrote Eq. (23) according to this rule as follows:

$$\eta_d = \frac{[gain_d]^\lambda}{dist_d} \tag{26}$$

where d is the two orientations (BOTTOM_ORIENT and TOP_ORIENT), $gain_d$ is the gain in orientation d, $dist_d$ is the distance from the closest vertex out of its tabu-list to the edge in orientation d, and λ is a constant that is the trade-off between the closest distance and gain. In this way, an ant can cross several edges of Hanan grid in each iteration and one ant will be removed from the set of alive ants in each movement.

This ACO-Steiner algorithm [47] can be used to construct initial trees for all nets and optimize the routing tree iteratively to further reduce wirelength. Moreover, this algorithm can generate different topologies with the same wirelength for a net, which is beneficial to reduce congestion scarifying only a little wirelength.

Reference [48] presents an Obstacle-Avoiding Rectilinear Steiner Minimum Tree (OARSMT) algorithm based on ACO. In this algorithm, an efficient graph reduction method, called *T-Reduction*, is adopted to reduce the searching space. This method is implemented by maintaining an FIFO queue. Firstly all 2-degree vertices are pushed into the queue. Then pop the first vertex from the queue, delete the vertex and its adjacent edges, and its adjacent non-terminal vertices are pushed into the queue while the queue is not empty. Later, non-terminal convex corner points are reduced. And finally reduce 1-degree terminals. Such terminals can be deleted along with their adjacent edges. It's worth mentioning that *T-Reduction* method is very efficient when the case scale is not large and is also suitable for escape graph reduction. And a greedy obstacle penalty distance (OP-distance) local heuristic is used in this OARSMT algorithm at the same time, where the OP-distance is used to estimate the distance between two vertices in the presence of obstacles. The proposed algorithm has a high optimization capability of wirelength and can handle complex obstacle cases including both convex and concave polygon obstacles.

2) APPLYING ACO TO GLOBAL ROUTING

ACO-Steiner can be easily extended to other routing problems such as power-driven routing. References [49] and [50] implement ACO algorithms on both grid-less Manhattan architectures and non-Manhattan routing architectures that use diagonal routing, which aim at limiting the power consumption of the chip. The difference is that, [50] actives power on the chip not only by minimizing wirelength and vias, but also capacitance.

In [50], the characteristics of ant colony's memory, stochastic decision making and strategies of collective and distributed learning are used to find the shortest possible routes, then choose the one that minimizes capacitance from those routes. Unlike [46], an ant will not die when meeting another ant, but rather a particular connection is marked as completed. Completing these paths can also reduce redundant paths, thereby decreasing the number of vias. The first step of the algorithm is to create a Hanan grid, then start routing nets using ACO. Notice that small nets are routed first in order to minimize obstruction from bigger nets. The specific ACO-Route approach can be described as Algorithm 2. The probability of unvisited nodes are calculated through the heuristic to minimize the distance between the ant and other ants and pheromone values (Step 10). If the ant meets another ant, for both the ants add the routes traveled by one to the route list of the other. The experiments prove that ACO is successful in solving the multi-constraint optimization problem of VLSI chip power minimization.

Algorithm 2 ACO for Routing

```
1  Create Hanan grid;
2  Order nets according to degree and then size;
3  Initialize a small amount of pheromone on the whole
   grid;
4  while termination condition is not met do
5      Route
6      for each ant do
7          Empty ant's memory;
8          Place ant at some terminal node;
9          Construct a complete tour for ant;
10         Calculate the probability of unvisited nodes;
11         Deposit pheromone on the paths taken by the
           ant;
12     Find the best ant of the iteration;
13     Update the global pheromone value;
14     End Route
15     Find the shortest ant routed solution, and check if
       any part of this solution interferes with any other
       solution.
```

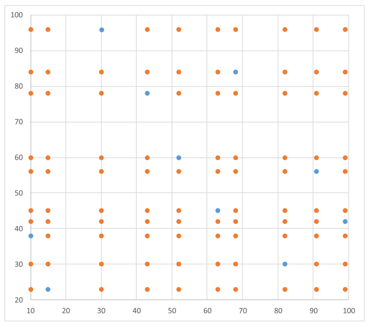
B. APPLICATION OF PSO

1) APPLYING PSO TO SMT CONSTRUCTION

The PSO algorithm introduced in Section II is usually used to solve continuous problems, while VLSI routing is a discrete problem. For this reason, many scholars have improved SPSO to solve actual discrete problems. At present, PSO has been widely used in SMT construction for VLSI routing problem, and has achieved good results. The key of SMT construction is the selection of Steiner points.

a: RST CONSTRUCTION USING PSO

Dong *et al.* [51] proposed a routing algorithm based on discrete particle swarm optimization (DPSO-RA), aiming at length and the bound of connectivity rate. The algorithm adopts a novel encoding and several update operations for DPSO, and achieves interconnection of all destination nodes in VLSI.



(a)

$$\begin{matrix}0&0&1&1&0&0&0&0&0&0\\0&0&0&0&0&0&1&0&0&0\\0&0&0&1&0&0&0&0&0&0\\0&0&0&1&1&0&1&0&0&0\\0&0&0&0&0&0&1&0&1&0\\0&1&0&0&0&1&1&0&0&0\\0&0&0&0&0&0&1&1&0&1\\1&1&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&1&0&0\\0&1&0&0&0&0&0&0&0&0\end{matrix}$$

(b)

FIGURE 7. (a) The establishment of Steiner matrix. (b) Particle coding of Steiner matrix.

Particle Encoding. Each particle represents an MST. DPSO-RA encodes the particle by establishing Steiner Matrix. A particle X_i is a matrix and is expressed as:

$$\begin{matrix}x_{11}&x_{12}&\cdots&x_{1n}\\x_{21}&x_{22}&\cdots&x_{2n}\\\vdots&\vdots&\ddots&\vdots\\x_{n1}&x_{n2}&\cdots&x_{nn}\end{matrix}$$

where n is the number of components waiting for interconnection. Each bit of the particle encoding corresponds a node in the Steiner Matrix, and is valued in binary. The nodes to be connected, called "terminal nodes", are taken as input from the user or are generated randomly in the form of x and y-coordinates. If a node is selected to construct the RST, the value of the bit is '1', otherwise '0'. Usually, the matrix will be reduced by deleting some rows and columns that do not have any terminal nodes.

Steiner Matrix is established by the horizontal and vertical lines' cross-points of the components. And it represents the possible locations of the Steiner nodes of RST. Figure 7(a) shows a Steiner Matrix of 10 components (blue) to be connected. Its corresponding particle is shown in Figure 7(b).

The velocity V_i of particle X_i is also a matrix, expressed as follows:

$$\begin{matrix}v_{11}&v_{12}&\cdots&v_{1n}\\v_{21}&v_{22}&\cdots&v_{2n}\\\vdots&\vdots&\ddots&\vdots\\v_{n1}&v_{n2}&\cdots&v_{nn}\end{matrix}$$

where each bit of the velocity represents the probability of selecting the corresponding node in Steiner Matrix.

Fitness Function. The cost of MST is the fitness value of this particle, which is calculated by Prim's algorithm.

Update Operation. DPSO-RA redefines the operations of PSO and uses Eq. (3) and (4) as the position and velocity of the particle respectively.

Definition 3: "$-$" Operation. The result of "$-$" operation between two particle locations is a velocity.

Definition 4: "$+$" Operation. The result of "$+$" operation between two velocities is a new velocity.

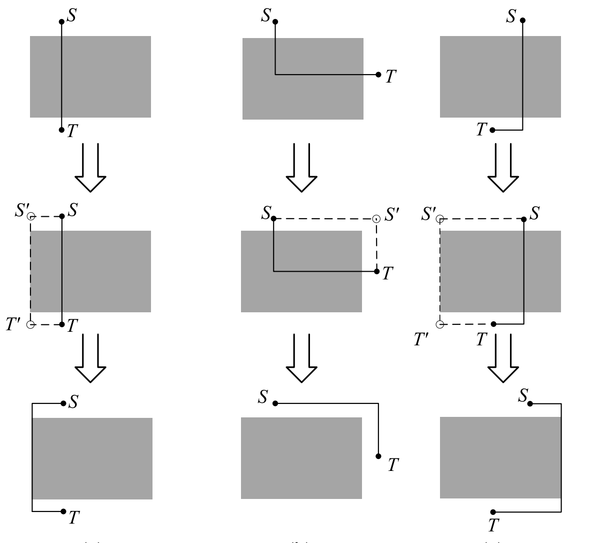


FIGURE 8. Obstacle-avoiding construction by generating virtual vertexes.

Definition 5: "$\times$" Operation. The parameters c_1 and c_2 are constants, r_1 and r_2 are random numbers between 0 and 1. The result of "$\times$" operation between a velocity and parameters is still a velocity.

Algorithm Processes.

Step 1. Initialize the particle swarm.

Step 2. Calculate the fitness value of each particle.

Step 3. Update *pbest* and *gbest*.

Step 4. Calculate the self-adaptive inertia weight for each particle.

Step 5. Update the velocity and position of particles.

Step 6. If the termination condition is met, the algorithm ends. Otherwise return to Step 2.

References [52] and [53] also use Steiner Matrix to encode the particles, and introduce the mutation operation of GA into DPSO. In [52], a matrix corresponding to a particle represents both position and velocity. Its velocity is updated by Eq. (3), while Eq. (4) is not used to update the position of the particle. The algorithm updates the position by selecting only some bits of the position matrix of the particles from the previous generation. This is the mutation process they introduced that can increase the efficiency and robustness of DPSO. Reference [53] uses PSO algorithm with constriction factor, that is, Eq. (7) is used to update the velocity and Eq. (4) is used to update the position. This algorithm adapts a novel mutation operation by selection of some bits of the previous position vector along with the information of the initial Steiner Matrix.

Shen *et al.* [54] considered wirelength and obstacles at the same time and then proposed a novel OARSMT construction algorithm, which avoids obstacles by generating virtual vertexes of the real vertexes. The algorithm designs the following obstacle-avoiding strategy for the case where the three types of routing across obstacles shown in Figure 8. Take Figure 8(a) as an example. First, vertices S and T

respectively produce corresponding virtual vertices, which have the same horizontal coordinate with the left (or right) edge of the obstacle and the same vertical coordinate with the original vertices. Next, sequentially connect S, S', T' and T, and dismantle the violate edges.

The OARSMT construction is divided into the following two steps:

(1) Divide all the pins of the net into k groups, and generate corresponding RSTs based on the above-mentioned obstacle-avoiding strategy. The algorithm places the pins that are close to each other in a group, while making the amount of each group of pins (at least two pins) as small as possible.

(2) Wirelength minimization is the optimization goal, and then [54] connects all of the groups considered as real vertexes to construct the final RST.

