X-architecture Steiner minimal tree algorithm based on multi-strategy optimization discrete differential evolution

Genggeng Liu [1], **Liliang Yang** [1], **Saijuan Xu** [2], **Zuoyong Li** [3], **Yeh-Cheng Chen** [4], **Chi-Hua Chen** [Corresp. 1]

[1] College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China

[2] Department of Information Engineering, Fujian Business University, Fuzhou, China

[3] Fujian Provincial Key Laboratory of Information Processing and Intelligent Control, Minjiang University, Fuzhou, China

[4] Department of Computer Science, University of California, Davis, Davis, United States

Corresponding Author: Chi-Hua Chen
Email address: chihua0826@gmail.com

Global routing is an important link in Very Large Scale Integration (VLSI) design. As the best model of global routing, X-architecture Steiner Minimal Tree (XSMT) has a good performance in wire length optimization. XSMT belongs to non-Manhattan structural model, and its construction process cannot be completed in polynomial time, so the generation of XSMT is an NP hard problem. In this paper, an X-architecture Steiner Minimal Tree algorithm based on Multi-strategy optimization Discrete Differential Evolution (XSMT-MoDDE) is proposed. Firstly, an effective encoding strategy, a fitness function of XSMT, and an initialization strategy of population are proposed to record the structure of XSMT, evaluate the cost of XSMT and obtain better initial particles, respectively. Secondly, elite selection and cloning strategy, multiple mutation strategies, and adaptive learning factor strategy are presented to improve the search process of discrete differential evolution algorithm. Thirdly, an effective refining strategy is proposed to further improve the quality of the final Steiner tree. Finally, the results of the comparative experiments prove that XSMT-MoDDE can get the shortest wire length so far, and achieve a better optimization degree in the larger-scale problem.

X-architecture Steiner minimal tree algorithm based on multi-strategy optimization discrete differential evolution

Genggeng Liu[1], Liliang Yang[1], Saijuan Xu[2], Zuoyong Li[3], Yeh-Cheng Chen[4], and Chi-Hua Chen[1]

[1]**College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China**
[2]**Department of Information Engineering, Fujian Business University, Fuzhou, China**
[3]**Fujian Provincial Key Laboratory of Information Processing and Intelligent Control, Minjiang University, Fuzhou, China**
[4]**Department of Computer Science, University of California, Davis, CA, USA**

Corresponding author:
Chi-Hua Chen[1]

Email address: chihua0826@gmail.com

ABSTRACT

Global routing is an important link in Very Large Scale Integration (VLSI) design. As the best model of global routing, X-architecture Steiner Minimal Tree (XSMT) has a good performance in wire length optimization. XSMT belongs to non-Manhattan structural model, and its construction process cannot be completed in polynomial time, so the generation of XSMT is an NP hard problem. In this paper, an X-architecture Steiner Minimal Tree algorithm based on Multi-strategy optimization Discrete Differential Evolution (XSMT-MoDDE) is proposed. Firstly, an effective encoding strategy, a fitness function of XSMT, and an initialization strategy of population are proposed to record the structure of XSMT, evaluate the cost of XSMT and obtain better initial particles, respectively. Secondly, elite selection and cloning strategy, multiple mutation strategies, and adaptive learning factor strategy are presented to improve the search process of discrete differential evolution algorithm. Thirdly, an effective refining strategy is proposed to further improve the quality of the final Steiner tree. Finally, the results of the comparative experiments prove that XSMT-MoDDE can get the shortest wire length so far, and achieve a better optimization degree in the larger-scale problem.

INTRODUCTION

At present, VLSI technology is developing at a high speed. Initially, the model to solve global routing problem was based on Manhattan structure (Held et al., 2017; Siddiqi and Sait, 2017; Chu and Wong, 2007). There are two ways to connect each pin in this structure, which are horizontal direction and vertical direction. In the development of this structure, limitation of the interconnect wire length optimization appeared, and in the actual situation, there is still a lot of optimization space for wire length of Steiner Minimum Tree (SMT). Wire length has a decisive influence on the chip performance. Based on this situation, non-Manhattan structure, which can make full use of the routing resources and shorten the wire length, has become the mainstream model of global routing.

X-architecture Steiner Minimal Tree (XSMT) is a representative model of non-Manhattan structure (Coulston, 2003; Chiang and Chiang, 2002). SMT problem is to find a minimum connection tree under a given set of pins by introducing additional Steiner points (Liu et al., 2014b). Because of SMT cannot be constructed in polynomial time, how to quickly and effectively construct an SMT is a key issue to be solved in VLSI manufacturing process. Heuristic search algorithm has a strong ability to solve NP-hard problem (Liu et al., 2018, 2020a). As a typical heuristic search algorithm, Differential Evolution (DE) algorithm has shown good optimization effect in many practical engineering problems. Therefore, based on DE algorithm, this paper designs relevant strengthening strategies to construct XSMT.

DE is a global optimization algorithm proposed by Storn and Price in 1997 (Storn and Price, 1997).

Each particle in DE corresponds to a solution vector, and the main process is composed of three steps: mutation, crossover, and selection. DE algorithm has many advantages, such as robustness, reliability, simple algorithm structure and few control parameters, etc., and it has been widely applied in global optimization (Zhao et al., 2020; Ge et al., 2017), artificial intelligence (Brest et al., 2006; Zhang et al., 2015), bioinformatics (Zhang et al., 2020), and other fields (Wu et al., 2019; Xue et al., 2014). Generation strategy of trial vector and setting method of control parameters will greatly affect the performance of DE algorithm. Many scholars have improved DE algorithm in these directions, and it has made great progress in recent years. DE was originally proposed for continuous problems and can not be directly used to solve discrete problems such as XSMT, therefore, this paper explores and formulates a Discrete Differential Evolution (DDE) algorithm for solving XSMT problem.

For this reason, this paper proposes X-architecture Steiner Minimal Tree algorithm based on Multi-strategy optimization Discrete Differential Evolution (XSMT-MoDDE). Firstly, we design an encoding strategy, a fitness function of XSMT, and a population initialization strategy based on Prim algorithm for DDE algorithm to record the structure of XSMT, evaluate XSMT and obtain high quality initial solution, respectively. Secondly, we design an elite selection and cloning strategy, a multiple mutation strategy, and an adaptive learning factor strategy to optimize the search process. At the end of the algorithm, an effective refining strategy is proposed to improve the quality of the final XSMT.

RELATED WORK

Research status of RSMT and XSMT

Optimizing the wire length of SMT is a popular research direction, and there are many important research achievements. In Liu et al. (2011), Rectilinear Steiner Minimal Tree (RSMT) based on Discrete Particle Swarm Optimization (DPSO) algorithm was proposed to effectively optimize the average wire length (Liu et al., 2011). Liu et al. (2014a) proposed a multi-layer obstacle avoidance RSMT construction method based on geometric reduction method (Liu et al., 2014a). Zhang et al. (2016) proposed a heuristic for constructing a RSMT with slew constraints to maximize routing resources over obstacles (Zhang et al., 2016).

Teig (2002) adopted XSMT, which is superior to RSMT in terms of average wire length optimization (Teig, 2002). In Zhu et al. (2005), an XSMT construction method was proposed by side substitution and triangle contraction methods (Zhu et al., 2005). Liu et al. (2020c) constructed a multi-layer global router based on the X-architecture. Compared with other global routers, it had better performance in overflow and wire length (Liu et al., 2020c). Liu et al. (2014b) proposed a PSO-based multi-layer obstacle-avoiding XSMT, which used an effective penalty mechanism to help particles to avoid obstacles (Liu et al., 2014b). In Liu et al. (2020b), a novel DPSO and multi-stage transformation were used to construct XSMT and RSMT. The simulation results on industrial circuits showed that this method could obtain high-quality routing solutions (Liu et al., 2020b).

The present situation of DE and DDE algorithm

DE algorithm has high efficiency and powerful search ability in solving continuous optimization problems. In the past 20 years after its emergence, many scholars have proposed improved versions of DE algorithm. These improvements better balance the exploitation and exploration ability of DE, and show strong optimization ability on many problems.

