Applied Soft Computing Journal 104 (2021) 107215

Contents lists available at ScienceDirect

Applied Soft Computing Journal

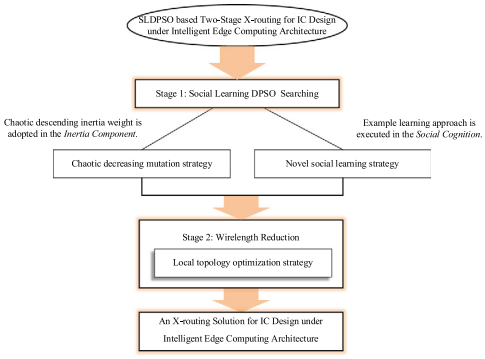
journal homepage: www.elsevier.com/locate/asoc

Social learning discrete Particle Swarm Optimization based two-stage X-routing for IC design under Intelligent Edge Computing architecture ☆

Genggeng Liu [a], Xiaohua Chen [a], Ruping Zhou [a], Saijuan Xu [b,*], Yeh-Cheng Chen [c], Guolong Chen [a]

[a] College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China
[b] Department of Information Engineering, Fujian Business University, Fuzhou, China
[c] Department of Computer Science, University of California, Davis, CA, USA

GRAPHICAL ABSTRACT



ARTICLE INFO

Article history:
Received 1 September 2020
Received in revised form 28 January 2021
Accepted 14 February 2021
Available online 24 February 2021

Keywords:
Intelligent edge computing
Social learning discrete particle swarm optimization
Soft computing
X-routing
Steiner minimum tree
Wirelength

ABSTRACT

One of the core features of Intelligent Edge Computing (IEC) is real-time decision making, therefore low delay is more important for IC design under IEC architecture. And in very large scale integration routing, wirelength is one of the most important indexes affecting the final delay of the IC design. Therefore, this paper introduces the X-routing with more potential for wirelength optimization and the Steiner Minimum Tree (SMT), which is the best routing model in multi-terminal nets. Then, based on Particle Swarm Optimization (PSO) technique which has the strong global optimization ability in Soft Computing, an effective Two-Stage X-routing Steiner minimum tree construction algorithm is proposed. The proposed algorithm is divided into two stages: *social learning discrete PSO searching* and *wirelength reduction*. In the first stage, two excellent strategies are proposed to maintain a good balance between exploration and exploitation capabilities of the PSO technique: (1) Chaotic decreasing inertia weight combined with mutation operator is set to enhance the exploration capability. (2) A new social learning approach combined with crossover operator is designed to ensure the diverse evolution of the swarm while maintaining the exploitation capability. In the second stage, a strategy based on local topology optimization is proposed to further reduce the length of X-routing Steiner tree. Experiments show that the proposed algorithm can achieve the best wirelength optimization and has a strong stability, especially for large-scale SMT problem, so as to better satisfy the demand of low delay of IC design under IEC architecture.

☆ This work was supported in part by National Natural Science Foundation of China (No. 61877010, 11501114), Natural Science Foundation of Fujian Province, China (2019J01243).

* Corresponding author.
E-mail addresses: liugenggeng@fzu.edu.cn (G. Liu), n180320033@fzu.edu.cn (X. Chen), 200320100@fzu.edu.cn (R. Zhou), xusaijuan@fjbu.edu.cn (S. Xu), ycch@ucdavis.edu (Y.-C. Chen), cgl@fzu.edu.cn (G. Chen).

https://doi.org/10.1016/j.asoc.2021.107215
1568-4946/

1. Introduction

In the world of the Internet-of-Things (IoT), a new concept is gaining popularity — Intelligent Edge Computing (IEC). The "edge" refers to the ability of every edge device in the Internet-of-Things to process data, rather than simply storing data in the cloud. IEC is a new computing technique, and one of its core features is real-time decision making and rapid response. Therefore, IC design under Intelligent Edge Computing architecture pays more attention to low delay. [1] finds that deep learning can solve the complex problems in emerging cloud computing architecture more efficiently because of its processing capacity for large-scale data sets. And the deep reinforcement learning method is combined with vehicle edge computing to solve the joint optimization problem of task scheduling and resource allocation in [2]. [3] investigates the neural morphological memory structures that can be used in edge computing in order to improve the ability of data processing under lower power requirements and adapt to the increasing data scale. In [4], the existing edge computing systems are comprehensively summarized, and the technology of deploying deep learning model at the edge is proposed. In Very Large Scale Integration (VLSI) routing, wirelength is the most important index that affects the final delay of the chip. And Steiner Minimum Tree (SMT) is the best connection model in VLSI routing with wirelength as the optimization goal. The SMT problem is to find a routing tree with the least cost to connect all given pins by introducing additional points (Steiner points). Therefore, the SMT construction is a most important issue in VLSI routing.

With the continuous progress and development of VLSI manufacturing processes, the interconnect wire effect has gradually become an important factor affecting the delay of the chip, which finally affects the chip performance. Most current researches on routing algorithms are based on the Manhattan structure [5–9], but these algorithms are limited in the ability to optimize the length of routing tree. The routing model based on Manhattan structure requires that the way of routing between pins can only be horizontal or vertical, which makes it more difficult to get high quality wirelength optimization results in the IC design. Therefore, the research focus of scholars has gradually shifted to non-Manhattan structure that can make full use of routing resources [10], thus it can better satisfy the demand of low delay of IC design under Intelligent Edge Computing architecture.

In recent years, Soft Computing (SC) has received great attention in the field of optimization. It simulates the biochemical processes of intelligent systems in nature, and can obtain low-cost solutions and robustness. By introducing hybrid soft computing technology, [11] reduces the impact of rapid data growth on the efficiency and accuracy of image and video retrieval process. [12] optimizes 3D IC interconnect delay problem with fixed shape constraints in two stages. In [13], aiming at an approximate optimal travel route, an optimal picking plan is constructed by random transposition to obtain a better solution. And the influence of parameter selection and implementation technology on Genetic Algorithm (GA) is investigated to further improve the performance of GA in [14]. Besides, Swarm Intelligence (SI) based algorithms have been gradually incorporated into SC because of their characteristics of both intelligent decision making and evolutionary computing. [15] and [16] made a relevant research on the application of several classic SI techniques in VLSI routing, and pointed out that Particle Swarm Optimization (PSO) with the simplicity and excellent search ability has obvious advantages in optimizing routing algorithms, especially improving the quality of Steiner tree solution. However, since the traditional PSO technique finishes the social learning of particles by learning only from the global best particle, which makes it difficult to jump out once the algorithm falls into a local extreme.

Therefore, this paper is dedicated to improving the PSO technique to make the algorithm well balance exploitation and exploration capabilities, so as to obtain high-performance routing results which are more suitable for low delay requirement in IC design under Intelligent Edge Computing architecture. This paper proposes a Two-Stage X-routing Steiner Minimum tree (TS-XSMT) construction algorithm based on Social Learning Discrete Particle Swarm Optimization (SLDPSO), consisting of the following two stages.

(1) SLDPSO searching

At this stage, the algorithm searches for a satisfactory X-routing Steiner Tree (XST) with a short wirelength through the PSO technique, which is a classical and efficient intelligent technique in Soft Computing. First, the edge-vertex encoding strategy and an effective fitness function are designed to make the algorithm better solve the discrete XSMT construction problem. Second, chaotic descending inertia weight is used to enhance the global search ability of the algorithm. Then a new social learning strategy is proposed to change the single learning method that the particle learns only from individual historically optimal particle (*pbest*) or the optimal particle of the swarm (*gbest*) in each iteration. And with the novel social learning strategy, the particles are continuously changed in each iteration through the learning example pool, which can enhance the diversity of population evolution. Finally, mutation and crossover operators are integrated into the particle update formula to realize the discretization of PSO, thereby constructing the Steiner tree with shortest length.

(2) wirelength reduction

At this stage, an effective strategy based on local topology optimization is used to reduce the length of XST. By continuously adjusting the local optimal structure near each pin, the length of the routing tree is further optimized to obtain the final XSMT. The proposed algorithm can obtain the best routing solution with shortest wirelength compared with similar work, which is helpful to further reduce the delay and satisfy the high-performance requirement of IC design under Intelligent Edge Computing architecture.

The rest of this paper is organized as follows. Section 2 presents the preliminaries. And Section 3 describes the design ideas and implementation of the TS-XSMT algorithm in detail, including *SLDPSO Searching* and *Wirelength Reduction*. In order to verify the good performance of the proposed TS-XSMT, the experimental comparisons of related strategies and algorithms are given in Section 4. Finally, Section 5 concludes this paper. In order to make it easier for reading, Table 1 lists the proper nouns and related technologies mentioned in this paper.

2. Related work

2.1. X-routing

X-routing is a non-Manhattan structure that has been studied by many scholars because it has more routing directions than the traditional Manhattan structure. In addition to horizontal and vertical directions, the X-routing tree can lay out 45° and 135° wires to make full use of routing resources and finally achieve superior topologies.

In order to obtain a shorter wirelength than the Manhattan structure, scholars used precise algorithm [17] to construct a non-Manhattan structure routing tree and can achieve 10% reduction in wirelength compared to the Rectilinear Steiner Minimum Tree (RSMT). However, the complexity of such algorithm is high, so

Table 1
The description of proper nouns and related technologies.

Abbreviation	Full name	Meaning
IC	Integrated Circuit	A circuit that integrates electronic components and the connections between these components through semiconductor technology.
IEC	Intelligent Edge Computing	A new computing technique which uses edge devices to process and analyze data.
IoT	Internet-of-Things	A huge network combines various information sensing devices with the Internet.
SC	Soft Computing	A new calculation method to solve the precise or imprecise modeling problem through low precision calculation.
SI	Swarm Intelligence	The swarm shows intelligent behavior by the simple cooperation among many non-intelligent individuals.
VLSI	Very Large Scale Integration	An integrated circuit which integrates thousands of transistors into a single chip.
PSP	Pseudo-Steiner Point	Additional points are added to the Steiner tree to build the XSMT.
SMT	Steiner Minimum Tree	A generated tree that additional points are allowed to be added besides the given points to minimize the cost.
RSMT	Rectilinear Steiner Tree	A Steiner minimum tree that can only route in 0° and 90° directions.
X-routing	–	A kind of non-Manhattan structure that the routing direction are not only 0° and 90°, but also 45° and 135°.
XST	X-routing Steiner Tree	A Steiner tree with X-routing.
XSMT	X-routing Steiner Minimum Tree	A Steiner minimum tree with X-routing.

some heuristic algorithms are proposed to solve larger-scale SMT problems. Chiang et al. proposed an X-routing Steiner tree construction algorithm with complexity $O(|V|+|E|)$ in [18]. Based on greedy strategy, edge replacement technology and triangular contraction technology are proposed to construct X-routing Steiner Tree in [19]. Yan [20] proposed an improved algorithm for delay-driven octagonal Steiner tree construction, which effectively reduced the delay of the longest path by adjusting the Y-shaped routing that the longest path passed. Tu et al. [21] proposed a timing-driven routing algorithm to balance the wirelength and timing, which is based on the properties of the shortest path tree and the minimum spanning tree. And X-routing is also introduced into multi-layer routing problem. The first multi-layer architecture using X-routing was proposed by Ho et al. [22]. They studied the optimal routing of the three-terminal nets on X-routing, and developed a general XSMT algorithm based on delaunay triangulation method. Wu et al. [23] proposed an X-routing Steiner Minimum Tree algorithm based on Discrete Differential Evolution (DDE-XSMT) and designed new crossover and mutation operators to obtain high-quality solutions. In addition to the optimization goal of wirelength, [24] and [25] consider the routing tree problem with obstacles and study the obstacle-avoiding Steiner tree algorithm under the non-Manhattan structure. In [26], with time delay as the optimization goal and considering bend reduction, a timing-driven X-routing Steiner tree algorithm is proposed.

The above research shows that X-routing can obtain a better solution than traditional Manhattan structure routing.

