Computer Science and Information Systems 21(4):1651–1672 https://doi.org/10.2298/CSIS240314052P

Remora Optimization Algorithm-based Adaptive Fusion via Ant Colony Optimization for Traveling Salesman Problem

Lin Piao

Department of Education, Liaoning National Normal College
No. 45, Chongdong Road, Huanggu District, Shenyang 110032 China
pllnco@163.com

Abstract. The traditional ant colony optimization (ACO) is easy to fall into local optimal when solving large-scale traveling salesman problem (TSP), and the convergence speed is slow. In order to enhance the local search ability of ACO, speed up the efficiency of ACO and avoid the premature problem, this paper proposes a novel remora optimization algorithm-based adaptive fusion via ant colony optimization for solving TSP. Firstly, an improved K-means clustering method is used to obtain the best clustering results and the optimal solutions of each class quickly by adaptive clustering strategy based on the maximum and minimum distance and class density. By using an improved Remora optimization algorithm, adjacent classes are fused to effectively improve the accuracy of the initial solution. In addition, the initial solution is optimized by the k-opt strategy. Finally, the random recombination strategy is used to recombine the pheromone and random excitation to make the algorithm jump out of the local optimal as far as possible and improve the accuracy of the algorithm. The experimental results show that the proposed algorithm not only guarantees the accuracy of solution, but also improves the stability when solving large-scale TSP.

Keywords: TSP, ACO, Remora optimization algorithm, K-means clustering, Adaptive fusion.

1. Introduction

The travelling salesman problem (TSP) is a classic problem in the field of combinatorial optimization. It can be described as a businessman who starts from any city, visits every city without repetition or omission, and finally returns to the starting place. Its goal is to find a shortest path that includes all cities [1]. TSP problem is a NP problem, which has important reference value in the fields of vehicle routing, network routing and shop scheduling etc. Traditional methods such as linear programming, dynamic programming, branch and bound and k-opt have been proposed by many scholars. These methods can obtain better solutions for small-scale TSP problems. With the gradual increase of data scale, the above methods are easy to fall into local optimal solution when solving medium- and large-scale TSP problems, and the time complexity increases greatly. For the medium and large scale TSP problem, swarm intelligence algorithms such as ant colony optimization algorithm (ACO) [2], particle swarm optimization algorithm (PSO) [3], simulated annealing algorithm (SA)[4] and genetic algorithm (GA) [5] can obtain acceptable solutions within an expected time range, so more scholars pay attention to them.

For a classical TSP problem, if the brute force solution is used, the time complexity is $O(n!)$. However, with the continuous development of TSP problems, the required time will increase geometrically when the scale is larger. Even if the improved traditional algorithm is used to solve large-scale problems, it still needs a very large time cost and low efficiency. Therefore, in recent years, many scholars have improved ant colony algorithm in combination with other algorithms [5,6].

Reference [7] proposed the combination of PSO and ACO, using PSO to improve the parameters of ACO, and set up an evaluation function to guide it to converge in the direction of high index. This algorithm was beneficial to improve the accuracy of the algorithm, but the running speed was slow. Starting from genetic algorithm, Reference [8] made use of the global search ability of genetic algorithm and the fast convergence of ant colony algorithm to plan the path in a short time. However, its scale was small and the accuracy of the algorithm needed to be further improved. When selecting correction points, reference [9] combined greedy strategy with ant colony algorithm, and then used algorithm A^* to search out a solution as the initial solution of ant colony algorithm, and modified the updating mode of pheromone to further improve the accuracy. However, the iteration times were too few and the accuracy was not improved enough, resulting in the ant colony algorithm ending the iteration without sufficient optimization.

Recently, Sharma et al. [10] proposed an improved discrete butterfly optimization algorithm. Greedy strategy was used to initialize the population. 2-opt method, 4-opt method and simulated annealing method were combined to improve the optimization ability of the algorithm. However, the introduction of two-bridge experiment made the time complexity of the algorithm very high. Panwar et al. [11] introduced Hamming distance into grey wolf optimizer (GWO) to make it suitable for combinatorial optimization problems, and used 2-opt method to enhance local optimization ability. In the discrete cuckoo search algorithm, Xu et al. [12] used k-means algorithm to divide medium- and large-scale TSP problems into multiple small-scale TSP problems, so as to accelerate the operation efficiency of medium- and large-scale TSP problems. However, due to the lack of interaction ability between cities of different sizes, the final solution effect was poor. Wu et al. [13] proposed a discrete sparrow search algorithm (DSSA), which improved the ability of the algorithm to leap out of the local optimal solution by utilizing the division of labor mechanism of sparrow and the exploration and development capability of the global disturbance strategy balance algorithm. At the same time, the 2-opt method was used to enhance the local search ability, and the results were better than the discrete gray Wolf algorithm. k-opt method or its deformation methods are effectively used in the above algorithms.

Some scholars combined new technology to improve ant colony algorithm. Reference [14] combined K-means algorithm and ant colony algorithm to optimize TSP problem. By clustering, the problem was decomposed into several sub-problems, which were then connected to obtain solutions, so that the algorithm had better performance in dealing with large-scale problems. However, because K-means itself needed to determine the value of K, and only the accuracy of the fusion subclass solution depended on the problem itself, the algorithm depended on the type of TSP problem, the application range was small, and the accuracy was insufficient. Reference [15] used the least variance method to cluster cities, and proposed a new city connection method, and then used the circular search method to connect subclass solutions to obtain the final solution. This algorithm had a good solution for small and medium-sized TSP problems. However, none of the above algorithms can dynamically determine the number of clusters, and the algorithm needs to be further improved when solving large-scale TSP problems.

In this paper, a novel remora optimization algorithm-based adaptive fusion via ant colony optimization for solving TSP is proposed. By using the characteristics of clustering, a large-scale TSP problem is decomposed into several sub-problems, and the subclass solutions are fused according to the fusion strategy of ant colony algorithm and Remora algorithm. Finally, k-opt strategy is used to further optimize the solution to obtain a higher precision solution. This algorithm can guarantee the accuracy of the solution and reduce the running time of the algorithm.