In this algorithm, Prüfer number [79] is used to encode the solution of RST, and 0-1 coding is used to select Steiner points. For example, a particle is represented by:

$$X_i^t = \begin{bmatrix} 2 & 5 & 4 & 5 & 4 & 100 \\ 0 & 1 & 0 & 0 & 1 & 15 \end{bmatrix}$$

where the Prüfer number is {2,5,4,5,4}, and the Steiner point selection scheme is {0,1,0,0,1}, the total length of this tree is 100 and the obstacles quantities of this particle is 15.

The algorithm introduces crossover and mutation operators inspired from GA to handle Prüfer numbers which are discrete magnitude. Based on SPSO (Eq. (3)), the following position update formula is given for solving the discrete problems:

$$X_i^t = c_2 \oplus F_2(c_1 \oplus F_1(\omega \oplus F(X_i^{t-1}), P_i^{t-1}), G_i^{t-1}) \qquad (27)$$

Eq. (27) uses the "+" operation to connect the following three parts, in other words two operations including mutation and crossover operation.

(1) Mutation operation:

$$\lambda_i^t = \omega \oplus F(X_i^{t-1}) = \begin{cases} F(X_i^{t-1}), & r_0 \le \omega \\ X_i^{t-1}, & else \end{cases} \qquad (28)$$

where F represents the mutation operator from GA with the probability of ω, and r_0 is a random number on the interval [0, 1).

(2) Crossover operation:

$$\delta_i^t = c_1 \oplus F_1(\lambda_i^t, P_i^{t-1}) = \begin{cases} F_1(\lambda_i^t, P_i^{t-1}), & r_1 \le c_1 \\ \lambda_i^t, & else \end{cases} \qquad (29)$$

$$X_i^t = c_2 \oplus F_2(\delta_i^t, G_i^{t-1}) = \begin{cases} F_2(\delta_i^t, G_i^{t-1}), & r_2 \le c_2 \\ \delta_i^t, & else \end{cases} \qquad (30)$$

where F_1, F_2 represent the crossover operators that influence self-cognitive and social-cognitive respectively, and the probabilities are c_1 and c_2. r_1 and r_2 are random numbers on the interval [0, 1).

This paper proposes a novel framework of PSO to solve OARMST problem and extends the application of SI algorithm in VLSI physical design. Many of the later works [55]–[61], [67], [68], [80], [81] are also based on this PSO framework of Eq. (27).

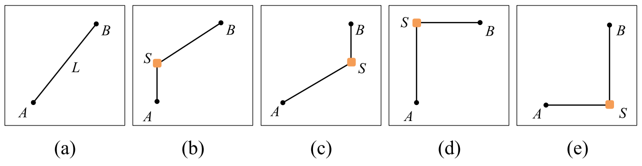


FIGURE 9. Four options of Steiner point for a given line segment.

b: OST CONSTRUCTION USING PSO

Liu *et al.* [55], [56], Huang *et al.* [56], [57], and Liu *et al.* [58]–[61] have done a lot of research work on the construction of Octagonal Steiner Minimum Tree (OSMT) based on the following interconnected structure, and have achieved excellent results.

Interconnected Structure. There are four choices for line segment L of connecting pins A and B as shown in Figure 9. And [55] gives four definitions about selection of Steiner points.

Definition 6: Choice 0 (as shown in Figure 9(b)). The Choice 0 of Steiner point corresponding to edge L is defined as leading rectilinear side first from A to Steiner point S, and then leading non-rectilinear side to B.

Definition 7: Choice 1 (as shown in Figure 9(c)). The Choice 1 of Steiner point corresponding to edge L is defined as leading non-rectilinear side first from A to Steiner point S, and then leading rectilinear side to B.

Definition 8: Choice 2 (as shown in Figure 9(d)). The Choice 2 of Steiner point corresponding to edge L is defined as leading vertical side first from A to Steiner point S, and then leading horizontal side to B.

Definition 9: Choice 3 (as shown in Figure 9(e)). The Choice 3 of Steiner point corresponding to edge L is defined as leading horizontal side first from A to Steiner point S, and then leading vertical side to B.

Particle Encoding. Based on the above design of interconnected wire, [55] uses numerical encoding called *Edge-to-Point* encoding to represent each candidate OST: For a net with n pins, a spanning tree would have n-1 edges, n-1 Steiner points and one extra bit which is the fitness of particle. Besides, two bits represent the two vertices of each edge. Namely, the particle is encoded with a numerical string of 3×(n-1)+1. For example, a particle can be expressed as follows:

7 6 **0** 6 4 **1** 7 5 **1** 5 1 **2** 1 3 **0** 1 8 **1** 5 2 **2** 0.0100

where the number '0.0100' is the fitness of the particle and each number in bold represents the Steiner point choice. Taking the first substring (7, 6, **0**) for instance, it represents one edge of the spanning tree which is composed of *Vertex 7*, *Vertex 6* and the Steiner point choice (*Choice 0*).

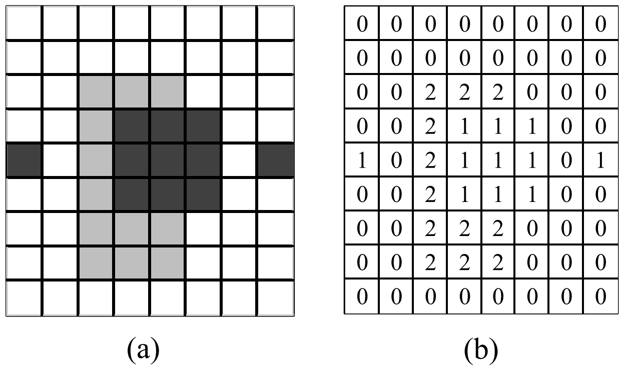
Reference [55] introduces mutation and crossover operations as particle update strategies like [54]. Like the problems of RST, we also study related problems of OST such as Obstacle-Avoiding Octagonal Steiner Minimum Tree (OAOSMT). References [56]–[58] consider the construction of OST with key issues such as obstacles, delay and number of bends, also based on this encoding method and particle update strategies, while using Union-Find Sets (UFS) to prevent the generation of invalid solutions. Reference [60] constructs OSMT based on self-adapting PSO with an effective Hybrid Transformation Strategy (HTS) to enlarge the search space. While [61] presents a unified algorithm for both octagonal and rectilinear SMT construction, and also applies HTS to PSO, which can achieve a better wirelength optimization. These two algorithms both have an ability to obtain multiple topologies of Steiner trees with the same wirelength, so that it can provide different topology options for congestion optimization in global routing.

2) APPLYING PSO TO GLOBAL ROUTING

PSO is also widely used for global routing because of its simplicity and the ability to quickly converge to an ideal solution.

Minimizing wirelength and reducing congestion are the most common and important tasks for global routing. Liu *et al.* [68] proposed a high-quality VLSI global router in X-architecture called XGRouter, which was based on ILP techniques, partition strategy and PSO. Their ILP formulation, called O-ILP, considers the congestion uniformity to get a more even routing distribution without generating too many hot spots and the solution has no overflowed edges. At the main stage of their algorithm, ILP is formulated from the original routing sub-region and an improved PSO is designed to solve the O-ILP formulation. In their proposed PSO, the 0-1 integer encoding is used for particle encoding, which can satisfy the completeness and non-redundancy principle. And based on the update formula of particles (Eq. (27)), a check strategy is incorporated in the crossover and mutation operators to meet the soundness principle. Finally, not only the wirelength but also the congestion uniformity is considered into the fitness function to generate better solutions. Reference [67] presents some enhancements based on XGRouter. In this paper, some new types of routing are introduced and PSO algorithm is combined with maze routing. This improved algorithm can be applied to multi-layer routing model, which can achieve good results on overflows and total cost of wirelength.

With the continuous development of the manufacturing process and design scale of IC, the feature size is continuously reduced, the parasitic effect of the connection is not negligible, and the interconnection delay exceeds the gate delay, which becomes the main factor determining the performance of the circuit. Therefore, it is necessary to consider other typical performance optimization targets except wirelength, such as delay and power dissipation. Several techniques have been used to reduce interconnection delay – *wire sizing*, *buffer insertion*, and *buffer sizing*. Among them, buffer insertion is the most effective interconnection optimization method to reduce delay.



0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	2	2	2	0	0	0
0	0	2	1	1	1	0	0
1	0	2	1	1	1	0	1
0	0	2	1	1	1	0	0
0	0	2	2	2	0	0	0
0	0	2	2	2	0	0	0
0	0	0	0	0	0	0	0

FIGURE 10. (a) Location of obstacles. (b) Corresponding vertex value.

Ayob *et al.* [63] employed PSO to solve buffer insertion problem in VLSI routing, with considerations on wire and buffer obstacles. They used distributed RC network as interconnected model and applied the Elmore Delay formulation to calculate the interconnection delay from source to sink.

a: ELMORE DELAY CALCULATION

For computation delay where the iterative calculation starts from source then advance to the sink, each node in the wire is labeled with a resistance-delay pair (r, t) where r is resistance of wire and t is delay time accumulated up to that node, respectively [82]. If the subsequent segment is wire, then the subsequence delay pair (r', t') is defined as:

$$r' = r_w + r \tag{31}$$
$$t' = (r + r_w/2)c_w + t \tag{32}$$

where r_w and c_w are the resistance and capacitance of the wire segment, respectively. If the segment consists of wire with buffer then (r', t') is defined as:

$$r' = r_b \tag{33}$$
$$t' = r(c_w + c_b) + r_w(c_w/2 + c_b) + d_b + t \tag{34}$$

where d_b, r_b and c_b are the intrinsic buffer delay, buffer output resistance and buffer input capacitance, respectively.

b: GRID-GRAPH MODEL

This algorithm utilizes a grid-graph model to represent routing path and buffer placement location. Figure 10(a) shows the location of source and sink (dark), buffer obstacle (gray) and wire obstacle (dark). And Figure 10(b) shows the corresponding value of each vertex location. The value is set to 1, which indicates the presence of wire obstacle for routing the path. The region whereby buffer is not permitted (buffer obstacle area) will be set to 2. Value = 0 represents the area that is possible for routing path and for buffer placement location.

c: PARTICLE ENCODING

In this problem model, each of the solution of path routing in each iteration is represented by one particle of PSO. If using a maximum number of 8 doglegs, each particle can be encoded

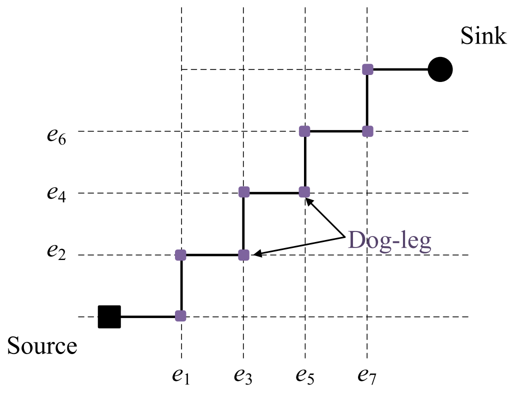


FIGURE 11. PSO VLSI routing mapping.

