Applied Intelligence (2022) 52:6552–6574
https://doi.org/10.1007/s10489-021-02774-9

Ant colony algorithm with Stackelberg game and multi-strategy fusion

Da Chen[1] · XiaoMing You[1] · Sheng Liu[2]

Accepted: 16 August 2021 / Published online: 13 September 2021
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2021

Abstract
Aiming at the disadvantages of the ant colony algorithm, such as slow convergence speed and easy to fall into local optimum, this paper proposes an ant colony algorithm with Stackelberg game and multi-strategy fusion. Firstly, Stackelberg game is established between ant colonies, and the population with the excellent performance is taken as the leader to increase the influence of excellent ant colony. Secondly, a multi-strategy fusion system is proposed, which is composed of three strategies: One is the pheromone fusion strategy, which selects the population whose entropy is less than the threshold value and the population with the highest similarity for pheromone fusion to increase the diversity of the algorithm. The second is the elite ant learning strategy, which speeds up the convergence rate by learning the elite ants of the elite population; The third is the pheromone recombination strategy, which helps the algorithm jump out of the local optimum. The simulation experiments of multiple cases in TSPLIB show that the improved algorithm balances diversity and the convergence speed, and effectively improves the quality of the solution.

Keywords TSP problem · Multiple populations · Stackelberg game · Multi-strategy fusion

1 Introduction

Traveling Salesman Problem [1–5] (TSP) is a classic NP-hard problem. There are many algorithms to solve the TSP problem, among which the ant colony algorithm is one of the best algorithms because it has good robustness and convergence speed. In addition, ant colony algorithms have a variety of applications in other fields, such as robot path planning [6], network routing problems [7] and scheduling problems [8, 9].

In the early 1990s, Italian scholars Dorigo et al. proposed the Ant System (AS) [10, 11] based on the foraging behavior of ants, but the algorithm has a slow convergence speed and is prone to stagnation in solving the TSP problem. Therefore, in 2006, Dorigo proposed the ant colony system (ACS) [12]. Compared with the AS algorithm, this algorithm greatly improves the convergence speed and solution accuracy in solving the TSP problem. At the same time, Stutzle et al. proposed the Max-min Ant System (MMAS) [13], which sets the upper and lower bounds as pheromone as a way to improve the solution efficiency of the algorithm. The above are some classical ant colony algorithms, which have greatly improved the solving ability but still have problems such as easy stagnation, slow convergence, and poor diversity.

Since then many experts have come up with their own improvements, Some algorithms improved the diversity of the algorithms; Yue Wu et al. helped the algorithm to improve the computational efficiency by designing a local search method [14]; Wei Gao proposed a premium penalty strategy that changes the distribution of pheromone as a way to increase the diversity on good paths and increase the search space [15]; Ye Ke et al. proposed a negative feedback mechanism to help the algorithm explore more unknown regions by continuously acquiring the experience of ant failures [16]; S.Li et al. proposed a collective action mechanism to improve the collaboration between individual

✉ XiaoMing You
yxm6301@163.com

Da Chen
1051860334@qq.com

Sheng Liu
ls6601@163.com

[1] College of Electronic and Electrical Engineering, Shanghai University of Engineering Science, Shanghai, 201620, China

[2] School of Management, Shanghai University of Engineering Science, Shanghai, 201620, China

ants [17]. The above algorithms enhance diversity through their respective methods, but suffer from the disadvantage of slow convergence of the algorithms.

Some algorithms improved the convergence of the algorithm; Qin et al. accelerated the convergence of the algorithm by enhancing the pheromone update and improving the positive feedback of the optimal path [18]. Sahar used clustering method to reduce the number of nodes in the TSP, which greatly improves the convergence speed. [19] These algorithms alleviate the problem of slow iteration of the ant colony algorithm to some extent, but do not have good diversity such that they tend to fall into local optima in the later stages of the algorithm.

There is a contradiction between the convergence speed and diversity of ant colony algorithms, and to improve this problem, some scholars have proposed many ingenious methods. Rafał; Skinderowicz used GPU parallel computing to speed up operations and increase the diversity of the algorithm by changing the roulette wheel [20]. Zhao D. et al. proposed a horizontal crossover strategy and a vertical crossover strategy for speeding up the convergence and expanding the search range of ants, respectively [21]. Liu et al. proposed a random reference mechanism to speed up the convergence and used a chaotic reinforcement strategy to improve the accuracy of the solution [22]. Han, ZP et al. combined the ant colony algorithm with symbiotic search to improve the search efficiency by adaptively changing the search of ants while rapidly selecting optimization parameters [23]. Wei Gao applied two meeting ants to construct the path and a strategy that polarized the pheromone on the path to improve the computational efficiency and effectiveness of the algorithm, respectively [24].

Other scholars applied multiple swarm algorithms and introduced knowledge from other fields to balance convergence and diversity and improve coordination between ants. Xiaoyu Wang et al. evaluated the uncertainty of pheromone through information entropy and improved population diversity through disturbance mechanism [25]. Zhou et al. proposed an ant colony algorithm that combines different search ranges and convergence speeds to greatly increase the diversity of the algorithm [26]. Dehui Zhang et al. proposed a collaborative filtering strategy to improve the efficiency of communication between populations [27]. Han Pan et al. smoothed the pheromones through a dynamic bootstrap mechanism and determined the communication frequency by comparing minimum spanning trees [28].

In this paper, we will propose ant colony algorithm with Stackelberg game and multi-strategy fusion(MSACS) for TSP problem; In order to improve the influence of high-quality populations, the algorithm will establish a Stackelberg game among multiple populations; In addition, a multi-strategy fusion system is proposed to communicate various information between populations. The main contributions of this paper are as follows:

1. In order to apply the superiority of the different populations, we choose ACS algorithm and MMAS algorithm to form heterogeneous multi-population ant colony algorithm.
2. The Stackelberg game model is established among multiple populations; Through the comprehensive evaluation of convergence, diversity and the overall state of the algorithm, the current best quality population is selected as the leader; Then the leader acts as a pioneer to explore the path for other populations, and forms a cooperative relationship with other populations through the exchange of information, so as to maximize the benefits of the whole system.
3. Multi-strategy fusion mechanism is used to improve the information exchange between populations. There are three strategies in this mechanism: The pheromone fusion strategy is used to improve the diversity of populations with low information entropy; The elite ant learning strategy is used to learn elite ants from excellent population to improve the convergence speed of the algorithm; The pheromone recombination strategy is used to smooth the over-high pheromone in non-public path, and local search is carried out to help the algorithm jump out of local optimal.

This paper is organized as follows: Section 2 introduces the background knowledge of traditional ant colony algorithm, information entropy and Stackelberg game; Section 3 introduces the main innovations and contributions of this paper; Section 4 describes the various comparative experiments and parameter Settings; Section 5 mainly summarizes our work and some of our future research directions.

2 Related research

2.1 Ant colony algorithm for solving Tsp problem

2.1.1 Path Selection

As shown in (1), the transfer of ant k from city i to city j conforms to the pseudo-random rule:

$$j = \begin{cases} \arg\max \left\{ \tau_{ij} \left[\eta_{ij} \right]^{\beta} \right\}, q \leq q_0 \\ J, q > q_0 \end{cases} \tag{1}$$

where q is a random number between [0,1]; q_0 is a deterministic parameter between [0,1];When the random number q is less than the parameter q_0, the ant k moves to the next city in the determined direction; When the random number q is greater than the parameter q_0, the ant k then moves to the next city by probability with the J; J is calculated from (2).

$$p_{ij}^k = \begin{cases} \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{ij}]^\alpha [\eta_{ij}]^\beta}, j \in N_i^k \\ 0, j \notin N_i^k \end{cases} \tag{2}$$

where α is the heuristic factor of pheromone; β is the heuristic factor of the greedy rule; N_i^k is the collection of cities that the ant k can choose from; τ_{ij} is the concentration of pheromone between node i and node j; η_{ij} is the heuristic function, whose expression is formula (3):

$$\eta_{ij} = \frac{1}{d_{ij}} \tag{3}$$

where d_{ij} is the distance between node i and node j;

2.1.2 Pheromone update

Local pheromone update rule: After the ant moves from city i to city j, the calculus will be updated for local pheromones as shown in (4):

$$\tau_{ij} \leftarrow (1-\rho)\,\tau_{ij} + \rho\tau_0 \tag{4}$$

where ρ is the volatile factor of local pheromone; τ_0 is the initial value of the pheromone;

Global pheromone update rule: When all ants have finished solving, the algorithm will perform a global pheromone update, as shown in (5):

$$\tau_{ij} \leftarrow (1-\xi)\,\tau_{ij} + \xi\Delta\tau_{ij}^{bs} \tag{5}$$

$$\Delta\tau_{ij}^{bs} = \frac{1}{C^{bs}} \tag{6}$$

where ξ is the volatility factor of the global pheromone; C^{bs} is the value of the global optimal solution; $\Delta\tau_{ij}^{bs}$ is the increment of global pheromone, as shown in (6):

2.2 Max-min ant colony algorithm

In order to improve the search efficiency of ant colony algorithm and avoid falling into local optimum prematurely, The MMAS algorithm sets upper and lower limits for the pheromone. The calculation methods of $\tau_{\max}$ and $\tau_{\min}$ are shown in (7) and (8):

$$\tau_{\max} = \left(1\big/\rho\right) * \left(1\big/T^{gb}\right) \tag{7}$$

$$\tau_{\min} = \tau_{\max}\big/2n \tag{8}$$

Where $\tau_{\max}$ is the upper limit set by the algorithm for the pheromone; $\tau_{\min}$ is the lower limit of pheromone; T^{gb} 1 is the current optimal solution value of the algorithm; ρ is the volatility factor of the pheromone; n is the number of nodes in the city.

2.2.1 Pheromone Update

The MMAS algorithm updates only the pheromone on the current optimal solution path; From (9) and (10), we can see the updating rules of pheromone.

$$\tau_{ij}\,(t+1) = (1+\rho)\,\tau_{ij}\,(t) + \Delta\tau_{ij}^{best} \tag{9}$$

$$\Delta\tau_{ij}^{best} = 1\big/f\left(s^{best}\right) \tag{10}$$

Where τ_{ij} is the value of the pheromone between node i and node j in the MMAS algorithm; t is the iteration count; $\Delta\tau_{ij}^{best}$ is the amount of change in the pheromone of the node through which the current optimal individual passes, which is obtained from (10); $f\left(s^{best}\right)$ is the size of the current optimal solution.

2.3 Lnformation entropy

Information entropy was put forward by Shannon [29] to evaluate the degree of disorder of information. The calculation method is shown in (11):

$$S\,(x) = -\sum_{i=1}^{n} P\,(x_i)\log_b\,(P\,(x_i)) \tag{11}$$

Where b is the base value of the logarithm; $P(x_i)$ is the quality function of probability. n is the number of solutions.

2.4 Stackelberg game

The Stackelberg game is a dynamic game of bounded rationality proposed by Stackelberg in 1953 [30]. First, the Stackelberg game divides the players into leaders and followers; Then, the leader goes first, and the followers make the decision that suits their own best interests on the basis of the leader's decision, and finally achieve dynamic equilibrium. The game model can be described as (12):

$$H = \{(L \cup \{F\})\,, \{P_l\}_{l\in L}, \{P_f\}_{f\in F}, R_l, R_f\} \tag{12}$$

where L is the selected leader, $\{L \cup \{F\}$ is all the participants; P_l is the leader's policy set, P_f is the follow policy set, R_l is the leader's revenue function, and R_f is the revenue of follows.