An Self-adaptive DE (SaDE) algorithm was proposed in Qin et al. (2008). In different stages of the evolution process, the value of control parameters is adjusted according to experience, which saves the trial and error cost of developers in the process of adjusting parameters (Qin et al., 2008). Rahnamayan et al. (2008) proposed an algorithm for accelerating DE, using opposition-based DE and opposition-based learning methods to initialize population and realize generation jumping to accelerate convergence of DE (Rahnamayan et al., 2008). Subsequently, Wang et al. (2011a) proposed an improved version of accelerated DE, which could be used to solve high-dimensional problems (Wang et al., 2011a). Wang et al. (2011b) proposed Composite DE (CoDE). The algorithm proposed three generation strategies of trial vector and three control parameter settings, and randomly combined the generation strategies and control parameters. The experimental results showed that the algorithm had strong competitiveness (Wang et al., 2011b). Wang et al. (2015) combined adaptive DE algorithm with Back Propagation Neural Network (BPNN) to improve its prediction accuracy (Wang et al., 2015).

Pin	p_1	p_2	p_3	p_4	p_5
Coordinate	$(01,22)$	$(05,05)$	$(12,10)$	$(18,03)$	$(22,16)$

Table 1. Coordinate information of pins

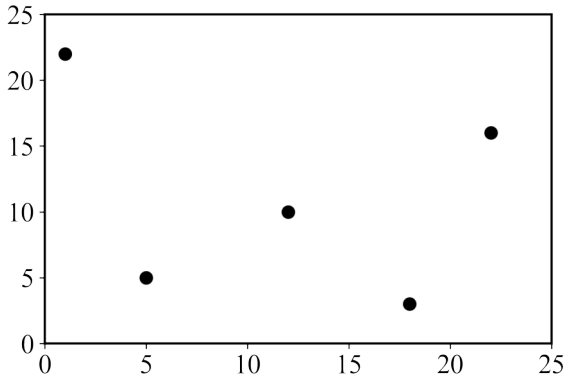


Figure 1. Distribution of pins.

DDE algorithm is a derivative of DE, which can solve discrete problems. Many existing results have applied DDE algorithm to solve practical problems. In Pan et al. (2008), DDE was used to solve the permutation flow shop scheduling problem with the total flow time criterion. For the total flow time criterion, its performance is better than the PSO algorithm proposed by predecessors (Pan et al., 2008). In Tasgetiren et al. (2010), an ensemble of DDE (eDDE) algorithms with parallel populations was presented. eDDE uses different parameter sets and crossover operators for each parallel population, and each parallel parent population has to compete with the offspring populations produced by this population and all other parallel populations (Tasgetiren et al., 2010). Deng and Gu (2012) presented a Hybrid DDE (HDDE) algorithm for the no-idle permutation flow shop scheduling problem with makespan criterion. A new acceleration method based on network representation was proposed and applied to HDDE, and the local search of the inserted neighborhood in HDDE was effectively improved to balance global search and local development (Deng and Gu, 2012).

PRELIMINARIES

XSMT problem

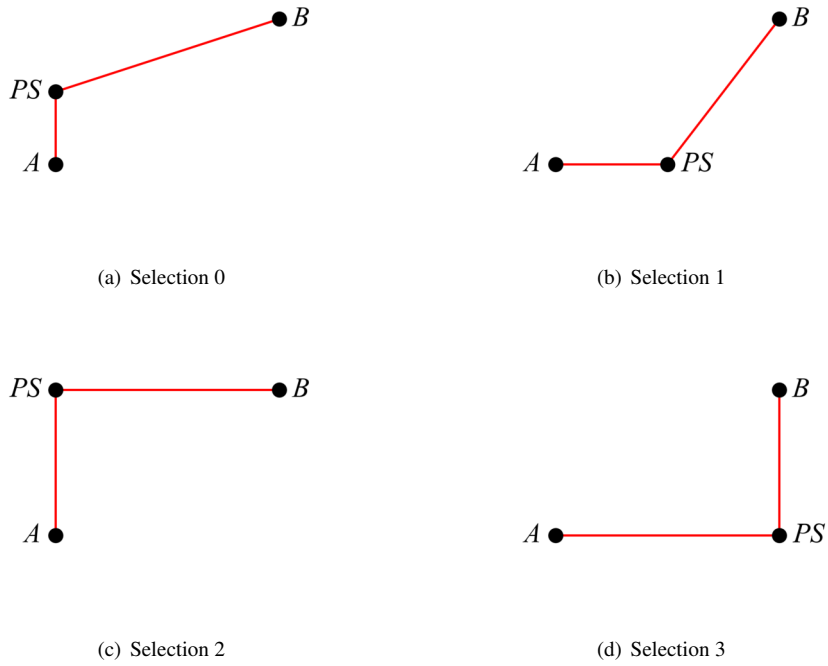
Unlike the traditional Manhattan structure, which only has horizontal and vertical connections, two connection methods of 45° and 135° are added to the XSMT problem (Liu et al., 2012, 2015). This paper introduces the concept of Pseudo-Steiner (PS) point (Definition 1). The PS point exists in two interconnected pins. The fixation of PS point determines the connection method (Definition 2-5) of two pins.

An example of XSMT problem model is as follows. In a given set of pins $\{p_1, p_2, ..., p_n\}$, p_i represents the $i-th$ pin to be connected, and the corresponding coordinate is (x_i, y_i). Given 5 pins, the corresponding coordinates are shown in Table 1, and the corresponding pin layout is shown in Figure 1.

Definition 1. *Pseudo-Steiner point. Except for pin points, other join points are called Pseudo-Steiner points, denoted as PS points.*

Definition 2. *Selection 0. As shown in Figure 2(a), draw the vertical edge from A to point PS, and then draw the X-architecture edge from PS to B.*

Definition 3. *Selection 1. As shown in Figure 2(b), draw the X-architecture edge from A to point PS, and then draw the vertical edge from PS to B.*



(a) Selection 0 (b) Selection 1

(c) Selection 2 (d) Selection 3

Figure 2. Four selections for connection method.

Definition 4. *Selection 2. As shown in Figure 2(c), draw the vertical edge from A to PS, and then draw the horizontal edge from PS to B.*

Definition 5. *Selection 3. As shown in Figure 2(d), draw the horizontal edge from A to PS, and then draw the vertical edge from PS to B.*

Differential evolution algorithm

DE algorithm is a heuristic search algorithm based on modern intelligence theory. The particles of population cooperate and compete with each other to determine the search direction.

The update process of DE

Initialization of the population: N particles are randomly generated, and the dimension of each particle is D. For example, X_i^0 represents the particle i, X_L is the lower limit of D-dimensional particles, and X_H is the upper limit of D-dimensional particles. The corresponding initialization method is as follows:

$$X_i^0 = X_L + randam(0,1) \times (X_H - X_L) \tag{1}$$

Mutation operator: In the process of the g-th iteration, mutation operator randomly select three particles X_a^g, X_b^g, and X_c^g in the population which are different from each other, and generate particles V_i^g according to the following mutation formula:

$$V_i^g = X_a^g + F \times (X_b^g - X_c^g) \tag{2}$$

where F is a learning factor, $F \in [0,2]$.

Crossover operator: In the process of crossover, the value of each dimension is selected from Particle X_i^g or Particle V_i^g. The probability of selection is cr. The formula of crossover is as follows:

$$u_i^j = \begin{cases} v_i^j & rand(0,1) \le cr \\ x_i^j & else \end{cases} \tag{3}$$

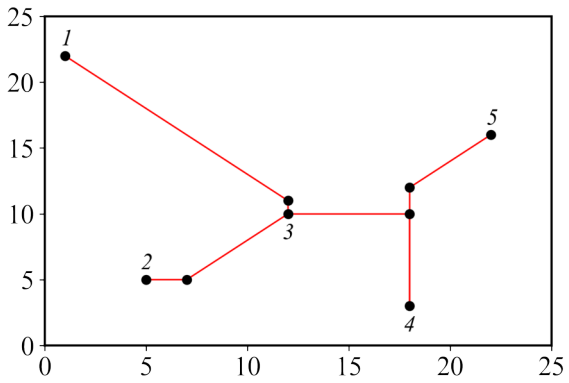


Figure 3. Steiner tree.

where j represents the dimension, cr is the crossover probability, $cr \in [0,1]$.

Selection operator: It adopts greedy strategy in the process of selection, that is, selecting the particle with the optimal adaptive value. The formula is as follows:

$$X_i^{(g+1)} = \begin{cases} V_i^g & f(V_i^g) < f(X_i^g) \\ X_i^g & else \end{cases} \tag{4}$$

where the value of Function $f(X)$ represents the fitness value of Particle X, and the fitness function definitions for each problem are different.

The flow of DE algorithm

Step 1. Initialize the population according to Formula 1, and initialize the parameters of DE algorithm.