by a seven-vector of position which is from e_1 to e_7 shown in Figure 11. And e_{even} is the location in grid graph measured from origin along x-axis and e_{odd} is the location in grid graph measured from origin along y-axis [63]. One path from source to sink is generated by connecting these continuous segments: source $\rightarrow e_1 \rightarrow e_2 \rightarrow e_3 \rightarrow e_4 \rightarrow e_5 \rightarrow e_6 \rightarrow e_7 \rightarrow$ sink. For a more complex case study, number of doglegs can be increased by increasing the vector position, e to n value. Thus particle i with the position and velocity are presented as:

$$x_i = \begin{bmatrix} e_1 \\ e_2 \\ \cdot \\ \cdot \\ \cdot \\ e_{n-1} \\ e_n \end{bmatrix}, v_i = \begin{bmatrix} v_{e_1} \\ v_{e_2} \\ \cdot \\ \cdot \\ \cdot \\ v_{e_{n-1}} \\ v_{e_n} \end{bmatrix} \tag{35}$$

d: FITNESS FUNCTION

The fitness of this algorithm is the delay time calculated accumulatively from source to sink, including both wire segment and wire segment terminated with buffer. Taking Figure 11 for instance, the number of total segments used can be calculated as Eq. (36)

$$p = (\sum_{a-1}^{n} |e_a - e_{a+2}|) + |x_{si} - e_{n-1}| + |x_{so} - e_1| \tag{36}$$

where n is a maximum number of dogleg and e_a represents the location of a node in grid graph. x_{so} and x_{si} are x-axis coordinates for source and sink. For every 6 segments, a buffer will be placed randomly. From the value of P and the corresponding buffer location, the delay time can be determined using Elmore delay within iterative calculation.

However, the algorithm did not perform wire and buffer sizing. Hence, Yusof *et al.* [64] extended Ayob's work by employing Binary Particle Swarm Optimization (BPSO) in buffer insertion and wire sizing in the presence of wire and buffer obstacles. In this algorithm, particle position X can be modelled as follows:

$$X = [x_1 x_2 \ldots x_q]^T$$

where x_q represents the type of wire with/without buffer for the respective to grid segment, q. For each x, seven bits are used to represent the case study, where the first 3 bits represent the type of wire used, the fourth bit indicates the use of buffer, and the last 3 bits represent the type of buffer used. The fitness function is determined by capacitance-delay pair (c, t), where c represents the total ground capacitance at node v in a delay model. And a new capacitance-delay pair (c', t') for the preceding interconnected segment is determined as follows:

$$c' = \begin{cases} c_w + c; & case1 \\ c_w + c_b; & case2 \end{cases} \tag{37}$$

$$t' = \begin{cases} r_w(c_w/2 + c) + t; & case1 \\ r_w(c_w/2 + c_b) + d_b + r_b c + t; & case2 \end{cases} \tag{38}$$

The particle velocity is updated by Eq. (3), and the position X, is updated for next iteration t for each bit d, using Eq. (39) based on the probability of the normal distribution.

$$X_{id}^{t+1} = \begin{cases} 1, & r < \dfrac{1}{1 + e^{-v_{id}^{t+1}}} \\ 0, & r \geq \dfrac{1}{1 + e^{-v_{id}^{t+1}}} \end{cases} \tag{39}$$

where r is random number of [0,1].

In addition, Yusof *et al.* [65] also proposed a two-step BPSO approach to obtain the best path of wire placement with buffer insertion from source to sink. The algorithm uses two BPSOs respectively to find the shortest path of wire placement and the best location of buffer insertion along the wire. The fitness function used for the second BPSO is the time delay formulation based on the iterative RLC delay model.

However, the low exploration capability of BPSO method sometimes leads to a low convergence rate of the routing algorithm. To overcome this drawback, Nath *et al.* [66] proposed a two-step modified constricted PSO with the integration of the mutation (CPSO-MU). PSO-MU discovers the minimal path of placing the wire at first and explores the finest location along the wire for inserting the buffer in the final step. Compared with [65], CPSO-MU can accomplish global convergence with a slighter number of iterations and produce a less interconnect delay.

C. APPLICATION OF DE

1) APPLYING DE TO SMT CONSTRUCTION

Manna *et al.* [69] described an improved DE for finding RSMT. The entire search space is represented by an n-by-n null matrix, where n is the dimension of search space. Then randomly change the element from '0' to '1' to produce an initial population. In order to better solve the discrete problem, they redesigned the three links of DE as follows:

(1) *Mutation*. Choose one particle randomly, then change the elements of the matrix randomly from '0' to '1'. The elements which are already equal to 1 are kept intact. This new matrix is treated as mutant matrix.

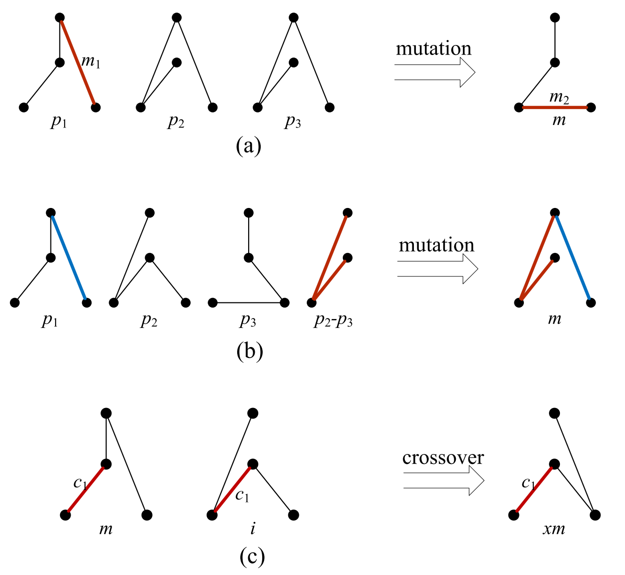


FIGURE 12. DDE for OSMT construction: (a) mutation operator when difference set is empty; (b) mutation operator when difference set is not empty; (c) crossover operator.

(2) *Crossover*. Crossover is performed between mutant matrix and target matrix to generate a trial Matrix using Eq. (13). If a generated random number is greater than the *CR* then an entire row (or column) is copied from target particle's position matrix to newly formed trial particle's position matrix. Otherwise the row (or column) is copied from the mutant matrix.

(3) *Selection*. The fitness values of trial matrix and target matrix are compared. The target matrix will be replaced by a matrix with better fitness value.

The proposed algorithm works well in both small and large dimensional routing problems. And because of DE's simplicity and flexibility, it can also be used in other discrete optimization problems.

In 2019, Wu *et al.* [70] proposed a feasible and effective X-architecture SMT construction algorithm based on Discrete Differential Evolution (DDE). And they designed new mutation and crossover operators combined with UFS for DDE. In this work, every particle is encoded as a numerical string by *Edge-to-Point* encoding like [55]. And the last bit represents the length of OSMT, while the fitness is set to the function of the length. Eqs. (8), (13) and (14) are adopted as particle updating formulas, respectively representing mutation, crossover and selection operations.

They designed novel operators and the corresponding evolution of using DE to search for OSMT is shown in Figure 12.

(1) *Mutation*. The subtraction operation in Eq. (8) refers to the difference set operation, and the addition operation refers to the union operation of the set. Choose three particles randomly, respectively p_1, p_2, p_3. So the subtraction operation can be divided into two cases:

- One is that the result of $p_2 - p_3$ is empty set. Edge m_1 to be mutated is randomly selected from particle p_1, then delete this edge, and p_1 is divided into two sub trees. Combining with UFS, two points are randomly selected from the two sub trees respectively and reconnected as the mutation particle (as shown in Figure 12(a)).
- The other is that the result of $p_2 - p_3$ is not empty. The elements of the difference set are as part of the edges of mutation result. Then select the remaining from the edge set of p_1 to reconstruct a new tree as the final mutation particle (as shown in Figure 12(b)).

(2) *Crossover*. When the mutated particle m crosses with the current particle i, the following action is performed: The common edges of the two trees are used as the starting set, and the remaining edges are used as the candidate set. Combining with UFS, constantly extract the remaining edges from candidate set until the crossover result is a legal tree (as shown in Figure 12(c)).

(3) *Selection*. A greedy strategy based on fitness is adopted in selection operation.

Compared with DPSO-based RSMT [29] and OSMT [55] construction algorithms, this kind of DDE can achieve a greater wirelength reduction on solving RSMT problem and in terms of constructing OSMT, the wirelength optimization ability is similar to that in [55]. It can be seen that DE has certain advantages over PSO in finding RSMT.

2) APPLYING DE TO GLOBAL AND DETAILED ROUTING

DE is also widely used in VLSI global routing. Reference [71] uses DE for 3D IC global routing to minimize wirelength with respect to Through Silicon Vias (TSV). The mutation operation of the algorithm uses Eq. (8), that is, the mutation operator creates mutant vectors by perturbing a randomly chosen vector X_{r_1} with the distinction of two other randomly chosen vectors X_{r_2} and X_{r_3}. The crossover and selection operations are performed by Eqs. (13) and (14), respectively. And [72] uses DE for restrictive channel routing and use the horizontal and vertical constraints in the solution vector encoding to eliminate unfeasible solution, which leads to lower complexity and reducing search space.

D. APPLICATION OF ABC

1) APPLYING ABC TO SMT CONSTRUCTION

[73] presents an ABC algorithm for solving the routing optimization problem, showing noteworthy improvements in reduction of the total interconnected length. The problem is to find a RSMT to connect all terminals or pins, without any severe loss of generality from a grid graph. And the search space is also represented by Steiner Matrix. Since ABC is good at exploration but poor at exploitation, they introduced two steps in the algorithm to improve the exploitation. One is to design a new probabilistic function for selecting food sources that takes into account the type and complexity of the problem, which is described as follows:

$$p_i = e^{-\frac{fit_i}{\rho}} \tag{40}$$

where ρ is a constant depending on the type and complexity of the problem dealt. The other is a new equation to exploit the food sources in onlooker bee phase of the algorithm, which is described as follows:

$$v_{ij} = x_{best} + \phi(x_{best} - x_{ij}) \tag{41}$$

where x_{best} is the best solution in the current iteration. Using Eq. (41), the worst solution will get the best chance for local search and the best solution in the current population can modify all other solutions in next generation.

The specific procedures for solving RSMT problem with ABC are shown in Algorithm 3.

