Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/mbe

MBE, 19(8): 8152–8186.
DOI: 10.3934/mbe.2022381
Received: 13 February 2022
Revised: 13 May 2022
Accepted: 24 May 2022
Published: 06 June 2022

Research article

Research on improved ant colony optimization for traveling salesman problem

Teng Fei[1], **Xinxin Wu**[2], **Liyi Zhang**[1], **Yong Zhang**[1] **and Lei Chen**[1,*]

[1] Institute of Information Engineering, Tianjin University of Commerce, Tianjin 300134, China
[2] College of Science, Tianjin University of Commerce, Tianjin 300134, China

* **Correspondence:** Email: chenlei@tjcu.edu.cn.

Abstract: As one of the most popular combinatorial optimization problems, Traveling Salesman Problem (TSP) has attracted lots of attention from academia since it was proposed. Numerous meta-heuristics and heuristics have been proposed and used to solve the TSP. Although Ant Colony Optimization (ACO) is a natural TSP solving algorithm, in the process of solving it, there are also some shortcomings such as slow convergence speed and prone to fall into local optimum. Therefore, this paper proposes an improved ant colony optimization based on graph convolutional network: Graph Convolutional Network Improved Ant Colony Optimization (GCNIACO). The graph convolutional network is introduced to generate a better solution, and the better solution is converted into the pheromone on the initial path of the ACO. Thereby, the guiding effect of the pheromone concentration for the ants at the beginning of the algorithm is enhanced. In the meantime, through adaptive dynamic adjustment of the pheromone volatility factor and the introduction of the 3-opt algorithm, the algorithm's ability to jump out of the local optimum is enhanced. Finally, GCNIACO is simulated on TSP datasets and engineering application example. Comparing the optimization results with other classical algorithms, it is verified that the graph convolutional network improved ant colony optimization has better performance in obtaining the optimal solution.

Keywords: traveling salesman problem; ant colony optimization; graph convolutional network; dynamic pheromone volatility factor; 3-opt algorithm

1. Introduction

This kind of optimization problem that finds the optimal solution in a finite set of feasible solutions is called combinatorial optimization problem. With the wide application in various industries and the rapid development of computer technology, the combinatorial optimization problem has developed into an independent branch of operational research. The research questions involve financial investment

[1], ecological environment [2], medical biology [3], logistics management [4], transportation [5], industrial engineering [6], medical image [7] and many other fields. Traveling salesman problem (TSP) is one of the most popular combinatorial optimization problems in recent years, and has been widely used as a benchmark for various optimization techniques and meta-heuristic searches. The traveling salesman problem was proposed in 1930, its goal is to find the shortest path to visit each city exactly once and return to the starting point city based on a given list of cities and the distance between cities. Since the feasible solution of this problem is the full permutation of all vertices, as the number of vertices increases, combinatorial explosion will occur, so this is also an NP-hard problem. Accurate algorithms such as cutting plane, branch and bound, the shortest spanning tree and the 2-matching can be used to solve the exact solution of the TSP [8]. However, when the traveling salesman problem is too large, the accuracy of the data sample is not enough, the problem target conflicts, etc., the accurate algorithm runs for a long time and it is difficult to obtain satisfactory results. Therefore, scholars use meta-heuristic and heuristic algorithms to effectively solve such problems. At this time, the algorithm does not insist on an accurate optimal solution, but finds an acceptable optimal solution in a reasonable time.

In the meta-heuristic algorithm, the principle and mechanism of ant colony optimization (ACO) [9] make it a natural traveling salesman problem solving algorithm. When ants are in search of food, they transmit information between individuals by releasing pheromone along the path they travel. At the same time, the ants will choose paths with a higher concentration of pheromone to move. With the continuous search of the ant colony, the shorter the path, the more ants will pass, and the higher the pheromone concentration left on the path, prompting more ants to choose this path to move, thus looping to form a positive feedback mechanism. This allows the ant colony to quickly find the shortest path to find food. In 1991, Italian scholar Dorigo et al. just imitated this foraging behavior of ant colonies in the biological world, and proposed a swarm intelligent bionic optimization algorithm: Ant Colony Optimization (ACO). But the ACO itself also has certain defects and shortcomings. With the continuous in-depth development of research, scholars have proposed many improvements and algorithm fusion schemes. Good results have been achieved and many practical problems in different situations have been solved.