Step 2. Calculate the fitness value of each particle in the population according to fitness function.

Step 3. During each iteration, mutation operation is performed on particles according to Formula 2 or other mutation operators to produce mutated particles.

Step 4. Check whether the algorithm reaches the termination condition. If so, the algorithm is terminated. Otherwise, return to Step 2 and update the related parameters.

XSMT-MODDE ALGORITHM

Encoding strategy

Property 1. *The encoding strategy of edge-point pairs is suitable for DDE algorithm, and it can well record the structure of XSMT.*

Suppose there are n pin points in the pin graph, and the corresponding Steiner tree has $n-1$ edges and $n-1$ PS points. Number each pin, determine an edge by recording two endpoints, and add a bit to record selection method of edge. Finally, a bit is added at the end to represent the fitness value of the particle, and the final encoding length is $3 \times (n-1)+1$. The Steiner tree corresponding to pins in Table 1 is shown in Figure 3, and the corresponding encoding is: 1 3 1 2 3 0 4 5 0 3 4 3 46.284.

Fitness function

Property 2. *The wire length of XSMT is a key factor that affects global routing results, and the fitness value based on the wire length of XSMT can make the algorithm go in the direction of optimal wire length to the greatest extent.*

In an edge set of a XSMT, all edges belong to one of the following four types: horizontal, vertical, 45° diagonal and 135° diagonal. Rotate a 45° diagonal counterclockwise 45° to form a vertical line and a 135° diagonal counterclockwise 45° to form a horizontal line, so that the four types of edges can be replaced by two types. Make the starting point number of all edges smaller than the ending point number, and then sort all edges according to the starting point number, and subtract the overlapping part of the edges. At this time, the total wire length of XSMT can be obtained.

Algorithm 1 Initialization strategy based on Prim algorithm

Require: V, N
Ensure: P
1: **function** PRIMALGORITHM(V)
2: $s \leftarrow random()/(maxnum+1) \times n+1$
3: $U \leftarrow \{s\}$
4: $T \leftarrow \emptyset$
5: **while** $(U != V)$ **do**
6: choose point $i \in U$
7: $mincost \leftarrow \infty$
8: **for** $k \in V-U$ **do**
9: **if** $cost(i,k) < mincost$ **then**
10: $mincost \leftarrow cost(i,k)$
11: $j \leftarrow k$
12: **end if**
13: **end for**
14: $T \cup \{(i,j)\}$
15: $U \cup \{j\}$
16: **end while**
17: **return** T
18: **end function**
19:
20: **function** GENERATEPOPULATION(V, N)
21: **for** $i \leftarrow 1$ to N **do**
22: $T \leftarrow$PRIMALGORITHM(V)
23: $P \cup \{T\}$
24: **end for**
25: **return** P
26: **end function**

The excellence of XSMT is determined by the total wire length. The smaller the wire length is, the higher the excellence of XSMT will be. Therefore, fitness value measured by XSMT-MoDDE is total wire length of particle. The fitness function of XSMT-MoDDE is shown in Formula 5.

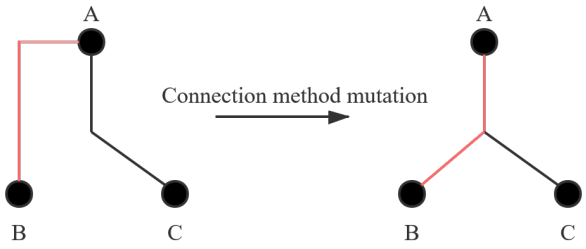
$$fitness(T_x) = \sum_{e_i \in T_x} length(e_i) \tag{5}$$

Initialization

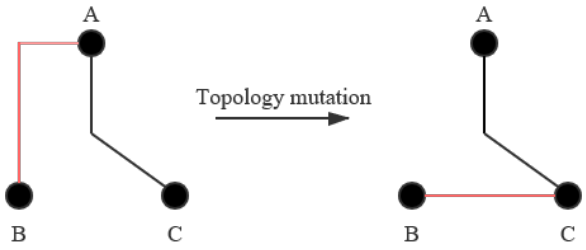
Property 3. *Prim algorithm can search an edge subset, which not only includes all the vertices in a connected graph, but also minimizes the sum of the weights of all the edges in subset. Selecting different starting points can get the same weight but different edge subsets. Prim algorithm is used to initialize population, so that particles in population have diversity and the solution space can be reduced at the same time.*

Traditional DE algorithm directly uses Formula 1 to initialize the population. However, for XSMT, if the random strategy is used to initialize each particle (i.e., randomly select a point as root, and use backtracking method to randomly select edges to build a legal tree), will lead to the problem that the solution space is too large to converge well. Therefore, this paper uses Prim algorithm to construct Minimum Spanning Tree (MST) to initialize population. The weight of each edge in MST is determined by Manhattan distance between each two pins. Each particle randomly selects a starting point s to generate a MST and randomly select a connection method for each edge of MST.

The relevant pseudo code is shown in Algorithm 1, where T is edge set of MST, s is starting point, U is point set of MST, V is pin set, P is population, and N is population size. From Lines 1-18 is the function to generate MST. Lines 2-3 randomly select a starting point s and add it to the set U. Line 4 initializes the edge set T. Line 6 selects a visited point i from the set U, and Line 7 sets the minimum cost to infinity. Lines 8-13 select a unvisited point j from the adjacent points of point i, the edge ij with



(a) Connection method mutation



(b) Topology mutation

Figure 4. Two ways of mutation

the least cost will be selected and added to set T, and the point j is marked as visited and added to set U. The MST algorithm ends when the set U is the same as the set V, and Line 17 returns a randomly generated MST. Lines 21-24 construct the population, and the initial particle is an MST generated by function PRIMALGORITHM.

Elite selection and cloning strategy

Property 4. *This strategy proposes two particle mutation strategies based on set, which can mutate elite particles in a very short time. The elite particles are cloned and mutated, and the optimal particle is selected based on greedy strategy to construct a elite buffer with high quality in a short time.*

Brief description

The elite selection and cloning strategy consists of four steps: selection, cloning, mutation, and extinction. Part of particles in the population are selected as elite particles, and then the elite particles are cloned to form cloned population. Cloned particles randomly mutate into mutated particles. Mutated particles are selected to enter the elite buffer according to extinction strategy. The elite buffer has the same size as the population and participates in the subsequent process of DE.

The elite selection and cloning strategy can effectively expand the search range of DDE, improve the global search ability of the algorithm, avoid falling into local peaks to a certain extent, and prevent the algorithm from premature convergence.

Algorithm flow

(1) Selection: Sort population according to fitness value, and select the first n particles to form an elite population, $n = k \times N$. k is elite ratio, and the best result can be obtained when k is selected as 0.2 after experimental verification.

(2) Cloning: Clone the particles of the elite population to form a cloned population C. The number of cloned particles is calculated according to Formula 6.

$$N_i = round\left(\frac{N}{i}\right) \tag{6}$$

Algorithm 2 Elite selection and cloning strategy

```
Require: P, N
Ensure: E
1: function SELECTION(P)
2:     n ← k × N
3:     S ← ∅
4:     H ← heap(P)
5:     for i ← 1 to n do
6:         S ∪ H.top()
7:     end for
8:     return S, n
9: end function
10:
11: function CLONEMUTATIONANDEXTINCTION(S, n)
12:     E ← ∅
13:     for i ← 1 to n do
14:         M ← ∅
15:         for j ← 1 to n/i do
16:             method ← random(0, 1)
17:             if method == 0 then m ← connection_method_mutation()
18:             else m ← topology_mutation()
19:             end if
20:             M ∪ m
21:         end for
22:         H1 ← heap(M)
23:         H2 ← heap(P)
24:         if H1.top() < H2.top() then E ∪ H1.top()
25:         end if
26:     end for
27:     return E
28: end function
```

where i is rank of the particle in original population, and $round()$ is rounding down function.

(3) Mutation: The mutation strategy adopts connection method mutation or topology mutation, and two strategies are shown in Figure 4. Each cloned particle is assigned to a mutation strategy to form a mutated particle.

For particles that adopt connection method, randomly select a edges, and the value of a is determined according to the number of edges, as shown in Formula 7, where n is the number of pins. Then change the connection method of the selected edge.

$$a = \max\left\{1, round\left(\frac{n-1}{10}\right)\right\} \tag{7}$$

For particles that adopt topology mutation, one edge is randomly disconnected in XSMT to form two sub-XSMTs, and then respectively select a point from the two sub-XSMTs to connect. This process adopts the idea of Disjoint Set Union (DSU) to ensure that a legal tree is obtained after mutation.