The goal of each game is to maximize the respective revenue, so the objective functions of leader and follow are (13) and (14):

$$\max_{p_l \in P_l} R\,(p_l) \tag{13}$$

$$\max_{p_f \in P_f} R_n\left(p_f\right) \tag{14}$$

where $R\left(p_l\right)$ and $R_n\left(p_f\right)$ are the objective functions of the leader and the follower respectively, and the whole master-slave game aims at maximising these two values.

To sum up, when both sides of the game reach Nash equilibrium, all participants gain the most and do not change their strategies Return matrix.

3 Ant colony algorithm with Stackelberg game and multi-strategy fusion

The Stackelberg game is a classic game model. In this game, the leader makes the decision first, and the followers make their own decisions after the leader makes the decision. Based on the traditional idea of Stackelberg game, we design a Stackelberg game among multi ant colonies. When the algorithm is running, the group that is beneficial to the whole system will be selected as leader, while the others will be viewed as followers. Then, the leader, a pioneer that explored other paths, trains a number of iterations after exchanging information with other followers. After the path has been explored, the most beneficial strategy will be applied by the followers to obtain the information explored by the leader. Finally, after exchanging information, the followers continue to explore the path. From the Fig. 1, we can see how the Stackelberg game between multiple populations works. The dynamic Stackelberg game model is divided into the following steps:

step1: Select the leader by (18);
step2: Leader make decisions first;
step3: Followers make decisions after the leader;
step4: back to Step1;

3.1 The establishment of Stackelberg game among ant colony algorithms

3.1.1 Parameters for evaluating population and algorithm state

Information entropy evaluates diversity

There are many indicators to evaluate the diversity of ant colony algorithm, such as standard deviation, information entropy and so on. For ant colony algorithms, keeping the variety of algorithms allows ants to choose more different paths. We use information entropy to measure the diversity of the algorithms, as shown in (15)

$$E\left(P\right) = -\sum_{x \in X} P\left(x\right) \log\left(P\left(x\right)\right) \tag{15}$$

Where $P\left(x\right)$ is the size of ant x in the current solution. $E\left(P\right)$ is the information entropy of all the solutions in the current population. The larger the deviation value of the solution is, the higher the information entropy will be, thus the higher the diversity will be.

Convergence evaluation

Convergence indicates the convergence ability of the current population, and the better the convergence is, the faster the current algorithm converges. In this section, we use (16) to evaluate the convergence of the current algorithm.

$$conv = \frac{length^i_{\min} - length_{\min}}{iter^i - iter^j} \tag{16}$$

Where $conv$ is the convergence of the current population, $length_{\min}$ is the optimal solution of the current population; $iter^j$ is the current number of iterations; $length^i_{\min}$ was the optimal solution last time; $iter^i$ is the number of iterations in which the last optimal solution is found.

Assess system status

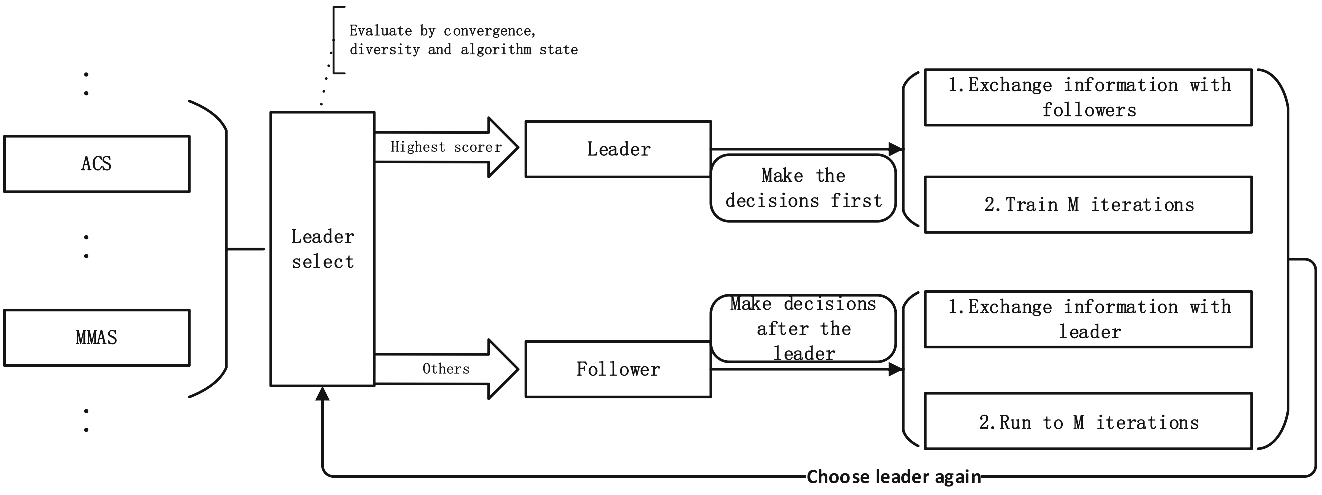

Evaluate by convergence, diversity and algorithm state
ACS
MMAS
Leader select
Highest scorer
Others
Leader
Follower
Make the decisions first
Make decisions after the leader
1. Exchange information with followers
2. Train M iterations
1. Exchange information with leader
2. Run to M iterations
Choose leader again

Fig. 1 The flow of the Stackelberg game

Similarly, we use the convergence of the optimal solution to measure the current state of the algorithm; If the optimal solution converges rapidly, it indicates that the current system is in a period of rapid convergence. At this time, the algorithm needs the population with better convergence as the leader; otherwise, it needs the population with better diversity. We use (17) to express the convergence of the overall algorithm.

$$CO = \frac{iter_{\min}^{i}}{iter_{\min}} \tag{17}$$

Where $iter_{\min}$ is the number of iterations of the current optimal solution; $iter_{\min}^{i}$ is the number of iterations of the previous optimal solution.

3.1.2 Multi-factor evaluation mechanism

After the number of planned iterations, the comprehensive evaluation index of each population will be calculated by (18), and the population with the highest score will be selected as the leader.

$$Y_i = (1 + CO)\, conv_i E(P)_i \tag{18}$$

Where Y_i is the measure of the comprehensive evaluation index of population i; CO reflects the overall convergence of the algorithm, which is obtained from (17); $conv_i$ is the convergence of population i; $E(P)_i$ is the information entropy of population i, which represents the diversity of population.

3.1.3 The choice of leader

In our system, leader exists to lead other groups because of their distinct advantages over other groups. Equation (18) is used to evaluate the overall performance of the population. There are many reasons for using these three variables as the standard of the comprehensive ability of the population; First, how leader is chosen depends on the current state of the system. If the current system is in a period of rapid convergence, it indicates that we need the population with better convergence and higher precision as the leader; If the convergence of the algorithm is slow at present, it means that the whole system needs those populations with good diversity as leaders to help the population find more different paths. From the (17), we can judge the convergence of the algorithm as a whole, and the current state of the algorithm can also be judged from another perspective. Secondly, we also need to consider the current state of the population, so we add the evaluation index of convergence and diversity of the algorithm into the formula, which can reflect the state index of a single ant colony to some extent. Therefore, by correlating these three parameters and considering the overall performance of the algorithm, we also consider the state of a single ant colony, which can reflect the comprehensive performance of the population.

3.1.4 Number of training iterations for leaders

In order to explore more paths, the leader trains M iterations in advance. There are many ways to determine M, such as specifying M as a fixed value directly, but that is not the best way. In this section, we use the similarity (We use cosine similarity to evaluate the population similarity, as shown in (20)) between populations to determine whether the leader has been trained; As shown in (19), when L is greater than l (l is the threshold), the leader training is considered to be completed, and then the number of training iterations of the leader is M. The selection steps for M are shown in Table 1 below.

$$L = \frac{\left|S_M^{leader} - S_M^{follower\_\min}\right|}{S_M^{leader}} \times 100\% \tag{19}$$

Where S_M^{leader} is the cosine similarity of the leaders; $S_M^{follower\_\min}$ is the minimum cosine similarity of the follower; L represents the degree of difference between leaders and followers. The greater the value of L, the more obvious the difference is.

3.1.5 Cosine similarity

In this section, we use cosine similarity to evaluate the similarity between populations, as shown in (20):

$$S_M = \frac{A \cdot B}{|A| \times |B|} \tag{20}$$

Where A and B is a 2-dimensional vector made up of $conv$ (Calculated by (16)) and E (Calculated by (15)), which represents the state of the current population to some extent; As can be seen from the (20), the higher the value of S_M, the more similar the two populations are.

3.2 Multi-strategy fusion mechanism

After the populations make their choices, they will exchange information, which is achieved by selecting various mechanisms. If the population chooses to cooperate after the game,

Table 1 The calculation process of M

```
1 M=0;
2 While termination condition is not satisfied do
3     M=M+1;
4     Calculate the similarity;
5     Calculate the difference between leaders and followers;
6 End-While
```

then the population will choose to exchange information with the cooperating population; For a population, there are many kinds of information that can be exchanged, such as pheromone, optimal solutions common paths, and so on; Therefore, the choice of information exchange is a key issue for the participants.

In this section, we define different types of information; For different information about an ant colony, it reflects the current state of the colony differently; For example, The pheromone of the population contains the path selection of the population history, and the communication of pheromone among the populations can help them to increase the choice of different paths, thus improving the diversity of the algorithm; Therefore, the population chooses different information to communicate, which usually results in different effects.

We will present three strategies for populations to communicate different information between populations in order to improve their required performance. First, the pheromone fusion strategy will be used to improve experience exchange between populations; In general, the pheromone matrix of an ant colony algorithm contains the path chosen in the history of the population, which reflects the fuzzy experience of the population to some extent, and has the path dependence of the population; If a population exchanges pheromone, it will gain experience from other populations and increase the probability of choosing alternative paths, which is shown in Section 3.2.1. Secondly, the elite ant learning strategy increases the influence of the dominant population. The elite individuals of the dominant population contain the excellent characteristics of the population, and communication and learning with them can improve the convergence of other populations, as shown in Section 3.2.2. Finally, we propose the pheromone recombination strategy to help the algorithm avoid stagnation when the population is trapped in a local optimum, which is shown in Section 3.2.3.

3.2.1 Pheromone fusion strategy

The pheromone fusion strategy is implemented when the population has reached the specified time for communication and has converged to a certain extent and its diversity has decreased. Firstly, the information entropy of each population is calculated by (15), which represents the diversity of the population. Secondly, the population with the highest value of information entropy is selected to form a new pheromone matrix through linear fusion by pheromone fusion rules. By reorganizing the pheromone of the population in this way, we can improve the exploration of other paths, which in turn can increase the diversity of the population; Since this strategy will reduce the convergence of the algorithm, we use the elite ant learning strategy in Section 3.2.2 to balance the diversity of the population and avoid the disadvantage of slow convergence of the algorithm. The pheromone sharing between ant colonies will be given by the following (21).

$$Ph^k = (1 - S_M)\, Ph^k + S_M Ph^m \tag{21}$$

Where Ph^k is the pheromone matrix of the current ant colony; Ph^m is the pheromone matrix of the population selected to communicate pheromone; S_M is the similarity value of the two populations, which is obtained from (20); According to (21), we form a new pheromone by linear combination of the pheromone matrix of the two populations in a certain proportion.

3.2.2 Elite ant learning strategy

After using pheromone sharing strategies to increase the population diversity, we need some strategies to increase the convergence rate of the algorithm to reduce the conflict between convergence and diversity; In this paper, (17) is used to evaluate the overall convergence of the algorithm. In addition, when the convergence value CO of the algorithm is less than $\omega_1(\omega_1$ is the current threshold of convergence), we will enable the elite ant learning strategies. We will select the population with the highest accuracy as the learning target, and learn the elite ants of this population to accelerate the convergence of the algorithm.