2.2. Intelligent edge computing

Edge Computing provides mobile and IoT app vendors with a new distributed computing paradigm that can deploy apps at hired edge servers distributed near users at the edge of the cloud, so as to promote the development of mobile terminal [27]. For the Internet-of-Things, the data procession of traditional cloud computing is completed in the cloud. However, [28,29] and [30] find that public key encryption as a useful technology in some "storage and service" cloud has important security risks such as data security and authenticity. The breakthrough in Edge Computing means that much of the control will be done on local devices rather than in the cloud, and the processing will be done at the local Edge Computing layer. Ngoko et al. [31] introduced an Edge Computing platform for the design of smart-buildings, which is based on a new model of servers that also serve as heaters. Ren et al. [32] investigated the collaboration between Cloud Computing and Edge Computing to effectively overcome the deficiencies of network congestion and long latency in cloud computing systems. Sangaiah et al. [33] proposed a method for conserving position confidentiality of roaming PBSs users using machine learning techniques, and a mobile edge computing service policy is followed in the proposed paradigm to ensure the timely delivery of PBSs. And in [34], a blockchain-enabled distributed security framework using edge cloud and Software-Defined Networking (SDN) is presented, which can efficiently and effectively meet the data confidentiality challenges introduced by the integration of blockchain, edge cloud and SDN paradigm.

Intelligent Edge Computing is a technology developed under the background of high bandwidth, time sensitive and IoT integration. By extending the computing resources, network bandwidth and storage capacity of cloud platform to the IOT edge, [35] enables intelligent robot factory to provide personalized and highly flexible emotional recognition and interactive experience. In order to solve the problem of resource management in edge computing, Wang et al. [36] proposed that cognitive agents use caching strategy to improve cache efficiency and reduce resource waste. [37] uses reinforcement learning based state action reward state action algorithm to make the best unloading decision and reduce the system cost to the greatest extent. And the main features of IEC are lower delay, higher security and lower power consumption, where low delay is an important feature of IEC. Therefore, IC design under Intelligent Edge Computing architecture pays more attention to low delay. In VLSI routing, wirelength is one of the most important indexes affecting the final delay of the chip, so it is necessary to study how to reduce the wirelength of the IC design under IEC architecture more effectively.

In this paper, the X-routing with more potential for wirelength optimization is introduced. And based on the best routing model SMT for multi-terminal nets, a routing algorithm of IC design under Intelligent Edge Computing architecture is constructed. Thus, the low delay requirement of Intelligent Edge Computing can be satisfied more effectively.

2.3. Soft computing

For large-scale problems, it is an arduous task to obtain better quality solutions with lower complexity. [38] proposes a series of novel mapping and optimization techniques to obtain low complexity and high performance. [39] proposes a new community detection algorithm based on influence nodes which can ensure the quality and efficiency of the algorithm. Soft Computing

Table 2
The comparison between PSO and other SI techniques on important key parameters.

SI techniques	Key parameters
ACO	Pheromone factor, heuristic factor, evaporation rate
DE	Scaling factor, crossover rate
ABC	Employed foragers, the threshold that a employed bee role is transformed into a scout bee
BEA	Inward force coefficient, attractive force coefficient, viscous coefficient
MA	Selected operator, crossover operator, mutation operator and local search operator
PSO/SLDPSO	Inertia weight, acceleration coefficients

simulates the biochemical processes of intelligent systems in nature, and can obtain low-cost solutions and robustness through fault tolerance for uncertainty, imprecision and incomplete truth values. Therefore, many researchers use SC to solve large-scale problems. [40] designs a hybrid SC model and predicts wind energy more accurately with lower cost.

As a typical representative technique of Soft Computing, SI algorithms have the characteristics of natural distribution and self-organizing characteristic [41]. And many common SI techniques, such as Ant Colony Optimization (ACO), Differential Evolution (DE), Artificial Bee Colony (ABC), Bacterial Evolutionary Algorithm (BEA), Memetic Algorithm (MA), PSO and others, have been widely applied in VLSI routing. Arora et al. [42,43] used ACO technology to construct Steiner trees and applied them to the global routing of Manhattan and non-Manhattan structures, respectively. And [44] and [45] use DE for 3D routing and restrictive channel routing, respectively. Bhattacharya et al. [46] proposed a ABC algorithm for solving the routing optimization problem, showing noteworthy improvements in reduction of the total interconnected length. Ayob et al. [47] applied firefly algorithm on VLSI routing to find minimum time delay by choosing the path and placing the buffers intelligently. And some work uses PSO technique to construct XSMT. [48] utilizes discrete PSO technique to construct an octagonal Steiner tree to short the wirelength. In [49], a hybrid transformation strategy is proposed to expand the search space based on self-adapting PSO. And an unified algorithm for constructing RSMT and XSMT is proposed in [50]. This algorithm can obtain multiple topologies of SMT, which is conducive to optimizing the congestion in global routing. Among them, PSO is favored because of its superior global search capability.

A large number of studies show that SC-based swarm intelligent algorithms, especially PSO can obtain the high-quality solution. Table 2 lists the key parameters of PSO and other SC-based SI technologies, where the SLDPSO method proposed in this paper has the same parameters as PSO. Besides, since PSO algorithm is easier to implement with high computational efficiency, it is utilized to better solve the X-routing SMT problem.

3. Preliminaries

3.1. Problem formulation

The SMT construction based on X-routing is more complicated than the traditional RSMT problem. Traditional RSMT has only horizontal and vertical routing directions. While the X-routing belongs to non-Manhattan structure, it can also route in 45° and 135° directions in addition to the horizontal and vertical directions. The XSMT problem can be described as follows: Given a set of pins $P = \{P_1, P_2, \ldots, P_n\}$, each pin is represented by a coordinate pair (x_i, y_i). Then connect all pins in P through some Steiner points to construct an XSMT. Taking a routing net with 8 pins as an example, Table 3 shows the input information of the pins. The layout distribution of the given pins is shown in Fig. 1.

Table 3
The input information for the pins of a net.

P_i	1	2	3	4	5	6	7	8
x_i	10	42	4	20	7	32	19	24
y_i	22	31	59	42	47	18	8	5

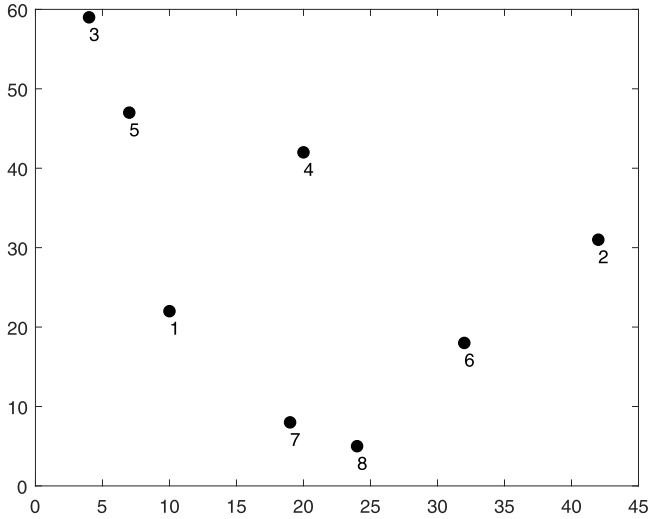


Fig. 1. The layout distribution of the given pins in Table 3.

3.2. Definitions

Definition 1 (*Pseudo-Steiner Point*)**.** In addition to original points formed by given pins, the final XMST can be constructed by introducing additional points called pseudo-Steiner points (PSP). In Fig. 2, the point S is PSP, and PSP contains the Steiner point.

Definition 2 (*Choice 0 (as Shown in Fig. 2(b))*)**.** The Choice 0 of PSP corresponding to edge L is defined as leading rectilinear side first from A to PSP S, and then leading non-rectilinear side to B.

Definition 3 (*Choice 1 (as Shown in Fig. 2(c))*)**.** The Choice 1 of PSP corresponding to edge L is defined as leading non-rectilinear side first from A to PSP S, and then leading rectilinear side to B.

Definition 4 (*Choice 2 (as Shown in Fig. 2(d))*)**.** The Choice 2 of PSP corresponding to edge L is defined as leading vertical side first from A to PSP S, and then leading horizontal side to B.

Definition 5 (*Choice 3 (as Shown in Fig. 2(e))*)**.** The Choice 3 of PSP corresponding to edge L is defined as leading horizontal side first from A to PSP S, and then leading vertical side to B.

3.3. Standard PSO

Particle swarm optimization is one of the typical representatives of SI algorithms. The main idea of PSO comes from the study of clustering behavior of birds, and it utilizes the characteristics of habitat to guide decision-making process. Since the PSO algorithm was proposed, due to its simple calculation, easy implementation, and few control parameters, it has been widely utilized in various fields. And it has achieved a wealth of theoretical analysis, performance improvement, and application of the algorithm. In the PSO algorithm, the particle updates its position by constantly learning the optimal historical experience of the individual and the optimal particle of the population, and no particle will be discarded throughout the iteration. Therefore, it has

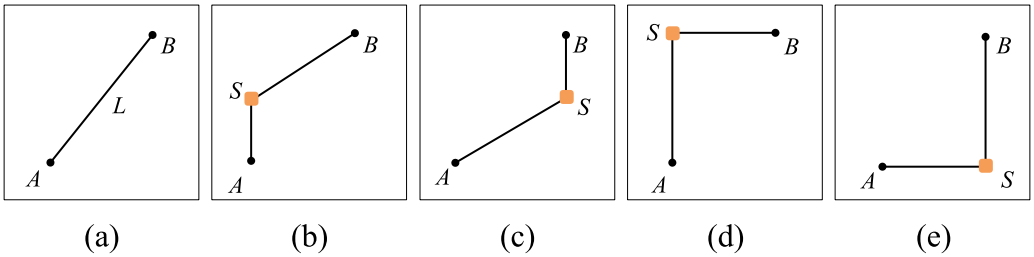


Fig. 2. Four choices of Steiner point for the given segment: (a) Line segment L; (b) Choice 0; (c) Choice 1; (d) Choice 2; (e) Choice 3.

superior global search capability and can maintain the diversity of population better, which is more conducive to avoid premature convergence. Eqs. (1) and (2) give the speed and position update formulas of the Standard Particle Swarm Optimization (SPSO) [51] (Consider a *D*-dimensional minimization problem):

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

$$X_{ij}^{t+1} = V_{ij}^{t+1} + X_{ij}^{t} \tag{2}$$

where $1 \le i \le M, 1 \le j \le D$, M is the population size, w is the inertia weight, c_1 and c_2 are acceleration coefficients. 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:

$$P_i^{t} = \begin{cases} X_i^{t}, \, if \; f(X_i^{t}) < f(P_i^{t}) \\ P_i^{t-1}, \, if \; f(X_i^{t}) \ge f(P_i^{t}) \end{cases} \tag{3}$$

$$G^{t} = P_g^{t}, g = \arg \min_{1 \le i \le M} [f(P_i^{t})] \tag{4}$$

In Eq. (1), $w \cdot V_{ij}^{t}$ is called inertia component, which determines the probability that the particle remains previous velocity. The *individual cognition* $c_1 \cdot r_1 \cdot (P_{ij}^{t} - X_{ij}^{t})$ reflects the process of continuous learning of particles from their own experience. And $c_2 \cdot r_2 \cdot (G_j^{t} - X_{ij}^{t})$ is called *social cognition*, which embodies information sharing and cooperation between particles. *Inertia component* determines the extent to which the particle explores unknown areas in the solution space, and reflects the exploration ability; *individual cognition* and *social cognition* enables particles to continuously search near the *pbest* and *gbest*, reflecting exploitation of the algorithm.

4. Details of SLDPSO based Two-Stage X-routing for IC design under IEC architecture

The Two-Stage XSMT construction algorithm proposed in this paper first finds a satisfactory XST with a short wirelength through *SLDPSO Searching*. At this stage, the chaotic descending mutation strategy and a novel social learning strategy combined with the crossover operator are added to the PSO to improve the search efficiency. And it can balance the exploitation and exploration capabilities of the algorithm. Then adjust the local optimal structure of the Steiner tree through *Wirelength Reduction* to get the final XSMT. The above framework of TS-XSMT is shown in Fig. 3.

4.1. SLDPSO searching

The XST problem is a discrete problem, so the particle update method of SPSO in Section 2 is no longer applicable to this problem. For this reason, the XSMT problem needs to be re-encoded, and the fitness value function is designed to optimize the wirelength. At the same time, the mutation and crossover operators are integrated into the particle update formula to complete the discretization of the PSO. Based on chaotic search and a novel social learning approach, this section proposes a social learning discrete particle swarm optimization algorithm for XSMT problems (SLDPSO-XSMT).

4.1.1. Particle encoding

Liu et al. [50] analyzed two commonly used encoding methods of spanning tree: Prufer number encoding and edge-vertex encoding, and proved that compared with Prufer number encoding, edge-vertex encoding is more suitable for evolutionary algorithms, especially PSO. And it can retain the optimal topological information of particles during the iterative process. Therefore, the following edge-vertex encoding strategy is adopted in this paper.