(4) Extinction: Select the trial elite particle m_{best} with the best fitness value in the mutated population. If $f(m_{best})$ is better than $f(g_{best})$, then m_{best} will be added to the elite buffer, and all other particles will die, otherwise, all particles in the mutation population will die. If the elite buffer is full, the particle with the worst fitness value will be popped and new particle will be pushed.

The pseudo code of the elite selection and cloning strategy is shown in Algorithm 2, where S represents elite population, M represents mutated population, the inputs are Population P and its size, and the output E represents the elite buffer. Lines 1-9 are selection function, Line 2 calculates the number n of elite particles, Line 3 initializes the Set S, Line 4 establishes a minimum heap according to the fitness value of

the population particles, and Lines 5-6 take n elite particles from the top of the minimum heap in turn. Lines 11-28 are the processes of cloning, mutation and extinction. Line 12 initializes Set E, Line 14 initializes Set M, Line 15-20 are cloning and mutation process, Line 15 clones elite particles, Line 16 selects a mutation strategy randomly, and Line 20 adds mutated elite particles to Set M. Lines 22-23 construct two minimum heaps through Set P and Set M. Line 24 compares the tops of the two minimum heaps to determine whether the trial elite particles are saved or died.

Novel multiple mutation strategy

Property 5. *The three novel mutation strategies proposed in this paper introduce the idea of set operations. Under the premise of reasonable computing time, through adjusting edge set of current particle and edge set of other particle, some substructures in XSMT are changed to search for a better combination of substructures.*

In DE algorithm, there are six commonly used mutation strategies (Epitropakis et al., 2011), and each strategy uses different basis vectors and differential vectors. The mutation formulas are shown below.

$$V_i^g = X_{r1}{}^g + F(X_{r2}{}^g - X_{r3}{}^g) \tag{8}$$

$$V_i^g = X_{r1}{}^g + F_1(X_{r2}{}^g - X_{r3}{}^g) + F_2(X_{r4}{}^g - X_{r5}{}^g) \tag{9}$$

$$V_i^g = X_{best}{}^g + F(X_{r1}{}^g - X_{r2}{}^g) \tag{10}$$

$$V_i^g = X_{best}{}^g + F_1(X_{r1}{}^g - X_{r2}{}^g) + F_2(X_{r3}{}^g - X_{r4}{}^g) \tag{11}$$

$$V_i^g = X_i^g + F(X_{best}{}^g - X_i^g) \tag{12}$$

$$V_i^g = X_{r0}{}^g + F_1(X_{best}{}^g - X_{r0}{}^g) + F_2(X_{r1}{}^g - X_{r2}{}^g) \tag{13}$$

where X_r^g represents a random particle in population, X_{best}^g represents the global optimal solution, and F represents learning factor.

Two operating rules

In XSMT-MoDDE algorithm, a particle represents a XSMT. Addition and subtraction operations in the above mutation formulas cannot be directly used in discrete problems. This paper defines two new calculation methods (Definition 6-7).

A is the edge set of particle X_1, B is the edge set of particle X_2, and the full set is $A \cup B$. There are two definitions as follows:

Definition 6. *$A \odot B$. $\odot$ is expressed as finding the symmetric difference of A and B, which is $(A \cup B) - (A \cap B)$, as shown in Figure 5(a).*

Definition 7. *$A \oplus B$. First calculate Set C, $C = A - B$, and then add the edges of Set B to Set C until Set C can form a legal tree, as shown in Figure 5(b).*

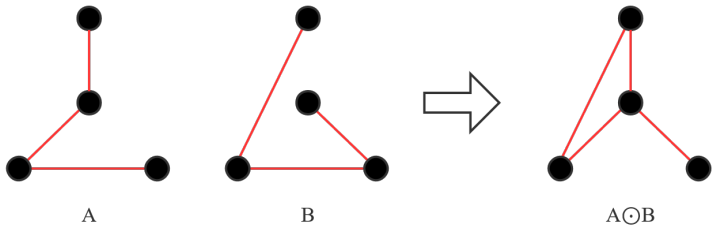
Three mutation strategies

In Mutation Strategy 1, basis vector is selected as current particle, and there are two differential vectors. The differential vector of the first stage is generated by the difference between the current particle and the corresponding local historical optimal particle, and Particle T is obtained by Formula 14. The differential vector in the second stage is generated by the difference between Particle T and the global optimal particle, and target mutated Particle V_i^g is obtained by Formula 15.

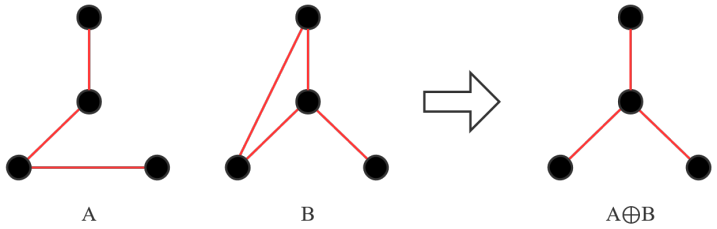
$$T = X_i^g \oplus F\left(X_{pbest}^g \odot X_i^g\right) \tag{14}$$

$$V_i^g = T \oplus F\left(X_{gbest}^g \odot T\right) \tag{15}$$

In Mutation Strategy 2, basis vector is still current particle, and there are two differential vectors. The differential vector in the first stage is generated by the difference between random particle and the corresponding local historical optimal particle, and Particle T is calculated by Formula 16. The differential vector in the second stage is generated by the difference between the random particle and global optimal particles, and target mutated Particle V_i^g is obtained by Formula 17.



(a) $A \odot B$



(b) $A \oplus B$

Figure 5. Operation process of two new operators

$$T = X_i^g \oplus F\left(X_{pbest}^g \odot X_r^g\right) \tag{16}$$

$$V_i^g = T \oplus F\left(X_{gbest}^g \odot X_r^g\right) \tag{17}$$

In Mutation Strategy 3, basis vector is current particle, and the differential vector is generated by the difference between the current particle and random particle in the population, and the mutated Particle V_i^g is obtained by Formula 18.

$$V_i^g = X_i^g \oplus F\left(X_i^g \odot X_r^g\right) \tag{18}$$

Mutation Strategy 1 can make particles obtain the partial structure of global optimal particle and the historical local optimal particle, and inherit the characteristics of the two optimal particles, which is a greedy strategy. The implementation of Mutation Strategy 3 can expand the search space and make the mutation direction completely get rid of the structure of the optimal particles, which is suitable for the early stage of iteration and increases the exploration ability of the algorithm. The exploratory ability of Mutation Strategy 2 is between Mutation Strategy 1 and Mutation Strategy 3.

In multiple mutation strategy, the iterative process is divided into two stages by setting a threshold. Three mutation strategies in the early stage are selected with equal probability, and the Mutation Strategy 3 is cancelled in the later stage. The pseudo-code of multiple mutation strategy is shown in Algorithm 3, where P represents population, N represents the size of the population, m represents the number of iterations, t represents threshold, and V represents mutated population. Line 5 judges whether the current iteration is in the early stage of the iteration. If it is in the early stage of the iteration, Mutation Strategy 1, Mutation Strategy 2, and Mutation Strategy 3 are adopted. Line 6 determines whether the current iteration is in the later stage of the iteration. If it is in the latter stage, Mutation Strategy 1 and Mutation Strategy 2 are adopted.

Adaptive learning factor

Property 6. *Learning factor is a key parameter to determine the performance of DDE algorithm, which has a decisive influence on the exploitation and exploration ability of algorithm. This paper proposes an adaptive learning factor based on set operation for the first time to effectively balance the search ability of XSMT-MoDDE algorithm.*

Algorithm 3 Multiple mutation strategy

Require: P, N, m, e
Ensure: V
function MUTIMUTATION(P, N, m, t)
  $V \leftarrow \emptyset$
  for $i \leftarrow 1$ to m **do**
    for $j \leftarrow 1$ to N **do**
      if $i <= t \times N$ **then** $s \leftarrow random(1,2,3)$
      else $s \leftarrow random(1,2)$
      end if
      if $s == 1$ **then** $v \leftarrow Mutation1(P[j])$
      else if $s == 2$ **then** $v \leftarrow Mutation2(P[j])$
      else if $s == 3$ **then** $v \leftarrow Mutation3(P[j])$
      end if
      $V[j] \leftarrow v$
    end for
  end for
  return V
end function

Operating rule for learning factors

As shown in Formula 2, the learning factor F acts on the difference vector and controls the global search capability of DDE algorithm (Wang et al., 2014; Gong et al., 2010; Brest et al., 2006). In discrete problems, simple multiplication operation cannot be used. This paper redefines the $*$ operation in Formula 2.