In this section, the ants in the population that are close to the top K% by the order of solutions are considered to be elite ants. When the population learns the elite ant, it learns by rewarding the pheromone in the path of the elite ant. We use (22). to reward pheromone in the path of the elite ants. The learning process is shown in Fig. 2

$$P_n = \left(1 + \frac{CO}{n}\right) P \tag{22}$$

Where P is the value of the pheromone in the public path; P_n is the new value of the rewarded pheromone in the common path; n is the number of nodes in the city; CO is the evaluation of the overall convergence of the algorithm, which is obtained from (17); At the beginning of the algorithm iteration, the convergence of the algorithm is fast and the value of CO is relatively large, which rewards more pheromone in the common path and speeds up the convergence speed of the algorithm; As the number of iterations increases, the convergence rate of the algorithm slows down, thus reducing the pheromone rewarding the path of elite ants.

3.2.3 Pheromone recombination strategy

The classical ant colony algorithm is easy to fall into local optimum (When the optimal solution does not change for

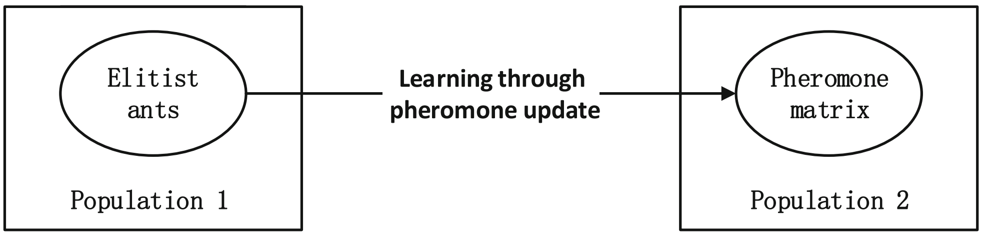


Fig. 2 Schematic diagram of learning strategies of elite ants

more than 200 iterations, it is considered to be trapped in the local optimal solution), and when the number of nodes to solve the problem is more, the algorithm is easier to fall into local optimum. Therefore, we need a strategy to help ant colony algorithm to jump out of local optimum. First of all, from the perspective of the solution mechanism of ant colony algorithm, the reason why ant colony algorithm falls into local optimum is usually that the concentration of pheromone matrix on the edge of some paths is too high, which reduces the possibility of ants in the ant colony to find other paths, so they fall into local optimum. From this perspective, we can help ACO jump out of local optimum by changing the distribution of pheromone. There are many ways to reset the pheromone of an ant colony. The simple way is to initialize the pheromone matrix, but the disadvantage is that there is no way to take advantage of the long experience of an ant colony, which leads to low efficiency. Therefore, we need to maintain the excellent experience of the ant colony in the pheromone matrix while removing the pheromone in the path with high concentrations. The specific operation of this policy is as follows:

step1: Find the common paths between populations.
step2: The pheromone concentration of the public path is maintained, and the pheromone smoothness of the non-public path is carried out by using (23);
step3: The order of public paths is maintained, and the local search for non-public paths is carried out by ant colony algorithm;
step4: Update the pheromone of a non-public path;
step5: All paths are searched again.

$$P_l = \frac{(P_{\max} + P_{\min})\, P_{best}}{2} \tag{23}$$

Where P_{max} is the maximum in the pheromone matrix; P_{min} is the minimum value in the colony's pheromone matrix; P_{best} is the pheromone on the optimal solution in the current population when the algorithm is stagnant. Through (23), we reduce the pheromone in the optimal solution path with high pheromone concentration in the stagnant population, and re-assign the original path by averaging the maximum and minimum values in the pheromone matrix; Doing so in this way has two benefits: One is that we can reduce pheromone on the pheromone pathway at very high concentrations. Second, the population can retain most of the original experience of finding paths to avoid reducing the efficiency of finding solutions.

3.3 Algorithm framework

First, the ACS algorithm and the MMAS algorithm are chosen as the ant colony algorithm for the various swarms; Then, a certain number of iterations are used to train the various parameters of each swarm; After the training is completed, the parameters are brought into the model of Stackelberg game, and the state of the various swarms is evaluated by a combination of the highest rated swarms as leaders and the rest as followers; The next training time for the various groups is calculated; Leaders take the lead in M iterations of training, after which information is obtained from followers through a multi-strategy fusion mechanism; Followers then conduct a search for solutions; Finally the algorithm continues to loop the loop until the requirements are met. The pseudo-code and flowcharts for the MSACS algorithm are shown below, as shown in Table 2 and Fig. 3.

Comparison with the literature [14] to [19] mentioned in the introduction, through a multi-strategy fusion mechanism, the MSACS algorithm balances convergence and

Table 2 Table of algorithmic framework

```
MSACS for TSP
1 Initialize the parameters of each population
2 Calculate the distance between nodes
3 While termination condition is not satisfied do
4     All populations run N iterations
5     Choosing leaders and followers
6     Calculating the number of leader training sessions M
7     Information sharing through multi-strategy fusion mechanism
8     for i=1:M
9         Leader seeking solutions
10        Pheromone Update
11    end
12    Information sharing through multi-strategy fusion mechanism
13    for i=1:M
14        Followers seeking solutions
15        Pheromone Update
16    end
17    nc = nc+M
18 End-While
19 Output best solution
```

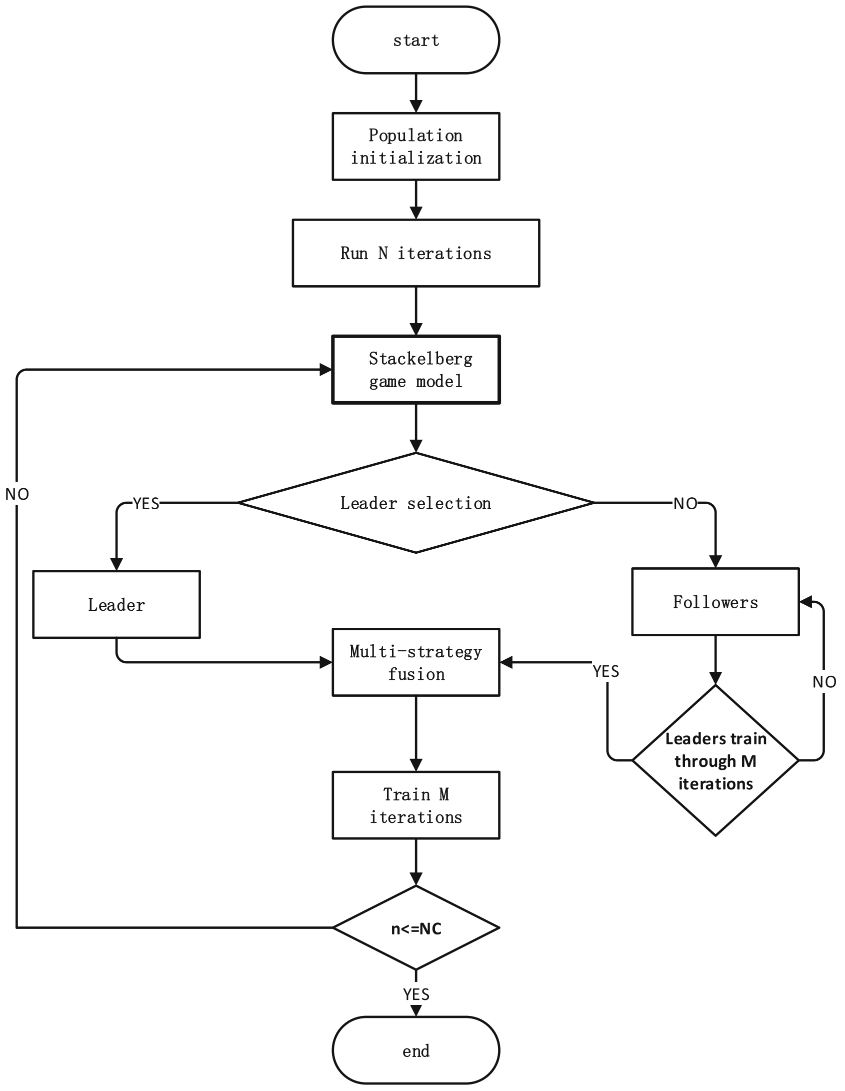


Fig. 3 Flowchart of the algorithm

diversity very well. In addition to this, modelling the master-slave game between multiple swarms can effectively exploit the characteristics of the various swarms and improve the synergy between ants compared to literature [20] to [24].

4 Experiment and simulation

In order to verify the performance of the MSACS algorithm, this paper selects 30 different TSP instances to test the algorithm from the TSPLIP library, and each example needs to carry out 20 experiments. The experimental test environment of this paper is Windows10 operating system, and Matlab2019A is used for simulation.

In Section 4.1, orthogonal experimental method is used to select the parameters of MMAS and ACS. The best combined parameters is selected through experiments. The multi-population ant colony algorithm we improved is composed of ACS algorithm and MMAS algorithm, therefore, the best parameters of the improved ant colony algorithm also choose these parameters.

In Section 4.2, we analyze the validity of the MSACS algorithm strategy; In Section 4.2.1, the effectiveness of multi-strategy fusion strategy is analyzed; In Section 4.2.2, the training time of leaders is analyzed.

In Section 4.3.1, Through 30 TSP examples, we compare the performance of MSACS, ACS and MMAS; In addition, we analyze the data differences of MSACS algorithm, ACS

Table 3 Experimental factors and levels of MMAS

	Level1	Leve2	Leve3	Leve4
α	1	2	3	4
β	1	2	3	4
ρ	0.1	0.2	0.3	0.4

Table 4 Results of MMAS orthogonal test

na	α	β	ρ	results
1	1	1	0.1	433.60
2	1	2	0.2	428.67
3	1	3	0.3	427.83
4	1	4	0.4	428.07
5	2	1	0.2	447.50
6	2	2	0.1	436.20
7	2	3	0.4	435.17
8	2	4	0.3	431.47
9	3	1	0.3	489.43
10	3	2	0.4	457.37
11	3	3	0.1	435.00
12	3	4	0.2	434.87
13	4	1	0.4	535.43
14	4	2	0.3	460.93
15	4	3	0.2	442.80
16	4	4	0.1	435.30

algorithm and MMAS algorithm in 30 groups of examples, through the rank sum test. In Section 4.3.2, The MSACS algorithm is in comparison with other ant colony algorithms and intelligent algorithms.

4.1 Parameter setting

In this section, we will choose the best parameter combination for MSACS algorithm, ACS algorithm and MMAS algorithm through orthogonal test; Tables 3, 4, and 5 shows the experimental results of parameter selection of MMAS; Tables 6, 7, and 8 shows the experimental results of parameter selection of ACS. Each experiment used eil51 as an example. For each set of parameters, we conducted experiments of 30 times respectively. Based on the above experiments, It can be obtained from the Table 5 that the best combination of parameters of the MMAS algorithm is: $\alpha = 1, \beta = 4, \rho = 0.1$; In addition, It can be obtained from the Table 5 that the best combination of parameters of the ACS algorithm is: $\alpha = 2, \beta = 4, \rho = 0.4, \zeta = 0.2, q_0 = 0.8$.

Table 5 Test results of MMAS

T	α	β	ρ
T_1	1718.17	1905.97	1740.10
T_2	1750.33	1783.17	1753.83
T_3	1816.67	1740.80	1809.67
T_4	1874.47	1729.70	1856.03
t_1	429.54	476.49	435.03
t_2	437.58	445.79	438.46
t_3	454.17	435.20	452.42
t_4	468.62	432.43	464.01
max	468.62	476.49	464.01
min	429.54	432.43	435.03
range	39.08	44.07	28.98
scheme	Level 1	Level 4	Level 1

Table 6 Experimental factors and levels of ACS

	Level1	Leve2	Leve3	Leve4
α	1	2	3	4
β	1	2	3	4
ρ	0.1	0.2	0.3	0.4
ζ	0.1	0.2	0.3	0.4
q_0	0.6	0.7	0.8	0.9

In Tables 5 and 8: T is the sum of the results. t is the average of each level range. *range* is the difference between the maximum minus the minimum, which will be used to determine which factor is important, and a larger range is usually more important. The *scheme* is to get the best result by orthogonal test of each factor.