Representing a candidate Steiner tree needs to contain all the edge information of the spanning tree and the PSP selection for each edge (shown in Definitions 2–5).

Given a net with n pins, the corresponding routing tree of this net has n-1 edges and one extra digit is the fitness of particle. For each three digits, the first two digits represent one edge that include two vertices and the last digit indicates the choice of PSP. Thus the length of one particle is $3 \times (n-1) + 1$. For example, the encoding of one routing tree (n = 8) shown in Fig. 4 can be expressed as follows:

3 5 *0* 5 4 *1* 5 1 *1* 1 7 *0* 7 8 *3* 8 6 *0* 6 2 *1* **106.497**

where the length of the particle is $3 \times (8-1) + 1 = 22$, and the fitness value is 106.497. In this numeric string, each italic number is the choice of PSP of corresponding edge. The first substring (3, 5, *0*) represents that Pin 3 and Pin 5 of the spanning tree in Fig. 4 are connected with Choice 0.

4.1.2. Fitness function

Definition 6. The length of an X-routing Steiner tree is the sum of the length of all the edge segments in the tree, which is calculated as follows:

$$L(T_x) = \sum_{e_i \in T_x} l(e_i) \tag{5}$$

where $l(e_i)$ represents the length of each segment e_i in the tree T_x.

The specific calculation process is as follows: (1) Divide all edge segments of the routing tree into four types: horizontal edge segments, vertical edge segments, 45° edge segments and 135° edge segments; (2) Rotate the 45-degree edge segments clockwise into the horizontal side, and rotate the 135-degree edge segments clockwise into the vertical side; (3) Each horizontal edge segment is arranged from bottom to top and then from left to right, according to the left terminal of this edge. And each vertical edge segment is arranged from left to right, and then from bottom to top, according to the bottom terminal of this edge; (4)

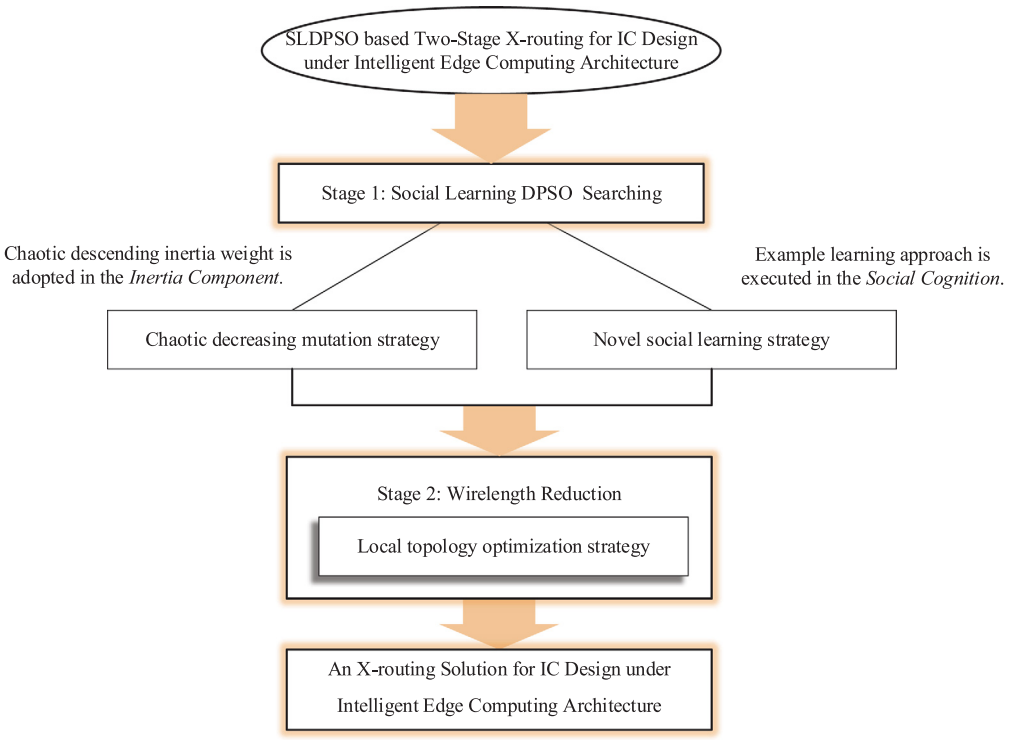


Fig. 3. The framework of TS-XSMT algorithm.

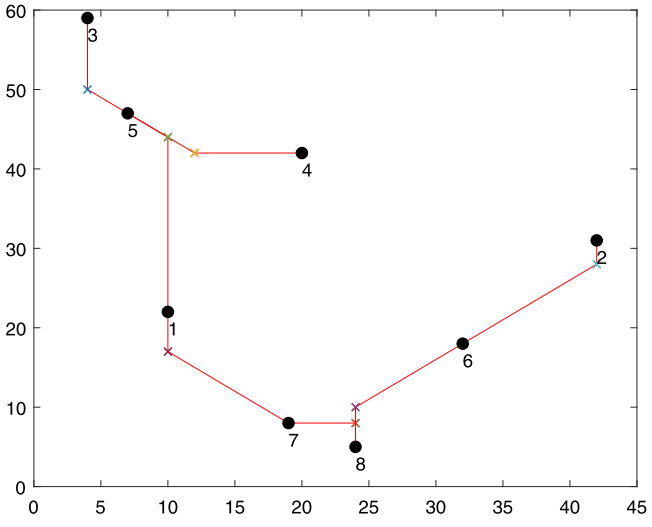


Fig. 4. A routing tree with eight pins given in Table 3.

The length of X-routing Steiner tree is the sum of the total length of these edges excluding overlap segments.

Therefore, aiming at the optimization goal of wirelength, the following fitness function is designed in this paper.

$$fitness = \frac{1}{L(T_x)+1} \times 10000 \tag{6}$$

4.1.3. Particle update formula

(1) Particle Update Formula for DPSO-XSMT

Based on SPSO and the mutation and crossover operators of genetic algorithm, we proposed a Discrete PSO based X-routing Steiner Minimum Tree (DPSO-XSMT) algorithm [48] in the early work, so that PSO can better solve this discrete problem. In DPSO-XSMT, particles follow the following update formula:

$$X_i^t = F_3(F_2(F_1(X_i^{t-1}, w), c_1), c_2) \tag{7}$$

where w is inertia weight, c_1 and c_2 are acceleration factors, F_1 is the mutation operator, and F_2 and F_3 are crossover operators. In DPSO-XSMT, F_1 is used to decide whether to keep the state of last iteration, which corresponds to the *inertia component* of PSO. Partial gene crossover with *pbest* through F_2, corresponding to the *individual cognition*. And F_3 is used to partial gene crossover with *gbest*, which corresponds to the *social cognition* of PSO.

The social cognition of DPSO-XSMT is realized by learning with *gbest* that is the best one among all particles in the swarm. *gbest* is the best learning object obtained by integrating the learning experience of all particles in the swarm. It is worth noting that in Eq. (7), the particle learns with *gbest* with a certain probability at each iteration. This means that once the algorithm finds a local best (*lbest*), it will immediately become the *gbest* in this generation, which is the object of social learning for all other particles. In addition, all particles learn from the same object at the same time. It is easy to fall into *lbest*, so that particles will repeatedly find the optimum near the *lbest* in next iteration. In order to make up for the shortcomings of this DPSO, a new social learning strategy is proposed to be integrated into DPSO, while using chaotic descending inertia weight to better solve the XSMT problem.

(2) Particle Update Formula for SLDPSO-XSMT

(a) Chaotic Descending Mutation Strategy

Inertia weight is a very important parameter in the PSO algorithm, and it will affect the accuracy of the algorithm's convergence. In SPSO, the inertia weight is a constant, resulting in weak global search capability of the PSO algorithm. To this end, Bansal et al. [52] conducted a more in-depth study of inertia weight. And the research indicates that chaotic inertia weights can obtain the best accuracy in the comparison of 15 forms of inertia weights.

Property 1. *Chaos search is a random movement with pseudorandomness, ergodicity, and regularity. And it is not sensitive to initial value and has high calculation accuracy [53].*

Definition 7. Logistic Mapping. The logistic mapping equation is used to generate a set of random sequences with the ergodicity:

$$z_{n+1} = \mu \cdot z_n \cdot (1 - z_n), n = 0, 1, 2, \ldots \tag{8}$$

In Eq. (8), $z \in (0, 1)$ is a chaotic variable, and the initial value $z_0 \neq \{0.25, 0.5, 0.75\}$, otherwise it would be eventually periodic. And $\mu \in [0, 4]$ is the control parameter. If $\mu = 4$, the logistic mapping will show entirely chaotic dynamics, and the trajectory of chaotic variable are dense over the whole search space, which means that its chaotic result besprinkles the interval of [0,1].

SLDPSO-XSMT algorithm uses chaotic descending inertia weight [54], which makes the PSO algorithm have a strong global search ability in the early stage of iteration, and can quickly converge in the later stage of iteration, while maintaining the randomness of particle mutation (The specific mutation process is described later). The specific formula is as follows:

$$w = (w_{init} - w_{end}) \cdot \frac{Maxiter - iter}{iter} + z \cdot w_{end} \tag{9}$$

where w_{init} and w_{end} are the initial value and final value of inertia weight w, *Maxiter* and *iter* are the maximum iterative time and the current iterative time, and z is chaotic variable, following Eq. (8). It makes inertia weight have chaotic characteristic as well as keep originally varying trend.

(b) Novel Social Learning Strategy

Social learning plays an important role in the learning behavior of SI, which helps individuals in the population to learn from other individuals without increasing the cost of their own trials and errors. Social Learning Particle Swarm Optimization (SLPSO) [55] is an optimized PSO algorithm. The proposed social learning mechanism greatly improves the global search capability and optimization performance. On the basis of SLPSO, Zhang et al. [56] proposed an alternative implementation based on example-mean learning and single-example learning to update particles, which enhances the exploitation of SLPSO. However, these update formulas cannot be directly applied to discrete problems. Therefore, inspired by the SLPSO algorithm, this section designs a new social learning strategy suitable for discrete problems to incorporate into the particle update formula, thereby improving the performance of the DPSO-XSMT algorithm.

Definition 8. Example Pool. All particles in the swarm $S=\{X_i | 1 \leq i \leq M\}$ are arranged in ascending order according to the fitness: $S=\{X_1, \ldots, X_{i-1}, X_i, X_{i+1}, \ldots, X_M\}$, and then the particle swarm $EP=\{X_1, \ldots, X_{i-1}\}$ constitutes an example pool of particle X_i.

Property 2. *Different particles have different learning example pools. And a particle also has different learning example pools in different iterations.*

Property 3. *For the particle X_i, a particle is randomly selected as the learning object from its current example pool in each social learning. Particularly, when the learning object selected by X_i is the first particle X_1 in the example pool, the particle X_i is learning the global optimal solution X_G at this time.*

Fig. 5 shows the example pool of a particle. The red pentagram is the unknown optimal solution to be explored, and the red particle X_G is the optimal solution found by the current population. For particle X_i, green particles are relatively backward particles, while yellow particles are closer to the global optimal solution than X_i and have better fitness values than X_i. These particles (including the global optimal solution X_G) constitute the example pool of particle X_i, and each particle has the possibility of becoming the object of its social learning. Particle X_i randomly selects any particle in the example pool in each iteration, and learns the historical experience of this particle (that is, learns the historical optimal value of this particle, which is called *kbest*) to complete its own social learning process. According to Property 2, it can be known that such a social learning process allows particles to improve themselves through continuous learning of different excellent individuals during the evolution process, which is conducive to the diversified development of the population. And the particle has a certain probability to learn the global optimal particle in the process of changing learning object, according to Property 3. This allows the algorithm to better balance exploration and exploitation.

(c) Particle Updating with Mutation and Crossover Operators

Based on DPSO-XSMT and the above-mentioned social learning strategy, this paper designs a new particle update process. SLDPSO-XSMT still uses Eq. (7) as the particle update formula, but F_3 has a different meaning. The specific operation is as follows.

(i) Inertia Component. The proposed algorithm completes the particle velocity is updated through F_1, which is expressed as follows:

$$W_i^t = F_1(X_i^{t-1}, w) = \begin{cases} M(X_i^{t-1}), r_1 < w \\ X_i^{t-1}, \quad \text{otherwise} \end{cases} \tag{10}$$

where, w is the probability of mutation operation, following the chaotic decreasing formula of Eq. (9). r_1 is a random number in [0,1].