Definition 8. $F * \left(X^g_{best} \odot X^g_r\right)$ $F < 1$. *Randomly eliminate n edges* $\{e_1, e_2, ..., e_n\}$ *from the edge set of difference particles, where* $e_i \in X^g_{best}$ *and* $e_i \notin X^g_i$*, and the value of n is calculated by Formula 19.*

Definition 9. $F * \left(X^g_{best} \odot X^g_r\right)$ $F > 1$. *Randomly eliminate n edges* $\{e_1, e_2, ..., e_n\}$ *from the edge set of difference particles, where* $e_i \in X^g_i$ *and* $e_i \notin X^g_{best}$*, and the value of n is calculated by Formula 20.*

Definition 10. $F * \left(X^g_{best} \odot X^g_r\right)$ $F = 1$. *No changes are made to the edge set.*

$$n = round\left((1-F) \times |X^g_{best}|\right) \tag{19}$$

$$n = round\left((F-1) \times |X^g_i|\right) \tag{20}$$

where $|X|$ represents the number of edge of Particle X.

Adaptive update process

Each Particle X_i corresponds to the adaptive learning factor F_i, which is initialized to 1. After each selection operation, the Parameter F_i is updated.

(1) Calculate reference Parameter r, $r \leftarrow k \times f_{best} + 1$, where k is 0.001 and f_{best} is the fitness value of the global optimal particle;

(2) Calculate difference value Δ between fitness value f_i of X^g_i and fitness value f_{best} of X^g_{best};

(3) Update F_i, the update formula is as follows:

$$F_i = \begin{cases} F_i + 0.05 & \Delta > r \\ F_i - 0.05 & \Delta \leq r \end{cases} \tag{21}$$

When the fitness value f_i is close enough to f_{best}, reduce F_i to preserve its structure to a greater extent, otherwise, increase F_i to expand the global search capability.

Algorithm 4 Refining strategy

```
Require: X, n
Ensure: R
1:  function REFINING(X, n)
2:      R ← ∅
3:      for i ← 1 to n do
4:          d ← CalculateDegree(X_i)
5:          Length ← 0
6:          Substructure ← ∅
7:          for j ← 1 to 4^d do
8:              s ← GetSubstructure()
9:              l ← GetCommonWireLength()
10:             if l > Length then
11:                 Substructure ← s
12:                 Length ← l
13:             end if
14:         end for
15:         for edge in Substructure do
16:             if edge not in R then
17:                 R ∪ edge
18:             end if
19:         end for
20:     end for
21:     return R
22: end function
```

Refining strategy

Property 7. *Refining strategy minimizes wire length of XSMT under the determined topology within a reasonable time.*

There may still be space for optimization for the optimal particles at the end of iteration. In order to search for a better result, a refining strategy is proposed. The steps of algorithm are as follows:

(1) Calculate degree of each Point p_i in the optimal particle. The degree is defined as the number of edges connected to point, denoted as d_i;

(2) There are 4 kinds of edges in X-architecture. If the degree of Point p_i is d_i, there are 4^{d_i} types of substructures corresponding to the point. The set of all substructures corresponding to Point p_i is S, and edge Set E is obtained when the substructures corresponding to Points $p_1 - p_{i-1}$ have been determined. Calculate common wire length l between Substructure s_i in Set S and Set E, select Substructure s_i corresponding to the largest l, and add the edges of s_i to the Set E. The algorithm ends until all points have been visited.

The pseudo code of the refining strategy algorithm is shown in Algorithm 4, where X represents the target particle obtained by the XSMT-MoDDE algorithm, n represents the point number of XSMT, and R represents the refined particle. Line 2 initializes Set R. Lines 3-20 search for the optimal substructure corresponding to each point. Line 4 calculates the degree of Point p_i, Line 5 initializes maximum common wire length, and Line 6 initializes the optimal substructure set. Lines 7-14 calculate common wire length and update the largest common wire length. Lines 15-19 store the edges in the optimal substructure into Set R.

Related parameters

The main parameters of the algorithm in this paper include population size n, iteration times m, threshold t, learning factor F, and crossover probability cr.

In the proposed algorithm, n is 50, m is 500, and t is 0.4. The adaptive strategy of learning factor F has been described in detail in Section 3.6. The crossover probability cr also adopts the adaptive strategy, which is as follows:

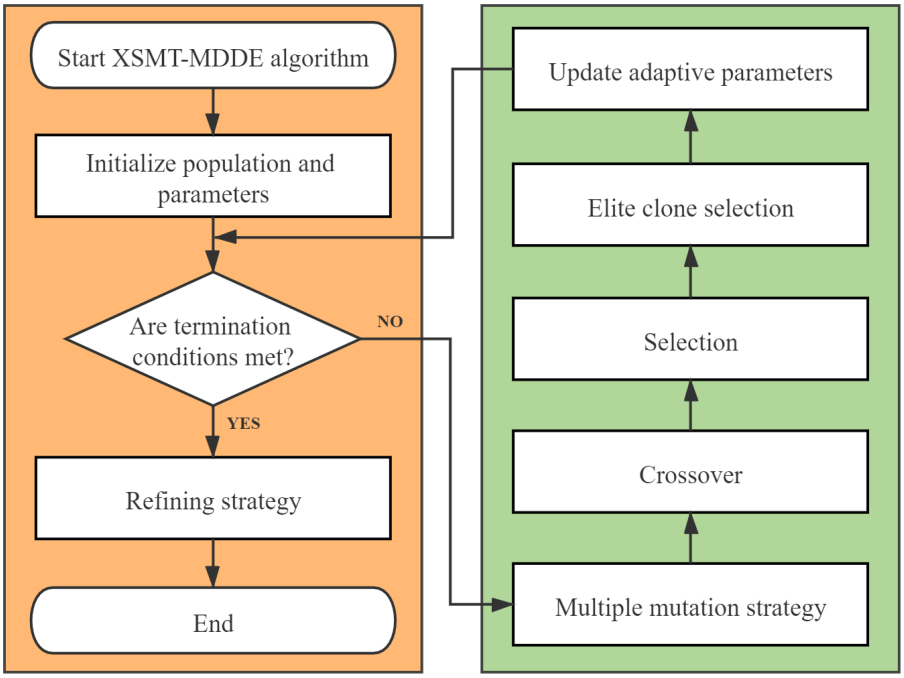


Figure 6. Algorithm flowchart.

$$cr_i = \begin{cases} cr_l + (cr_u - cr_l)\frac{f_i - f_{\min}}{f_{\max} - f_{\min}} & f_i > \overline{f} \\ cr_l & else \end{cases} \tag{22}$$

where cr_i=0.1, cr_u=0.6, f_i represents the fitness value of the current particle, f_{min} represents the minimum historical fitness value, f_{max} represents the maximum historical fitness value, and $\overline{f}$ represents the average historical fitness value.

The algorithm flow of XSMT-MoDDE

The algorithm flow chart of XSMT-MoDDE is shown in Figure 6, and the detailed flow is as follows:

(1) Initialize parameters.

(2) Use Prim algorithm to initialize population.

(3) Check the current stage: early stage or late stage of iteration.

(4) Select a mutation strategy from the corresponding mutation strategy pool according to the current stage.

(5) Obtain the trial particles according to the crossover operator.

(6) Obtain the next generation of particles according to the selection operator.

(7) Adopt elite selection and cloning strategy, and update the elite buffer after four steps of selection, clone, mutation and extinction.

(8) Check the number of iterations, and end the iteration if the termination condition is met, otherwise, return to Step (3).

(9) At the end of XSMT-MoDDE algorithm, a refining strategy is adopted to obtain the target solution.

Complexity analysis of XSMT-MoDDE algorithm

Property 8. *When the population size is m and the number of pins is n, the time complexity of one iteration is* $O(mnlogn)$.

Complexity analysis of multiple mutation operator

The mutation process is divided into two stages. First, difference vector is constructed, and then difference vector and the basis vector are used to construct the trial particles.