4.2 Policy testing and performance analysis

4.2.1 Analysis of the effectiveness of the mechanism

The multi-strategy fusion strategy proposed includes three mechanisms: the pheromone sharing mechanism, the elite ant learning mechanism and the pheromone recombination mechanism. Firstly the pheromone sharing mechanism is applied to increase the diversity of the population; Secondly the elite ant learning mechanism helps the algorithm improve the convergence speed by learning the elite individuals of the

Table 7 Results of ACS orthogonal test

na	α	β	ρ	ζ	q_0	results
1	1	1	0.1	0.1	0.6	431.33
2	1	2	0.2	0.2	0.7	429.07
3	1	3	0.3	0.3	0.8	429.13
4	1	4	0.4	0.4	0.9	428.77
5	2	1	0.2	0.3	0.9	4293.10
6	2	2	0.1	0.4	0.8	429.17
7	2	3	0.4	0.1	0.7	428.90
8	2	4	0.3	0.2	0.6	428.67
9	3	1	0.3	0.4	0.7	431.77
10	3	2	0.4	0.3	0.6	431.50
11	3	3	0.1	0.2	0.9	428.43
12	3	4	0.2	0.1	0.8	429.47
13	4	1	0.4	0.2	0.8	430.20
14	4	2	0.3	0.1	0.9	431.53
15	4	3	0.2	0.4	0.6	432.3
16	4	4	0.1	0.3	0.7	430.33

Table 8 Test results of ACS

T	α	β	ρ	ζ	q_0
T_1	1718.3	1722.4	1719.3	1721.2	1723.8
T_2	1715.8	1721.3	1719.9	1716.4	1720.1
T_3	1721.2	1718.8	1721.1	1720.1	1717.9
T_4	1724.4	1717.2	1719.4	1722.0	1717.8
t_1	429.6	430.6	429.8	430.3	430.9
t_2	428.9	430.3	429.9	429.1	430.0
t_3	430.9	429.7	430.3	430.0	429.5
t_4	431.1	429.3	429.8	430.5	429.5
max	431.1	430.6	430.3	430.5	430.9
min	428.9	429.3	429.8	429.1	429.5
range	2.2	1.3	0.5	1.4	1.4
scheme	Level 2	Level 4	Level 4	Level 2	Level 3

dominant population; Finally The pheromone recombination mechanism is applied to improve the probability of the algorithm jumping out of local optimum.

We use MSACS-1 to label the MSACS algorithm without the pheromone fusion strategy, MSACS-2 to label the MSACS algorithm without the elite ant learning strategy, and MSACS-3 to label the MSACS algorithm without the pheromone recombination strategy; We used three TSP instances (kroA100, kroB150, and A280) to test the effectiveness of the strategy. Each instance ran 20 experiments respectively, and each instance ran 2000 generations; In the comparison experiment, the following parameters are used to judge: the optimal solution (best), the worst solution (worst), the average solution (mean) and the average deviation (MeanError(%)); The experimental results are shown in Fig. 4 and Table 9.

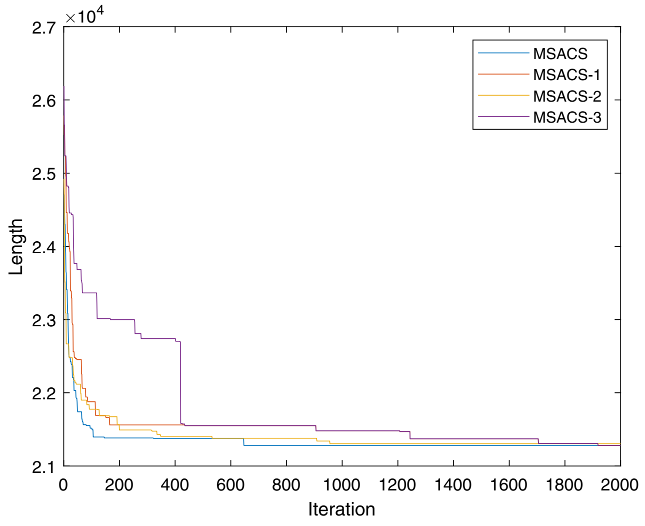
As shown in Table 9 and Fig. 4, the convergence speed and solving quality of MSACS are better than MSACS-1,MSACS-2 and MSACS-3; In addition, due to the absence of the elite ant learning strategy, the convergence speed of MSACS-2 is the slowest among several algorithms, which verifies that this mechanism can enhance the speed of convergence; Finally, in the experimental comparison of A280, it can be seen that the accuracy of MSACS-3 is the lowest among several algorithms, which is due to the lack of the pheromone recombination strategy, resulting in it falling into local optimum in the late stage of most experiments.

4.2.2 Analysis of the frequency of information exchange

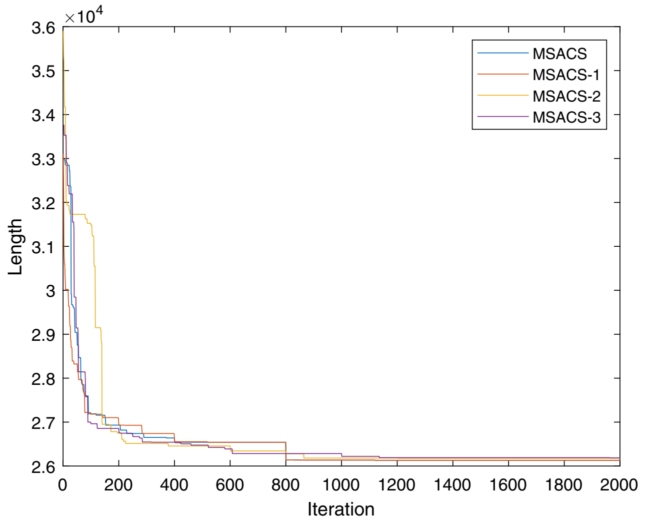
In this paper, the leader selected through stackelberg game needs to run for a period of time M before other populations. In order to determine the optimal running time, experiments were conducted with 50 iterations, 100 iterations, 150 iterations, 200 iterations, 250 iterations, 300 iterations, 350 and fq (The number of dynamic iterations is determined by Section 3.1.4) iterations respectively; TSP instances EIL51, EIL76, KROA100, and CH130 were used to run

Table 9 Comparison of MSACS, MSACS-1, MSACS-2 and MSACS-3

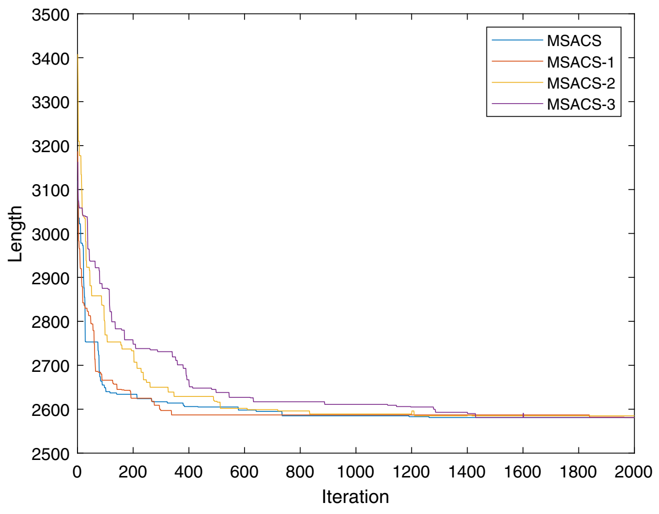
TSP instances	Algorithm	Best	Worst	Mean	MeanError(%)
	MSACS	21282	21292	21282.55	0.00
kroA100	MSACS-1	21282	21292	21307.54	0.12
	MSACS-2	21282	21305	21309.67	0.13
	MSACS-3	21282	21331	21308.39	0.12
	MSACS	26130	26228	26153.60	0.09
kroB150	MSACS-1	26130	26268	26166.58	0.14
	MSACS-2	26130	26423	26161.36	0.12
	MSACS-3	26183	26644	26184.87	0.21
	MSACS	2579	2610	2595.00	0.64
a280	MSACS-1	2580	2642	2598.34	0.75
	MSACS-2	2582	2626	2599.89	0.81
	MSACS-3	2590	2653	2605.05	1.01



(a) Convergence curves in kroA100



(b) Convergence curves in kroB150



(c) Convergence curves in a280

Fig. 4 Convergence comparison of MSACS, MSACS-1, MSACS-2 and MSACS-3

Table 10 The choice of M value

	Instance			
Choose	Eil51	Eil76	KroA100	Ch130
50	436.80	546.05	21310.84	6204.87
100	431.52	546.21	21294.55	6195.54
150	432.22	544.64	21286.21	6177.42
200	429.55	541.17	21290.91	6151.24
250	430.45	542.11	21289.65	6164.47
300	432.13	543.31	21293.33	6184.33
350	433.77	543.56	21305.24	6205.84
fq	426.05	538.15	21282.55	6143.75

20 groups each and run 2000 iterations. The results are shown in Table 10, which shows the average solution of 20 experiments. According to the table, the best parameter is fq, which is determined by the similarity between the populations.

4.3 Comparative experimental analysis

4.3.1 Contrastive analysis with classical ant colony algorithm

In order to compare the abilities of ACS, MMAS and MSACS algorithms, we selected 30 different TSP instances for comparative analysis. The experiment is analyzed in terms of the following evaluation parameters: the optimal solution ($best$), the minimum error(PD_{best}), the worst solution ($Worst$), the worst error(PD_{worst}), the average solution ($Average$), the average error(PD_{avg}), the optimal iteration number ($iters$), the Standard deviation (std), and the rank sum test (P). The results of the experiment are presented in Tables 11, 12, and 13. The minimum error rate is obtained from (24); The average error rate is calculated by (25); The maximum error rate is obtained from (26); The standard deviation is obtained from (27).

$$PD\_best = \frac{L_B - L_{\min}}{L_{\min}} \times 100\% \tag{24}$$

$$PD\_avg = \frac{L_{AVG} - L_{\min}}{L_{\min}} \times 100\% \tag{25}$$

$$PD\_worst = \frac{L_W - L_{\min}}{L_{\min}} \times 100\% \tag{26}$$

Where L_B is the optimal solution found by the algorithm; L_A is the average of N optimal solutions; L_W is the worst of N optimal solutions; L_{min} is the optimized solution for the TSP instance.

$$dev = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(l_i - l_{avg}\right)^2} \tag{27}$$

Where N is the number of ants; l_i is the best solution for the ith experiment; l_{avg} is the average of the current N solutions.