Algorithm 3 Construct RSMT by ABC

```
1  Define objective function, $f(x)$, maximum number of
   iterations, $maxit$ and a counter, $c$ ;
2  Initialize a population of bees $x_i$ ($i = 1,2,\ldots,NP$) and
   iteration counter $iter = 0$;
3  while $iter < maxiter$ do
4      for $i = 1:NP/2$ (employed bees) do
5          Calculate new solution by Eq. (16);
6          Calculate $f(x_i)$ for all $i$;
7          Calculate $p_i$ by Eq. (40);
8      for $i = 1:NP/2$ (onlooker bees) do
9          Select solution based on $p_i$;
10         Calculate $f(x_i)$;
11         Calculate $p_i$ by Eq. (40);
12         Use greedy selection;
13     if for particular $x_i$, $f(x_i)$ doesn't improve until $c$ then
14         Scout produces new solution using Eq. (18);
15     $iter = iter + 1$;
```

Later in 2015, Zhang and Ye [74] proposed a discrete ABC for the OARSMT construction which is used for routing the multi-terminal nets in VLSI design. The algorithm generates the escape segments for OARSMT firstly, and then use the discrete ABC to search a near-optimal solution. Compared with the basic ABC algorithm, their algorithm has the following improvements: (1) A key-node-based encoding scheme is adopted to build a tight searching scope for representing the feasible solution; (2) For Steiner tree construction, a modified classic heuristic is presented as the encoder that can construct the Steiner tree with a good solution quality; (3) A key-node neighborhood configuration and two local search operators with an indefinite search step length are applied to the local search strategy; (4) In addition, a global search operator is designed to avoid the generation of the same initial solution every time and a merging operation is used to enhance the global search.

2) APPLYING ABC TO GLOBAL ROUTING

Bhattacharya *et al.* [75] utilized ABC to find the optimum wire length of a chip during global routing. The search space of the problem is represented by the Steiner Matrix. In this case, the solutions act as the food sources and the nectar amount of the food sources decides their fitness.

Three phases are used to obtain a solution:

a: EMPLOYED BEE PHASE

Initialize a number of solutions (equal to half of the population size). Each solution is generated by randomly selecting some elements of its corresponding matrix and setting it to 1 (These points can be regarded as Steiner points in the original global routing graph), while the original terminal nodes are kept unchanged.

b: ONLOOKER BEE PHASE

The fitness value of each matrix is obtained using the Prim's algorithm. The probability value of each solution (matrix) is then obtained by Eq. (17). Depending on this probability, the solutions are selected in the onlooker bee phase.

c: SCOUT BEE PHASE

For any food source (matrix), the solution is discarded and the algorithm will enter the scout bee phase if the solution is not improved within the set threshold range. And the new solutions are generated by randomly selecting points in the original matrix.

Repeat the above steps until the termination condition is satisfied. Then the best solution is determined according to the fitness value and the optimal wirelength is obtained. In this paper, ABC can work efficiently because the candidate solutions are obtained from the parents by means of a simple and elementary operation method. This method is implemented based on finding the difference between randomly chosen parts of parent solution and a random solution from the population. So that the convergence of the algorithm is hastened. Compared with a quick and robust global router NTHU 2.0 [83], which solves all International Symposium on Physical Design (ISPD) benchmarks, [75] can generate better results and more optimum routing configurations.

Zhang and Ye [76] applied the ABC algorithm to the routing for two-terminal nets and compared its performance with the maze algorithm. In order to adapt the ABC algorithm for solving the routing problem of two-terminal nets, several parts of ABC are altered as follows: (1) The algorithm adopts a sequence encoding method, which divides the solutions of the problem into two parts: the row-based solutions and the column-based solutions; (2) Half of the initial solution is generated by pattern routing, and the other half is randomly generated to speed up the algorithm convergence; (3) A local search operation is designed to exchange the front part sequences or behind part sequences of the auxiliary solution and the current solution to generate two new solutions, and then select the best solution from the new solutions and the current solutions as the updated current solution; (4) A dynamic attractiveness probability is designed for adjusting the positive feedback of this algorithm during the different stages. The experimental results demonstrate that ABC algorithm can find the less cost routing paths for the routing problem of two-terminal nets than the maze algorithm.

E. APPLICATION OF FA

1) APPLYING FA TO GLOBAL ROUTING

Nasir *et al.* [78] applied FA on VLSI routing to find the minimum time delay by choosing the path and placing the buffers intelligently. The proposed algorithm employs the location of doglegs to model the firefly that represents the routing solution. Like the previous model of using PSO to solve routing problem [63], in this work, a distributed RC network is still used as interconnected model and the Elmore Delay formulation is applied to compute the interconnect delay from source to sink. And the same grid-graph model, as shown in Figure 8(a) and 8(b), is used to represent the areas that are possible for routing path and information about obstacles and buffers. In the model of Figure 9, each firefly represents a candidate solution of the problem. Similar to the particle swarm, the position of each firefly in FA is also a seven-vector from e_1 to e_7. Thus the general representation of the firefly in VLSI routing problem as follows:

$$x_i = \begin{bmatrix} e_1 \\ e_2 \\ \cdot \\ \cdot \\ \cdot \\ e_{n-1} \\ e_n \end{bmatrix}$$

Algorithm 4 FA for VLSI Routing Problem

```
1  Define objective function $f(x)$, and maximum number of
   iterations, $maxit$;
2  Initialize a population of fireflies $x_i (i = 1, 2, \ldots, q)$,
   parameters: $\gamma$, $\beta_0$, $\alpha$, and iteration counter $iter = 0$;
3  Place the buffers randomly for the population;
4  while $iter < maxiter$ do
5      for $i=1 : q$ (all $q$ fireflies) do
6          for $j = 1 : q$ do
7              if $I_i < I_j$ then
8                  Move firefly $i$ towards $j$ using Eq. (21);
9                  Place the buffers randomly for firefly $i$;
10                 Perform correction if necessary;
11                 Evaluate new solution according to the
                   $f(x)$, update $I_i$ and global best if
                   necessary;
12 Output the firefly with the highest light intensity;
```

Algorithm 4 gives the FA algorithm for VLSI routing optimization problem. The algorithm generates the initial firefly population randomly, and the location of a firefly is evaluated by the fitness function. And light intensity of firefly I_i at x_i is determined by fitness $f(x)$ using Eq. (31) to Eq. (34). At each iteration, each firefly will fly to a firefly with larger light intensity, using Eq. (21) as the position update formula, where distance r is the Cartesian distance between the two fireflies. However, after the firefly moves towards another agent, the new position suggested might be invalid due to inexact coordinate location. Therefore, the correction strategy in Step 10 is proposed to mainly focus on two situations: one is that the solution is a decimal, and the easy way is to round it off; the other is that the solution exceeds the coordinate boundary, and the measure taken is to ignore this iteration. Next (Step 11) is the fitness evaluation of the new firefly and the update of the light intensity. The final solution is obtained by continuously iterating and updating the global best. The author compared this method with PSO-based routing algorithm [63] and the same optimal solution was obtained. For the routing problem without buffer insertion, the algorithm process is roughly the same as Algorithm 4, only needing to omit Step 3 to Step 9.

In [62], FA, the PSO with inertia weight (PSO-W), the PSO with constriction factor (PSO-C) and the PSO with mutation (PSO-MU) are used to minimum wirelength in global routing. And both FA and PSO-MU yield lowest minimum wirelength, but the calculation time consumed by FA is about 6 times that of PSO-MU and 10 times that of PSO-W. It can be seen that traditional FA algorithm has a better optimization capability than traditional PSO algorithm, but a good improved PSO algorithm such as PSO-MU can achieve the same optimization results as traditional FA, and the time complexity is much lower.

In [77], FA and ABC are used to connect all terminal nodes under the initial setting of the randomly generated terminal nodes coordinates. And the performance of the two algorithms in terms of minimizing the interconnected length is compared. The experimental results show that FA can achieve shorter wirelength in the search for terminal nodes of RSMT, but the computational cost is larger. Therefore, if our ultimate demand is only time delay reduction by minimizing interconnected wirelength, FA is no doubt a better choice. However if the runtime requirement is higher, it is recommended to choose ABC.

V. RELATED DISCUSSIONS

At present, most of routing problems are aiming at minimizing interconnected wirelengths, while at the same time considering more and more factors in the routing process such as: obstacle-avoiding routing, minimizing the number of vias, delay and congestion optimization, etc. Chip performance can be further optimized through these routing optimization efforts. Therefore, a distribution of SI techniques per VLSI routing research problem is given in Figure 13, respectively *SMT construction*, *wirelength-driven routing*, *obstacle-avoiding routing*, *timing-driven routing*, *power-driven routing*. As can be seen that most of the routing problems with SI techniques aim at optimizing the wirelength, while the optimization for delay, obstacle and power

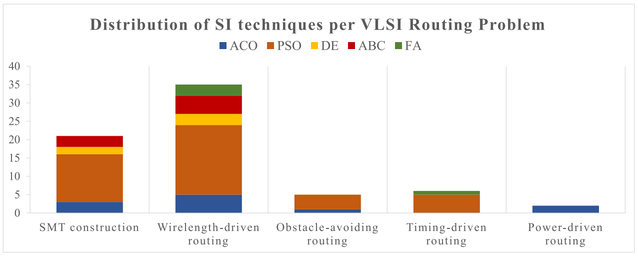


FIGURE 13. Distribution of SI techniques per VLSI routing subproblem in the surveyed works.

consumption is rare. As 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 have built a series of routing models and evaluation models under various new processes, such as multi-layer routing based on X-architecture, global routing for MDSV design, new problem models under the advanced via-pillar technology, etc.

A. MULTI-LAYER ROUTING BASED ON X-ARCHITECTURE

In recent years, more and more researches [55]–[58], [60], [61], [70], [84], [85] have been conducted on X-architecture SMT construction. However, most of the SI algorithms are used to solve the RSMT problem. Because the early global routing algorithms are based on Manhattan architecture, the wirelength and interconnection delay are optimized by optimizing the Steiner tree topology, transforming the wire width, and inserting buffers, thereby helping to upgrade performance of the chip. Moreover, when the global routing based on Manhattan architecture optimizes the interconnect length, its optimization ability is limited since the routing direction can only be horizontal and vertical. Therefore, it is necessary to change the traditional Manhattan architecture from the root of the problem. The proposed X-architecture model makes up for the limitation of the RST routing directions, which can make full use of the routing area and interconnection resources. So that it can further reduce the chip area, shorten the wirelength, reduce power consumption, and make many performance indicators in the physical design get promoted. At present, most of the research works on solving non-Manhattan architecture Steiner trees are based on exact algorithms [86], [87] and traditional heuristic algorithms [30], [31]. The time complexity of the exact algorithms grows exponentially with the size of the problem, while the traditional heuristic algorithms are mostly based on greedy strategies and are prone to local extrema. And when constructing Steiner trees, they do not make full use of the geometric properties of non-Manhattan architecture, and cannot guarantee the quality of Steiner trees. Therefore, it is of great practical significance to explore routing algorithms based on various SI techniques. At present, some works [55]–[58], [60], [61] have applied PSO to X-architecture routing and achieved good results.

As IC design enters the nano-field, the number of routing metal layers continues to increase, making multi-layer global routing face a huge challenge. However, the research work on non-Manhattan architecture mainly focuses on the construction of Steiner tree, and there is a lack of research on multi-layer global routing algorithms. The current research status at home and abroad [88]–[90] shows that although the non-Manhattan structure can effectively reduce the wirelength and chip area, greatly improve the performance and density of chips, but the cost of the number of vias are increased at the same time. In multi-layer routing, the number and size of vias are key optimization goals. Therefore, constructing an efficient multi-layer global router under the non-Manhattan architecture is an important theoretical and practical work.