Construction of difference vector: Sort the edges of two edge sets according to the number of edge start point, and use binary search to construct the non-common edges. The complexity of this process is $O(nlog(n))$, and the non-common edge set is the difference vector.

Construction of mutation particle: Construct the difference set of basis vector and difference vector according to the above-mentioned similar idea. Then the edges in the difference set are stored in DSU, and edges are randomly selected from difference vector to be added to DSU until a complete tree is constructed. The time complexity of this process is $O(nlog(n))$.

Complexity analysis of elite selection and cloning strategy

A minimum heap is established according to the fitness value of particles, and the heap top is selected for cloning each time. The time complexity required for this process is $O(n)$.

The mutation process adopts connection method mutation and topology mutation. The connection method mutation selects two different edges randomly from the edge set to modify the connection method of the edges. The time complexity required is $O(1)$. In topology mutation, one edge is randomly disconnected to form two sub-XSMTs, which are recorded using the DSU. It takes $O(nlog(n))$ time to construct two sub-XSMTs with DSU, and randomly select one point from each of two sub-XSMTs to establish connection, this process takes O(1) time.

The particles obtained by the elite selection and cloning strategy need to be stored in an elite buffer with a size of m. The population particles and the particles of elite buffer participate in mutation, crossover, and selection operations together.

Complexity analysis of refining strategy

The degree of Point i is recorded as d_i. We always keep d_i within 4, even if there is a minimum probability greater than 4, only four connected edges will be considered in refining strategy. The adjacent edges of a point select a connection method respectively to form a substructure. An X-architecture edge has four selection methods, so one point corresponds to 4^{d_i} substructures, where $4^{d_i} \leq 256$.

Refining strategy takes out the optimal particle constructed by XSMT-MoDDE algorithm, enumerates substructures for each point of the particle, and obtain the substructure with the largest common wire length. So for the case of n points, the required time is $\sum_{i=1}^{n} \left(d_i \times 4^{d_i}\right)$.

EXPERIMENTAL RESULTS

The proposed XSMT-MoDDE has been implemented in C++ language on a windows computer with 3.5 GHz Intel CPU. To compare the experimental results fairly, we run all programs in the same experimental environment and use the same benchmarks from GEO and IBM. The population size and iteration size of all heuristic algorithms are set to 50 and 500 respectively. Calculation formula of optimization rate is shown in Formula 23.

$$rate = \frac{b-a}{b} \times 100\% \tag{23}$$

where a is the experimental result of the XSMT-MoDDE algorithm, and b is the experimental result of other algorithms.

Verify the effectiveness of multi-strategy optimization

Experiment 1: In order to verify the effectiveness of the multi-strategy optimization DDE algorithm in constructing XSMT, this experiment will compare the results of XSMT-MoDDE algorithm and XSMT-DDE algorithm. Experimental results are shown in Table 2 and Table 3. Table 2 is the optimization results of wire length, and Table 3 is the optimization results of standard deviation. The results show that multi-strategy optimization can achieve an average wire length optimization rate of 2.35% and a standard deviation optimization rate of 95.69%. This experiment proves that multi-strategy optimization has a powerful effect on wire length reduction, and at the same time greatly increases the stability of DDE.

Circuit	Pins	XSMT-DDE	XSMT-MoDDE	Reduction(%)
1	8	16956	16900	0.33%
2	9	18083	18023	0.33%
3	10	19430	19397	0.17%
4	15	25728	25614	0.44%
5	20	32434	32171	0.81%
6	50	49103	48090	2.06%
7	70	57386	56397	1.72%
8	100	70407	68917	2.12%
9	400	145183	139871	3.66%
10	410	146680	141571	3.48%
11	500	160031	154406	3.51%
12	1000	232057	220577	4.95%
Average				1.97%

Table 2. Average wire length optimization results of multi-strategy optimization

Circuit	Pins	XSMT-DDE	XSMT-MoDDE	Reduction(%)
1	8	56	0	100.00%
2	9	58	0	100.00%
3	10	42	0	100.00%
4	15	198	10	94.95%
5	20	343	51	85.13%
6	50	1036	147	85.81%
7	70	1082	102	90.57%
8	100	1905	279	85.35%
9	400	3221	120	96.27%
10	410	3222	178	94.48%
11	500	3193	139	95.65%
12	1000	3977	106	97.33%
Average				93.80%

Table 3. Standard deviation optimization results of multi-strategy optimization

Circuit	Pins	XSMT-DDE	Refining	Reduction(%)
1	8	16900	16900	0.00%
2	9	18023	18023	0.00%
3	10	19397	19397	0.00%
4	15	25614	25624	-0.04%
5	20	32171	32091	0.25%
6	50	48090	48090	0.00%
7	70	56397	56105	0.52%
8	100	68917	68457	0.67%
9	400	139871	138512	0.97%
10	410	141571	140359	0.86%
11	500	154406	152649	1.14%
12	1000	220577	217060	1.59%
Average				0.50%

Table 4. Average wire length optimization results of refining strategy

Circuit	Pins	XSMT-DDE	Refining	Reduction(%)
1	8	0	0	-
2	9	0	0	-
3	10	0	0	-
4	15	10	8	20.00%
5	20	51	22	56.86%
6	50	147	119	19.05%
7	70	170	136	20.00%
8	100	279	187	32.97%
9	400	120	57	52.50%
10	410	178	56	68.54%
11	500	139	50	64.03%
12	1000	115	113	1.74%
Average				37.30%

Table 5. Standard deviation optimization results of refining strategy

Circuit	Pins	Mean value				Reduction(%)		
		DDE	ABC	GA	MoDDE	DDE	ABC	GA
1	8	16956	16918	16918	16900	0.33%	0.00%	0.00%
2	9	18083	18041	18041	18023	0.33%	0.10%	0.10%
3	10	19430	19696	19696	19397	0.17%	1.52%	1.52%
4	15	25728	25919	25989	25624	0.40%	1.14%	1.40%
5	20	32434	32488	32767	32091	1.06%	1.22%	2.06%
6	50	49103	48940	48997	48090	2.06%	1.74%	1.85%
7	70	57386	57620	57476	56105	2.23%	2.63%	2.39%
8	100	70407	70532	70277	68457	2.77%	2.94%	2.59%
9	400	145183	141835	141823	138512	4.59%	2.40%	2.40%
10	410	146680	143642	143445	140359	4.31%	2.29%	2.15%
11	500	160031	156457	156394	152649	4.61%	2.43%	2.39%
12	1000	232057	222547	222487	217060	5.90%	2.47%	2.44%
Average						2.40%	1.74%	1.77%

Table 6. Comparison results of average wire length in GEO dataset

Circuit	Pins	Best value				Reduction(%)		
		DDE	ABC	GA	MoDDE	DDE	ABC	GA
1	8	16918	16918	16918	16900	0.11%	0.11%	0.11%
2	9	18041	18041	18041	18023	0.10%	0.10%	0.10%
3	10	19415	19696	19696	19397	0.09%	1.52%	1.52%
4	15	25627	25627	25897	25605	0.09%	0.09%	1.13%
5	20	32209	32344	32767	32091	0.37%	0.78%	2.06%
6	50	47987	48637	48783	47975	0.03%	1.36%	1.66%
7	70	56408	57227	57445	55919	0.87%	2.29%	2.66%
8	100	68829	70382	70092	68039	1.15%	3.33%	2.93%
9	400	141967	141490	141467	138382	2.53%	2.20%	2.18%
10	410	144033	143310	143282	140179	2.68%	2.18%	2.17%
11	500	156950	156034	156110	152591	2.78%	2.21%	2.25%
12	1000	226654	222262	222285	216824	4.34%	2.45%	2.46%
Average						1.26%	1.55%	1.77%

Table 7. Comparison results of best wire length in GEO dataset

Circuit	Pins	Standard deviation				Reduction(%)		
		DDE	ABC	GA	MoDDE	DDE	ABC	GA
1	8	56	0	0	0	100.00%	-	-
2	9	58	0	0	0	100.00%	-	-
3	10	42	0	0	0	100.00%	-	-
4	15	198	148	46	8	95.96%	94.59%	82.61%
5	20	343	118	45	22	93.59%	81.36%	51.11%
6	50	1036	242	133	119	88.51%	50.83%	10.53%
7	70	1082	195	140	136	87.43%	30.26%	2.86%
8	100	1905	69	112	187	90.18%	-171.01%	-66.96%
9	400	3221	200	170	57	98.23%	71.50%	66.47%
10	410	3222	146	122	56	98.26%	61.64%	54.10%
11	500	3193	160	133	50	98.43%	68.75%	62.41%
12	1000	3977	131	107	113	97.16%	13.74%	-5.61%
Mean						95.65%	33.52%	28.61%