As shown in Table 11, 21 groups of optimal solutions were found by the MSACS algorithm in 30 groups of TSP instances. For the other TSP instances, the percentage deviation from the optimal solution is also relatively small, with 7 TSP instances having an error of less than 1%; In addition, the percentage deviation of the optimal solution for the remaining three TSP instances is less than 2%. In addition, in the average error comparison, in most instances, the value of P_{avg} is less than 1% (kroB200, lin318, pr439, att532, p654, and d1291 whose P_{avg} is 1.47%, 1.89%, 2.53%, 2.64%, and 3.63%); Most values of P_{worst} have percentage deviations of less than 4% (p654 and d1291 whose P_{worst} is 6.08% and 7.44%);

As can be seen from Tables 12 and 13, The MSACS algorithm outperforms the traditional MMAS algorithm and the ACS algorithm in all TSP instances for all properties. It can be seen that in urban TSP instances with less than 200 nodes, most MACS algorithms can find optimal solutions compared to ACS algorithms and MMAS algorithms; In addition, due to the elite ant learning strategy, the MSACS algorithm has the fastest algorithm convergence speed compared to the other two algorithms; In the experimental comparison of the number of city nodes in the TSP instances between 200 and 500, MSACS still has the high accuracy. This is due to the influence of the pheromone fusion strategy and the elite ant learning strategy, which makes the algorithm both convergent and maintain good diversity; In the TSP example, the MSACS algorithm is far more accurate than the other two algorithms in the experimental comparison for city nodes greater than 500(especially in the comparison of TSP instances d1291, the error of the optimal solution of MSACS is 1.34%, while the error of MMAS algorithm and ACS algorithm are 6.98% and 8.47% respectively). What's more, in the comparison of average deviation, We can see that the average error of MSACS is lower than the two other algorithms, which indicates that the solution stability of MSACS algorithm is also better.

Figure 5 shows the comparison of the standard deviation distribution of MSACS algorithm, ACS algorithm and MMAS algorithm; It can be seen from the figure that, compared with the other two algorithms, the standard deviation of MSACS algorithm is the smallest in most TSP instances; In addition, standard deviation can reflect the stability of the algorithm to some extent, and the smaller the standard deviation is, the better the stability of the algorithm is; From the comparison in the figures, we can see that the stability of MSACS algorithm is the best among the three algorithms.

Figure 6 shows the convergence variation of MSACS, ACS, and MMAS in the set of TSP instances; As can be seen from the figure, MSACS algorithm has higher convergence speed and accuracy than MMAS algorithm and ACS algorithm; Since MSACS uses pheromone recombination strategy, it improves the disadvantage of ACS algorithm and MMAS algorithm that it is easy to fall into local optimum, and has better accuracy in various TSP instances.

Figure 7 shows the optimal paths found by the MSACS algorithm for eight different TSP instances.

Table 11 Performance comparison of MSACS in different TSP instances

Instances	Size (cities)	Optimal solution	Solutions by MSACS							
			Best	Average	Worst	PD_best	PD_avg	PD_worst	Std	iters
Eil51	51	426	426	426.05	427	0.00%	0.01%	0.23%	0.22	56
Eil76	76	538	538	538.15	540	0.00%	0.03%	0.37%	0.49	124
Pr76	76	108159	108159	108243.1	109102	0.00%	0.08%	0.87%	231.98	651
Rat99	99	1211	1211	1212	1221	0.00%	0.08%	0.83%	3.08	77
KroA100	100	21282	21282	21282.55	21292	0.00%	0.00%	0.05%	1.88	142
KroB100	100	22141	22141	22192.45	22337	0.00%	0.23%	0.89%	63.44	137
KroC100	100	20749	20749	20760.7	20880	0.00%	0.06%	0.63%	31.78	354
KroD100	100	21294	21294	21313.3	21389	0.00%	0.09%	0.45%	36.05	874
KroE100	100	22068	22068	22110.65	22137	0.00%	0.19%	0.31%	32.60	645
Eil101	101	629	629	629.7	632	0.00%	0.11%	0.48%	0.80	845
Lin105	105	14379	14379	14384.2	14483	0.00%	0.04%	0.72%	23.26	632
Pr124	124	59030	59030	59067.3	59159	0.00%	0.06%	0.22%	39.39	1447
Ch130	130	6110	6110	6143.75	6177	0.00%	0.55%	1.10%	22.20	1564
Ch150	150	6528	6528	6530	6539	0.00%	0.03%	0.17%	3.46	1014
KroA150	150	26524	26524	26537.5	26628	0.00%	0.05%	0.39%	28.19	1432
KroB150	150	26130	26130	26153.6	26228	0.00%	0.09%	0.38%	32.07	1346
Pr152	152	73682	73682	73993.75	74397	0.00%	0.42%	0.97%	209.06	1500
Rat195	195	2323	2332	2336.9	2346	0.39%	0.60%	0.99%	3.63	455
d198	198	15780	15844	15917.8	15992	0.41%	0.87%	1.34%	42.56	1306
KroA200	200	29368	29368	29435.4	29478	0.00%	0.23%	0.37%	39.53	1746
KroB200	200	29437	29437	29870.2	30025	0.00%	1.47%	2.00%	139.01	1769
Tsp225	225	3916	3916	3948	3970	0.00%	0.82%	1.38%	14.59	1364
Pr226	226	80369	80422	80530.35	80595	0.07%	0.20%	0.28%	53.81	1032
A280	280	2579	2579	2595.5	2610	0.00%	0.64%	1.20%	8.77	1822
Lin318	318	42029	42286	42824.15	43298	0.61%	1.89%	3.02%	235.24	1714
Fl1417	417	11861	11901	11969.4	12000	0.34%	0.91%	1.17%	24.43	1224
Pr439	439	107217	107916	108403.6	108970	0.65%	1.11%	1.64%	314.26	1568
Att532	532	86729	88403	88921.4	89185	1.93%	2.53%	2.83%	225.32	1771
P654	654	34643	34939	35558.45	36749	0.85%	2.64%	6.08%	589.26	1689
D1291	1291	50801	51482	52645.35	54582	1.34%	3.63%	7.44%	771.96	1889

Table 12 Performance comparison of MSACS and ACS in different TSP instances

Instances	Size (cities)	Optimal solution	Solutions by ACS				Solutions by MSACS				
			PD_best	PD_avg	PD_worst	iter	PD_best	PD_avg	PD_worst	iter	P
Eil51	51	426	0.00%	0.41%	1.88%	195	0.00%	0.01%	0.23%	56	0.000
Eil76	76	538	0.00%	0.80%	2.23%	450	0.00%	0.03%	0.37%	124	0.000
Pr76	76	108159	1.21%	2.75%	6.66%	662	0.00%	0.08%	0.87%	651	0.000
Rat99	99	1211	0.00%	0.83%	1.90%	1251	0.00%	0.08%	0.83%	77	0.000
KroA100	100	21282	0.00%	0.80%	3.03%	1114	0.00%	0.00%	0.05%	142	0.000
KroB100	100	22141	0.43%	0.93%	1.43%	1755	0.00%	0.23%	0.89%	137	0.000
KroC100	100	20749	0.00%	0.59%	1.27%	1463	0.00%	0.06%	0.63%	354	0.000
KroD100	100	21294	0.07%	1.73%	4.46%	852	0.00%	0.09%	0.45%	874	0.000
KroE100	100	22068	0.00%	0.86%	3.85%	547	0.00%	0.19%	0.31%	645	0.000
Eil101	101	629	0.32%	2.27%	4.45%	774	0.00%	0.11%	0.48%	845	0.000
Lin105	105	14379	0.00%	0.67%	2.25%	1254	0.00%	0.04%	0.72%	632	0.000
Pr124	124	59030	0.10%	1.10%	2.10%	1488	0.00%	0.06%	0.22%	1447	0.000
Ch130	130	6110	0.88%	2.23%	3.76%	894	0.00%	0.55%	1.10%	1564	0.000
Ch150	150	6528	0.40%	0.97%	2.62%	634	0.00%	0.03%	0.17%	1014	0.000
KroA150	150	26524	1.13%	2.34%	4.51%	1543	0.00%	0.05%	0.39%	1432	0.000
KroB150	150	26130	0.04%	1.87%	4.73%	744	0.00%	0.09%	0.38%	1346	0.000
Pr152	152	73682	0.61%	1.18%	2.36%	1254	0.00%	0.42%	0.97%	1500	0.000
Rat195	195	2323	0.82%	1.72%	3.62%	899	0.39%	0.60%	0.99%	455	0.000
d198	198	15780	4.94%	7.12%	9.99%	1854	0.41%	0.87%	1.34%	1306	0.000
KroA200	200	29368	0.42%	1.18%	3.19%	365	0.00%	0.23%	0.37%	1746	0.000
KroB200	200	29437	0.87%	2.16%	4.74%	1125	0.00%	1.47%	2.00%	1769	0.003
Tsp225	225	3916	0.79%	2.44%	4.85%	1875	0.00%	0.82%	1.38%	1364	0.000
Pr226	226	80369	0.29%	0.65%	1.81%	1335	0.07%	0.20%	0.28%	1032	0.000
A280	280	2579	0.70%	2.80%	7.33%	1442	0.00%	0.64%	1.20%	1822	0.000
Lin318	318	42029	1.66%	3.60%	6.34%	1541	0.61%	1.89%	3.02%	1714	0.000
Fl1417	417	11861	4.55%	6.22%	8.37%	1744	0.34%	0.91%	1.17%	1224	0.000
Pr439	439	107217	1.75%	4.32%	7.95%	1846	0.65%	1.11%	1.64%	1568	0.000
Att532	532	86729	3.12%	5.31%	8.23%	1944	1.93%	2.53%	2.83%	1771	0.000
P654	654	34643	6.00%	8.57%	14.36%	1992	0.85%	2.64%	6.08%	1689	0.000
D1291	1291	50801	8.47%	11.07%	15.62%	986	1.34%	3.63%	7.44%	1889	0.000

Table 13 Performance comparison of MSACS and MMAS in different TSP instances

Instances	Size (cities)	Optimal solution	Solutions by MMAS				Solutions by MSACS				
			PD_best	PD_avg	PD_worst	iter	PD_best	PD_avg	PD_worst	iter	P
Eil51	51	426	0.00%	0.56%	1.41%	200	0.00%	0.01%	0.23%	56	0.000
Eil76	76	538	0.00%	0.82%	1.49%	332	0.00%	0.03%	0.37%	124	0.000
Pr76	76	108159	1.42%	2.53%	3.94%	562	0.00%	0.08%	0.87%	651	0.000
Rat99	99	1211	0.17%	1.23%	2.81%	1036	0.00%	0.08%	0.83%	77	0.000
KroA100	100	21282	0.97%	1.63%	3.41%	1275	0.00%	0.00%	0.05%	142	0.000
KroB100	100	22141	0.19%	0.99%	1.67%	1865	0.00%	0.23%	0.89%	137	0.000
KroC100	100	20749	0.03%	0.89%	1.90%	1443	0.00%	0.06%	0.63%	354	0.000
KroD100	100	21294	0.18%	1.67%	3.72%	1365	0.00%	0.09%	0.45%	874	0.000
KroE100	100	22068	0.62%	1.31%	2.37%	547	0.00%	0.19%	0.31%	645	0.000
Eil101	101	629	0.48%	2.31%	3.50%	1204	0.00%	0.11%	0.48%	845	0.000
Lin105	105	14379	0.00%	0.36%	0.88%	1323	0.00%	0.04%	0.72%	632	0.000
Pr124	124	59030	0.63%	1.34%	2.26%	1566	0.00%	0.06%	0.22%	1447	0.000
Ch130	130	6110	1.75%	2.24%	2.98%	1336	0.00%	0.55%	1.10%	1564	0.000
Ch150	150	6528	0.37%	1.25%	2.37%	1145	0.00%	0.03%	0.17%	1014	0.000
KroA150	150	26524	1.49%	2.79%	4.64%	1641	0.00%	0.05%	0.39%	1432	0.000
KroB150	150	26130	2.12%	2.68%	3.45%	1559	0.00%	0.09%	0.38%	1346	0.000
Pr152	152	73682	0.34%	1.14%	2.81%	1366	0.00%	0.42%	0.97%	1500	0.000
Rat195	195	2323	0.47%	0.76%	1.64%	423	0.39%	0.60%	0.99%	455	0.160
d198	198	15780	3.01%	4.90%	6.35%	1544	0.41%	0.87%	1.34%	1306	0.000
KroA200	200	29368	1.32%	2.12%	2.92%	1778	0.00%	0.23%	0.37%	1746	0.000
KroB200	200	29437	2.36%	3.53%	4.55%	1841	0.00%	1.47%	2.00%	1769	0.000
Tsp225	225	3916	2.09%	4.09%	6.36%	1988	0.00%	0.82%	1.38%	1364	0.000
Pr226	226	80369	1.16%	1.90%	2.46%	1256	0.07%	0.20%	0.28%	1032	0.000
A280	280	2579	1.16%	3.48%	5.66%	1887	0.00%	0.64%	1.20%	1822	0.000
Lin318	318	42029	4.03%	5.39%	6.16%	1745	0.61%	1.89%	3.02%	1714	0.000
Fl1417	417	11861	4.87%	6.22%	7.55%	1651	0.34%	0.91%	1.17%	1224	0.000
Pr439	439	107217	3.08%	4.35%	5.46%	1855	0.65%	1.11%	1.64%	1568	0.000
Att532	532	86729	6.03%	7.79%	8.92%	1961	1.93%	2.53%	2.83%	1771	0.000
P654	654	34643	7.83%	10.00%	12.26%	1774	0.85%	2.64%	6.08%	1689	0.000
D1291	1291	50801	6.98%	9.20%	11.23%	1204	1.36%	3.63%	7.44%	1889	0.000