2. Related Works

2.1. TSP

The traveling salesman problem can be expressed as an undirected graph $G = (S, E)$. Where $S = 1, 2, \cdots, N$ represents the set of vertices. E is the edge set, edge $e_{ij} \in (i, j \in S, i \neq j)$. Let c_{ij} be the distance of e_{ij} and the decision variable be:

$$y(x) = \{1 \quad e_{ij} incurrentsolution 0 \quad otherwise \tag{1}$$

The optimization objectives of this problem are as follows:

$$min \sum_{i=1}^{N} \sum_{j=1}^{N} x_{ij} c_{ij}. \tag{2}$$

$$s.t. \sum_{j \neq i} x_{ij} = \sum_{i \neq j} x_{ji} = 1, i \in S. \tag{3}$$

$$\sum_{j \neq i} x_{ij} \leq |V| - 1, V \subset S. \tag{4}$$

Where $|V|$ represents the number of vertices in the set S. The first constraint means that each vertex is connected to only two edges, and the second constraint means that no sub-loop is generated. Equation (2) is the target of optimization in this paper. This problem is a typical combinatorial optimization problem, and no definite algorithm can find the global optimal solution in the polynomial time range.

2.2. ACO for Solving TSP

TSP is a classical combinatorial optimization problem. Combinatorial optimization is to organize, group and sort some discrete events according to mathematical methods. The specific content of the traveling salesman problem is as follows: Suppose a traveling businessman needs to sell goods, he needs to visit all the cities and finally return to the origin, on the premise that these cities can only be visited once, and he needs to find the shortest visit path and its order. When the number of cities to be planned is larger, the time complexity and spatial complexity of the solution will rise and increase exponentially. With the increasing complexity of the problem, it is no longer effective to use the traditional method of solving [16,17].

Nowadays, the mainstream algorithms for solving TSP generally fall into two categories: precise algorithms and heuristic algorithms. The former is suitable for small scale TCP, while the latter is suitable for medium and large scale TSP. The precise algorithms mainly include delimiting method and planning method. Heuristic algorithms include Genetic Algorithm (GA), Simulated Annealing algorithm (SA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), etc. Among them, ACO provides a new way to solve the TCP by using the inspiration of biology due to its evolutionary thought.

In the classical ant colony algorithm, the initial N ants are randomly placed in the initial city. The ants need to move on to the next city according to a certain probability. The ants decide on the next city according to the state transition probability rule (5).

$$P_{ij}^{k}(t) = \frac{[\tau_{ij}(t)^{\alpha}][\eta_{ij}]^{\beta}}{\sum_{k}[\tau_{ij}(t)^{\alpha}][\eta_{ij}]^{\beta}}. \tag{5}$$

Where, τ_{ij} is the pheromone value of the city and connection path in the process of t iteration. η_{ij} is the reciprocal value of the length of the connecting path between city i and j, and is a heuristic information. If the path length between city i and j is shorter, then the next city chosen by ants is more likely to be j. $Allowed_k$ represents the city that ant $k \in Allowed_k$ has not visited. α and β determine the importance of pheromones and heuristics, respectively. When all ants have completed a complete path construction, that is, an iteration is completed, the pheromone update will be carried out according to formulas (6) and (7).

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t). \tag{6}$$

$$\Delta\tau_{ij}^{k} = \{1/L_k(t) \quad if \quad ant \quad passes \quad path \quad (i,j) 0 \quad otherwise \tag{7}$$

Where ρ represents pheromone volatilization parameter. m is the total number of ants in each iteration, represents the pheromone value released by ant k on the path (i,j), and it is the travel path length of ant k. By setting the maximum number of iterations, the running of the algorithm ends when the number of iterations reaches the maximum value, and the optimal solution of the algorithm is obtained. If stagnation occurs during iteration, the process is initialized to get rid of stagnation and continue iteration until the set iteration upper bound is reached.

2.3. K-means Clustering Algorithm

K-means clustering algorithm is a widely used clustering algorithm, and it is also one of the most basic clustering algorithms [18]. The basic flow is as follows:

1. First, we need to determine the value of k, where k represents the number of classes to be clustered.
2. k data points are randomly selected from the data set as the center of mass.
3. For each point in the data, the distance between it and each centroid is calculated, k distances are obtained, the centroid of the smallest distance is selected, and the point is divided into this class.

4. After traversing all points, the center of mass of each class is recalculated and compared with the existing center of mass.
5. If the distance between the newly calculated centroid and the existing centroid is less than a certain set threshold, it means that the centroid tends to be stable. Without repeating the above steps, we can consider that the clustering has reached the expected result and the algorithm terminates. If not, it needs to iterate step 3 to step 5 until the center of mass meets the above conditions.

2.4. Max-min Distance Strategy

Max-min distance strategy is a common strategy for optimizing K-means clustering algorithm. The basic idea is to limit the distance between classes by setting thresholds, so as to achieve the purpose of solving different clustering problems without setting k value.

The basic steps are as follows:

1. Randomly select a point from the data as the initial clustering centroid z_l.
2. The sample point z_2 farthest from the first cluster centroid is selected as the second cluster centroid.
3. Calculate the distance between the remaining sample points and z_l and z_2, and find the minimum value between the two values, that is:

$$d_{ij} = ||x_i - z_j||, j = 1, 2. \tag{8}$$

$$d_i = min[d_{ij}], i = 1, 2, \cdots, N. \tag{9}$$

4. If

$$d_l = \max_i [min[d_{i1}, d_{i2}]] > \theta^* ||z_l - z_2||. \tag{10}$$

5. Assuming that there are k clustering centers, it calculates the distance between the sample point and each clustering center:

$$d_l = \max_i [min[d_{i1}, d_{i2}, \cdots, d_{ik}]] > \theta^* ||z_l - z_2||. \tag{11}$$

   If this is true, the fifth step is repeated until the above conditions are not met, that is, no new clustering centers appear.
6. According to the principle of minimum distance, all sample points are divided into each cluster center, that is, it calculates:

$$d_{ij} = ||x_i - z_j||. \tag{12}$$

3. Proposed TSP Solution Method

Aiming at the problem that the size of the traditional clustering algorithm k needs to be set in advance and the number of clusters cannot be changed according to the situation, the K-means clustering algorithm based on the maximum and minimum distance is chosen. The number of classes can be changed according to the set distance size, and once the specified threshold is exceeded, the cluster is no longer. However, it is still necessary to set the threshold value in advance, which cannot achieve the function of adaptive clustering. And when the number of nodes in TSP problem is too large, it is difficult to find a balance between running time and precision by ant colony algorithm. This paper proposes a novel remora optimization algorithm-based adaptive fusion via ant colony optimization for solving TSP, which can effectively ensure the accuracy of the solution and improve the convergence speed of the algorithm.