The SLDPSO-XSMT algorithm uses two-point mutation. If the generated random number $r_1 < w$, the algorithm will randomly select two edges and replace the PSP choices of these two edges, so as to achieve the purpose of randomly changing the topology of the routing tree. Otherwise, keep the routing tree unchanged. Fig. 6 gives a routing tree with 6 pins. Below the tree is the code for the particle (the fitness value of particle is not shown in Fig. 6). It can be seen that the algorithm randomly selects two edges m_1 (3, 2, 1) and m_2 (2, 5, 3) of particle X_i: 321 122 253 and 423 260 673. After F_1 operation, the PSP choices of m_1 and m_2 are replaced to Choice 2 and 0, respectively.

(ii) Individual Cognition. The proposed algorithm uses F_2 to complete the individual cognition of particles, which is expressed as follows:

$$S_i^t = F_2(W_i^t, c_1) = \begin{cases} C_p(W_i^t), r_2 < c_1 \\ W_i^t, \quad \text{otherwise} \end{cases} \tag{11}$$

where c_1 represents the probability that the particle crosses with its historical optimum (X_i^P), and r_2 is a random number in [0,1). This part reflects the process of particles learning their own experience.

(iii) Social Cognition. The proposed algorithm uses F_3 to complete the social cognition of particles, which is expressed as follows:

$$X_i^t = F_3(S_i^t, c_2) = \begin{cases} C_p(S_i^t), r_3 < c_2 \\ S_i^t, \quad \text{otherwise} \end{cases} \tag{12}$$

where c_2 represents the probability that the particle crosses with the historical optimum of any particle X_k^P in the example pool, and r_3 is a random number in [0,1). This part reflects the information sharing and communication between particles.

The crossover operators in Eqs. (11) and (12) are as follows. When the generated random number $r_2 < c_1$ (or $r_3 < c_2$), F_2 performs the crossover operation on a Steiner tree with n pins. The algorithm first randomly generates the required crossed continuous interval $[C_{start}, C_{end}]$, where C_{start} and C_{end} are random

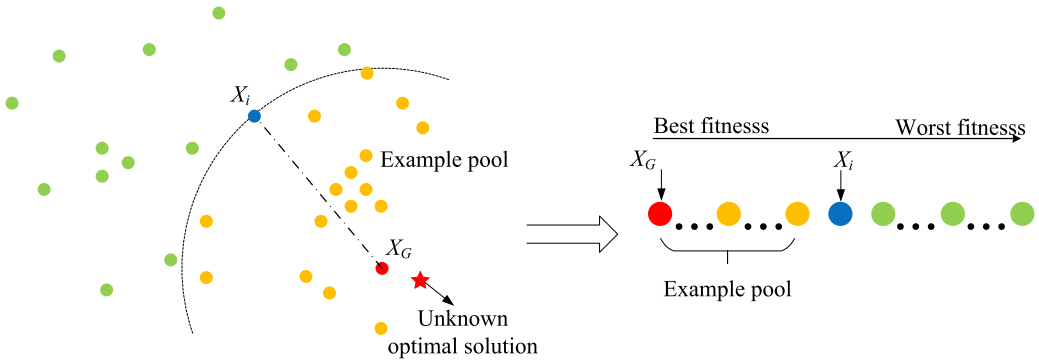


Fig. 5. Example pool of the particle learning.

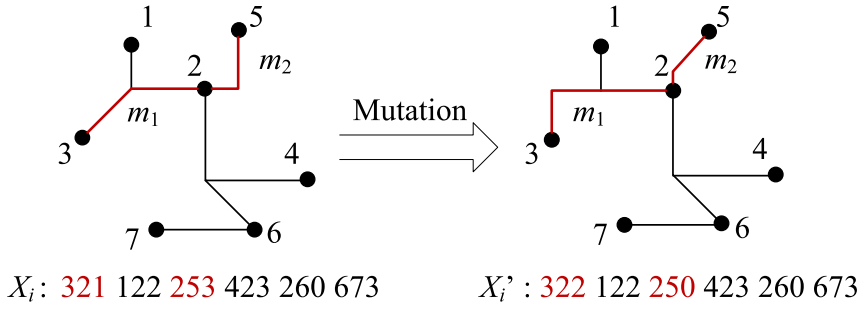


Fig. 6. Mutation operator of SLDPSO-XSMT.

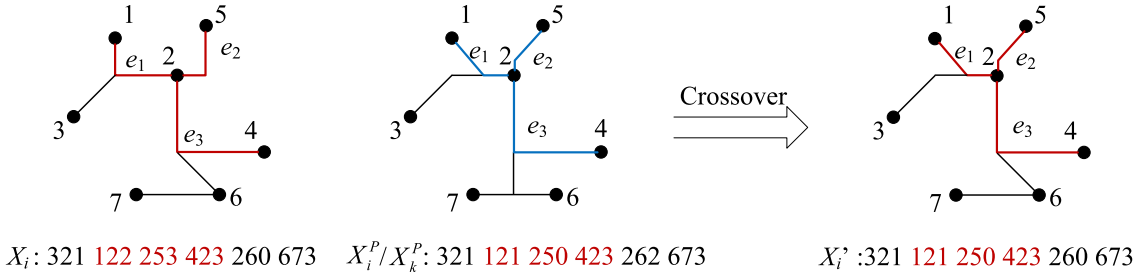


Fig. 7. Crossover operator of SLDPSO-XSMT.

integers between [1, n-1]. Then, find the encoding of the learning object (X_i^P or X_k^P) of particle X_i on the interval $[3 \times (C_{start} - 1), 3 \times C_{end}]$. Finally, replace the number on the interval of particle X_i with the code on the interval of the learning object. As shown in Fig. 7, particle X_i (321 122 253 423 260 673) is the particle to be crossed, and particle (321 121 250 423 262 673) is its learning object. It is presented as X_i^P in Eq. (11), while in Eq. (12), it is presented as X_k^P. The algorithm first generates a continuous interval $[C_{start}, C_{end}] = [2, 4]$, and the corresponding edges to be crossed are e_1, e_2, and e_3. Then the code of learning object in the interval $[3 \times (2 - 1), 3 \times 4] = [3, 12]$ is 121 250 423, corresponding to the blue solid line segment in Fig. 7. Finally, replace the encoding on the interval of particle X_i with the above encoding string. After the crossover operation, the PSP choices of edges e_1, e_2, and e_3 in X_i are respectively changed from Choice 2, Choice 3, and Choice 3 to Choice 1, Choice 0, and Choice 3, while the topology of the remaining edges remains unchanged.

After the above crossover operators, particle X_i can learn some genes from better particles. Repeated iterative learning can gradually make particle X_i move closer to the global optimal position. Moreover, the acceleration factor c_1 is set to decrease linearly and c_2 is set to increase linearly, so that the algorithm has a higher probability to learn its own historical experience in the early iteration to enhance global search ability. While it has a higher probability to learn outstanding particles in the later iteration to enhance exploitation ability, so as to quickly converge to a position close to the global optimum.

4.1.4. Procedure of SLDPSO searching

The steps for *SLDPSO Searching* can be summarized as follows.

Step 1. Initialize the population and parameters. The minimum spanning tree construction method is proposed to construct initial routing tree, and the PSP choices of each edge is randomly initialized to Choice 0 or 1.

Step 2. Calculate the fitness value of each particle according to Eq. (6), and sort them in ascending order: S={$X_1, \ldots, X_{i-1}, X_i, X_{i+1}, \ldots, X_M$} .

Step 3. Initialize each particle's *pbest* and *gbest* to itself, and initialize the example pool of each particle .

Step 4. Update the velocity and position of each particle according to Eqs. (10)–(12). The update of w adopts the chaotic decreasing strategy (Eqs. (8)–(9)), and the update of c_1 and c_2 respectively adopt linear decreasing and linear increasing strategy.

Step 5. Calculate the fitness value of each particle.

Step 6. Update each particle's *pbest* and its learning example pool *EP*, as well as *gbest*.

Step 7. If the termination condition is met (the set maximum number of iterations is reached), end the algorithm. Otherwise, return to **Step 4**.

Property 4. *When initializing the PSP choices for all edges of the routing tree, compared with the random initialization of 4 choices (Choice 0–3), the initial edge strategy of 2 choices (Choice 0 or Choice 1) can effectively improve the quality of the solution.*

Since the rectangular routing way (Choice 2–3) limits the optimization of wirelength, while the 45° and 135° routing way can make better use of limited routing resources. Therefore, to a certain extent, initializing the PSP choices of all edges to Choice 0 or Choice 1 can generate better initial particles and speed up the algorithm to find the optimal solution. The experiments in Section 4 also prove the correctness of Property 4.

Property 5. *The proposed SLDPSO method based on the novel social learning strategy has a good balance between global exploration and local exploitation ability so as to effectively solve the XSMT problem.*

Different from DPSO-XSMT, the particles of the proposed SLDPSO-XSMT will not only learn form their own historical experience, but also learn from any better particle in the current population, having possibility learning from global optimal solution. At the same time, the chaotic descending inertia weight also increases the diversity in the evolution process of the population, which is conducive to enhancing the exploration ability of the algorithm, thereby breaking the local optimum. On the other hand, particle X_k have better fitness than particle X_i. X_i does not learn from X_k but directly learn the best historical experience of X_k, namely X_k^P (at iteration t, $f(X_k^{P,t}) \leq f(X_k^t) \leq f(X_i^t)$, which is conducive to strengthening the exploration ability and speeding up the algorithm convergence.

4.2. Wirelength reduction

4.2.1. Analysis of local optimal topology

The randomness of particle flight is one of the main reasons why the PSO algorithm has such a powerful search ability. For large-scale discrete problems, PSO can find satisfactory solutions in a limited time. But in most cases, there will still be a certain distance from the unknown optimal solution. In addition, the introduction of the X-routing has reduced the wirelength of the Steiner tree to some extent, but the X-routing is not better than the rectangle architecture in some cases. Therefore, a local topology optimization strategy is proposed to make more precise adjustments to the Steiner tree found during the *SLDPSO Searching* stage.

As shown in Fig. 8, v_2 is a 2-degree terminal connected to v_1 and v_3, respectively. Based on the existing topology (v_2, v_3, s_1) where s_1 is the PSP connecting v_2 and v_3, the PSP choice of edge (v_2, v_1) needs to be considered so that this 2-degree terminal v_2 has the best topology. Fig. 8 lists several typical cases, where s_2 and s_3 are PSPs formed by v_2 connected to v_1 through a rectangle architecture and an X-routing, respectively.

- When s_2 is on the right side of s_1 (Fig. 8(a)): If v_2 is connected to v_1 by Choice 3, a new edge segment $e(s_2, v_1)$ is added. If v_2 is connected to v_1 by Choice 0, a new edge segment $e(s_3, v_1)$ is added. Obviously, (v_2, v_1, s_2) is the optimal structure, because the right-angle edge $e(s_2, v_1)$ is always smaller than the hypotenuse $e(s_3, v_1)$.
- When s_2 is on the left of s_1 (Fig. 8(b)(c)(d)): In Fig. 8(b), there is $e(s_1, s_2) = d$, so that $e(s_1, s_2) + e(s_2, v_1) = e(s_3, v_1)$. In this case, for the given topology, the total wirelength obtained by (v_2, v_1, s_2) and (v_2, v_1, s_3) are the same. And if $e(s_1, s_2) > d$ (Fig. 8(c)), since there is always $e(s_1, s_3) + (e(s_1, s_2) + e(s_2, v_1)) > e(s_1, s_3) + e(s_3, v_1)$, so v_2 is connected to v_1 by Choice 0, that is (v_2, v_1, s_3) is the optimal structure. If $e(s_1, s_2) < d$ (Fig. 8(d)), it is obvious that compared to the X-routing, the rectangle architecture can obtain a shorter wirelength, that is, (v_2, v_1, s_2) is the optimal structure.

The above analysis shows that, on the one hand, neither the X-routing nor the rectangle architecture can obtain the best topology, only the effective combination of the two can obtain a shorter wirelength. On the other hand, based on this thinking mode, a corresponding local topology optimization strategy is designed to further optimize the XSMT obtained by *SLDPSO Searching*.