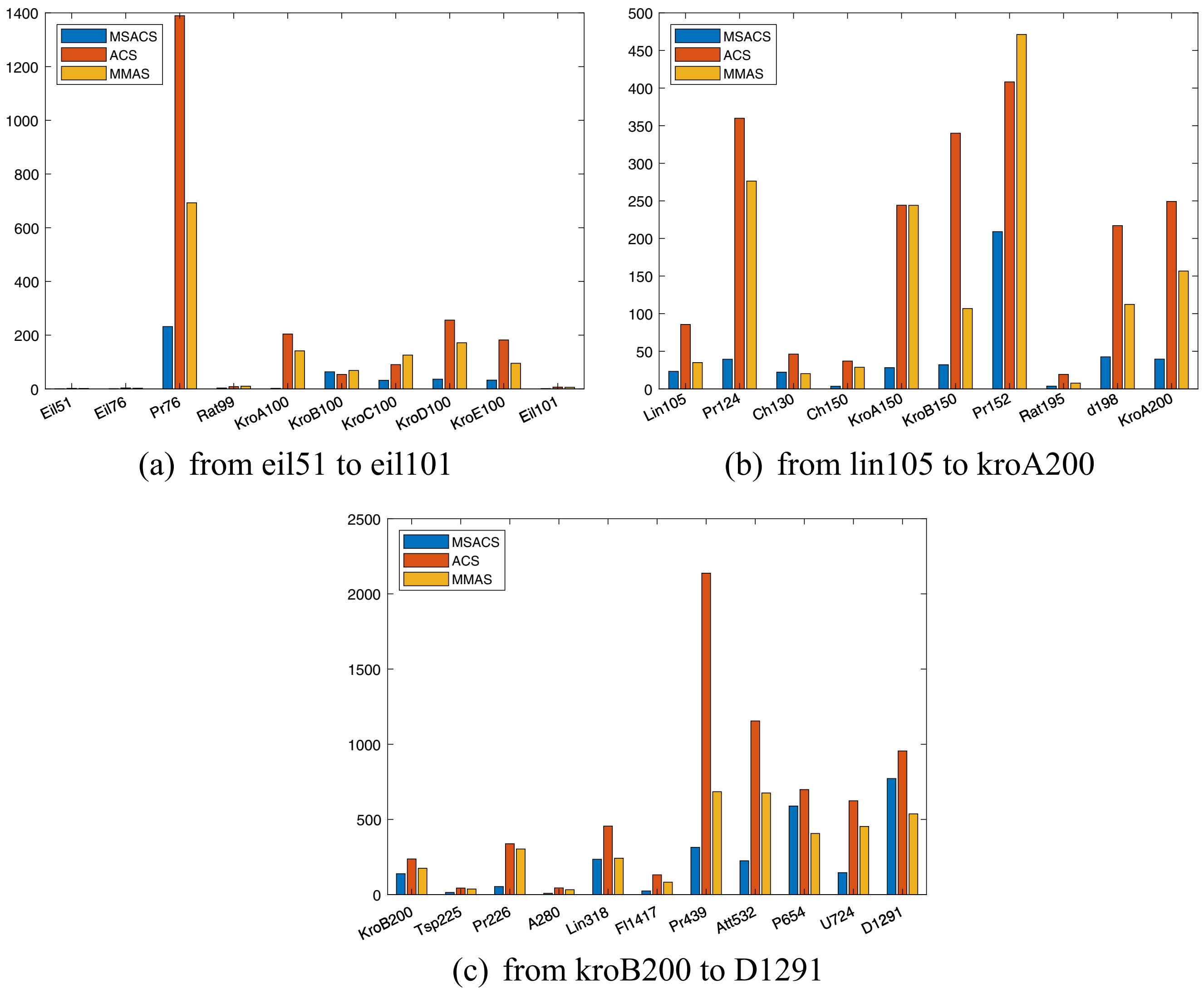


Fig. 5 The stability of the different algorithms

The P values in Tables 12 and 13 are the results of rank sum test, which reflects the differences of samples; As can be seen from the table, the results of MSACS are significantly different from those of MMAS and ACS in most cases.

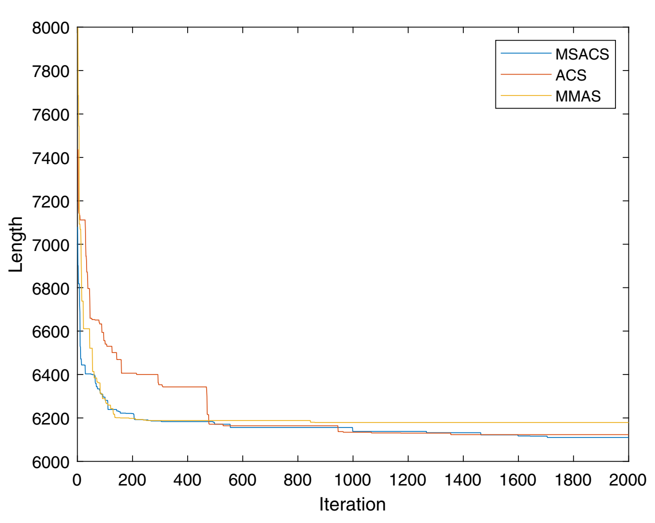
Overall, the MSACS algorithm enhances the accuracy of the algorithm, speeds up the convergence, and reduces the possibility of algorithm stagnation. The solving ability of MSACS is better than that of MMAS and ACS.

4.3.2 Comparative analysis with other ant colony algorithms and other intelligent algorithms

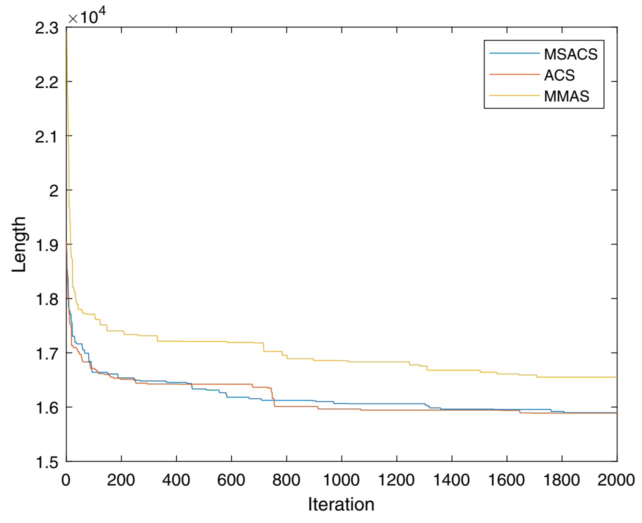
To further verify the effectiveness of the improved MSACS algorithm, we compare it with other improved ACS algorithms and other intelligent algorithms in this section, and the results are shown in Tables 14, 15, 16, 17, 18, and 19.

In Table 14 MSACS algorithm and PACO-3OPT [31] algorithm were compared, and from the table we can see: In cities with less than 200 nodes, the MSACS algorithm finds the optimal solution while the PACO-3OPT algorithm finds the optimal solution only partially. In the comparison of optimal, worst and average solutions, the MSACS algorithm performs better than the PACO-3OPT algorithm in most of the TSP instances. In spite of that, MSACS algorithm is compared with HAACO [32] algorithm in all aspects in detail in Table 15. Similarly, we can see that the MSACS algorithm outperforms the HAACO algorithm in the comparison of all city instances.

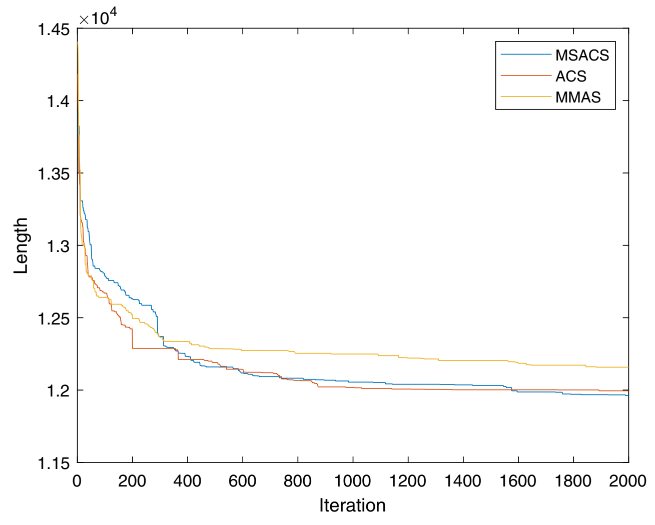
Further, other intelligent algorithms are used to compare with MSACS. In Tables 16 and 17, the DSFLA and DJAYA algorithms are compared with the MSACS algorithm, respectively. In addition to being better in the comparison



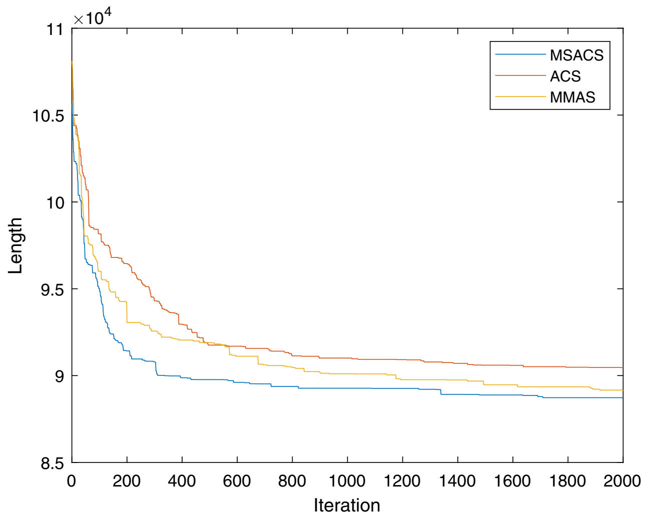
(a) Convergence curves in ch130



(b) Convergence curves in d198



(c) Convergence curves in fl417



(d) Convergence curves in att532

Fig. 6 Convergence comparison of MSACS, ACS and MMAS

of optimal and average solutions, the MSACS algorithm is smoother than the other two algorithms in the comparison of variance.

Finally, to better evaluate the capability of the MSACS algorithm, more comparisons will be shown in Tables 18 and 19. DFFO [4], DEACO [19], CCMACO [33], HDACO [34], HGA [35], HDABC [36] GA-MARL+NICH-LS [37], SSABC [38], AG-BSO [39], PSO-ACO-3opt [40], MARL+NICH-LS [37], IBA [41], DWCA [42], HMMA [43], DBACS [44], DCS [45], ABC-3OPT [46], DSMO [47], JCACO [27], and MGACACO [48]will be compared with the MSACS algorithm in the optimal solution. In the comparison, the solution accuracy of MSACS algorithm is better than the other algorithms. This shows that the MSACS algorithm expands the search space by selecting the best strategy through the Stackelberg game, which results in better accuracy.