B. MDSV DESIGNS FOR GLOBAL ROUTING

With the development of process technologies, nanoscale Complementary Metal-Oxide-Semiconductor (CMOS) circuits have experienced a dramatic increase in transistor density over the past three decades, resulting in increased density of power consumption in the circuit. According to related research, the power density on the microprocessor increases at twice the rate every three years. Such high-density power consumption will cause the temperature of the wafer to overheat, which will reduce the reliability of the circuit. Therefore, the problem of power consumption has to be taken seriously. However, most current routing algorithms are proposed to reduce the number of overflows, wirelength and computation time, and all functions operate in the same voltage mode. This traditional voltage supply mode can easily cause excessive unnecessary power consumption. This is because all of the functions of the chip operate in the same high-voltage mode, while other devices that can operate in the lower-voltage mode also operate at high voltages, increasing the power consumption of the chip, thereby reducing battery life. In order to change this voltage supply model, industry chip companies and researchers have proposed Multiple Supply Voltage (MSV) design modes that can control the voltage of different functional components to effectively reduce power consumption through complex control strategies. So multi-voltage design is widely used in advanced applications or low power applications. Compared to MSV designs, MDSV technology can further reduce power consumption. In the MDSV design, the voltage of each power domain can be dynamically changed according to the corresponding power modes. In some power modes, such as standby mode and sleep mode, some power domain can even be set to completely off to save power.

The introduction of MDSV brings new opportunities and challenges to the physical design of VLSI circuits, and causes the update of the entire layout domain algorithms. It puts forward new requirements for the physical design process with MDSV as the voltage supply model including layout planning and placement, routing, parameters extraction, etc. It also poses a great challenge to the study of EDA

(Electronics Design Automation) tools, which is closely related to the global routing. Reference [91] is the first time to build a corresponding global router for a MDSV design mode. However, this work has oversimplified its mathematical problems, and failed to conduct corresponding experimental research from the goal of reducing power. In the MDSV design mode, a net may pass through more than one power domain, some of which may be off and the other power domains are still in active modes. For an active net that has long-distance routing in a closed power domain, a functional conflict may occur if its repeater is placed in a closed power domain. Therefore, limiting the routing length of the active net in the closed power domain is an extremely important global routing problem in the MDSV design. In [92], a length-limited maze routing algorithm is proposed to control the wirelength of the routing path. At the same time, [93] also proposes a fast SMT construction algorithm that considers the limit of the routable wirelength within obstacles. But none of them considers that the length constraints under different power domains are different. Therefore, the two global routers [92], [93] may still produce some illegal routing results under the MDSV design mode.

The existing research work on the MDSV design mode mainly focuses on some local stages such as layout and clock tree construction, and lack an effective and complete solution of global routing under the MDSV design. If the novel MDSV is introduced into global routing phase, new global routing issues will arise, including new constraints and mathematical models. It is necessary to propose more effective algorithms. In the research of our team and the industry companies including Cadence Design Systems in the US, we found that the global routing algorithms under MDSV design can effectively reduce the large amount of power consumption, but the current academic community lacks research on the global routing under MDSV design mode. In fact, in order to solve the dynamic power problem, scholars have also developed some new tools based on MSV and other related technologies. However, in these tools, the relevant routing algorithms need to be changed to solve the routing problems in the new design mode. Therefore, it is of great theoretical value and practical significance to find an effective global routing algorithm under the MDSV design and construct an efficient low-power global router.

C. MULTILAYER ROUTING UNDER ADVANCED VIA-PILLAR PROCESS

Under current manufacturing processes, interconnect delay has exceeded gate delay and is a major factor in determining circuit performance. The interconnect delay is mainly optimized during the routing phase. Therefore, it is necessary to consider more performance optimization targets such as delay in routing process in order to optimize the performance of the chip. With the development of process technology, the resistance of metal wires and vias increases exponentially, which brings more stringent constraints to traditional routing-related algorithms. It leads to the existing methods being prone to excessive delay in the solution process, and increases the difficulty of timing convergence, thereby seriously affecting the performance of chips. To this end, Synopsys and TSMC (Taiwan Semiconductor Manufacturing Company) jointly launched the key process of via-pillar in 2017 as a representative technology for chip performance in designs of 7 nm and below [94]. The via-pillar process is based on the fundamental problem of high resistance affecting the delay problem, which can greatly reduce the resistance of the metal wires and vias, thereby greatly providing the optimization of the delay and optimizing the overall performance of chips.

After introducing the via-pillar process, many problems in the traditional routing stage need to be updated, including layer assignment, track assignment and detailed routing. 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, and delay optimization issues. However, the research works on the traditional routing problems mostly regard the vias as having no shape and size. The size and position of the via 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 increased greatly. At the same time, the industry still adopts manual implementation for some of the related routing problems under the via-pillar process, and there is still lacking an effective automated design process. Therefore, it is of great theoretical and practical significance to seek an effective and complete automated routing algorithm under the via-pillar process to construct an efficient performance-driven multi-layer router.

1) LAYER ASSIGNMENT

In recent years, the work of layer assignment not only aims at minimizing the number of vias, but further attempts to reduce the delay [95], [96]. However, these works with considerations for delay optimization do not take into account the size of vias. It is assumed that the vias do not occupy routing resources and thus do not affect the routing of metal wires, which will seriously affect the accuracy of the final global routing results guiding for detailed routing. Existing research works on layer assignment either do not consider the delay problem, or the via size, which are 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 algorithms are no longer applicable to the problems under via-pillar process. And it is necessary to design a corresponding effective algorithm.

2) TRACK ASSIGNMENT

In order to speed up the detailed routing and enhance the accuracy of global routing guidance performance, track assignment is added between global and detailed routing. However, the existing research works on track assignment problems do not consider the local nets, or the location of vias, or time delay, which will further aggravate the mismatch of global routing to the detailed routing. Reference [97] applies DPSO on track assignment problem, considering local nets, overlapped conflict, wirelength and blockages. And compared with the negotiation-based optimization algorithm [98], it achieves less overlap cost. Thus, it can be seen that applying SI techniques to track assignment problem is a wise approach. For track assignment under the via-pillar process, it is necessary to re-establish a new model considering the via location, delay optimization, and avoiding open, while further reducing the total overlap cost, wirelength and delay. In order to effectively improve the global routing solution's guiding role for detailed routing, it is very important to construct an effective, high-quality, and easily parallelized track assignment algorithm.

3) DETAILED ROUTING

Detailed routing methods can be divided into serial algorithms and parallel algorithms. In order to make up for the effect of the order of nets on global results, the serial routing usually refines the routing results by means of rip-up and reroute, and can quickly obtain a good quality solution. But these serial routing methods [99], [100] may generate some invalid solutions, while the parallel detailed routing algorithms based on the multi-commodity flow model [44] can simultaneously route multiple nets. However, these algorithms do not take into account the delay optimization problem and the complexity is too high, which cannot solve the performance-driven routing problem well. While the introduction of via-pillar process can effectively optimize the delay. Therefore, finding an efficient algorithm for detailed routing under the via-pillar process is also a problem worth exploring.

VI. PROSPECTS OF FUTURE STUDY

As one of the emerging optimization methods in recent years, SI is becoming one of the most concerned optimization areas. The simulation of the intelligent behaviors exhibited by natural organisms provides new ideas and methods for solving various complex optimization problems. The five SI techniques mentioned above have been widely used in various fields of VLSI, especially in routing problems. Therefore, the effectiveness and efficiency of SI in the routing filed deserve scholars to further explore its potential.

A. ROUTING FOR ADVANCED TECHNOLOGY MODELS USING SI

With the advent of various new processes and new technologies in the field of IC, more possibilities are provided for the optimization of routing problems. Figure 14 portrays the frequency of published SI-based works for each surveyed VLSI routing subproblem per year. To a certain extent, this survey reflects the research trend of VLSI routing problems in recent years. It can be seen that since 2009, scholars have begun to focus on optimizing the time delay and power consumption in routing. And the highest paper output was achieved in 2015, while in the past four years, there have been few studies on solving routing problems by SI techniques. So the appearance of new processes brings new research prospect for routing field, which can further optimize the wirelength, time delay, power consumption and other important objectives, so as to improve the overall performance of the chip.

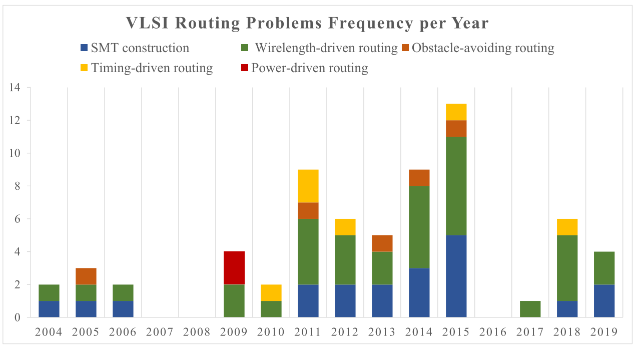


FIGURE 14. Yearly distribution of VLSI Routing subproblems solved by SI techniques in the surveyed works.

1) USE SI TO SOLVE NON-MANHATTAN ARCHITECTURE ROUTING

From the application of SI in the routing problems in Section IV, most of routing algorithms are based on grid graph. However, in the multi-layer routing model, there are many kinds of routing directions of some metal layers. If the original methods are directly applied to non-Manhattan architecture routing problems, the algorithms will become more complicated and have certain limitations. Therefore, non-Manhattan architecture and the grid routing model are not a good combination. Especially in the global routing phase, it is necessary to rethink some essential problems like the cost function of the routing edges, allocation of routing resource, etc. In order to better solve non-Manhattan construction routing problem of VLSI, it is of great significance to explore effective strategies on the SMT construction based on non-Manhattan architecture, rerouting of the congested area, delay optimization, and the construction of layer assignment algorithm.

2) USE SI TO SOLVE LOW-POWER GLOBAL ROUTING IN MDSV DESIGN MODE

Compared with a single voltage mode, the global routing in MSV design mode are more complicated, either facing a computational explosion or falling into local extremum and not being able to approach the global optimal solution, so that people are beginning to seek various heuristic algorithms.

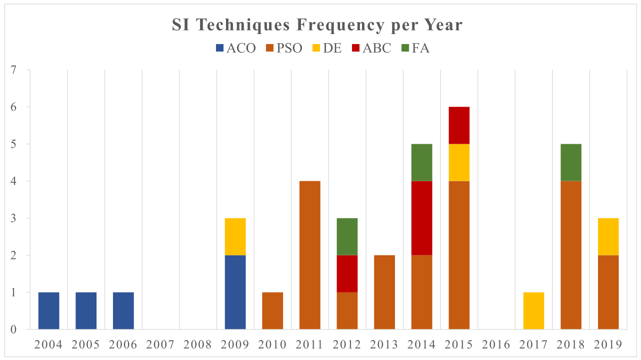


FIGURE 15. Yearly distribution of the SI techniques applied to VLSI routing subproblems in the surveyed works.