3.1. Adaptive Clustering Strategy Based on Area Density

Aiming at the problems of slow convergence and poor quality of traditional ant colony algorithm in dealing with large-scale problems, adaptive clustering is adopted in this paper to exploit the advantages of ant colony algorithm in dealing with small-scale problems.

Because the maximum minimum distance strategy still needs to determine the clustering threshold, the concept of area density is proposed.

Using a rectangle so that all points are inside the rectangle, it determines the density according to the formula (13):

$$density = A \times S/count. \tag{13}$$

Here, $density$ represents the area density, S represents the rectangular area, $count$ represents the number of points in the TSP problem, and A represents the coefficient.

The results obtained by the area density strategy are used to replace the distance threshold in the max-min distance K-means clustering strategy, so that the clustering strategy can change the number of classes according to different types of problems and achieve the purpose of dynamically determining k.

Considering the impact of the size of each class on the total TSP problem, a class that is too large or too small will have a greater impact on the final result. In order to avoid the error caused by this situation, the amount of data within the class needs to be controlled.

Considering the influence of the number of points within a class on the result, this paper limits the number of points for each class to between 50 and 120, as shown in **Algorithm 1** for specific operations.

Specific treatment methods are as follows:

- First, the number of points for each class is calculated, if $p - number > 120$, the class is pulled out to re-cluster, the distance threshold is 0.9 times the previous, and the above process is repeated until all the class points are less than 120.
- Second, the number of points for each class is calculated here, if the number of points $p - number < 30$, the class nearest the center of mass is forced to merge, if the number of points is $30 < p - number < 50$, then try to merge with nearby classes. If the fusion result number $p - number < 120$, then fusion is allowed, if the number $p_n umber > 120$, it does not allow fusion, separates them into classes. Here, $p - number$ represents the number of city points in the class.

3.2. Initialization Optimization Based on ROA

ROA is a meta-heuristic optimization algorithm inspired by the parasitic feeding behavior of Remora in the ocean. Remora does this action by attaching itself to other hosts (such

Algorithm 1 Interclass point processing flow

```
Require: Input the results after the initial clustering
Ensure: Output the final clustering results
 1: input data
 2: if $count > 120$ then
 3:     Narrowing $\tau$, re-clustering
 4: end if
 5: if $count < 50$ then
 6:     if $count < 30$ then
 7:         Fusion directly
 8:     else
 9:         if Points after fusion $< 120$ then
10:             Perform fusion
11:         else
12:             Separate class
13:         end if
14:     end if
15: end if
16: Output result
```

as humpback whales or swordfish) to move, hunt, and evade predators. Similar to other meta-heuristic algorithms, ROA is mainly divided into three stages, namely initialization, exploration, and development.

Initialization stage At this stage, ROA generates initial populations in the search space in a random manner. The population vector is represented by X and contains N search agents with dimension d. The specific mathematical model is:

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,d} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,d} \\ \vdots & \vdots & \cdots & \vdots \\ x_{N,1} & x_{N,2} & \cdots & x_{N,d} \end{bmatrix} \tag{14}$$

Where N represents population size and d represents problem dimension.

Exploration stage By attaching to a swordfish, the Remora can move long distances within the search space, so the Remora's movement trajectory is the same as that of a swordfish. In addition, Remora also moves around its host in a small area to probe the surrounding feeding environment and determine whether to change the host.

– Swordfish strategy. Swordfish is one of the fastest fish in the world. By attaching to the swordfish, Remora mora quickly moves in the search space at a long distance. Its position update mode is as follows:

$$X(t+1) = X_{best}(t) - (r_1 \times (\frac{X_{best}(t) + X_{rand}(t)}{2}) - X_{rand}(t)). \tag{15}$$

Where $X_{best}(t)$ represents the global optimal position. r_1 indicates a random number between 0 and 1. $X_{rand}(t)$ represents random individuals in a population.

– Empirical attack. To maximize the benefits of feeding, Remora also moves around its host in small areas, surveying the surrounding prey distribution to determine if it needs to change hosts. Mathematically expressed as:

$$X(t+1) = X(t) + r_G \times (X(t) - X_{pre}(t)). \tag{16}$$

Where $r_G \in [0,1]$ represents random numbers that follow Gaussian distribution. $X_{pre}(t)$ represents the population position of the last iteration.

Development stage Remora does this process by attaching itself to a whale, so it hunts in the same way that whales do. If the host does not change, the remora attaches itself to the proper location and waits for the host to feed it.

– Whale strategy. At this point, the whales complete their hunting activities through the bubble net attack strategy. In turn, the remora performs its feeding by attaching itself to the whale. The mathematical model is as follows:

$$X(t+1) = D \times e^a \times \cos(2\pi\alpha) + X(t). \tag{17}$$

$$\alpha = r_2(a-1)+1. \tag{18}$$

$$a = -(1+\frac{t}{T}). \tag{19}$$

$$D = |X_{best}(t) - X(t)|. \tag{20}$$

Where D represents the position difference between prey and predator. $r_2 \in [0,1]$ represents a random number between 0 and 1. t and T represent the current and maximum number of iterations, respectively.

– Host feeding. When the host is not changed, Remora uses factor C to select a suitable adsorption site for the host to feed. The mathematical model is as follows:

$$X(t+1) = X(t) + A. \tag{21}$$

$$A = B \times (X(t) - C \times X_{best}(t)). \tag{22}$$

$$B = 2 \times V \times r_3 - V. \tag{23}$$

$$V = 2 \times (1 - \frac{t}{T}). \tag{24}$$

Where C represents remora factor, and its value is 0.1. r_3 represents a random number between 0 and 1. The procedure for executing the ROA algorithm is shown in **Algorithm 2**.