4.2.2. Procedure of wirelength reduction

Algorithm 1 Local topology optimization

Require: XST
Ensure: XSMT
1: **for** each terminal v_i in P of XST **do**
2: Initialize $v_i.degree = 0$, $v_i.adj\_list[] = \emptyset$;
3: **end for**
4: **for** each edge (v_i, v_j) of XST **do**
5: **if** $v_i.degree < q$ **then**
6: $v_i.degree += 1$;
7: add v_i to $v_j.adj\_list[]$;
8: **end if**
9: **if** $v_j.degree < q$ **then**
10: $v_j.degree += 1$;
11: add v_j to $v_i.adj\_list[]$;
12: **end if**
13: **end for**
14: $Sort(P)$ in decreasing order according to $v_i.degree$;
15: **for** each terminal v_i in P of XST **do**
16: $Sort(v_i.adj\_list)$ in decreasing order according to $v_i.adj\_list[k].degree$; // $1 \leq k \leq vi.degree$
17: **end for**
18: **for** each terminal v_i in P **do**
19: **for** each edge $(v_i, v_i.adj\_list[k])$ connected to v_i **do**
20: *bestChoice* = getChoice(v_i, $v_i.adj\_list[k]$); //Get the PSP choice of the edge $(v_i, v_i.adj\_list[k])$.
21: *bestFitness* = getFitness(*bestChoice*); //Get current fitness value.
22: **for** *choice* = 0 to 3 **do**
23: **if** *choice* == *bestChoice* **then**
24: Continue;
25: **else**
26: *curFitness* = getFitness(*choice*);
27: **if** *bestFitness* > *curFitness* **then**
28: ResetChoice(v_i, $v_i.adj\_list[k]$, *choice*); //Update the local topology.
29: *bestFitness* = *curFitness*;
30: **end if**
31: **end if**
32: **end for**
33: **end for**
34: **end for**

Property 6. *The proposed local topology optimization strategy can effectively reduce the wirelength of XST by adjusting the local optimal structure of all terminals in the routing tree.*

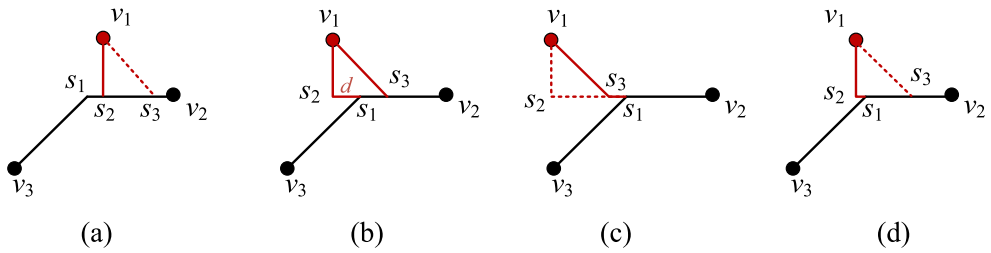


Fig. 8. The several optimal structure of a 2-degree terminal.

It is assumed that the terminal with most degrees in a tree is q-degree terminal, and q is a user-defined parameter, which meets $1 \leq q \leq Q$. Considering the complexity of actual structures of nets, it is not recommended to adjust all the adjacent edges of each pin. Because it will pay a time price, especially when the Q is very large and there are many pins with degrees close to Q in the net. Therefore, the local topology optimization strategy traverses at most q adjacent edges of each pin and makes appropriate adjustments. At the same time, the algorithm can make a compromise between optimization performance and run time by adjusting the value of parameter q. The closer the value of q is to Q, the better the optimization performance. $q = \lceil 0.8 \times Q \rceil$ is set in this paper.

The pseudo code of the local topology optimization strategy is shown in Algorithm 1. Each terminal has two attributes, one is *degree*, that is, the number of adjacent terminals; the other is the *adj_list*[], which is used to store each adjacent terminal. The specific implementation steps of this strategy are as follows.

- Record the degree of each terminal in XST and corresponding list of its adjacent terminals (lines 1–13). In this step, the algorithm records the degree of each terminal by traversing each edge of XST, and adds its adjacent terminals to the corresponding list. Particularly, for a terminal whose degree is greater than q, the algorithm only records q adjacent terminals.
- All terminals are arranged in descending order of *degree*, and the terminals in *adj_list* of each terminal is also arranged in descending order of degree (lines 14–17). So as to optimize the topology of the dense area first, and then optimize the topology of the sparse area.
- Optimize the local topology of each terminal in turn (lines 18–34). For each point v_i in P, the algorithm will traverse each adjacent edge $(v_i, v_i.adj\_list[k])$ of the point, where $1 \leq k \leq v_i$.degree, and adjust the PSP choice of the edge (from Choice 0 to Choice 3), from which the PSP choice with the smallest fitness is retained.

The local topology optimization strategy is through continuously adjusting the PSP Choice of each adjacent edge for each terminal under the current XST topology to obtain the local optimal structure, thereby further optimizing the wirelength of XST.

4.3. Overall flow of SLDPSO based Two-Stage X-routing for IC design under IEC architecture

For an initial routing tree, the proposed TS-XSMT algorithm defines a two-stage operation to optimize the wirelength. In the *SLDPSO Searching* stage, the particle is updated through mutation operator and crossover operators combined with novel social learning strategy. If the termination condition is met, exit the search stage, otherwise the iteration will continue. After the *SLDPSO Searching* stage, the algorithm will enter the stage of *Wirelength Reduction*. At this time, the XST obtained through the last stage will be optimized by the local topology optimization strategy. The local topology structure of all pins in the Steiner tree may be adjusted, and the wirelength of the routing tree will be further optimized.

Fig. 9 shows an example of the complete flow of the proposed TS-XSMT algorithm. A particle in the swarm represents a kind of topological structure of the Steiner tree. The process indicated by blue arrow is the *SLDPSO Searching* stage, and the process indicated by green arrow is the *Wirelength Reduction* stage. In the first stage, the particle will repeatedly perform three operations: F_1, F_2 and F_3 (see Section 4.1.3 for specific definitions). After the mutation operation of F_1, the PSP choices of edges e_1 and e_2 will be randomly changed, and then the particle will realize the learning process through crossover operations with *pbest* and *kbest*, respectively. In F_2, the particle will learn the PSP choices of e_1 and e_3 from *pbest*, and in F_3, it will learn the PSP choices of e_4 and e_5 from *kbest*. After repeated iterations, a Steiner tree with a shorter length can be obtained, and then the proposed algorithm will enter the second stage. At this stage, the proposed algorithm gradually optimizes the local topology of each point from the point with the largest degree to the point with the smallest degree. As can be seen from Fig. 9, the point with the largest degree is v_1, whose degree is 4, and there is only e_5 to be optimized in its adjacent edge. After changing the PSP choice of e_5 from Choice 3 to Choice 0, the local topology of point v_1 reaches the best. Then, after optimizing the point v_2 with a degree of 3 and modifying the PSP choice of its adjacent edge e_6, the topology of the routing tree corresponding to the particle will no longer be optimized, because the remaining points already have the optimal local topology.

4.4. Complexity of SLDPSO based Two-Stage X-routing for IC design under IEC architecture

Lemma 1. *Assuming the population size is M, the number of iterations is T, the number of pins is n, and* $m = max\{M \times T, n\}$, *and then the complexity of TS-XSMT algorithm is* $O(mn\log_2 n)$.

Proof (*1*). In the *SLDPSO Searching* stage: From Step 2 to Step 6, mutation operators, crossover operators, the calculation of fitness value and update of the example pool are included. The time complexity of mutation and crossover operators are both $O(n)$. As for the calculation of fitness value, it is realized by calculating the length of the routing tree. And its complexity is mainly determined by the complexity of the sorting method $O(n\log_2 n)$. Since the example pool of each particle would change at the end of each iteration, the time for updating example pool is mainly spent on sorting, that is, its time complexity is also $O(n\log_2 n)$. Therefore, the complexity of the internal loop of the SLDPSO algorithm is $O(n\log_2 n)$. Then, considering the size of the population and the number of iterations, the complexity of proposed SLDPSO algorithm is $O(MT \times n\log_2 n)$.

(2) In the *Wirelength Reduction stage*: This stage consists of three steps. First, the degree of each terminal in XST and corresponding list are recorded, which only needs to traverse all

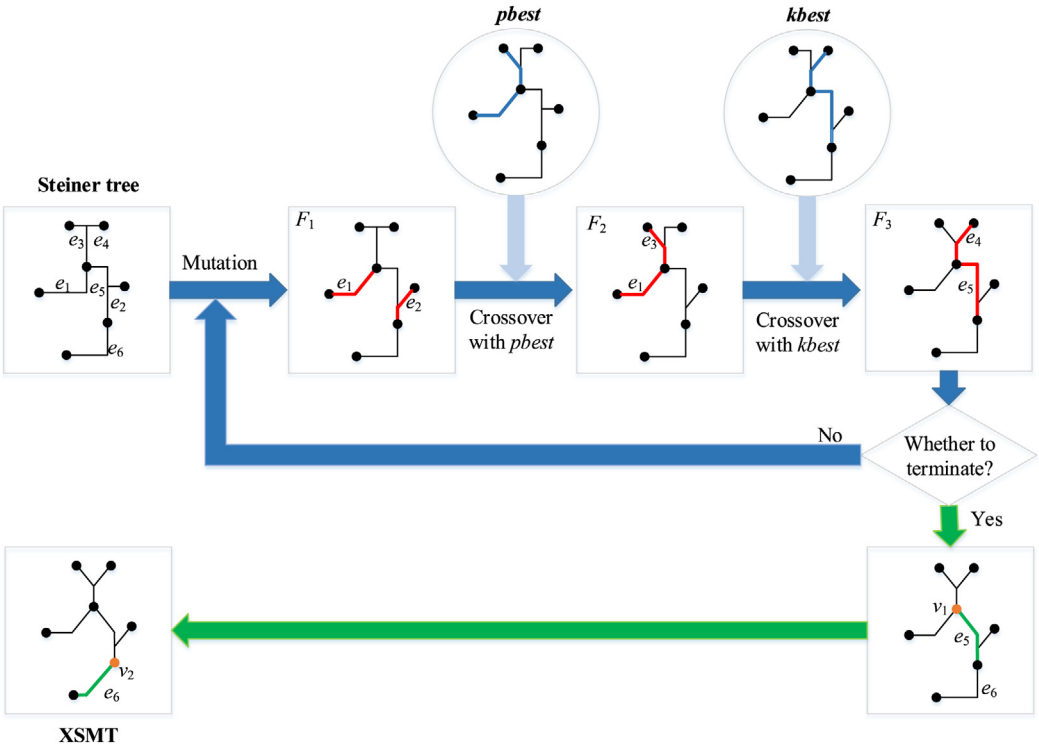


Fig. 9. An example of SLDPSO based TS-XSMT algorithm.

terminals and edges of XST, and the time complexity is $O(n)$. Next, all terminals are arranged in descending order of degree with a time complexity of $O(n^2\log_2 n)$. Finally, all adjacent edges of each terminal need to be traversed when optimizing the local topology, which is completed in $O(n^2)$. Therefore, the complexity of this stage is $O(n^2\log_2 n)$.

In summary, the complexity of TS-XSMT algorithm is $O(mn\log_2 n)$. □

5. Experimental results

In order to verify the performance and effectiveness of the proposed algorithm and related strategies in this paper, experiments are performed on the benchmark circuit suite [57], and detailed comparison results are given in this section. The scale of this benchmark includes nets with pins from 8 to 1000. The parameter settings in this paper are consistent with the literature [50], that is, w chaotically decreases from 0.95 to 0.4, c_1 decreases linearly from 0.82 to 0.5, and c_2 increases linearly from 0.4 to 0.83. The population size is set to 100 and the maximum number of iterations is 1000. Considering the randomness of the PSO algorithm, the mean values in all experiments are obtained by independent run 20 times.

5.1. Validation of initial edge strategy

To verify the effectiveness of the initial edge strategy, this section compares the wirelength of routing tree obtained by the initial edge strategy of two choices (2-init) and four choices (4-init). In Table 4, *bestWL* is wirelength of the routing tree corresponding to the optimal particle in the initial population, namely the shortest wirelength obtained. And *aveWL* is the average wirelength of all the original routing trees. Obviously, *aveWL* reflects the quality of the initial population, and *bestWL* represents the most potential particle in the population. It can be seen that compared with the 4-init strategy, the 2-init strategy can optimize 6.762% in the average wirelength of the initial particles, and the obtained optimal particle has a wirelength that is about 3.208% shorter than 4-init strategy. And as you can see from the *aveWL* metric, the 2-init strategy is able to get the initial routing trees with better quality for all test cases.

Table 4
Comparison between the initial edge strategy of two choices (2-init) and four choices (4-init).