5 Conclusion

In this paper, we propose An ant colony algorithm with Stackelberg game and multi-strategy fusion. First of all, MMAS algorithm and ACS algorithm are used to construct heterogeneous multi-population ant colony algorithm, so that the advantages of different algorithms are used to balance the convergence and diversity of the algorithm; Secondly, Stackelberg game with dynamic changes of leaders among multiple populations is established. In this game, the overall consideration algorithm and the attributes

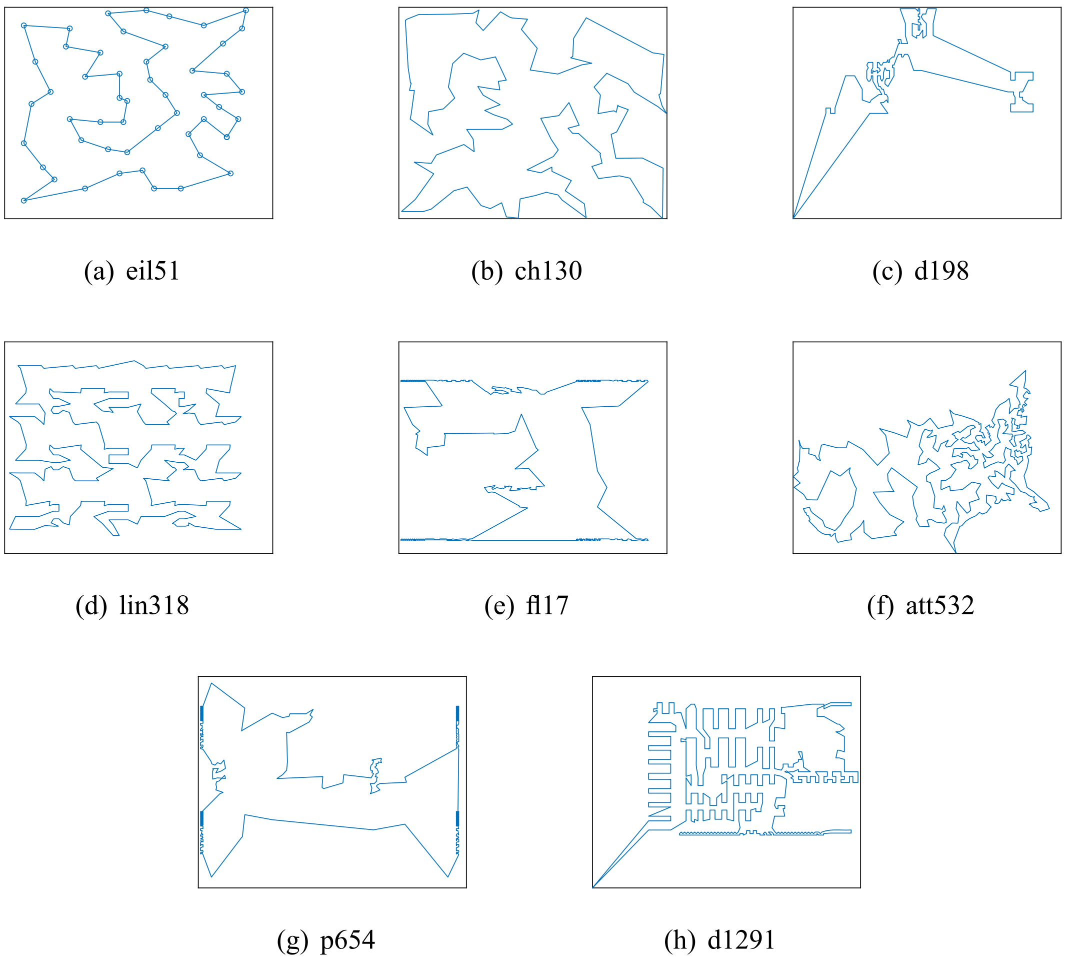


Fig. 7 Best tours for each TSP instance found by MSACS

Table 14 Comparison of MSACS and PACO-3OPT in TSP instances

instance		PACO-3OPT(2018)				MSACS			
name	optima	Avg.	Best	Worst	PD(%)	Avg.	Best	Worst	PD(%)
Eil51	426	426	426	427	0.00	426	426	427	0.00
Berlin52	7542	7542	7542	7542	0.00	7542	7542	7542	0.00
St70	675	678	676	679	0.15	676	675	679	0.00
Eil76	538	540	538	542	0.00	538	538	540	0.00
Rat99	1211	1217	1213	1225	0.17	1212	1211	1221	0.00
KroA100	21282	21327	21282	21382	0.00	21282	21282	21292	0.00
Eil101	629	631	629	639	0.00	630	629	632	0.00
Lin105	14379	14393	14379	14422	0.00	14384	14379	14483	0.00
Ch150	6528	6601	6570	6627	0.64	6530	6528	6539	0.00
KroA200	29368	29644	29533	29721	0.56	29435	29368	29478	0.00
Fl417	11861	11987	11972	NA	0.94	11969	11901	12000	0.34
Pr439	107217	108702	108482	NA	1.18	108403	107916	108970	0.65
P654	34643	35075	35045	NA	1.16	34939	35558	36749	0.85
U724	41919	43123	42764	NA	2.02	43313	42978	43474	2.55

Table 15 Comparison of MSACS and HAACO in TSP instances

instance		HAACO(2020)				MSACS			
name	optima	Avg.	Best	Worst	PD(%)	Avg.	Best	Worst	PD(%)
Eil51	426	428	426	431	0.00	426	426	427	0.00
Eil76	538	542	538	545	0.00	538	538	540	0.00
Rat99	1211	1214	1211	1218	0.00	1212	1211	1221	0.00
KroA100	21282	21364	21282	21445	0.00	21282	21282	21292	0.00
Eil101	629	632.5	630	635	0.00	630	629	632	0.00
Lin105	14379	14412	14379	14483	0.00	14384	14379	14483	0.00
Ch150	6528	6579	6554	6595	0.40	6530	6528	6539	0.00
KroA200	29368	29633	29483	29755	0.39	29435	29368	29478	0.00
Fl417	11861	NA	11960	NA	0.83	11969	11901	12000	0.34
Pr439	107217	NA	108730	NA	1.41	108403	107916	108970	0.65

Table 16 Comparison of MSACS and DSFLA in TSP instances

instance		DSFLA(2021)				MSACS			
name	optima	Avg.	Best	SD.	PD(%)	Avg.	Best	SD.	PD(%)
Eil51	426	426	426	0.5	0.00	426	426	0.22	0.00
Eil76	538	539	538	1.6	0.00	538	538	0.49	0.00
Rat99	1211	1216	1211	5.04	0.00	1212	1211	3.08	0.00
KroA100	21282	21312	21282	50.01	0.00	21282	21282	1.88	0.00
KroC100	20749	20772	20749	56.37	0.00	20760	20749	31.78	0.00
Eil101	629	632	629	3.65	0.00	630	629	0.8	0.00
Lin105	14379	14423	14379	55.82	0.00	14384	14379	23.26	0.00
Pr124	59030	59503	59030	185.88	0.00	59067	59030	39.39	0.00
Ch130	6110	6140	6110	47.66	0.49	6143	6110	22.2	0.00
Ch150	6528	6562	6533	12.95	0.08	6530	6528	3.46	0.00
Pr152	73682	73970	73682	271.23	0.00	73994	73682	209.06	0.00
KroA200	29368	29671	29499	135.84	0.45	29435	29368	39.53	0.00

Table 17 Comparison of MSACS and DJAYA in TSP instances

instance		DJAYA(2021)			MSACS		
name	optima	Avg.	SD.	PD(%)	Avg.	SD.	PD(%)
Eil51	426	440	4.95	2.64	426	0.22	0.00
Eil76	538	573	6.33	5.10	538	0.49	0.00
KroA100	21282	21735	331.33	2.13	21282	1.88	0.00
Eil101	629	677	4.87	5.46	630	0.8	0.00
Ch150	6528	6638	52.79	1.63	6530	3.46	0.00
Tsp225	3916	4095	42.54	6.12	3948	14.59	0.00

Table 18 Comparison of MSACS and other algorithms in TSP instances

TSP	Eil51	Berlin52	St70	Eil76	Rat99	kroA100	Lin105
MSACS	426	7542	675	538	1211	21282	14379
PD_best(%)	0.00	0.00	0.00	0.00	0.00	0.00	0.00
DFFA(2020)	426	7542	675	538	1211	21282	14379
PD_best(%)	0.00	0.00	0.00	0..00	0.00	0.00	0.00
DEACO(2020)	426	7542	675	538	1211	21282	14379
PD_best(%)	0.00	0.00	0.00	0..00	0.00	0.00	0.00
CCMACO(2019)	426	7542	675	538	NA	21282	NA
PD_best(%)	0.00	0.00	0.00	0.00	NA	0.00	NA
HDACO(2018)	426	7542	675	538	NA	21282	NA
PD_best(%)	0.00	0.00	0.00	0.00	NA	0.00	NA
HGA(2014)	428	7544	677	544	1219	21285	14382
PD_best(%)	0.47	0.03	0.30	1.12	0.66	0.01	0.02
HDABC(2017)	426	7542	675	538	NA	21282	NA
PD_best(%)	0.00	0.00	0.00	0.00	NA	0.00	NA
GMNL(2017)	426	7542	675	538	1211	21282	14379
PD_best(%)	0.00	0.00	0.00	0.00	0.00	0.00	0.00
SSABC(2018)	426	7542	675	538	1211	21282	14379
PD_best(%)	0.00	0.00	0.00	0.00	0.00	0.00	0.00
AG-BSO(2020)	428	7542	678	541	1211	21070	NA
PD_best(%)	0.61	0.00	0.44	0.50	0.00	0.00	NA
PSO-ACO(2015)	426	7542	676	538	1221	21301	14379
PD_best(%)	0.00	0.00	0.15	0.00	1.07	0.09	0.00
JCACO(2019)	426	NA	NA	538	1211	21282	NA
PD_best(%)	0.00	NA	NA	0.00	0.00	0.00	NA
MNL(2017)	426	7542	675	538	1211	21282	14379
PD_best(%)	0.00	0.00	0.00	0.00	0.00	0.00	0.00
IBA(2016)	426	7542	675	539	NA	21282	NA
PD_best(%)	0.00	0.00	0.00	0.19	NA	0.00	NA
DWCA(2018)	426	7542	675	543	NA	21282	NA
PD_best(%)	0.00	0.00	0.00	0.93	NA	0.00	NA
ABC3OPT(2019)	427	7542	675	538	1211	21282	14379
PD_best(%)	0.23	0.00	0.00	0.00	0.00	0.00	0.00
MGACACO	427	7544	NA	539	1216	NA	14381
PD_best(%)	0.23	0.02	NA	0.18	0.41	NA	0.01

of each population are comprehensively used to select leader. Then, the leader improves the search efficiency by training a certain number of iterations in advance to explore more paths; Finally, the multi-strategy fusion strategy is used to improve the information exchange among the populations, among which three strategies are proposed: Strategy 1 is the pheromone fusion strategy, under which the pheromone matrices between populations that need to communicate are combined into a new pheromone matrix through a linear combination of certain parameters. This strategy can improve the ability of ants to explore different paths and increase the diversity between populations; Strategy 2 is the elite ant learning strategies. Under this strategy, the population learns from the experience of the elite ants of the dominant population through the updating of pheromone to improve the convergence rate of the population; Strategy 3 is the pheromone recombination strategy, which is used to help the population jump out of local optimality. When the population is in the local optimal state, the pheromone of common paths between the populations are retained, and the pheromone of non-public paths are smoothed, and then the local search is carried out.