Algorithm 2 ROA

```
1: The search space, population size $N$, maximum number of iterations $T$, Remora factor $C$ are
   initialized and the initial population is generated.
2: Perform boundary revision on the search agent and calculate its fitness value.
3: If $H(i) = 0$, the remora attaches to the whale and uses the bubble net attack strategy to prey.
   When $H(i) = 1$, Remora attaches to the swordfish to complete the long-distance movement.
4: Perform an empirical attack, replace if the new position produced is better than the original,
   and reset the random number set $H$, otherwise perform host feeding.
5: Determine whether the program meets the termination condition, if so, jump out of the loop
   and output the current best solution, otherwise return to step 2.
```

Adaptive weight factor improvement The exploration stage of ROA mainly relies on attachment to swordfish, as shown in Equation (15), which utilizes the best position of the population X_{best} and the random position X_{rand} respectively. However, X_{best} has good convergence performance but does not have extensive search ability, which cannot guide the population to fully explore the solution space, and it is easy to ignore potential areas. Therefore, an adaptive weight factor is proposed in this paper, which enables X_{best} to have adaptive disturbance ability, ensure convergence ability, and explore the surrounding neighborhood extensively, so as to find more potential solutions. The mathematical model is as follows:

$$w(t+1) = r_G \times \sin(\frac{\pi \times t}{4T}). \tag{25}$$

Generally, the traditional adaptive weight factor has less fluctuation in the curve in the early stage of iteration, which gives the particles less disturbance. The large fluctuation in the middle and late iteration provides the algorithm with superior global exploration performance and can guide the population to approach the optimal region or adjacent region. The improved location update method is:

$$X(t+1) = w(t) \times X_{best}(t) - (r_1 \times (\frac{X_{best}(t) + X_{rand}(t)}{2}) - X_{rand}(t)). \tag{26}$$

Lens opposition-based learning ROA implements iterative optimization by attaching to different hosts. Specifically, Remora attaches itself to a swordfish and a whale for global exploration and local exploitation, respectively. In addition, Remora can autonomously decide whether to change the host according to the surrounding environment to improve the success rate of feeding. However, the above search strategy mainly depends on the host. If the host's predatory ability is not strong (falling into the local optimal trap), then the Remora will also fall into it, resulting in a "precocious" phenomenon, which affects the optimization performance of the algorithm. In order to alleviate this trend, lens opposition-based learning (LOBL) [19] is introduced to generate the lens reverse solution in the search space by using the convex lens imaging principle, and the optimal individual is selected to enter the next iteration through the greedy strategy. The introduction of this strategy is conducive to improving the diversity of the whole population, that is, using the rich search information of the lens reverse solution to improve the ability of the population to jump out of the local extreme value and accelerate the convergence of the algorithm.

As shown in Figure 1, assuming that point x exists in one-dimensional space $[L_B, U_B]$, a lens with a height of h^* and a position of x^* is formed under the action of a lens. The y axis represents a convex lens, located at a midpoint in space.

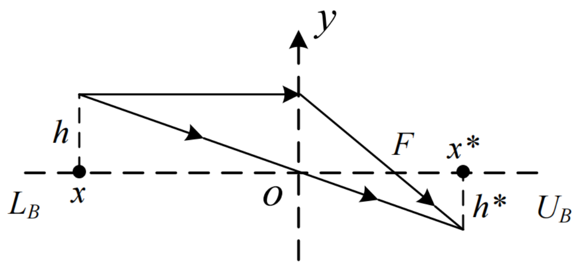

Fig. 1. LOBL

According to the principle of lens imaging:

$$\frac{(L_B, U_B)/2 - x}{x^* - (L_B, U_B)/2} = \frac{h}{h^*}. \tag{27}$$

L_B and U_B represent the upper and lower bounds of the search space. Let $k = h/h^*$, that is, formula (27) can be transformed as follows:

$$x^* = \frac{L_B + U_B}{2} + \frac{L_B + U_B}{2k} - \frac{x}{k}. \tag{28}$$

It can be seen from equation (28) that if $k = 1$, then it is equivalent to the reverse learning strategy, and it can also be said that the reverse learning of lens imaging is a special case of the reverse learning strategy.

The optimization process of the initial value based on the modified ROA is shown in **Algorithm 3**.

Algorithm 3 Optimization process of initial value based on modified ROA

```
1: Initializing parameter population size $N$, maximum number of iterations $T$, factor $C$ of Remora.
2: The search agent boundary is modified, the fitness value is calculated, and the best individual and fitness value are recorded.
3: Calculate the lens inverse solution of the search agent, the fitness value and save the optimal individual according to the greedy strategy.
4: If $H(i) = 0$, the remora attaches to the whale and uses the bubble net attack strategy to prey. When $H(i) = 1$, Remora attaches to the swordfish to complete the middle and long distance adaptive movement.
5: Perform an empirical attack, replace if the new position produced is better than the original, and reset the random number set H, otherwise perform host feeding.
6: Judge whether the program meets the termination condition, if yes, jump out of the loop and output the current optimal segmentation threshold, otherwise return to step (2).
```

From the above introduction, we can see that ROA mainly has three stages, namely, the initialization stage, the exploration stage and the development stage. In the initialization stage, a search agent with population size N is generated in the search space, where the spatial dimension is d, then the operation complexity of this stage is $O(N \times d)$. In the exploration phase, ROA completes the position update through swordfish strategy and experience attack, then the operation complexity of this part is $O(N \times d)$. In the development phase, the location is updated through whale strategy and host feeding, and the computational complexity is $O(N \times d)$. If the exploration phase and development phase are repeated T times, the ROA operation complexity is $O(N \times d \times T)$. In the improved ROA, LOBL strategy is introduced to improve algorithm performance, and its complexity is $O(N \times d \times T)$. In addition, adaptive weight factors are introduced to enhance inter-group communication learning, and the complexity is $O(T)$. The rest is the same as the original algorithm. In summary, the improved ROA operation complexity is $O(N \times d \times T)$, which shows that the algorithm proposed in this paper does not improve the operation complexity.

3.3. Improved k-opt Method

Figure 2 shows the 2-opt optimization process. This method selects two non-adjacent points i and j to find a shorter path according to the following formula.

$$D(i, i+1) + D(j, j+1) > D(i, j) + D(i+1, j+1). \tag{29}$$

Where $D(i, j)$ represents the distance between two points i and j. If the condition is true, then the sequence $L(i+1, j)$ is reversed. $L(i, j)$ represents a sequence of paths in an existing path that starts with i and ends with j. Through continuous optimization of the existing sequence, a shorter sequence is finally constructed. Similarly, Figure 3 shows the 3-opt optimization process. This method selects three non-adjacent points i, j, and k to optimize the current path according to the following formula. If the condition is true, then the sequence $L(j+1, k)$ is reversed.

$$D(i, i+1) + D(j, j+1) + D(k, k+1) > D(i, k) + D(i+1, j+1) + D(j, k+1). \tag{30}$$

Isoart et al. [20] systematically discussed the advantages and disadvantages of different k values in the k-opt method. The 2-opt and 3-opt methods are optimization processes based on greedy search, so the above methods also inherit the shortcomings of greedy search, that is, they eventually lead to local optimal solutions. At the same time, the time complexity of 2-opt and 3-opt methods is closely related to the data size. In theory, the maximum number of sequence reversals by 2-opt method is $O(n^2)$, and the maximum number of sequence reversals by 3-opt method is $O(n^3)$.