Table 8. Comparison results of standard deviation in GEO dataset

Circuit	Nets	Pins	Value			Reduction(%)	
			SAT	KNN	MoDDE	SAT	KNN
ibm01	11507	44266	61005	61071	56080	8.07%	8.17%
ibm02	18429	78171	172518	167359	154868	10.23%	7.46%
ibm03	21621	75710	150138	147982	133999	10.75%	9.45%
ibm04	26263	89591	164998	164828	149727	9.26%	9.16%
ibm06	33354	124299	289705	280998	256674	11.40%	8.66%
ibm07	44394	164369	368015	368015	335556	8.82%	8.82%
ibm08	47944	198180	431879	413201	371948	13.88%	9.98%
ibm09	53039	187872	418382	417543	382282	8.63%	8.44%
ibm10	64227	269000	588079	589102	532644	9.43%	9.58%
Mean						10.05%	8.86%

Table 9. Comparison results of wire length in IBM dataset

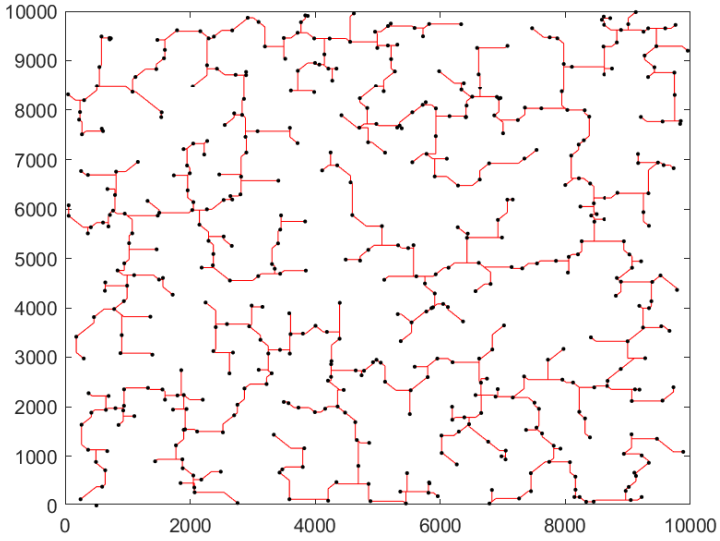
Verify the effectiveness of refining strategy

Experiment 2: In order to verify the effectiveness of the refining strategy, this experiment will compare the results of refined XSMT-MoDDE algorithm and XSMT-MoDDE algorithm. The experiment result is shown in Table 4 and Table 5. Table 4 is the optimization results of wire length, and Table 5 is the optimization results of standard deviation. The results show that refining strategy can achieve an average wire length optimization rate of 0.50% and a standard deviation optimization rate of 37.30%. From the experimental results and the above complexity analysis, it can be seen that after XSMT-MoDDE algorithm is over, refining strategy only takes a short time to obtain a lot of optimization of wire length and standard deviation. Regardless of whether refining strategy is added or not, both can always obtain accurate solutions in circuits with less than 10 pins. Refining strategy has more significant optimization effects in larger circuits.

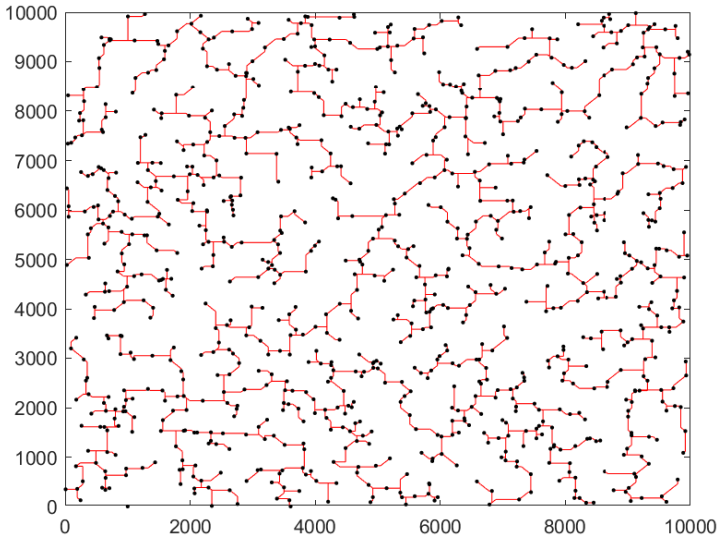
Algorithm comparison experiment

Experiment 3: To compare the performance of XSMT-MoDDE algorithm with other heuristic algorithms, we compare the results of XSMT constructed by MoDDE algorithm, DDE algorithm, Artificial Bee Colony (ABC) algorithm, and Genetic Algorithm (GA). The experimental results are shown in Table 6, Table 7, and Table 8. XSMT-MoDDE compares with XSMT-DDE, XSMT-ABC, and XSMT-GA, the average wire length is reduced by 2.40%, 1.74%, and 1.77%, the optimal wire length is reduced by 1.26%, 1.55%, and 1.77%, and the standard deviation is reduced by 95.65%, 33.52%, and 28.61%. Experimental results show that XSMT-MoDDE is better than XSMT-DE, XSMT-ABC, and XSMT-GA in both the wire length and standard deviation indicators. Compared with other algorithms, this algorithm still has excellent stability on the basis of having better wire length results.

Experiment 4: In the stage of global routing, there are tens of thousands of nets on the circuit board, and pins inside net need to be interconnected. This paper uses XSMT-MoDDE algorithm to optimize wire length of global routing. This experiment adopts the benchmark provided by IBM, and XSMT-MoDDE algorithm, SAT algorithm, and KNN algorithm are used to construct XSMT. The experimental results are shown in Table 9. Compared with SAT and KNN, XSMT-MoDDE optimizes wire length by 10.05% and 8.86% respectively. Experimental results show that XSMT-MoDDE can greatly shorten the wire length in the construction of multi-nets XSMT problem, and provide effective guidance for global routing.



(a) Steiner tree with 500 pins



(b) Steiner tree with 1000 pins

Figure 7. Steiner tree generated by XSMT-MoDDE.

Finally, for a better understanding the results of XSMT-MoDDE algorithm, we use Matlab to simulate the final XSMT diagrams. We choose Circuit 11 and Circuit 12 in Table 7 as representatives, as shown in Figure 7(a) and 7(b).

CONCLUSIONS

This paper designs four optimization strategies. The first three optimization strategies are used to strengthen DDE algorithm, and the fourth optimization strategy is used to reduce the wire length of final particle to the greatest extent.

Elite selection and cloning strategy expands the search range and enhances the diversity of the population particles. The elite particles are cloned and mutated, and the most excellent particle is selected greedily. This strategy enables the algorithm to quickly converge to a better state. Novel multi-mutation strategy introduces the idea of set operation. Through the interaction between edge sets, the corresponding shape of XSMT is changed. Three mutation strategies have different exploitation and exploration capabilities, and the three strategies are used alternately to avoid the algorithm from converging to the local peak prematurely. Adaptive learning factor dynamically adjusts and retains the ratio between the current particle edge set and the optimal particle edge set. Effectively improve global exploitation and local exploitation capabilities, and seek a balance between random strategy and greedy strategy.

XSMT-MoDDE algorithm proposed in this paper uses three indicators to measure algorithm results which are average wire length, optimal wire length, and standard deviation as evaluation. The proposed algorithm has achieved better optimization results compared with other algorithms. Moreover, XSMT-MoDDE has a stronger optimization ability in circuits with large-scale circuits. It is better than the results of the SAT and KNN algorithms in the case of multi-nets. Therefore, XSMT-MoDDE algorithm has good application prospect in the stage of global routing.

ADDITIONAL INFORMATION AND DECLARATIONS

Funding

This work was supported in part by National Natural Science Foundation of China (No. 61877010, 11501114), Natural Science Foundation of Fujian Province, China (2019J01243), Open Fund Project of Fujian Provincial Key Laboratory of Information Processing and Intelligent Control (Minjiang University) (No. MJUKF-IPIC201910).

Competing Interests

Chi-Hua Chen is an Academic Editor for PeerJ Computer Science.