The experimental results show that Compared with the traditional ant colony algorithm, the improved ant colony algorithm and other intelligent algorithms, the MSACS

Table 19 Comparison of MSACS and other algorithms in TSP instances

TSP	KroB100	Pr152	Pr226	Fl417	Pr439	P654	U724
MSACS	21241	73682	80369	11901	107916	34939	42978
PD_best(%)	0.00	0.00	0.07	0.34	0.65	0.85	2.55
DSMO(2020)	22308	74244	83588	12219	112105	NA	NA
PD_best(%)	0.75	0.18	4.00	3.02	4.56	NA	NA
PSO-ACO(2015)	NA	NA	NA	11941	108530	35052	43172
PD_best(%)	NA	NA	NA	0.73	1.20	1.20	3.00
HMMA(2015)	22388	73892	83972	12543	114095	37044	46662
PD_best(%)	1.11	0.28	4.48	5.70	6.40	6.90	11.30
DBACS(2020)	NA	73682	80369	11949	108036	34939	42983
PD_best(%)	NA	0.00	0.00	0.74	0.76	0.85	2.56
AG-BSO(2020)	22152	NA	80961	11936	113074	NA	NA
PD_best(%)	0.05	NA	0.74	0.63	5.46	NA	NA
DCS(2014)	22139	73683	80440	11949	108136	NA	42860
PD_best(%)	0.00	0.00	0.09	0.75	0.86	NA	2.26

algorithm has better convergence speed and precision, and has better quality in the solution of large-scale examples.

In addition, the ant colony algorithm mentioned in this paper can also be applied to specific practical applications, such as robot path planning: in a general solution, the map rasterization makes path planning for robots a nodal optimization problem, which is similar to the TSP problem. In our algorithm, Stackelberg game model can improve the collaboration of ants in path planning; The multi-strategy fusion mechanism can help the algorithm balance diversity and convergence, and increase its probability of jumping out of the local optimum when stalled.

In the future work, we will further investigate the essence of solution seeking of ant colony algorithm, improve its convergence speed and solving precision for large-scale node problems, and apply it to some practical engineering problems.

References

1. Durbin R, Willshaw D (1987) An analogue approach to the travelling salesman problem using an elastic net method. Nature 326(6114):689–691
2. Dong X, Zhang H, Xu M, Shen F (2021) Hybrid genetic algorithm with variable neighborhood search for multi-scale multiple bottleneck traveling salesmen problem. Futur Gener Comput Syst 114:229–242
3. Zelinka I, Das S (2020) Gamesourcing: an unconventional tool to assist the solution of the traveling salesman problem. Nat Comput (Suppl. 2):1–11
4. Benyamin A, Farhad SG, Saeid B (2020) Discrete farmland fertility optimization algorithm with metropolis acceptance criterion for traveling salesman problems. International Journal of Intelligent Systems
5. Zhong Y, Wang L, Lin M, Zhang H (2019) Discrete pigeon-inspired optimization algorithm with metropolis acceptance criterion for large-scale traveling salesman problem. Swarm Evol Comput:48
6. Xiang X, Qiu J, Xiao J, Zhang X (2020) Demand coverage diversity based ant colony optimization for dynamic vehicle routing problems. Eng Appl Artif Intell 91:103582
7. Shokouhifar M (2021) Fh-aco: Fuzzy heuristic-based ant colony optimization for joint virtual network function placement and routing. Appl Soft Comput 107(2):107401
8. Jia YH, Chen WN, Yuan H, Gu T, Zhang H, Gao Y, Zhang J (2018) An intelligent cloud workflow scheduling system with time estimation and adaptive ant colony optimization. IEEE Trans Syst Man Cybern Syst:1–16
9. Liang D, Zhan ZH, Zhang Y, Zhang J (2019) An efficient ant colony system approach for new energy vehicle dispatch problem. IEEE Trans Intell Transp Syst PP(99):1–14
10. Dorigo M, Maniezzo V, Colorni A (1996) Ant system: Optimization by a colony of cooperating agents. ieee trans syst man cybernetics - part b. IEEE Trans Cybern 26(1):29–41
11. Dorigo M, Gambardella LM (1997) Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans Ec 1(1):53–66
12. Dorigo M, Birattari M, Stützle T (2006) Ant colony optimization. IEEE Comput Intell Mag 1(4):28–39
13. Stützle T, Hoos HH (2000) Max-min ant system. Futur Gener Comput Syst 16(8):889–914
14. Wu Y, Ma W, Miao Q, Wang S (2019) Multimodal continuous ant colony optimization for multisensor remote sensing image registration with local search. Swarm Evol Comput 47:89–95
15. Gao W (2016) Premium-penalty ant colony optimization and its application in slope stability analysis. Appl Soft Comput 43:480–488
16. Ye K, Zhang C, Ning J, Liu X (2017) Ant-colony algorithm with a strengthened negative-feedback mechanism for constraint-satisfaction problems. Inf Sci 406–407:29–41
17. Li S, Cai S, Li L, Sun R, Yuan G (2020) Caas: a novel collective action-based ant system algorithm for solving tsp problem. Soft Comput 24(12):9257–9278
18. Zhang Q, Zhang C (2018) An improved ant colony optimization algorithm with strengthened pheromone updating mechanism for constraint satisfaction problem. Neural Comput Appl 30:3209–3220

19. Ebadinezhad S (2020) Deaco: Adopting dynamic evaporation strategy to enhance aco algorithm for the traveling salesman problem. Eng Appl Artif Intell 92:103649
20. Skinderowicz R (2020) Implementing a gpu-based parallel max-min ant system. Fut Gener Comput Syst 106:277–295
21. Zhao D, Liu L, Yu F, Heidari AA, Wang M, Oliva D, Muhammad K, Chen H (2021a) Ant colony optimization with horizontal and vertical crossover search: Fundamental visions for multi-threshold image segmentation. Expert Syst Appl 167:114122
22. Zhao D, Liu L, Yu F, Heidari AA, Wang M, Liang G, Muhammad K, Chen H (2021b) Chaotic random spare ant colony optimization for multi-threshold image segmentation of 2d kapur entropy. Knowl-Based Syst 216:106510
23. Han Z, Wang Y, Tian D (2021) Ant colony optimization for assembly sequence planning based on parameters optimization. Frontiers of Mechanical Engineering
24. Gao W (2020) Modified ant colony optimization with improved tour construction and pheromone updating strategies for traveling salesman problem. Soft Comput:1–27
25. Wang X, Yang K, Yang L (2018) Application research of inner-plant economical operation by multi-colony ant optimization. Water Resources Management
26. Zhou J, Wang C, Li Y, Wang P, Li C, Lu P, Mo L (2017) A multi-objective multi-population ant colony optimization for economic emission dispatch considering power system security. Appl Math Model 45:684–704
27. Zhang D, You X, Liu S, Yang K (2019) Multi-colony ant colony optimization based on generalized jaccard similarity recommendation strategy. IEEE Access 7:157303–157317
28. Pan H, You X, Liu S, Zhang D (2020) Pearson correlation coefficient-based pheromone refactoring mechanism for multi-colony ant colony optimization. Appl Intell:1–23
29. Shannon CE (1948) A mathematical theory of communication. The Bell Syst Techn J 27(3):379–423
30. Emile J (1953) Stackelberg (Heinrich von) - The Theory of the Market Economy, translated from the German and with an introduction by Alan T. PEACOCK. Revue Écon 4(6):944–945
31. Gùlcù S, Mahi M, Baykan ÒK, Kodaz H (2018) A parallel cooperative hybrid method based on ant colony optimization and 3-opt algorithm for solving traveling salesman problem. Soft Comput 22(5):1669–1685
32. Tuani AF, Keedwell E, Collett M (2020) Heterogenous adaptive ant colony optimization with 3-opt local search for the travelling salesman problem. Appl Soft Comput 97:106720
33. Zhang H, You X (2019) Multi-population ant colony optimization algorithm based on congestion factor and co-evolution mechanism. IEEE Access 7:1–1
34. Liao E, Liu C (2018) A hierarchical algorithm based on density peaks clustering and ant colony optimization for traveling salesman problem. IEEE Access PP(99):1–1
35. Yong W (2014) The hybrid genetic algorithm with two local optimization strategies for traveling salesman problem. Comput Ind Eng 70(apr.):124–133
36. Zhong Y, Lin J, Wang L, Hui Z (2017) Hybrid discrete artificial bee colony algorithm with threshold acceptance criterion for traveling salesman problem. Inf Sci:421
37. Alipour MM, Razavi SN, Derakhshi MRF, Balafar MA (2018) A hybrid algorithm using a genetic algorithm and multiagent reinforcement learning heuristic to solve the traveling salesman problem. Neural Comput Appl 30:2935–2951
38. Khan I, Maiti MK (2018) A swap sequence based artificial bee colony algorithm for traveling salesman problem. Swarm Evol Comput:S2210650216304588
39. Wu C, Fu X (2020) An agglomerative greedy brain storm optimization algorithm for solving the tsp. IEEE Access 8:201606–201621
40. Mahi M, Baykan K, Kodaz H (2015) A new hybrid method based on particle swarm optimization, ant colony optimization and 3-opt algorithms for traveling salesman problem. Appl Soft Comput 30:484–490
41. Osaba E, Yang XS, Diaz F, Lopez-Garcia P, Carballedo R (2016) An improved discrete bat algorithm for symmetric and asymmetric traveling salesman problems. Eng Appl Artif Intell 48:59–71
42. Eneko O, Del SJ, Ali S, Nekane BM, DCavid B (2018) A discrete water cycle algorithm for solving the symmetric and asymmetric traveling salesman problem. Appl Soft Comput 71:S1568494618303818–
43. Wang Y (2015) Hybrid max–min ant system with four vertices and three lines inequality for traveling salesman problem. Soft Computing
44. Yu J, You X, Liu S (2020) Dynamic density clustering ant colony algorithm with filtering recommendation backtracking mechanism. IEEE Access PP(99):1–1
45. Zhou Y, Ouyang X, Jian X (2014) A discrete cuckoo search algorithm for travelling salesman problem. Int J Collab Intell 1(1):68
46. Khan I, Maiti MK (2019) A swap sequence based artificial bee colony algorithm for traveling salesman problem. Swarm Evol Comput 44:428–438
47. Akhand M, Ayon SI, Shahriyar S, Siddique N, Adeli H (2020) Discrete spider monkey optimization for travelling salesman problem. Appl Soft Comput 86:105887
48. Deng W, Zhao H, Zou L, Li G, Yang X, Wu D (2017) A novel collaborative optimization algorithm in solving complex optimization problems. Soft Computing

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Da Chen was born in Yancheng, Jiangsu, China, in 1997. He is currently pursuing the M.S. degree with the Shanghai University of Engineering Science. His research interests include intelligent algorithm, path planning of mobile robot, and embedded systems.

XiaoMing You was born in Huaihua, Jiangsu, China, in 1963. She received the Ph.D. degree in computer science from the East China University of Science and Technology, in 2007. She is currently a Professor and a M.S. degree Supervisor with the Shanghai University of Engineering Science. Her research interests include swarm intelligent systems, distributed parallel processing, and evolutionary computing.

Sheng Liu was born in Daye, Hubei, China, in 1966. He received the Ph.D. degree in computer science from the East China University of Science and Technology, in 2008. He is currently a Professor and a M.S. degree Supervisor with the Shanghai University of Engineering Science. His research interests include quantum inspired evolutionary computation, distributed parallel processing, and evolutionary computing.