Improved 2-opt and 3-opt methods are proposed in order to alleviate the local optimal solution of 2-opt and 3-opt methods. In the improved 2-opt method, the set J of all subsequent nodes that are not adjacent to the current node i is found, and the set is constructed as follows:

$$Diff_2 = D(i, i+1) + D(J, J+1) - D(i, J) - D(i+1, J+1). \tag{31}$$

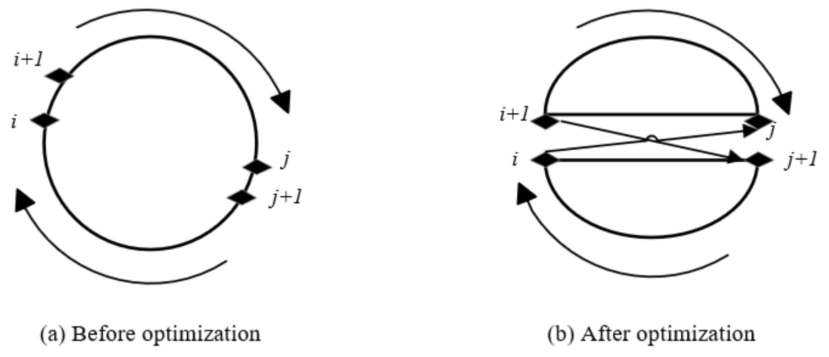

Fig. 2. Optimization process of 2-opt method

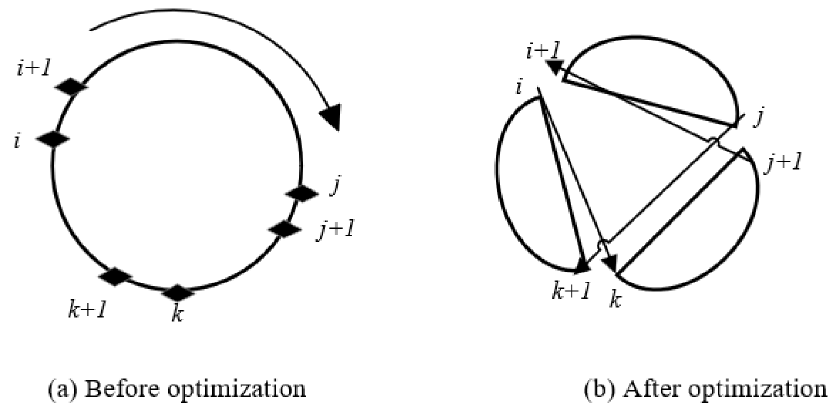

Fig. 3. Optimization process of 3-opt method

Where, the set $Diff_2$ represents the difference between all subsequent nodes and the current node after the sequence exchange. Find the maximum $max - S_2$ in the set $Diff_2$, if the condition $max - S_2 > 0$ is true, the sequence between node j and node i corresponding to $max - S_2$ is reversed, that is, the current sequence $L(i+1, j)$ is reversed. In the improved 3-opt method, the set K of all nodes that are not adjacent to the current node i and j is found and the set $Diff_3$ is constructed

$$Diff_3 = D(i, i+1) + D(j, Jj+1) + D(K, K+1) - D(i, K) - D(i+1, j+1) - D(j, K+1). \tag{32}$$

Compared with the original 2-opt method, the sequence reversal times of the improved 2-opt algorithm do not exceed $O(n)$. The improved 3-opt method reverses the sequence no more than $O(n/2)$. Secondly, for any node, the improved 2-opt method reverses the sequence at most 1 times, while the original 2-opt algorithm reverses the sequence at most $n - 2$ times.

3.4. Random Recombination Strategy

When using the ant colony guidance strategy, it is easy to fall into a local optimal situation. In order to avoid falling into local optimality as much as possible, this paper adopts random recombination strategy to make the algorithm jump out of stagnation. The details are as follows:

1. After N stagnation, the original pheromone matrix will be abandoned and a new one will be established.
2. The new pheromone matrix is set to the initial pheromone matrix, that is, every value except the diagonal is the initial value τ_0.
3. Each initial value of the new factor matrix is given a random multiplier incentive, and finally the new information matrix replaces the existing pheromone matrix, as shown in formula (33). Where the upper limit of magnification is the quotient of the current highest pheromone and the initial value τ_0.

$$\begin{bmatrix} 0 & x_{1,1}\tau_0 & x_{1,2}\tau_0 & \cdots & x_{i,j}\tau_0 \\ x_{1,1}\tau_0 & 0 & x_{2,1}\tau_0 & x_{2,2}\tau_0 & x_{i,j}\tau_0 \\ x_{1,2}\tau_0 & x_{2,1}\tau_0 & \cdots & x_{3,1}\tau_0 & x_{i,j}\tau_0 \\ x_{1,3}\tau_0 & x_{2,2}\tau_0 & x_{3,1}\tau_0 & 0 & x_{i,j}\tau_0 \\ x_{i,j}\tau_0 & x_{i,j}\tau_0 & x_{i,j}\tau_0 & x_{i,j}\tau_0 & 0 \end{bmatrix} \tag{33}$$

4. Experiment and Result Analysis

This algorithm adopts Matlab2017a simulation software, based on AMD Ryzen 5 3550H with Radeon Vega Mobile Gfx2.10 GHz processor, 8GB memory, and 64-bit Win10 operating system. This paper selects typical problem sets from TSPLIB, which are commonly used to test and optimize algorithms.

4.1. Parameter Setting

In this paper, the optimal parameters of ant colony algorithm are deeply studied, which is conducive to the application of ant colony algorithm in optimization problems. In order to make the experiment more reasonable, this paper conducts several tests, and the statistical results show that the parameter Settings in Table 1 are more reasonable and the experimental results are better.