Test	Pins	*bestWL*			*aveWL*		
		4-init	2-init	Imp(%)	4-init	2-init	Imp(%)
1	8	17 434	17 506	−0.411	18 939	17 751	6.269
2	9	18 377	18 041	1.831	20 211	18 283	9.540
3	10	19 823	19 703	0.603	21 710	19 944	8.134
4	20	33 874	32 810	3.142	35 624	33 072	7.163
5	50	51 022	49 138	3.692	52 884	49 340	6.702
6	70	59 494	57 547	3.272	61 093	57 721	5.519
7	100	73 271	70 365	3.966	75 104	70 569	6.038
8	410	151 451	143 302	5.381	153 109	143 683	6.156
9	500	164 033	155 888	4.965	166 340	156 481	5.927
10	1000	235 507	222 235	5.636	237 172	222 542	6.168
Average				3.208			6.762

Therefore, the initial edge strategy of two choices can bring better initial particles to the proposed algorithm, which is conducive to find a better solution.

5.2. Validation of SLDPSO searching

In order to validate the effectiveness of the first stage of the TS-XSMT algorithm, that is, the DPSO searching based on a novel social learning strategy, this paper compares the proposed SLDPSO-XSMT with DPSO-XSMT [48]. For the sake of experiment fairness, the data of DPSO-XSMT in Table 5 are all obtained under the parameters setting in this paper, where the inertia weight is set to decrease linearly from 0.95 to 0.4. And the wirelength of obtained best XSMT (best value), the average wirelength of obtained XSMTs in 20 runs (mean value), and standard deviation are

compared between SLDPSO and DPSO methods. Among the three performance indexes, the first two indexes reflect the wirelength optimization ability, while the latter reflects the stability of the algorithm in multiple experiments.

Compared with DPSO method, the proposed SLDPSO method can obtain shorter wirelength and has stronger stability, and respectively achieves 0.121%, 0.199% and 32.80% optimization in best value, mean value and standard deviation on average. Among all test cases, the average wirelength is optimized the most in Test 8 (410 pins), where the wirelength of the best XSMT obtained is optimized by 0.371%, the average wirelength is reduced by 0.499%, and the standard deviation is optimized by 54.50%. And for Test 6, the DPSO method produces a wirelength standard deviation of 98 in 20 executions, while the standard deviation of SLDPSO method is only 35, which is 64.73% optimized. It means that our algorithm has a stronger stability. The experimental results in Table 5 prove the effectiveness of the novel social learning strategy proposed in this paper.

Additionally, it can be seen that the standard deviation of SLDPSO is much lower than that of the DPSO method. One is because the chaotic decreasing inertia weight is adopted. Since the inertia component of DPSO is realized by mutation operator, the value of inertia weight determines the probability of mutation. The linear decreasing method makes the frequency of particle mutation to a large extent depend on the initial and final values. Therefore, the frequency of mutation in the early iterations is very high, and it gets very low in the later iterations. In contrast, the pseudo-randomness, ergodicity, and insensitivity of initial value of the chaotic search make the value of inertial weight more evenly distributed in a linearly decreasing region, making the probability of mutation relatively stable, and at the same time it would gradually decrease as the growth number of iteration. The second is because the social learning of particles through their example pools expands the scope of learning objects, while DPSO only performs social learning by learning the best particle in all individuals (*gbest*). For the backward particles in the population, due to the large gap with *gbest*, one update may cause a large position change, resulting in poor stability. The random selection of learning objects through the example pool alleviates this kind of jumping to a certain extent, thereby improving the stability of the algorithm.

5.3. Validation of local topology optimization strategy

The wirelength of the obtained Steiner trees before and after using local topology optimization strategy are compared in Table 6. The experiment compares wirelength of the best XST (best value), wirelength of the worst XST (worst value) and average wirelength of all XSTs (mean value) obtained in 20 executions. As can be seen from the experimental results, local topology optimization strategy can reduce wirelength by 2.381% on average for the net with 1000 pins. Among them, the length of the worst Steiner tree is 220629, and after local topology optimization, the wirelength is 214950, reducing by 2.57%. And for the best Steiner tree obtained by SLDPSO searching, its wirelength is optimized by 2.263% after local topology optimization. However, local topology optimization strategy does not play a role in wirelength optimization for nets with 20 pins or less. This is because the SLDPSO method has been able to find a great solution in the limited search space. This also shows that the SLDPSO method can obtain satisfactory solutions for small and medium-sized Steiner tree problems without going through the second stage. Experimental results show that the local topology optimization strategy has a significant optimization on test cases with more than 50 pins, and the more pins of the net, the more wirelength is optimized.

In addition, it can be clearly seen that for all test cases, the wirelength reduction of the worst XST is higher than the best, and it is possible for the worst XST to obtain a shorter length than the best XST after local topology optimization. For example, for Test case 9, the wirelength of best XST obtained by *SLDPSO Searching* is 153889 and the worst is 154196. After this optimization strategy, the length of the worst XST is 151441, which is shorter than the best that the length is 151455. The experimental results in Table 6 demonstrate the effectiveness of the local topology optimization strategy in reducing wirelength.

5.4. Validation of SLDPSO based Two-Stage X-routing for IC design under IEC architecture

In order to verify the effectiveness of the TS-XSMT algorithm proposed in this paper, this section compares the wirelength optimization capabilities of the TS-XSMT algorithm with the existing Steiner minimum tree construction algorithms, including RSMT and XSMT algorithms.

Table 7 shows the comparison between the TS-XSMT algorithm and the two RSMT algorithms, DPSO (R) [58] and HTS-PSO (R) [50], respectively. And the mean wirelength of the obtained SMT and normalized value from 20 executions of these three algorithms are compared. From Table 7, the average wirelength ratio of the Steiner minimum tree obtained by DPSO (R), HTS-PSO (R) and TS-XSMT algorithm is 1.114:1.089:1. This means that the wirelength obtained by DPSO(R) and HTS-PSO(R) algorithms will be 11.4% and 8.9% longer than that of ours, respectively.

For Test 1 with the least pins, the wirelength of SMT obtained by the DPSO (R) and HTS-PSO (R) algorithms are 6.0% and 4.6% longer than ours. For Test 10 with the most pins, the mean wirelength obtained by DPSO (R) algorithm is 245201, and is 1.141 times that of ours, while the mean wirelength obtained by the HTS-PSO (R) algorithm is 239824, which is 11.6% longer than ours. It can be seen that our algorithm can greatly optimize the wirelength of routing tree. In addition, it is analyzed that for the net with more pins, the wirelength of Steiner minimum tree obtained by the proposed TS-XSMT algorithm has more obvious wirelength optimization than the above two algorithms.

At the same time, this paper also compares TS-XSMT to some XSMT algorithms with good wirelength optimization capabilities in recent years, including: DDE (X) [23], DPSO (X) [48] and HTS-PSO (X) [50]. And the comparison of the above algorithms on the mean wirelength and standard deviation indicators are respectively shown in Tables 8 and 9, which respectively reflect the wirelength optimization capability and stability of the proposed algorithm.

As shown in Table 8, the average wirelength ratio of the XSMT generated by the three XSMT algorithms to our algorithm is 1.025:1.008:1.004:1, indicating that the TS-XSMT algorithm is generally superior to these three XSMT algorithms. Compared with the DDE (X) algorithm, ours has better wirelength optimization for the net with 20 or more pins. For example, In Test 10, the wirelength of XSMT generated by TS-XSMT algorithm is only 214990, while the wirelength generated by DDE (X) algorithm is 232359, which is 8.1% longer than the wirelength obtained by ours. And compared with the DPSO (X) algorithm, TS-XSMT algorithm can achieve the same or shorter wirelength on all test cases. And the HTS-PSO (X) algorithm is slightly better than our algorithm for the test cases with 20 or fewer pins. While in large-scale nets (more than 50 pins), TS-XSMT algorithm can obtain shorter wirelength. In summary, compared with the above three XSMT algorithms, the proposed algorithm can achieve an obvious wirelength optimization on Test 5-10, and the more pins the net contains, the stronger wirelength optimization capability of the TS-XSMT algorithm. Particularly, for the net with 1000 pins, the wirelength of corresponding SMTs obtained by the DDE (X), DPSO (X) and HTS-PSO (X) algorithms are 8.1%, 2.9%, and 2.3% longer than ours, respectively.

Table 5
Comparison between SLDPSO and DPSO methods.

Test	Pins	Best value			Mean value			Standard deviation		
		DPSO	SLDPSO	Imp(%)	DPSO	SLDPSO	Imp(%)	DPSO	SLDPSO	Imp(%)
1	8	16 918	16 918	0.000	16 918	16 918	0.000	0	0	0.00
2	9	18 041	18 041	0.000	18 041	18 041	0.000	0	0	0.00
3	10	19 696	19 696	0.000	19 696	19 696	0.000	0	0	0.00
4	20	32 193	32 193	0.000	32 213	32 207	0.019	11	10	11.29
5	50	47 960	47 953	0.014	48 038	47 982	0.118	83	32	61.45
6	70	56 279	56 278	0.002	56 448	56 341	0.191	98	35	64.73
7	100	68 578	68 486	0.133	68 911	68 697	0.311	182	104	42.69
8	410	141 427	140 902	0.371	141 852	141 143	0.499	238	108	54.50
9	500	154 365	153 889	0.308	154 742	154 056	0.443	204	97	52.70
10	1000	220 795	219 955	0.380	221 142	220 240	0.408	282	167	40.68
Average				0.121			0.199			32.80

Table 6
Comparison before and after local topology optimization strategy.

Test	Pins	Best value			Worst value			Mean value		
		Before	After	Imp(%)	Before	After	Imp(%)	Before	After	Imp(%)
1	8	16 918	16 918	0.000	16 918	16 918	0.000	16 918	16 918	0.000
2	9	18 041	18 041	0.000	18 041	18 041	0.000	18 041	18 041	0.000
3	10	19 696	19 696	0.000	19 696	19 696	0.000	19 696	19 696	0.000
4	20	32 193	32 193	0.000	32 214	32 214	0.000	32 205	32 205	0.001
5	50	47 960	47 953	0.014	48 032	47 953	0.165	47 977	47 953	0.051
6	70	56 281	56 271	0.018	56 489	56 307	0.323	56 351	56 280	0.125
7	100	68 486	68 335	0.221	69 019	68 335	0.991	68 697	68 347	0.510
8	410	140 902	139 082	1.291	141 332	139 122	1.564	141 143	139 074	1.466
9	500	153 889	151 455	1.582	154 196	151 441	1.786	154 056	151 408	1.719
10	1000	219 970	214 991	2.263	220 629	214 950	2.574	220 233	214 990	2.381
Average				0.539			0.740			0.625

Table 7
Comparison between TS-XSMT and RSMT algorithms.

Test	Pins	Mean wirelength			Normalized value		
		DPSO (R)	HTS-PSO (R)	Ours	DPSO (R)	HTS-PSO (R)	Ours
1	8	17 931	17 693	16 918	1.060	1.046	1.000
2	9	20 503	19 797	18 041	1.136	1.097	1.000
3	10	21 910	21 226	19 696	1.112	1.078	1.000
4	20	35 723	35 072	32 205	1.109	1.089	1.000
5	50	53 383	52 025	47 953	1.113	1.085	1.000
6	70	61 987	61 129	56 280	1.101	1.086	1.000
7	100	76 016	74 416	68 347	1.112	1.089	1.000
8	410	156 520	153 672	139 074	1.125	1.105	1.000
9	500	170 273	166 592	151 408	1.125	1.100	1.000
10	1000	245 201	239 824	214 990	1.141	1.116	1.000
Average					1.114	1.089	1.000

Table 8
Comparison between TS-XSMT and other XSMT algorithms (1).