Author Contributions

- Genggeng Liu contributed to the conception of the study, conceived and designed the experiments, analyzed the data, authored and reviewed drafts of the paper, and approved the final draft.
- Liliang Yang contributed to the conception of the study, conceived and designed the experiments, analyzed the data, authored and reviewed drafts of the paper.
- Saijuan Xu contributed significantly to analysis and manuscript preparation, conceived and designed the experiments, analyzed the data, authored and reviewed drafts of the paper.
- Zuoyong Li contributed significantly to analysis and manuscript preparation, conceived and designed the experiments, analyzed the data, authored and reviewed drafts of the paper.
- Yeh-Cheng Chen performed the data analyses and wrote the manuscript, conceived and designed the experiments, analyzed the data, authored and reviewed drafts of the paper.
- Chi-Hua Chen helped perform the analysis with constructive discussions, conceived and designed the experiments, analyzed the data, authored and reviewed drafts of the paper.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request. The source codes could be downloaded on the webpage of GitHub (https://github.com/yll7960/XSMT_MoDDE).

REFERENCES

Brest, J., Greiner, S., Boskovic, B., Mernik, M., and Zumer, V. (2006). Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE Transactions on Evolutionary Computation*, 10(6):646–657.

Chiang, C. and Chiang, C.-S. (2002). Octilinear steiner tree construction. In *The 2002 45th Midwest Symposium on Circuits and Systems, 2002. MWSCAS-2002.*, volume 1, pages I–603. IEEE.

Chu, C. and Wong, Y.-C. (2007). Flute: Fast lookup table based rectilinear steiner minimal tree algorithm for vlsi design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(1):70–83.

Coulston, C. S. (2003). Constructing exact octagonal steiner minimal trees. In *Proceedings of the 13th ACM Great Lakes symposium on VLSI*, pages 1–6.

Deng, G. and Gu, X. (2012). A hybrid discrete differential evolution algorithm for the no-idle permutation flow shop scheduling problem with makespan criterion. *Computers & Operations Research*, 39(9):2152–2160.

Epitropakis, M. G., Tasoulis, D. K., Pavlidis, N. G., Plagianakos, V. P., and Vrahatis, M. N. (2011). Enhancing differential evolution utilizing proximity-based mutation operators. *IEEE Transactions on Evolutionary Computation*, 15(1):99–119.

Ge, Y.-F., Yu, W.-J., Lin, Y., Gong, Y.-J., Zhan, Z.-H., Chen, W.-N., and Zhang, J. (2017). Distributed differential evolution based on adaptive mergence and split for large-scale optimization. *IEEE Transactions on Cybernetics*, 48(7):2166–2180.

Gong, W., Cai, Z., Ling, C. X., and Li, H. (2010). Enhanced differential evolution with adaptive strategies for numerical optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 41(2):397–413.

Held, S., Müller, D., Rotter, D., Scheifele, R., Traub, V., and Vygen, J. (2017). Global routing with timing constraints. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(2):406–419.

Liu, C.-H., Lin, C.-X., Chen, I.-C., Lee, D., and Wang, T.-C. (2014a). Efficient multilayer obstacle-avoiding rectilinear steiner tree construction based on geometric reduction. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 33(12):1928–1941.

Liu, G., Chen, G., and Guo, W. (2012). Dpso based octagonal steiner tree algorithm for vlsi routing. In *2012 IEEE Fifth International Conference on Advanced Computational Intelligence (ICACI)*, pages 383–387. IEEE.

Liu, G., Chen, G., Guo, W., and Chen, Z. (2011). Dpso-based rectilinear steiner minimal tree construction considering bend reduction. In *2011 Seventh International Conference on Natural Computation*, volume 2, pages 1161–1165. IEEE.

Liu, G., Chen, Z., Guo, W., Chen, G., et al. (2018). Self-adapting pso algorithm with efficient hybrid transformation strategy for x-architecture steiner minimal tree construction algorithm(in chinese). *Pattern Recognition and Artificial Intelligence*, 31(5):398–408.

Liu, G., Chen, Z., Zhuang, Z., Guo, W., and Chen, G. (2020a). A unified algorithm based on hts and self-adapting pso for the construction of octagonal and rectilinear smt. *Soft Computing*, 24(6):3943–3961.

Liu, G., Guo, W., Niu, Y., Chen, G., and Huang, X. (2015). A pso-based timing-driven octilinear steiner tree algorithm for vlsi routing considering bend reduction. *Soft Computing*, 19(5):1153–1169.

Liu, G., Huang, X., Guo, W., Niu, Y., and Chen, G. (2014b). Multilayer obstacle-avoiding x-architecture steiner minimal tree construction based on particle swarm optimization. *IEEE Transactions on Cybernetics*, 45(5):1003–1016.

Liu, G., Zhu, W., Xu, S., Zhuang, Z., Chen, Y.-C., and Chen, G. (2020b). Efficient vlsi routing algorithm employing novel discrete pso and multi-stage transformation. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–16.

Liu, G., Zhuang, Z., Guo, W., and Chen, G. (2020c). A high performance x-architecture multilayer global router for vlsi(in chinese). *Acta Automatica Sinica*, 46(1):79–93.

Pan, Q.-K., Tasgetiren, M. F., and Liang, Y.-C. (2008). A discrete differential evolution algorithm for the permutation flowshop scheduling problem. *Computers & Industrial Engineering*, 55(4):795–816.

Qin, A. K., Huang, V. L., and Suganthan, P. N. (2008). Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE transactions on Evolutionary Computation*, 13(2):398–417.

Rahnamayan, S., Tizhoosh, H. R., and Salama, M. M. (2008). Opposition-based differential evolution. *IEEE Transactions on Evolutionary computation*, 12(1):64–79.

Siddiqi, U. F. and Sait, S. M. (2017). A game theory based post-processing method to enhance vlsi global routers. *IEEE Access*, 5:1328–1339.

Storn, R. and Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359.

Tasgetiren, M. F., Suganthan, P. N., and Pan, Q.-K. (2010). An ensemble of discrete differential evolution algorithms for solving the generalized traveling salesman problem. *Applied Mathematics and Computation*, 215(9):3356–3368.

Teig, S. L. (2002). The x architecture: not your father's diagonal wiring. In *Proceedings of the 2002 international workshop on System-level interconnect prediction*, pages 33–37.

Wang, H., Wu, Z., and Rahnamayan, S. (2011a). Enhanced opposition-based differential evolution for solving high-dimensional continuous optimization problems. *Soft Computing*, 15(11):2127–2140.

Wang, J., Liao, J., Zhou, Y., and Cai, Y. (2014). Differential evolution enhanced with multiobjective sorting-based mutation operators. *IEEE Transactions on Cybernetics*, 44(12):2792–2805.

Wang, L., Zeng, Y., and Chen, T. (2015). Back propagation neural network with adaptive differential evolution algorithm for time series forecasting. *Expert Systems with Applications*, 42(2):855–863.

Wang, Y., Cai, Z., and Zhang, Q. (2011b). Differential evolution with composite trial vector generation strategies and control parameters. *IEEE Transactions on Evolutionary Computation*, 15(1):55–66.

Wu, H., Xu, S., Zhuang, Z., and Liu, G. (2019). X-architecture steiner minimal tree construction based on discrete differential evolution. In *The International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery*, pages 433–442. Springer.

Xue, Y., Zhuang, Y., Gu, J., Chang, X., and Wang, Z. (2014). Strategy selecting problem of self-adaptive discrete differential evolution algorithm(in chinese). *Journal of Software*, 25(5):984–996.

Zhang, G., Ma, L., Wang, X., and Zhou, X. (2020). Secondary structure and contact guided differential evolution for protein structure prediction. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 17(3):1068–1081.

Zhang, H., Ye, D.-y., and Guo, W.-z. (2016). A heuristic for constructing a rectilinear steiner tree by reusing routing resources over obstacles. *Integration*, 55:162–175.

Zhang, Y., Gong, Z., and Chen, Q. (2015). Community detection in complex networks using immune discrete differential evolution algorithm(in chinese). *Acta Automatica Sinica*, 41(4):749–757.

Zhao, H., Zhan, Z. H., Lin, Y., Chen, X., Luo, X. N., Zhang, J., Kwong, S., and Zhang, J. (2020). Local binary pattern-based adaptive differential evolution for multimodal optimization problems. *IEEE Transactions on Cybernetics*, 50(7):3343–3357.

Zhu, Q., Zhou, H., Jing, T., Hong, X.-L., and Yang, Y. (2005). Spanning graph-based nonrectilinear steiner tree algorithms. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(7):1066–1075.