Table 1. Basic parameter setting

α	β	ξ	ρ	$q0$	m	Tc
1	4	0.1	0.3	0.7	50	2000

α is the information heuristic factor, and the smaller α denotes the stronger randomness, but it is easy to fall into local optimal. The greater β denotes the greater probability of ants to choose the local shorter path, which can accelerate the convergence speed, but the accuracy of the algorithm diversity and settlement is poor. ξ is the volatilization rate of local pheromone. ρ is the global pheromone volatilization rate. $q0$ is a pseudo-random factor that affects the algorithm's exploration of new paths. The smaller the $q0$, the greater the probability of ants to explore new paths; conversely, the larger $q0$ denotes the stronger greed. Ants concentrate on the shortest path, which speeds up the convergence speed, but results in poor search ability. m is the number of ants, and m ants search in parallel. The larger m denotes more solutions that can be obtained, and the accuracy of the algorithm is improved correspondingly, while the time complexity is increased. Tc indicates the total number of iterations.

The pheromone matrix needs to be initially stimulated in the ant guidance strategy. The value of N multiplier needs to integrate convergence speed, local optimization, initial solution accuracy, so the value of N should not be too high or too low. According to the experiment results in Table 2 and Table 3, N in this paper is set at 10.

Table 2. kroA200 solutions under different excitation multiples (Standard solution 29368)

N	Average solution	Optimal solution	Error rate %	Iteration number
2	29712	29564	0.66	908
5	29631	29548	0.62	1058
10	29631	29548	0.62	1058
20	29682	29564	0.66	1849
30	29789	29564	0.66	1843
40	29634	29548	0.62	1058
50	29682	29564	0.66	318

Table 3. Lin318 solutions under different excitation multiples (Standard solution 42029)

N	Average solution	Optimal solution	Error rate %	Iteration number
2	43839	43130	2.63	1979
5	43557	42775	1.79	1788
10	43167	42704	1.61	1842
20	43782	42834	1.92	1821
30	43786	42999	2.31	1988
40	43649	43001	2.32	1525
50	43709	43229	2.86	1459

4.2. Determination of Area Density Strategy Coefficient A

The area density strategy provides a limiting condition for the first clustering and a basic sample for the subsequent clustering adjustment, that is, the area density determines the number of iterations required for the subsequent adjustment. Therefore, in order to enable subsequent operations to quickly reach the set standards, a reasonable coefficient of area density strategy should be selected to make the first clustering result as close as possible to the required result, that is, the number of each class is close to the value range of 50-120.

In this paper, different coefficients A will be selected, and the selection of coefficients will be judged by the results of the first clustering. As can be seen from Table 4 and Table 5, the optimal value range is [0.01,0.001]. In order to find better results, the value of this range is further refined to facilitate the solution of the optimal value.

As can be seen from Table 6 and Table 7, when $A = 0.002$, there is a better clustering result, and the number of iterations required for subsequent optimization will be effectively reduced. Therefore, the A of this paper is 0.002.

Table 4. kroA200 solutions under different excitation multiples

A value	Cluster number	Minimum number	Maximum number
1	2	100	100
0.1	2	100	100
0.01	2	100	100
0.001	11	12	27
0.0001	101	1	6

Table 5. Lin318 solutions under different excitation multiples

A value	Cluster number	Minimum number	Maximum number
1	1	318	318
0.1	2	150	168
0.01	2	150	168
0.001	9	19	57
0.0001	101	1	11

Table 6. kroA200 solutions under different excitation multiples

A value	Cluster number	Minimum number	Maximum number
0.002	5	32	49
0.004	2	100	100
0.006	2	100	100
0.008	2	100	100

4.3. Algorithm Performance Analysis

A. Performance analysis of adaptive clustering strategy

Table 7. Lin318 solutions under different excitation multiples

A value	Cluster number	Minimum number	Maximum number
0.002	4	61	89
0.004	2	150	168
0.006	2	150	168
0.008	2	150	168

In order to show the influence of adaptive clustering strategy on this algorithm, two sets of experiments were set in this paper to compare the accuracy of the solution, namely ROA+ant colony guidance strategy+random recombination strategy, and ROA+adaptive clustering strategy+ant colony guidance strategy+random recombination strategy, as shown in Table 8. The formula for calculating the error rate is:

$$\theta = \frac{L_{ROA-best} - L_{best}}{L_{best}}. \tag{34}$$

Where $L_{ROA-best}$ represents the theoretical optimal solution, L_{best} represents the optimal solution of the algorithm, and θ represents the error rate. Each group is tested 10 times, with the maximum iteration number of 2000 each time, and the excitation rate N is selected to be 10 times, and the average solution and error rate are calculated according to the results.

Table 8. Optimization scheme

Group number	Name	Scheme combination
A	ROA+ACO+RRS	ROA+ACO+random recombination strategy
B	ROA+AC+ACO+RRS	ROA+adaptive clustering+ACO+random recombination strategy

As can be seen from Table 9, in terms of average solution, optimal solution and error rate, the data in group B is better than that in group A, because group B provides a better initial solution, which makes the algorithm optimized around the initial solution at the initial stage, resulting in improved accuracy of the solution. In addition, the average solution of group B algorithm is better, which also shows that group B algorithm is more stable.

B. Performance analysis of ant colony guidance strategy

In order to show the influence of ant colony guidance strategy on this algorithm, three groups of experiments are set up in this paper to compare the accuracy of the solution, which are divided into specific ROA, ROA+ adaptive clustering+random recombination strategy, ROA+adaptive clustering+ant colony guidance strategy+ random recombination strategy. The calculation formula of error rate is consistent with that mentioned above, as shown in Table 10. Each group is tested 10 times, with the maximum iteration number of 2000 each time, and the excitation rate N is selected to be 10 times, and the average solution and error rate are calculated according to the results.

It can be seen from Table 11 that the optimal solution of group C is improved to A certain extent compared with group A and Group B. In addition, the solution of group A

Table 9. Performance comparison results of different optimization strategies

TSP instance	opt	scheme	average solution	optimal solution	error rate%
KroB150	26131	A	26352	26142	0.04
KroB150	26131	B	26217	26131	0
KroA200	29369	A	29705	29495	0.41
KroA200	29369	B	29506	29402	0.12
Pr226	80369	A	80982	80529	0.78
Pr226	80369	B	80658	80486	0.14
F1417	11861	A	12089	12016	1.31
F1417	11861	B	12128	11963	0.86

is fluctuated greatly because the ant colony guidance strategy is not added. In contrast, the average solution of group C is better than that of group A due to the addition of ant colony guidance strategy. Under the condition of ensuring the error rate, the addition of ant colony guidance strategy makes the algorithm more stable.