Test	Pins	Mean wirelength				Normalized value			
		DDE (X)	DPSO (X)	HTS-PSO (X)	Ours	DDE (X)	DPSO (X)	HTS-PSO (X)	Ours
1	8	16 911	16 918	16 921	16 918	1.000	1.000	1.000	1.000
2	9	18 039	18 041	18 023	18 041	1.000	1.000	0.999	1.000
3	10	19 469	19 696	19 397	19 696	0.988	1.000	0.985	1.000
4	20	32 342	32 213	32 063	32 202	1.004	1.000	0.996	1.000
5	50	48 668	48 038	48 027	47 953	1.015	1.002	1.002	1.000
6	70	57 255	56 448	56 350	56 278	1.017	1.003	1.001	1.000
7	100	70 686	68 911	68 625	68 347	1.034	1.008	1.004	1.000
8	410	147 115	141 852	140 898	139 074	1.058	1.020	1.013	1.000
9	500	159 672	154 742	153 708	151 408	1.055	1.022	1.015	1.000
10	1000	232 359	221 142	219 954	214 990	1.081	1.029	1.023	1.000
Average						1.025	1.008	1.004	1.000

As shown in Table 9, our algorithm can perfectly reduce the standard deviation to 0 on Test 1–3 and 5. For other test cases, the ratio of the average standard deviation of DDE (X), DPSO (X), HTS-PSO (X) and ours is 87.92:6.76:6.43:1. Particularly, the above three algorithms all produce the largest standard deviation on Test 7 (100 pins), which is 130.94 times, 13.22 times, and 10.86 times than our algorithm respectively. The experimental results show that TS-XSMT has absolute advantage in stability of the algorithm.

It can be seen that the proposed TS-XSMT algorithm has the strongest stability and superior wirelength optimization capability, especially for large-scale nets.

Table 9
Comparison between TS-XSMT and other XSMT algorithms (2).

Test	Pins	Standard deviation				Normalized value			
		DDE (X)	DPSO (X)	HTS-PSO (X)	Ours	DDE (X)	DPSO (X)	HTS-PSO (X)	Ours
1	8	34	0	16	0	–	–	–	–
2	9	41	0	0	0	–	–	–	–
3	10	133	0	0	0	–	–	–	–
4	20	165	11	31	10	16.01	1.10	3.01	1.00
5	50	827	83	94	0	–	–	–	–
6	70	1135	98	145	14	79.63	6.89	10.16	1.00
7	100	1802	182	150	14	130.94	13.22	10.86	1.00
8	410	3615	238	188	36	100.80	6.64	5.23	1.00
9	500	3688	204	181	46	80.21	4.44	3.93	1.00
10	1000	4089	282	184	34	119.95	8.28	5.40	1.00
Average						87.92	6.76	6.43	1.00

Table 10
Comparison between TS-XSMT and other optimization algorithms (1).

Test	Pins	Mean wirelength					Normalized value				
		ABC	GA	MA	BEA	Ours	ABC	GA	MA	BEA	Ours
1	8	16 918	17 183	17 038	17 676	16 918	1.000	1.016	1.007	1.045	1.000
2	9	18 041	18 096	18 078	18 260	18 041	1.000	1.003	1.002	1.012	1.000
3	10	19 696	19 751	19 708	19 947	19 696	1.000	1.003	1.001	1.013	1.000
4	20	32 366	32 535	32 665	33 030	32 202	1.005	1.010	1.014	1.026	1.000
5	50	48 689	48 903	48 961	49 343	47 953	1.015	1.020	1.021	1.029	1.000
6	70	57 273	57 356	57 372	57 724	56 278	1.018	1.019	1.019	1.026	1.000
7	100	70 461	70 195	70 232	70 585	68 347	1.031	1.027	1.028	1.033	1.000
8	410	143 700	143 448	143 448	143 642	139 074	1.033	1.031	1.031	1.033	1.000
9	500	156 502	156 314	156 249	156 467	151 408	1.034	1.032	1.032	1.033	1.000
10	1000	222 537	222 448	222 372	222 576	214 990	1.035	1.035	1.034	1.035	1.000
Average							1.017	1.020	1.019	1.028	1.000

Table 11
Comparison between TS-XSMT and other optimization algorithms (2).

Test	Pins	Standard deviation					Normalized value				
		ABC	GA	MA	BEA	Ours	ABC	GA	MA	BEA	Ours
1	8	0	292	129	148	0	–	–	–	–	1.00
2	9	0	61	81	161	0	–	–	–	–	1.00
3	10	0	61	118	136	0	–	–	–	–	1.00
4	20	78	182	118	156	10	7.80	18.20	11.80	15.60	1.00
5	50	268	144	113	88	0	–	–	–	–	1.00
6	70	267	87	123	50	14	19.07	6.21	8.79	3.57	1.00
7	100	267	90	114	69	14	19.07	6.43	8.14	4.93	1.00
8	410	204	154	143	140	36	5.67	4.28	3.97	3.89	1.00
9	500	140	152	199	164	46	3.04	3.30	4.33	3.57	1.00
10	1000	147	104	158	138	34	4.32	3.06	4.65	4.06	1.00
Average							9.83	6.91	6.95	5.94	1.00

5.5. Comparison with different optimization algorithms

In order to further verify the effectiveness and superiority of the proposed algorithm, this section compares TS-XSMT algorithm with four optimization algorithms, respectively ABC, GA, MA and BEA in Tables 10 and 11.

In Table 10, the average wirelength ratio of the XSMT generated by the four optimization algorithms to our algorithm is 1.017:1.020:1.019:1.028:1, which means the average wirelength obtained by the four algorithms is 1.7%, 2.0%, 1.9%, 2.8% longer than that of ours. Compare with ABC, our algorithm achieves shorter wirelength except for Test 1, 2 and 3 that ABC has the same results with TS-XSMT. For example, for the net with 1000 pins, TS-XSMT algorithm obtains an XSMT with the wirelength only 214990, while the wirelength obtained by ABC is 222537, which is 3.5% longer than the wirelength generated by ours. And compared with GA, MA and BEA, TS-XSMT algorithm can obtain the best wirelength optimization on all the test cases, especially for large-scale nets. In summary, the data of mean wirelength clearly shows that TS-XSMT algorithm is generally superior to the four optimization algorithms in wirelength optimization.

Meanwhile, as shown in Table 11, the standard deviation of our algorithm is undoubtedly the best. Specifically, for Test 1, 2, 3 and 5, TS-XSMT algorithm can reduce the standard deviation to 0. For the rest test cases, the ratio of the average standard deviation of ABC, GA, MA, BEA and ours is 9.83:6.91:6.95:5.94:1. When in Test 7 (100 pins), the ratio between ABC and our algorithm reaches the maximum, that the standard deviation of ABC is 19.07 times TS-XSMT algorithm. Table 11 clarifies that our algorithm has better stability than other optimization algorithms. It shows TS-XSMT algorithm can maintain a stable wirelength optimization in many times of experiments.

The experimental results validate that the proposed algorithm has the superior performance both in stability and wirelength optimization. And among these optimization algorithms, the TS-XSMT algorithm shows a clear advantage.

6. Conclusion

Intelligent Edge Computing plays an important role in the application of Internet of Things, and one of its core features is real-time decision making. Therefore, IC design under Intelligent

Edge Computing should pay more attention to low delay. And in VLSI routing, wirelength is one of the most important indexes affecting the final delay of the chip. Therefore, aiming at SMT routing model and utilizing the X-routing with better wirelength optimization ability, this paper proposes a two-step X-routing Steiner minimum tree construction algorithm based on SLDPSO, which is an efficient intelligent technique in Soft Computing. Firstly, the X-routing is utilized as the interconnected model for the edges of Steiner tree to make full use of routing resources. Secondly, in order to overcome the shortcomings of traditional PSO algorithms that are easy to fall into local extremes, two effective stages, including *SLDPSO Searching* and *Wirelength Reduction*, are proposed to construct a high-quality XSMT. Experimental results prove that the TS-XSMT algorithm can obtain the best routing scheme with shortest wirelength compared with similar work, so as to further reduce the chip delay and can better satisfy the high performance requirements of IC design under Intelligent Edge Computing architecture.

The proposed algorithm can be applied to industrial circuits and achieve the high-quality solutions with the best wirelength which are more suitable for low delay requirement in IC design under IEC architecture, and can be helpful for the construction of VLSI router in EDA design. Based on the best model of X-routing in this paper, the future work will solve the multi-layer X-routing, and then give a complete design of multi-layer X-routing. Finally, an efficient router can be constructed, which can be directly used to solve the multi-layer X-routing problem of VLSI, and it has important theoretical value and practical significance.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] F. Jauro, H. Chiroma, A.Y. Gital, M. Almutairi, S.M. Abdulhamid, J.H. Abawajy, Deep learning architectures in emerging cloud computing architectures: Recent development, challenges and next research trend, Appl. Soft Comput. 96 (2020) 106582, http://dx.doi.org/10.1016/j.asoc.2020.106582.

[2] Z. Ning, P. Dong, X. Wang, J.J.P.C. Rodrigues, F. Xia, Deep reinforcement learning for vehicular edge computing: An intelligent offloading system, ACM Trans. Intell. Syst. Technol. (TIST) 10 (6) (2019) 1–24, http://dx.doi.org/10.1145/3317572.

[3] O. Krestinskaya, A.P. James, L.O. Chua, Neuromemristive circuits for edge computing: A review, IEEE Trans. Neural Netw. Learn. Syst. 31 (1) (2020) 4–23, http://dx.doi.org/10.1109/TNNLS.2019.2899262.

[4] F. Liu, G. Tang, Y. Li, Z. Cai, X. Zhang, T. Zhou, A survey on edge computing systems and tools, Proc. IEEE 107 (8) (2019) 1537–1562, http://dx.doi.org/10.1109/JPROC.2019.2920341.

[5] K.-W. Lin, Y.-S. Lin, Y.-L. Li, R.-B. Lin, A maze routing-based methodology with bounded exploration and path-assessed retracing for constrained multilayer obstacle-avoiding rectilinear steiner tree construction, ACM Trans. Des. Autom. Electron. Syst. (TODAES) 23 (4) (2018) 1–26, http://dx.doi.org/10.1145/3177878.

[6] S.-E.D. Lin, D.H. Kim, Construction of all rectilinear steiner minimum trees on the Hanan grid, in: Proceedings of the 2018 International Symposium on Physical Design, 2018, pp. 18–25, http://dx.doi.org/10.1145/3177540.3178240.

[7] E. Wuerges, 3-step rectilinear minimum spanning tree construction for obstacle-avoiding component-to-component routing, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. (2020) 1, http://dx.doi.org/10.1109/TCAD.2020.2972534.

[8] S. Held, D. Müller, D. Rotter, R. Scheifele, V. Traub, J. Vygen, Global routing with timing constraints, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 37 (2) (2018) 406–419, http://dx.doi.org/10.1109/TCAD.2017.2697964.

[9] W. Guo, X. Huang, PORA: A Physarum-inspired obstacle-avoiding routing algorithm for integrated circuit design, Appl. Math. Model. 78 (2020) 268–286, http://dx.doi.org/10.1016/j.apm.2019.10.027.

[10] G. Liu, W. Zhu, S. Xu, Z. Zhuang, Y.-C. Chen, G. Chen, Efficient VLSI routing algorithm employing novel discrete PSO and multi-stage transformation, J. Ambient Intell. Humaniz. Comput. (2020) http://dx.doi.org/10.1007/s12652-020-02659-8.

[11] H. Bhaumik, S. Bhattacharyya, M.D. Nath, S. Chakraborty, Hybrid soft computing approaches to content based video retrieval: A brief review, Appl. Soft Comput. 46 (2016) 1008–1029, http://dx.doi.org/10.1016/j.asoc.2016.03.022.

[12] Q. Xu, S. Chen, B. Li, Combining the ant system algorithm and simulated annealing for 3D/2D fixed-outline floorplanning, Appl. Soft Comput. 40 (2016) 150–160, http://dx.doi.org/10.1016/j.asoc.2015.10.045.

[13] A. Maity, S. Das, Efficient hybrid local search heuristics for solving the travelling thief problem, Appl. Soft Comput. 93 (2020) 106284, http://dx.doi.org/10.1016/j.asoc.2020.106284.

[14] S. Saleh, K. Ibrahim, M.M. Eiteba, Study of genetic algorithm performance through design of multi-step LC compensator for time-varying nonlinear loads, Appl. Soft Comput. 48 (2016) 535–545, http://dx.doi.org/10.1016/J.ASOC.2016.07.043.

[15] X. Chen, G. Liu, N. Xiong, Y. Su, G. Chen, A survey of swarm intelligence techniques in VLSI routing problems, IEEE Access 8 (2020) 26266–26292, http://dx.doi.org/10.1109/access.2020.2971574.

[16] H. Tang, G. Liu, X. Chen, N. Xiong, A survey on steiner tree construction and global routing for VLSI design, IEEE Access 8 (2020) 68593–68622, http://dx.doi.org/10.1109/ACCESS.2020.2986138.

[17] C.S. Coulston, Constructing exact octagonal steiner minimal trees, in: Proceedings of the 13th ACM Great Lakes Symposium on VLSI, ACM Washington, DC, USA, 2003, pp. 1–6, http://dx.doi.org/10.1145/764808.764810.