A large number of research results show that the SI algorithm is indeed a powerful optimization tool and has strong vitality in solving various NP-hard problems. Considering the use of SI techniques to construct an effective strategy for solving the low-power global routing problem in MDSV design mode, to a certain extent, can promote the construction of an efficient VLSI global router in a MDSV chip design environment.

3) USE SI TO SOLVE ROUTING UNDER THE VIA-PILLAR PROCESS

For performance-driven multi-layer routing under the complex via-pillar process, SI techniques are used reasonably and efficiently to solve problems such as layer assignment, track assignment and detailed routing. For example, in the track assignment stage, it is a smart idea to find the optimal assignment scheme by establishing a multi-objective SI optimization framework when considering the location of vias, local nets and delay.

B. EXPLORE NEW AND AVAILABLE SI TECHNIQUE

The exploration and exploitation abilities of the SI algorithms are well played in the VLSI routing problems, and have contributed greatly to improving the quality and runtime of the routing algorithms. Among the above-mentioned SI techniques, PSO is simple and easy to implement, and has strong global optimization ability. Figure 15 gives a yearly distribution of the SI techniques applied to VLSI routing subproblems in surveyed works. It shows that PSO has become the most widely used SI technique in surveyed VLSI routing problems since 2009. Since this technology was proposed, it has attracted great attention from many scholars. In just a few years, a research hotspot has been formed and a large number of research results have been obtained. In recent years, PSO has been widely used in VLSI *partitioning* [101], *floorplanning* and *placement* [81], [102], *routing* [51]–[58] and other fields [103], [104]. Moreover, in our preliminary work [29], [55]–[61], [67], [68], [70], [80], [81], [97], PSO has also been well applied to solve related problems in VLSI physical design. A large number of experimental results show that the PSO algorithm is indeed a beneficial optimization tool and has strong vitality. Therefore, exploring new and available SI techniques is also one of the research directions for future VLSI routing problems.

1) QUANTUM PARTICLE SWARM OPTIMIZATION (QPSO)

In the PSO algorithm, the motion state of a particle is described by position and velocity. With the evolution of time, the motion track of the particle is fixed. At the same time, the velocity of the particle is limited, so that the search space is a finite and gradually decreasing region, and global convergence cannot be guaranteed. Therefore, QPSO based on δ-potential well model [105] is proposed and becomes the most widely researched and used version. In QPSO, all particles have quantum behavioral properties and no longer follow the Newtonian equations for the motion state of particles in PSO, which is a deterministic description. Compared with PSO, QPSO uses a simpler position-only moving model with fewer control parameters. More importantly, particles of a quantum system can appear anywhere at a certain probability distribution to achieve global search.

In recent years, QPSO has been successfully applied in various fields such as neural network [106], power systems [107], electronics and electromagnetics [108]. Therefore, QPSO has an opportunity to better solve VLSI routing problems that PSO can solve.

2) CPSO

The traditional PSO needs a mass of particles to deal with high-dimensional problems, so that the calculation is quite complicated and it is difficult to get a satisfactory solution. The general approach to solving high-dimensional problems is to adopt a divide-and-conquer strategy. Early CPSO algorithms are also based on this strategy. The CPSO proposed by Van den Bergh and Engelbrecht [20] decomposes the input vector into multiple sub-vectors and performs PSO for each sub-vector, and then integrates the search results into a global swarm. In 2005, Niu *et al.* [109] introduced the master-slave mode into PSO. A swarm is formed from one master swarm and multiple slave swarms. The slave swarms execute PSO (or its variants) independently to maintain the diversity of particles, while the master swarm enhances its particles based on its own knowledge and also the knowledge of the particles in the slave swarms. Reference [110] combines random grouping and adaptive weighting, and proposes Cooperatively Coevolving Particle Swarm Optimization (CCPSO), which can perform reasonably well with only a small number of evaluations. Later based on CCPSO, they adopted a new PSO position update rule that relies on Cauchy and Gaussian distributions to sample new points in the search space, and a scheme to dynamically determine the coevolving subcomponent sizes of the variables [111]. This method has become a very competitive method to solve complex multimodal optimization problems.

PSO based on collaborative strategy is a powerful way to deal with high-dimensional problems. Parallel execution of multiple populations can speed up the efficiency.

And information exchange in subgroups can balance exploration and exploitation, reducing the risk of premature convergence and local optimal. In dealing with complex VLSI routing issues, collaborative PSO has an opportunity to achieve better solutions.

3) HYBRID SI ALGORITHMS

Each SI technique has its own unique properties and will have different effects in different application scenarios. In recent years, in order to improve the performance of various SI techniques, and further improve the quality of the solution, various hybrid SI techniques: PSO-ACO [112]–[114], ACO-ABC [115], [116], DE-ABC [117], [118], FA-DE [119], PSO-DE [120], [121] and other hybrid SI algorithms [122], [123] are proposed and successfully applied in various fields. Reference [112] proposes a two-stage hybrid swarm intelligence optimization algorithm, using the randomicity, rapidity and wholeness of PSO and GA for rough searching; and for detailed searching, they made use of the parallel, positive feedback and high accuracy of solution of ACO. Kefayat *et al.* [115] took the advantages of the global search ability of ABC and the local search ability of ACO to make up for the shortcomings of ACO that are easy to fall into local optimum. In the hybrid DE-ABC algorithm proposed by Yang *et al.* [117], employed bees employ the mutation and crossover strategies of DE to enforce their exploration ability while onlooker bees keep their original updating strategy to retain the exploitation ability. This algorithm improves the convergence speed and searching ability of ABC.

Using a single SI technique is easily affected by the initial population and parameter settings, while mixing different SI technologies, or communicating information through the co-evolution of different types of populations, or learning some mechanisms from other SI techniques can greatly improve the performance of the algorithm and avoid falling into local extremes as much as possible. The above successful cases show that the hybrid SI algorithms have opportunity to be better applied to VLSI routing problems than the single SI technique.

VII. CONCLUSION AND FUTURE WORK

SI techniques can solve a variety of NP-hard problems in VLSI routing including *Steiner tree construction*, *wirelength-driven routing*, *obstacles-avoiding routing*, *timing-driven routing*, *power-driven routing*, etc. In the Steiner tree construction problem, the Manhattan architecture cannot fully utilize the routing resources because of its limited routing orientation, while the non-Manhattan architecture was proposed to solve this problem. And more and more research works are based on non-Manhattan architecture, especially the X-architecture. In this problem, SI techniques are mainly used for the selection of Steiner points, and the wirelength is used as an evaluation index. These evaluation indexes guide the individuals to develop towards the optimal goal in the form of functions. For example, in PSO and DE, they exist in the form of fitness function and in ACO, ABC and FA, they are respectively reflected as pheromone, food sources and intensity. In the obstacles-avoiding routing, presence of obstacles may affect the selection of Steiner points. By designing effective obstacle-avoiding strategies, meanwhile considering wirelength and the cost of routing through obstacles, both the wirelength and congestion are minimized. Timing-driven routing is usually done with a distributed RC network as the interconnected model, using the Elmore delay formula to calculate time delay from source to sink. Power-driven routing, in addition to considering wirelength, as well as the number of vias, capacitance and other constraints. The later study also introduced MDSV to further reduce power consumption.

We investigate five classic and commonly used SI techniques: PSO, DE, ACO, ABC, and FA in the field of VLSI routing. Among them, ACO and PSO have been widely used in most of the routing problems mentioned above, while the remaining SI techniques are used less, most of which are only used in the selection of Steiner points and the routing problem considering the minimum interconnected length. Therefore, we expect to see more available SI techniques that will improve the quality of the solution can be applied to the routing algorithms considering obstacles, delay, and power consumption. Especially the QPSO, collaborative PSO and hybrid SI algorithms, which are potential techniques, are promising for better results.

Furthermore, it is crucial to introduce relevant new techniques and new processes to achieve breakthroughs in the field of VLSI routing, such as the X-architecture introduced to make full use of the routing resources, the MDSV designed to solve the low-power global routing, and the via-pillar process introduced to improve the performance of the multi-layer routing. The traditional routing models are not applicable in these new design backgrounds, and the complexity is too high. Therefore, exploring new routing models under these new technologies and new processes is the development trend of VLSI routing in the future. As a powerful optimization tool, SI will continue to play a significant role in VLSI routing.

REFERENCES

[1] S. V. Kumar, P. Rao, H. Sharath, B. Sachin, U. Ravi, and B. Monica, "Review on vlsi design using optimization and self-adaptive particle swarm optimization," *J. King Saud Univ.-Comput. Inf. Sci.*, to be published. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1319157817302793

[2] 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.

[3] M. Mavrovouniotis, C. Li, and S. Yang, "A survey of swarm intelligence for dynamic optimization: Algorithms and applications," *Swarm Evol. Comput.*, vol. 33, pp. 1–17, Apr. 2017.

[4] Y.-F. Zhu and X.-M. Tang, "Overview of swarm intelligence," in *Proc. Int. Conf. Comput. Appl. Syst. Modeling (ICCASM)*, vol. 9, Oct. 2010, pp. 400–403.

[5] W. Wisittipanich and P. Hengmeechai, "Comparison of PSO and DE for truck scheduling in multi-door cross docking terminals," in *Proc. IEEE Int. Conf. Ind. Eng. Eng. Manage. (IEEM)*, Dec. 2017, pp. 50–54.

[6] Z. Han, Y. Li, and J. Liang, "Numerical improvement for the mechanical performance of bikes based on an intelligent PSO-ABC algorithm and WSN technology," *IEEE Access*, vol. 6, pp. 32890–32898, 2018.

[7] S. S. Sebtahmadi, H. B. Azad, S. H. A. Kaboli, M. D. Islam, and S. Mekhilef, "A PSO-DQ current control scheme for performance enhancement of Z-source matrix converter to drive IM fed by abnormal voltage," *IEEE Trans. Power Electron.*, vol. 33, no. 2, pp. 1666–1681, Feb. 2018.

[8] W. Cao, K. Liu, M. Wu, S. Xu, and J. Zhao, "An improved current control strategy based on particle swarm optimization and steady-state error correction for SAPF," *IEEE Trans. Ind. Appl.*, vol. 55, no. 4, pp. 4268–4274, Jul. 2019.

[9] Y. Li, H. Soleimani, and M. Zohal, "An improved ant colony optimization algorithm for the multi-depot green vehicle routing problem with multiple objectives," *J. Cleaner Prod.*, vol. 227, pp. 1161–1172, Aug. 2019.

[10] F. Yang, P. Wang, Y. Zhang, L. Zheng, and J. Lu, "Survey of swarm intelligence optimization algorithms," in *Proc. IEEE Int. Conf. Unmanned Syst. (ICUS)*, Oct. 2017, pp. 544–549.