Table 10. Optimization scheme

Group number	Name	Scheme combination
A	ROA	ROA
B	ROA+AC+RRS	ROA+adaptive clustering+random recombination strategy
C	ROA+AC+ACO+RRS	ROA+adaptive clustering+ACO+random recombination strategy

Table 11. Performance comparison results of different optimization strategies

TSP instance	opt	scheme	average solution	optimal solution	error rate%
KroB150	26131	A	26625	26306	0.67
KroB150	26131	B	28817	28817	10.29
KroB150	26131	C	26217	26131	0
KroA200	29369	A	29772	29573	0.68
KroA200	29369	B	30341	30255	3.03
KroA200	29369	C	29506	29402	0.12
Pr226	80369	A	80982	80529	0.78
Pr226	80369	B	90195	90121	12.13
Pr226	80369	C	80658	80476	0.13

C. Performance analysis of random recombination strategy

In order to show the influence of random recombination strategies on this algorithm, two sets of experiments are set up to compare the accuracy of the solution, namely ROA+adaptive clustering+ant colony guidance strategy, ROA+adaptive clustering+ant colony guidance strategy+random recombination strategy. The calculation formula of error rate is consistent with the previous. Each group is tested 10 times, with the maximum iteration number of 2000 each time.

Table 12. Optimization scheme

Group number	Name	Scheme combination
A	ROA+AC	ROA+adaptive clustering+ACO
B	ROA+AC+ACO+RRS	ROA+adaptive clustering+ACO+random recombination strategy

As can be seen from Table 13, when the optimal solution is the same, the random recombination strategy improves the average solution significantly. As can be seen from U574 and Rat575, compared with group A, the random recombination strategy has a great improvement in the average solution and optimal solution for the evenly distributed TSP problem, which enables the algorithm to solve a wider range of problems with better solutions.

Table 13. Performance comparison results of different optimization strategies

TSP instance	opt	scheme	average solution	optimal solution	error rate%
F1417	11861	A	12565	11964	0.86
F1417	11861	B	12129	11964	0.86
U574	36905	A	41255	39588	7.27
U574	36905	B	39857	38751	5.01
Rat575	6773	A	7246	7009	3.47
Rat575	6773	B	7183	6924	2.22

4.4. Comparison with Traditional ACO

In order to compare proposed algorithm with traditional ACO, typical TSP problems such as eil51, eil76, KroA100, KroB100, KroB150, KraoA200, pr226, Lin318, fl417, pr439 are compared in this paper. Where, the time length refers to the time taken from the beginning of the algorithm to finding the optimal solution of the algorithm, as shown in Table 14.

Table 14. Comparison between traditional ACO and proposed method

No. TSP instance	No. opt	Proposed best	Proposed error rate%	Proposed time/m	ACO best	ACO error rate%	ACO time/m
eil51	426	**426**	0	5.31	426	0	3.86
eil76	538	**538**	0	6.86	538	0	4.96
KroA100	21282	**21282**	0	12.55	21282	0	9.79
KroB100	22141	**22141**	0	12.57	22141	0	10.71
KroB150	26131	**26131**	0	14.62	26306	0.68	12.65
KraoA200	29368	**29402**	0.11	15.95	29564	0.67	16.79
pr226	80369	**80475**	0.14	15.65	80529	0.79	17.46
Lin318	42029	**42496**	1.11	17.98	42705	1.61	25.77
fl417	11861	**11963**	0.85	18.61	12032	1.43	29.64
pr439	107217	**108602**	1.29	21.43	109422	2.07	30.65

It can be seen from Table 14 that in small-scale cities Eil51 and Eil76, proposed method and ACO can both find theoretical optimal solutions. In the medium-scale cities KroA100, KroB100, KroB150 and KroA200, the theoretical optimal solution can be found by proposed method except KroA200, while the traditional ACO can not find the theoretical optimal solution. In large-scale cities Lin318, F1417 and Pr439, none of the two algorithms can find the theoretical optimal solution, but proposed method has the highest accuracy. In addition, in terms of time, proposed method takes significantly longer than traditional algorithm to find initial solutions in small and medium scale due to the need to find current initial solutions. However, as the scale increases, the time complexity of traditional algorithm becomes higher and higher, while proposed method can rapidly converge and quickly find optimal solutions.

The rank sum test method can be used to clarify the differences between algorithms, and the results are shown in Table 15. When the data is less than 0.05, there is a significant difference between the two samples. It can be seen from the results in the table that there are obvious differences between the proposed algorithm and the traditional ACO in most TSP problems.

Table 15. Comparison table of rank sum tests

TSP instance	ACO	Proposed
eil51	**0.054**	0.049
eil76	0.013	0.037
KroA100	0.041	0.049
KroB100	**0.266**	0.063
KroB150	0.015	0.037
KraoA200	0.016	0.017
pr226	0.014	0.033
Lin318	0.003	0.007
fl417	0.001	0.004

4.5. Comparison with Other Advanced Methods

In order to further verify the performance of the algorithm, this paper compares the proposed method with the advanced algorithms in recent years. The comparison data includes NNM [21], DRSO [22] and SASL [23] algorithms, which all have better results in solving typical TSP problems. The experimental results are shown in Table 16. It can be seen from table 16 that the accuracy of proposed method is obviously improved when compared with the current newer algorithm, especially in the medium scale and large scale, the quality of the solution is obviously better than other algorithms. It can be concluded that the performance of proposed algorithm is better than that of other optimization algorithms.