[18] C. Chiang, C.-S. Chiang, Octilinear Steiner tree construction, in: The 2002 45th Midwest Symposium on Circuits and Systems, 2002, Vol. 1, MWSCAS-2002, IEEE, 2002, pp. I–603, http://dx.doi.org/10.1109/MWSCAS.2002.1187293.

[19] Q. Zhu, H. Zhou, T. Jing, X.L. Hong, Y. Yang, Spanning graph-based nonrectilinear steiner tree algorithms, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 24 (7) (2005) 1066–1075, http://dx.doi.org/10.1109/TCAD.2005.850862.

[20] J.-T. Yan, Timing-driven octilinear steiner tree construction based on Steiner-point reassignment and path reconstruction, ACM Trans. Des. Autom. Electron. Syst. (TODAES) 13 (2) (2008) 1–18, http://dx.doi.org/10.1145/1344418.1344422.

[21] P. Tu, W. Chow, E.F.Y. Young, Timing driven routing tree construction, in: 2017 ACM/IEEE International Workshop on System Level Interconnect Prediction, SLIP, 2017, pp. 1–8, http://dx.doi.org/10.1109/SLIP.2017.7974908.

[22] T.-Y. Ho, C.-F. Chang, Y.-W. Chang, S.-J. Chen, Multilevel full-chip routing for the X-based architecture, in: Proceedings of the 42nd Design Automation Conference, 2005, pp. 597–602, http://dx.doi.org/10.1109/DAC.2005.193880.

[23] H. Wu, S. Xu, Z. Zhuang, G. Liu, X-Architecture Steiner minimal tree construction based on discrete differential evolution, in: The International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery, Vol. 1074, Springer, 2019, pp. 433–442, http://dx.doi.org/10.1007/978-3-030-32456-8_47.

[24] X. Huang, G. Liu, W. Guo, G. Chen, Obstacle-avoiding octagonal Steiner tree construction based on particle swarm optimization, in: 2013 Ninth International Conference on Natural Computation, ICNC, IEEE, 2013, pp. 539–543, http://dx.doi.org/10.1109/ICNC.2013.6818035.

[25] X. Huang, G. Liu, W. Guo, Y. Niu, G. Chen, Obstacle-avoiding algorithm in X-architecture based on discrete particle swarm optimization for VLSI design, ACM Trans. Des. Autom. Electron. Syst. (TODAES) 20 (2) (2015) 1–28, http://dx.doi.org/10.1145/2699862.

[26] G. Liu, W. Guo, Y. Niu, G. Chen, X. Huang, A PSO-based timing-driven Octilinear Steiner tree algorithm for VLSI routing considering bend reduction, Soft Comput. 19 (5) (2015) 1153–1169, http://dx.doi.org/10.1007/s00500-014-1329-2.

[27] Q. He, G. Cui, X. Zhang, F. Chen, S. Deng, H. Jin, Y. Li, Y. Yang, A game-theoretical approach for user allocation in edge computing environment, IEEE Trans. Parallel Distrib. Syst. 31 (3) (2020) 515–529, http://dx.doi.org/10.1109/TPDS.2019.2938944.

[28] T.-Y. Wu, C.-M. Chen, K.-H. Wang, C. Meng, E.K. Wang, A provably secure certificateless public key encryption with keyword search, J. Chin. Inst. Eng. 42 (1) (2019) 20–28, http://dx.doi.org/10.1080/02533839.2018.1537807.

[29] C.-M. Chen, K.-H. Wang, K.-H. Yeh, B. Xiang, T.-Y. Wu, Attacks and solutions on a three-party password-based authenticated key exchange protocol for wireless communications, J. Ambient Intell. Humaniz. Comput. 10 (2019) 3133–3142, http://dx.doi.org/10.1007/s12652-018-1029-3.

[30] C. Chen, B. Xiang, Y. Liu, K. Wang, A secure authentication protocol for internet of vehicles, IEEE Access 7 (2019) 12047–12057, http://dx.doi.org/10.1109/ACCESS.2019.2891105.
[31] Y. Ngoko, C. Cérin, An edge computing platform for the detection of acoustic events, in: 2017 IEEE International Conference on Edge Computing, EDGE, IEEE, 2017, pp. 240–243, http://dx.doi.org/10.1109/IEEE.EDGE.2017.44.
[32] J. Ren, G. Yu, Y. He, G.Y. Li, Collaborative cloud and edge computing for latency minimization, IEEE Trans. Veh. Technol. 68 (5) (2019) 5031–5044, http://dx.doi.org/10.1109/TVT.2019.2904244.
[33] A.K. Sangaiah, D.V. Medhane, T. Han, M.S. Hossain, G. Muhammad, Enforcing position-based confidentiality with machine learning paradigm through mobile edge computing in real-time industrial informatics, IEEE Trans. Ind. Inf. 15 (7) (2019) 4189–4196, http://dx.doi.org/10.1109/TII.2019.2898174.
[34] D.V. Medhane, A.K. Sangaiah, M.S. Hossain, G. Muhammad, J. Wang, Blockchain-enabled distributed security framework for next-generation IoT: An edge cloud and software-defined network-integrated approach, IEEE Internet Things J. 7 (7) (2020) 6143–6149, http://dx.doi.org/10.1109/JIOT.2020.2977196.
[35] L. Hu, Y. Miao, G. Wu, M.M. Hassan, I. Humar, iRobot-Factory: An intelligent robot factory based on cognitive manufacturing and edge computing, Future Gener. Comput. Syst. 90 (2019) 569–577, http://dx.doi.org/10.1016/j.future.2018.08.006.
[36] R. Wang, M. Li, L. Peng, Y. Hu, M.M. Hassan, A. Alelaiwi, Cognitive multi-agent empowering mobile edge computing for resource caching and collaboration, Future Gener. Comput. Syst. 102 (2020) 66–74, http://dx.doi.org/10.1016/j.future.2019.08.001.
[37] T. Alfakih, M.M. Hassan, A. Gumaei, C. Savaglio, G. Fortino, Task offloading and resource allocation for mobile edge computing by deep reinforcement learning based on SARSA, IEEE Access 8 (2020) 54074–54084, http://dx.doi.org/10.1109/ACCESS.2020.2981434.
[38] J. Pan, C. Lee, A. Sghaier, M. Zeghid, J. Xie, Novel systolization of subquadratic space complexity multipliers based on Toeplitz matrix–vector product approach, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 27 (7) (2019) 1614–1622, http://dx.doi.org/10.1109/TVLSI.2019.2903289.
[39] T. Ma, Q. Liu, J. Cao, Y. Tian, A. Al-Dhelaan, M. Al-Rodhaan, LGIEM: Global and local node influence based community detection, Future Gener. Comput. Syst. 105 (2020) 533–546, http://dx.doi.org/10.1016/j.future.2019.12.022.
[40] J.-S. Pan, P. Hu, S.-C. Chu, Novel parallel heterogeneous meta-heuristic and its communication strategies for the prediction of wind power, Processes 7 (2020) http://dx.doi.org/10.3390/pr7110845.
[41] F. Yang, P. Wang, Y. Zhang, L. Zheng, J. Lu, Survey of swarm intelligence optimization algorithms, in: 2017 IEEE International Conference on Unmanned Systems, ICUS, IEEE, 2017, pp. 544–549, http://dx.doi.org/10.1109/ICUS.2017.8278405.
[42] T. Arora, M.E. Moses, Ant Colony Optimization for power efficient routing in manhattan and non-manhattan VLSI architectures, in: 2009 IEEE Swarm Intelligence Symposium, IEEE, 2009, pp. 137–144, http://dx.doi.org/10.1109/SIS.2009.4937856.
[43] T. Arora, M. Moses, Using ant colony optimization for routing in VLSI chips, in: AIP Conference Proceedings, Vol. 1117, American Institute of Physics, 2009, pp. 145–156, http://dx.doi.org/10.1063/1.3130617.
[44] K. Pandiaraj, P. Sivakumar, R. Sridevi, Minimizationof wirelength in 3d IC routing by using differential evolution algorithm, in: 2017 IEEE International Conference on Electrical, Instrumentation and Communication Engineering, ICEICE, IEEE, 2017, pp. 1–5, http://dx.doi.org/10.1109/ICEICE.2017.8191950.
[45] S. Vijayakumar, J.G. Sudhakar, G. Muthukumar, T.A.A. Victoire, A differential evolution algorithm for restrictive channel routing problem in VLSI circuit design, in: 2009 World Congress on Nature & Biologically Inspired Computing, NaBIC, IEEE, 2009, pp. 1258–1263, http://dx.doi.org/10.1109/NABIC.2009.5393755.
[46] P. Bhattacharya, A. Khan, S.K. Sarkar, A global routing optimization scheme based on ABC algorithm, in: Advanced Computing, Networking and Informatics, Vol. 2, Springer, 2014, pp. 189–197, http://dx.doi.org/10.1007/978-3-319-07350-7_21.
[47] A. Mohd Nasir, H. Ahmad Fariz, I. Abdul Halim, H.B. Hassrizal, M.A. Muhamad Safwan, Z.A. Amar Faiz, A Firefly Algorithm approach for routing in VLSI, in: International Symposium on Computer Applications and Industrial Electronics, IEEE, 2012, pp. 43–47, http://dx.doi.org/10.1109/ISCAIE.2012.6482066.
[48] G. Liu, G. Chen, W. Guo, DPSO based octagonal Steiner tree algorithm for VLSI routing, in: 2012 IEEE Fifth International Conference on Advanced Computational Intelligence, ICACI, IEEE, 2012, pp. 383–387, http://dx.doi.org/10.1109/ICACI.2012.6463191.
[49] G. Liu, Z. Chen, W. Guo, G. Chen, Self-adapting PSO algorithm with efficient hybrid transformation strategy for X-Architecture Steiner minimal tree construction algorithm, Pattern Recognit. Artif. Intell. 31 (5) (2018) 398–408, http://dx.doi.org/10.16451/j.cnki.issn1003-6059.201805002 (in Chinese).
[50] G. Liu, Z. Chen, Z. Zhuang, W. Guo, G. Chen, A unified algorithm based on HTS and self-adapting PSO for the construction of octagonal and rectilinear SMT, Soft Comput. 24 (6) (2020) 3943–3961, http://dx.doi.org/10.1007/s00500-019-04165-2.
[51] Y. Shi, R.C. Eberhart, Empirical study of particle swarm optimization, in: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), Vol. 3, IEEE, 1999, pp. 1945–1950, http://dx.doi.org/10.1109/cec.1999.785511.
[52] J.C. Bansal, P. Singh, M. Saraswat, A. Verma, S.S. Jadon, A. Abraham, Inertia weight strategies in particle swarm optimization, in: 2011 Third World Congress on Nature and Biologically Inspired Computing, IEEE, 2011, pp. 633–640, http://dx.doi.org/10.1109/NaBIC.2011.6089659.
[53] X. Xu, H. Rong, M. Trovati, M. Liptrott, N. Bessis, CS-PSO: chaotic particle swarm optimization algorithm for solving combinatorial optimization problems, Soft Comput. 22 (3) (2018) 783–795, http://dx.doi.org/10.1007/s00500-016-2383-8.
[54] Y. Feng, G.-F. Teng, A.-X. Wang, Y.-M. Yao, Chaotic inertia weight in particle swarm optimization, in: Second International Conference on Innovative Computing, Informatio and Control, ICICIC 2007, IEEE, 2007, p. 475, http://dx.doi.org/10.1109/ICICIC.2007.209.
[55] R. Cheng, Y. Jin, A social learning particle swarm optimization algorithm for scalable optimization, Inform. Sci. 291 (2015) 43–60, http://dx.doi.org/10.1016/j.ins.2014.08.039.
[56] X. Zhang, X. Wang, Q. Kang, J. Cheng, Differential mutation and novel social learning particle swarm optimization algorithm, Inform. Sci. 480 (2019) 109–129, http://dx.doi.org/10.1016/j.ins.2018.12.030.
[57] D. Warme, P. Winter, M. Zachariasen, GeoSteiner software for computing Steiner trees, 2003, http://geosteiner.net.
[58] G. Liu, G. Chen, W. Guo, Z. Chen, DPSO-based rectilinear steiner minimal tree construction considering bend reduction, in: 2011 Seventh International Conference on Natural Computation, Vol. 2, IEEE, 2011, pp. 1161–1165, http://dx.doi.org/10.1109/ICNC.2011.6022221.