[11] M. Dorigo, "Optimization, learning and natural algorithms," Ph.D. dissertation, Politecnico di Milano, Milan, Italy, 1992.

[12] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, Perth, WA, Australia, Nov./Dec. 1995, pp. 1942–1948.

[13] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, 1997.

[14] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Dept. Comput. Eng., Erciyes Univ., Kayseri, Turkey, Tech. Rep. TR06, 2005.

[15] X.-S. Yang, *Nature-Inspired Metaheuristic Algorithms*. London, U.K.: Luniver Press, 2008.

[16] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: Optimization by a colony of cooperating agents," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 26, no. 1, pp. 29–41, Feb. 1996.

[17] K. Socha and M. Dorigo, "Ant colony optimization for continuous domains," *Eur. J. Oper. Res.*, vol. 185, no. 3, pp. 1155–1173, Mar. 2008.

[18] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Proc. Congr. Evol. Comput. (CEC)*, vol. 3, Jan. 2003, pp. 1945–1950.

[19] M. Clerc and J. Kennedy, "The particle swarm—Explosion, stability, and convergence in a multidimensional complex space," *IEEE Trans. Evol. Comput.*, vol. 6, no. 1, pp. 58–73, Feb. 2002.

[20] F. Vandenbergh and A. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 225–239, Jun. 2004.

[21] J. Liang, A. Qin, P. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Trans. Evol. Comput.*, vol. 10, no. 3, pp. 281–295, Jun. 2006.

[22] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 4–31, Feb. 2011.

[23] A. Qin, V. Huang, and P. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 398–417, Apr. 2009.

[24] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (ABC) algorithm," *Appl. Soft Comput.*, vol. 8, no. 1, pp. 687–697, Jan. 2008.

[25] J. Parkhurst, N. Sherwani, S. Maturi, D. Ahrams, and E. Chiprout, "SRC physical design top ten problem," in *Proc. Int. Symp. Phys. Design (ISPD)*, 1999, pp. 55–58.

[26] 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.

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

[28] 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.

[29] G. Liu, G. Chen, W. Guo, and Z. Chen, "DPSO-based rectilinear Steiner minimal tree construction considering bend reduction," in *Proc. 7th Int. Conf. Natural Comput.*, vol. 2, Jul. 2011, pp. 1161–1165.

[30] T. Samanta, P. Ghosal, H. Rahaman, and P. Dasgupta, "A heuristic method for constructing hexagonal Stesiner minimal trees for routing in vlsi," in *Proc. IEEE Int. Symp. Circuits Syst.*, Sep. 2006, pp. 1788–1791.

[31] 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.

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

[33] 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.

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

[35] 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.

[36] 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 Des. Integr. Circuits Syst.*, vol. 29, no. 12, pp. 1931–1944, Dec. 2010.

[37] 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.

[38] 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, pp. 1–21, Mar. 2009.

[39] Z. Cao, T. T. Jing, J. Xiong, Y. Hu, Z. Feng, L. He, and 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.

[40] L. Mcmurchie and C. Ebeling, "PathFinder: A negotiation-based performance-driven router for FPGAs," in *Proc. 3rd Int. ACM Symp. Field-Program. Gate Arrays*, 1995, pp. 111–117.

[41] H. Shin and A. Sangiovanni-Vincentelli, "A detailed router based on incremental routing modifications: Mighty," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 6, no. 6, pp. 942–955, Nov. 1987.

[42] J. Soukup, "Fast maze router," in *Proc. 15th Design Autom. Conf.*, Jun. 1978, pp. 100–102.

[43] Y. Zhang and C. Chu, "Regularroute: An efficient detailed router with regular routing patterns," in *Proc. Int. Symp. Phys. Design*, 2011, pp. 45–52.

[44] X. Jia, Y. Cai, Q. Zhou, and B. Yu, "A multicommodity flow-based detailed router with efficient acceleration techniques," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 1, pp. 217–230, Jan. 2018.

[45] M. Ozdal, "Detailed-routing algorithms for dense pin clusters in integrated circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 28, no. 3, pp. 340–349, Mar. 2009.

[46] Y. Hu, T. Jing, X. Hong, Z. Feng, X. Hu, and G. Yan, "An efficient rectilinear steiner minimum tree algorithm based on ant colony optimization," in *Proc. Int. Conf. Commun., Circuits Syst.*, vol. 2, Oct. 2004, pp. 1276–1280.

[47] Y. Hu, T. Jing, Z. Feng, X.-L. Hong, X.-D. Hu, and G.-Y. Yan, "ACO-Steiner: Ant colony optimization based rectilinear Steiner minimal tree algorithm," *J. Comput. Sci. Technol.*, vol. 21, no. 1, pp. 147–152, Jan. 2006.

[48] 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. (ASP-DAC)*, Jul. 2005, pp. 7–12.

[49] 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.

[50] T. Arora and M. Moses, "Using ant colony optimization for routing in vlsi chips," in *Proc. AIP Conf.*, 2009, vol. 1117, no. 1, pp. 145–156.

[51] C. Dong, G. Wang, Z. Chen, S. Sun, and D. Wang, "A VLSI routing algorithm based on improved DPSO," in *Proc. IEEE Int. Conf. Intell. Comput. Intell. Syst.*, vol. 1, Nov. 2009, pp. 802–805.

[52] A. Khan, S. Laha, and S. K. Sarkar, "A novel particle swarm optimization approach for VLSI routing," in *Proc. 3rd IEEE Int. Adv. Comput. Conf. (IACC)*, Feb. 2013, pp. 258–262.

[53] S. Nath, S. Ghosh, and S. K. Sarkar, "A novel approach to discrete particle swarm optimization for efficient routing in vlsi design," in *Proc. 4th Int. Conf. Rel., INFOCOM Technol. Optim. (ICRITO)(Trends Future Directions)*, Sep. 2015, pp. 1–4.

[54] Y. Shen, Q. Liu, and W. Guo, "Obstacle-avoiding rectilinear Steiner minimum tree construction based on discrete particle swarm optimization," in *Proc. 7th Int. Conf. Natural Comput.*, vol. 4, Jul. 2011, pp. 2179–2183.

[55] G. Liu, G. Chen, and W. Guo, "DPSO based Octagonal Steiner tree algorithm for VLSI routing," in *Proc. IEEE 5th Int. Conf. Adv. Comput. Intell. (ICACI)*, Oct. 2012, pp. 383–387.

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

[57] 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. Des. Autom. Electron. Syst.*, vol. 20, no. 2, pp. 1–28, Mar. 2015.

[58] 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.

[59] 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.

[60] G. Liu, Z. Chen, W. Guo, and G. Chen, "Self-adapting pso algorithm with efficient hybrid transformation strategy for x-architecture steiner minimal tree construction algorithm," (in Chinese), *Pattern Recognit. Artif. Intell.*, vol. 31, no. 5, pp. 398–408, 2018.

[61] 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.*, pp. 1–19, Jun. 2019.

[62] S. Ghosh, S. Nath, R. Biswas, P. Venkateswaran, J. K. Sing, and S. K. Sarkar, "PSO variants and its comparison with firefly algorithm in solving VLSI global routing problem," in *Proc. IEEE Electron Devices Kolkata Conf. (EDKCON)*, Nov. 2018, pp. 513–518.

[63] M. N. Ayob, Z. M. Yusof, A. Adam, A. F. Z. Abidin, I. Ibrahim, Z. Ibrahim, S. Sudin, N. Shaikh-Husin, and M. K. Hani, "A particle swarm optimization approach for routing in VLSI," in *Proc. 2nd Int. Conf. Comput. Intell., Commun. Syst. Netw.*, Jul. 2010, pp. 49–53.

[64] Z. M. Yusof, A. F. Z. Abidin, M. Salam, K. Khalil, J. A. A. Mukred, M. K. Hani, and Z. Ibrahim, "A binary particle swarm optimization approach for buffer insertion in vlsi routing," *Int. J. Innov. Manage., Inf. Prod.*, vol. 2, no. 3, pp. 34–39, 2011.

[65] Z. M. Yusof, T. Z. Hong, A. F. Z. Abidin, M. N. A. Salam, A. Adam, K. Khalil, J. A. A. Mukred, N. Shaikh-Husin, and Z. Ibrahim, "A two-step binary particle swarm optimization approach for routing in VLSI with iterative RLC delay model," in *Proc. 3rd Int. Conf. Comput. Intell., Modelling Simul.*, Sep. 2011, pp. 63–67.

[66] S. Nath, P. P. G. Neogi, J. K. Sing, and S. K. Sarkar, "VLSI routing optimization based on modified constricted PSO with iterative RLC delay model," in *Proc. Int. Conf. Current Trends Towards Converging Technol. (ICCTCT)*, Mar. 2018, pp. 1–7.

[67] 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.

[68] 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, 2015.

[69] S. Manna, T. Chakrabarti, U. Sharma, and S. K. Sarkar, "Efficient VLSI routing optimization employing discrete differential evolution technique," in *Proc. IEEE 2nd Int. Conf. Recent Trends Inf. Syst. (ReTIS)*, Jul. 2015, pp. 461–464.

[70] H. Wu, S. Xu, Z. Zhuang, and G. Liu, "X-architecture Steiner minimal tree construction based on discrete differential evolution," in *Proc. ICNC-FSKD ICHSA*, 2019, pp. 433–442.

[71] K. Pandiaraj, P. Sivakumar, and R. Sridevi, "Minimizationof wirelength in 3D IC routing by using differential evolution algorithm," in *Proc. IEEE Int. Conf. Electr., Instrum. Commun. Eng. (ICEICE)*, Apr. 2017, pp. 1–5.

[72] S. Vijayakumar, J. G. Sudhakar, G. Muthukumar, and T. A. A. Victoire, "A differential evolution algorithm for restrictive channel routing problem in VLSI circuit design," in *Proc. World Congr. Nature Biol. Inspired Comput. (NaBIC)*, Dec. 2009, pp. 1258–1263.

[73] P. Bhattacharya, A. Khan, and S. K. Sarkar, "A global routing optimization scheme based on ABC algorithm," in *Proc. Adv. Comput., Netw. Inform.*, vol. 2. Springer, 2014, pp. 189–197.

[74] H. Zhang and D.-Y. Ye, "Key-node-based local search discrete artificial bee colony algorithm for obstacle-avoiding rectilinear Steiner tree construction," *Neural Comput. Appl.*, vol. 26, no. 4, pp. 875–898, May 2015.

[75] P. Bhattacharya, A. Khan, S. K. Sarkar, and S. Sarkar, "An artificial bee colony optimization based global routing technique," in *Proc. Int. Conf. Control, Instrum., Energy Commun. (CIEC)*, Jan. 2014, pp. 621–625.

[76] H. Zhang and D. Ye, "An artificial bee colony algorithm approach for routing in VLSI," in *Proc. Int. Conf. Swarm Intell.* Springer, 2012, pp. 334–341.