Table 16. Comparison between different methods

No. TSP instance	No. opt	Proposed best	Proposed error rate%	NNM best	NNM error rate%	DRSO best	DRSO error rate%	SASL best	SASL error rate%
eil51	426	**426**	0	437	2.58	431	1.08	426	0
berlin52	7542	**7542**	0	7635	1.23	7545	0.03	7542	0
eil76	538	**538**	0	561	4.28	548	1.77	543	0.91
KroA100	21282	**21282**	0	22025	3.50	21286	0.02	21320	0.11
KroB100	22141	**22141**	0	23021	3.99	22372	1.03	22141	0.11
eil101	629	**629**	0	674	7.22	644	2.35	641	1.91
KroB150	26131	**26131**	0	27577	5.53	263508	0.83	26454	1.20
Rat195	2323	**2337**	0.6	2489	7.15	2599	8.12	2355	1.31
KraoA200	29368	**29402**	0.11	31829	8.83	29518	0.51	29599	0.71
pr439	107217	**108602**	1.29	116379	8.55	12148	10.21	99788	3.65

5. Conclusions

In order to solve the problem that ant colony algorithm convergence speed is slow and the effect is not ideal when solving large-scale problems, this paper proposes an ant colony algorithm-based ROA that integrates adaptive clustering and ant colony guidance strategy. Firstly, the large-scale problem is decomposed into several sub-problems through adaptive clustering, which speeds up the convergence speed and improves the accuracy of understanding. Secondly, several sub-problems are fused by ROA fusion strategy. Finally, the ant colony guidance strategy and opt were used to optimize the fusion results to further improve the accuracy of understanding. In the typical TSP problem, it is shown that the improved ant colony algorithm has higher convergence speed and solving precision than the traditional ant colony algorithm.

Nowadays, ant colony algorithm is widely used in travel salesman problem, robot path planning and other problems. The improved ant colony algorithm proposed in this paper can be applied in these fields, and has higher solving accuracy and convergence speed than other algorithms. The improved ant colony algorithm proposed in this paper has good performance on a large scale, but there is still room for improvement. The accuracy of this paper and the speed of the improved algorithm still need to be further improved when solving the very large scale traveling salesman problem. Next, we will consider optimizing the mother ant guidance strategy, so as to further improve the solution accuracy and convergence speed when dealing with very large scale problems.

References

1. Halim A H, Ismail I. Combinatorial optimization: comparison of heuristic algorithms in travelling salesman problem[J]. Archives of Computational Methods in Engineering, 2019, 26: 367-380.
2. Wang Y, Han Z. Ant colony optimization for traveling salesman problem based on parameters optimization[J]. Applied Soft Computing, 2021, 107: 107439.
3. Zheng R, Zhang Y, Yang K. A transfer learning-based particle swarm optimization algorithm for travelling salesman problem[J]. Journal of Computational Design and Engineering, 2022, 9(3): 933-948.
4. lhan $\dot{I}$, Gökmen G. A list-based simulated annealing algorithm with crossover operator for the traveling salesman problem[J]. Neural Computing and Applications, 2022: 1-26.

5. Stodola P, Michenka K, Nohel J, et al. Hybrid algorithm based on ant colony optimization and simulated annealing applied to the dynamic traveling salesman problem[J]. Entropy, 2020, 22(8): 884.
6. Hao T, Yingnian W, Jiaxing Z, et al. Study on a hybrid algorithm combining enhanced ant colony optimization and double improved simulated annealing via clustering in the Traveling Salesman Problem (TSP)[J]. PeerJ Computer Science, 2023, 9: e1609.
7. Ilin V, Simić D, Simić S D, et al. A hybrid genetic algorithm, list-based simulated annealing algorithm, and different heuristic algorithms for travelling salesman problem[J]. Logic Journal of the IGPL, 2022: jzac028.
8. Zhang P, Wang J, Tian Z, et al. A genetic algorithm with jumping gene and heuristic operators for traveling salesman problem[J]. Applied Soft Computing, 2022, 127: 109339.
9. Li W, Xia L, Huang Y, et al. An ant colony optimization algorithm with adaptive greedy strategy to optimize path problems[J]. Journal of Ambient Intelligence and Humanized Computing, 2022: 1-15.
10. Sharma S, Chakraborty S, Saha A K, et al. mLBOA: A modified butterfly optimization algorithm with lagrange interpolation for global optimization[J]. Journal of Bionic Engineering, 2022, 19(4): 1161-1176.
11. Panwar K, Deep K. Discrete Grey Wolf Optimizer for symmetric travelling salesman problem[J]. Applied Soft Computing, 2021, 105: 107298.
12. Xu H, Lan H. An Adaptive Layered Clustering Framework with Improved Genetic Algorithm for Solving Large-Scale Traveling Salesman Problems[J]. Electronics, 2023, 12(7): 1681.
13. Wu C, Fu X, Pei J, et al. A novel sparrow search algorithm for the traveling salesman problem[J]. IEEE Access, 2021, 9: 153456-153471.
14. Kusumahardhini N, Hertono G F, Handari B D. Implementation of K-Means and crossover ant colony optimization algorithm on multiple traveling salesman problem[C]//Journal of Physics: Conference Series. IOP Publishing, 2020, 1442(1): 012035.
15. Hamdan B, Bashir H, Cheaitou A. A novel clustering method for breaking down the symmetric multiple traveling salesman problem[J]. Journal of Industrial Engineering and Management, 2021, 14(2): 199-218.
16. Sun Yang, Teng Lin, Yin Shoulin, Li Hang. A new Wolf colony search algorithm based on search strategy for solving travelling salesman problem[J]. International Journal of Computational Science and Engineering, 18(1), pp:1-11, 2019.
17. Teng L. Brief Review of Medical Image Segmentation Based on Deep Learning[J]. IJLAI Transactions on Science and Engineering, 2023, 1(02): 01-08.
18. Jing Yu, Hang Li, Shoulin Yin. New intelligent interface study based on K-means gaze tracking[J]. International Journal of Computational Science and Engineering, vol. 18, no. 1, pp. 12-20, 2019.
19. Majumdar P, Mitra S, Bhattacharya D. Honey Badger algorithm using lens opposition based learning and local search algorithm[J]. Evolving Systems, 2023: 1-26.
20. Isoart N, Régin J C. A k-Opt Based Constraint for the TSP[C]//27th International Conference on Principles and Practice of Constraint Programming (CP 2021). Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.
21. Shi Y, Zhang Y. The neural network methods for solving Traveling Salesman Problem[J]. Procedia Computer Science, 2022, 199: 681-686.
22. Mzili T, Riffi M E, Mzili I, et al. A novel discrete Rat swarm optimization (DRSO) algorithm for solving the traveling salesman problem[J]. Decision making: applications in management and engineering, 2022, 5(2): 287-299.
23. Gong X, Rong Z, Wang J, et al. A hybrid algorithm based on state-adaptive slime mold model and fractional-order ant system for the travelling salesman problem[J]. Complex & Intelligent Systems, 2023, 9(4): 3951-3970.

Lin Piao is with the Department of Education, Liaoning National Normal College. He had published several papers related to his works. Research direction: math analysis, data analysis, artificial intelligence.

Received: March 14, 2024; Accepted: June 29, 2024.