[77] A. Khan, P. Bhattacharya, and S. K. Sarkar, "A swarm based global routing optimization scheme," in *Proc. Int. Conf. Adv. Electr. Eng. (ICAEE)*, Jan. 2014, pp. 1–4.

[78] M. N. Ayob, F. Hassan, A. H. Ismail, H. H. Basri, M. S. Azmi, and A. F. Z. Abidin, "A firefly algorithm approach for routing in VLSI," in *Proc. Int. Symp. Comput. Appl. Ind. Electron. (ISCAIE)*, Dec. 2012, pp. 43–47.

[79] H. Prufer, "Neuer bewis eines satzes uber permutationnen," *Arch. Math. Phys.*, vol. 27, pp. 742–744, 1918.

[80] 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, 2014.

[81] G. Chen, W. Guo, and Y. Chen, "A PSO-based intelligent decision algorithm for vlsi floorplanning," *Soft Comput.*, vol. 14, no. 12, pp. 1329–1337, 2010.

[82] H. Zhou, D. Wong, I.-M. Liu, and A. Aziz, "Simultaneous routing and buffer insertion with restrictions on buffer locations," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 19, no. 7, pp. 819–824, Jul. 2000.

[83] Y.-J. Chang, Y.-T. Lee, and T.-C. Wang, "NTHU-route 2.0: A fast and stable global router," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2008, pp. 338–343.

[84] 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. Des. Autom. Electron. Syst.*, vol. 21, no. 3, pp. 1–31, Apr. 2016.

[85] 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, pp. 1–3, 2017.

[86] A. Thurber and G. Xue, "Computing hexagonal Steiner trees using PCX [for VLSI]," in *Proc. 6th IEEE Int. Conf. Electron., Circuits Syst. (ICECS)*, vol. 1, Jan. 2003, pp. 381–384.

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

[88] E. Hursey, N. Jayakumar, and S. Khatri, "Non-Manhattan routing using a Manhattan router," in *Proc. 18th Int. Conf. VLSI Design Held Jointly 4th Int. Conf. Embedded Syst. Design*, Apr. 2005, pp. 445–450.

[89] C. Chang and Y. Chang, "X-route: An X-architecture full-chip multilevel router," in *Proc. IEEE Int. SOC Conf.*, Sep. 2007, pp. 229–232.

[90] A. R. Agnihotri and P. H. Madden, "Congestion reduction in traditional and new routing architectures," in *Proc. 13th ACM Great Lakes Symp. VLSI*, 2003, pp. 211–214.

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

[92] 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.*, 2010, pp. 200–205.

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

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

[95] 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.

[96] 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.

[97] W. Guo, X. Chen, G. Liu, and G. Chen, "Track assignment algorithm based on hybrid discrete particle swarm optimization," (in Chinese), *Pattern Recognit. Artif. Intell.*, vol. 32, no. 8, pp. 758–770, 2019.

[98] M.-P. Wong, W.-H. Liu, and T.-C. Wang, "Negotiation-based track assignment considering local nets," in *Proc. 21st Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2016, pp. 378–383.

[99] Y. Zhang and C. Chu, "Gdrouter: Interleaved global routing and detailed routing for ultimate routability," in *Proc. 49th Annu. Design Autom. Conf.*, 2012, pp. 597–602.

[100] M. Ahrens, M. Gester, N. Klewinghaus, D. Muller, S. Peyer, C. Schulte, and G. Tellez, "Detailed routing algorithms for advanced technology nodes," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 4, pp. 563–576, Apr. 2015.

[101] A. Prakash and R. K. Lal, "PSO: An approach to multiobjective VLSI partitioning," in *Proc. Int. Conf. Innov. Inf., Embedded Commun. Syst. (ICIIECS)*, Mar. 2015, pp. 1–7.

[102] G. Karimi, H. Akbarpour, and A. Sadeghzadeh, "Multi objective particle swarm optimization based mixed size module placement in VLSI circuit design," *Appl. Math. Inf. Sci.*, vol. 9, no. 3, pp. 1485–1492, 2015.

[103] H. Wang, Y. Jin, and J. Doherty, "Committee-based active learning for surrogate-assisted particle swarm optimization of expensive problems," *IEEE Trans. Cybern.*, vol. 47, no. 9, pp. 2664–2677, Sep. 2017.

[104] L. Zhao and J. Wei, "A new bi-level PSO algorithm based on dynamic constraint processing and approximate navigation," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2019, pp. 1659–1663.

[105] J. Sun, B. Feng, and W. Xu, "Particle swarm optimization with particles having quantum behavior," in *Proc. Congr. Evol. Comput.*, Sep. 2004, pp. 325–331.

[106] G. Das, S. Panda, and S. K. Padhy, "Quantum particle swarm optimization tuned artificial neural network equalizer," in *Proc. Soft Comput., Theories Appl.* Springer, 2018, pp. 579–585.

[107] W. Zhang, W. Shi, and J. Zhuo, "BDI-agent-based quantum-behaved PSO for shipboard power system reconfiguration," *Int. J. Comput. Appl. Technol.*, vol. 55, no. 1, pp. 4–11, 2017.

[108] L. Dos Santos Coelho and P. Alotto, "Global optimization of electromagnetic devices using an exponential quantum-behaved particle swarm optimizer," *IEEE Trans. Magn.*, vol. 44, no. 6, pp. 1074–1077, Jun. 2008.

[109] B. Niu, Y. Zhu, and X. He, "Multi-population cooperative particle swarm optimization," in *Proc. Eur. Conf. Artif. Life* Springer, 2005, pp. 874–883.

[110] X. Li and X. Yao, "Tackling high dimensional nonseparable optimization problems by cooperatively coevolving particle swarms," in *Proc. IEEE Congr. Evol. Comput.*, May 2009, pp. 1546–1553.

[111] X. Li and X. Yao, "Cooperatively coevolving particle swarms for large scale optimization," *IEEE Trans. Evol. Comput.*, vol. 16, no. 2, pp. 210–224, Apr. 2012.

[112] W. Deng, R. Chen, B. He, Y. Liu, L. Yin, and J. Guo, "A novel two-stage hybrid swarm intelligence optimization algorithm and application," *Soft Comput.*, vol. 16, no. 10, pp. 1707–1722, Oct. 2012.

[113] A. M. Aaref, S. Mohammed, A. B. Noori, and A. M. Humada, "A hybrid (ACO-PSO) algorithm based on maximum power point tracking and its performance improvement within shadow conditions," *Int. J. Eng. Technol.*, vol. 7, no. 4.37, pp. 43–47, 2018.

[114] N. Shivaraman and S. Mohan, "A reactive hybrid metaheuristic energy-efficient algorithm for wireless sensor networks," in *Smart Network Inspired Paradigm and Approaches in IoT Applications*. Springer, 2019, pp. 1–13.

[115] M. Kefayat, A. Lashkar Ara, and S. Nabavi Niaki, "A hybrid of ant colony optimization and artificial bee colony algorithm for probabilistic optimal placement and sizing of distributed energy resources," *Energy Convers. Manage.*, vol. 92, pp. 149–161, Mar. 2015.

[116] P. Shunmugapriya and S. Kanmani, "A hybrid algorithm using ant and bee colony optimization for feature selection and classification (AC-ABC Hybrid)," *Swarm Evol. Comput.*, vol. 36, pp. 27–36, Oct. 2017.

[117] J. Yang, W.-T. Li, X.-W. Shi, L. Xin, and J.-F. Yu, "A hybrid ABC-DE algorithm and its application for time-modulated arrays pattern synthesis," *IEEE Trans. Antennas Propag.*, vol. 61, no. 11, pp. 5485–5495, Nov. 2013.

[118] L. D. Long, D.-H. Tran, and P. T. Nguyen, "Hybrid multiple objective evolutionary algorithms for optimising multi-mode time, cost and risk trade-off problem," *Int. J. Comput. Appl. Technol.*, vol. 60, no. 3, pp. 203–214, 2019.

[119] L. Zhang, L. Liu, X.-S. Yang, and Y. Dai, "A novel hybrid firefly algorithm for global optimization," *PLoS ONE*, vol. 11, no. 9, Sep. 2016, Art. no. e0163230.

[120] S. Varshney and M. Mehrotra, "A hybrid particle swarm optimization and differential evolution based test data generation algorithm for data-flow coverage using neighbourhood search strategy," *Informatica*, vol. 42, no. 3, pp. 417–438, 2018.

[121] N. K. Yadav, "Hybridization of particle swarm optimization with differential evolution for solving combined economic emission dispatch model for smart grid," *J. Eng. Res.*, vol. 7, no. 3, 2019.

[122] M. Naz, Z. Iqbal, N. Javaid, Z. Khan, W. Abdul, A. Almogren, and A. Alamri, "Efficient power scheduling in smart homes using hybrid grey wolf differential evolution optimization technique with real time and critical peak pricing schemes," *Energies*, vol. 11, no. 2, p. 384, Feb. 2018.

[123] T. P. Jacob and K. Pradeep, "A multi-objective optimal task scheduling in cloud environment using cuckoo particle swarm optimization," *Wireless Pers. Commun.*, vol. 109, no. 1, pp. 315–331, Nov. 2019.

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.

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.

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 (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. He has published over 300 international journal articles and over 100 international conference papers. Some of his works were published in the IEEE JSAC, the IEEE or ACM Transactions, the ACM Sigcomm Workshop, IEEE INFOCOM, ICDCS, and IPDPS. His research interests include cloud computing, security and dependability, parallel and distributed computing, networks, and optimization theory.

Dr. Xiong 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 in the 28th North American Fuzzy Information Processing Society Annual Conference, in 2009. He is also the Chair of Trusted Cloud Computing Task Force, the 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 about 100 international journals, including the IEEE JSAC, the IEEE SMC (Park: A/B/C), the IEEE Transactions on Communications, the IEEE Transactions on Mobile Computing, and the IEEE Transactions on Parallel and Distributed Systems. He has been serving as the Editor-in-Chief, an Associate Editor, or an Editorial Member for over ten international journals, including as an Associate Editor for the IEEE Transactions on Systems, Man and Cybernetics: Systems and the *Information Science* and the Editor-in-Chief for the *Journal of Internet Technology* and the *Journal of Parallel and Cloud Computing*, and a Guest Editor for over ten international journals, including the *Sensor Journal*, WINET, and MONET.

YARU SU received the B.S. degree in automation from Xiamen University, in 2006, and the Ph.D. degree in pattern recognition and intelligent system from the University of Science and Technology of China, in 2012. She is currently an Assistant Professor with the College of Mathematics and Computer Science, Fuzhou University. Her research interests include machine learning, pattern recognition, and data mining.

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 was a Postdoctoral Fellow with the Institute of Computer Science, National University of Defense and Technology, Changsha, China, in 2007. He is currently a Professor with the Institute of Mathematics and Computer Science, Fuzhou University. His current research interests include VLSI layout design, information security, big data analysis, and computational intelligence. He is a member of ACM.