With the rise of machine learning, especially the development of graph neural networks, the learning-based approach is also effective in solving the TSP. The motivation of the paper is that although the ACO has defects and deficiencies, it is possible to achieve good results after improvement. At the same time, the graph convolutional network also has the ability to solve the TSP. Therefore, the article proposes a new improved ant colony optimization based on graph convolutional network: Graph Convolutional Network Improved Ant Colony Optimization (GCNIACO). Aiming at shortcomings of the lack of pheromone and slow convergence in the initial stage of the ACO, the graph convolutional network is introduced to generate a better solution, and this better solution is converted into the initial pheromone of the ant colony through a pheromone conversion strategy. Thereby, the initial solving speed of the algorithm has been improved. Then, aiming at the shortcomings of ant colony optimization that it is easy to stagnation and fall into local optimum in the later stage of iterative stage, the ability of the algorithm to jump out of local optimum is enhanced by dynamically adjusting the pheromone volatility factor and introducing the 3-opt algorithm.

Table 1. Summary of the literature review for solving the traveling salesman problem.

Methodology	Mechanism	Features	Challenges
DJAYA [10]	A new discrete optimization method based on Jaya algorithm and transformation operators is proposed.	In terms of mean, standard deviation and relative error, the results show that DJAYA is competitive and alternative.	Inferior to DSTAII in some cases.
D-GWO [11]	Combined with the 2-opt algorithm, a new discrete GWO algorithm is proposed.	D-GWO is able to provide adequate and comparable solutions for symmetric TSP.	Poor performance on instances kroc100 and kroe100.
DHOA [12]	A new hybrid optimization algorithm is developed by integrating the excellent performance of deer hunting optimization algorithm and earthworm optimization algorithm.	Demonstrate better convergence performance and lower computational complexity.	The applicability of the algorithm in practical engineering problems has not been verified.
S-ROA [13]	A new hybrid optimization algorithm is developed by integrating the excellent performance of spotted hyena optimizer algorithm and rider optimization algorithm.	Able to get rid of premature convergence.	It takes longer time in solving for large scale number of cities.
DBAL [14]	Lévy's flights and neutral crossover operator are introduced.	The ability of the algorithm to jump out of the local optimum is enhanced.	DBAL is not suitable for very large-scale TSP datasets (over 1002 cities); and the running time also needs to be further optimized.
neural combinatorial optimization framework [19]	Design their own critic to compute a baseline for the tour length; and introduce 2-opt heuristic.	Near-optimal results are obtained on a 2D Euclidean graph.	The applicability of the algorithm on real TSP instances has not been verified.
GNN [20]	Train the model to be an efficient message-passing algorithm.	GNNs can learn the decision variables of solving the traveling salesman problem with very little supervision.	Model is not trained and evaluated on the comprehensive set of real and random graphs.
model based on attention layers [21]	An attention layer-based model is proposed and the model is trained by using REINFORCE with a simple baseline based on a deterministic greedy rollout.	Significantly improve the learning heuristic results of TSP.	The simple masking procedure in the model cannot meet the feasibility constraints of practical problems.
BGNN [22]	Design a bidirectional message passing layer and train the model with imitation learning.	It can balance running time and performance, and has the ability to jump out of local optimum.	The applicability of the algorithm on real TSP instances has not been verified.

This study's major contributions for the literature are as follows:

- A new improved ant colony optimization is proposed to improve its performance when solving the TSP.
- The simulation test results indicate that GCNIACO can provide the solutions with good competitive performance for the TSP.
- Provide a reference method for the combination of swarm intelligence algorithm and machine learning.

Table 2. Summary of the literature review on ant colony optimization improvement.

Methodology	Mechanism	Features	Challenges
Fuzzy-ACO [23]	Introduce fuzzy logic module to calculate pheromone value; and introduce Taguchi concept to optimize algorithm parameters.	The algorithm is more appropriate for handling complex network.	The pheromone formula and algorithm operation in the actual environment are not perfect.
DEACO [24]	Dynamically adjust the parameters of the ant colony optimization.	The convergence speed and search accuracy are improved.	Theoretical analysis is difficult, and the experimental basis is more than theoretical research.
PACON [25]	Introduce an angle and angle calculation rule in pheromone transfer rule; and increase the weight of pheromone concentration on the optimal path.	The algorithm convergence speed and convergence accuracy have been improved.	The applicability of the algorithm in practical engineering problems has not been verified.
HAACO [26]	Introduce the heterogeneity of ACO; introduce parameter adaptation rules; and integrate 3-opt algorithm.	HAACO has better algorithm performance relative to the compared algorithms.	Solve poorly on some very large instances of TSP.
GA-ACO [27]	The ant population is initialized using the genetic algorithm.	It has stronger global search performance and fast convergence.	No simulation comparisons with other excellent algorithms have been performed.
improved ant colony algorithm [28]	The heuristic function is improved; the pheromone update mechanism is improved, and the path selection strategy is improved.	It outperforms traditional algorithms in terms of path length and task waiting time.	The simultaneous working of multiple cranes has not been analyzed.
improved ant colony algorithm [29]	The initial pheromone differential distribution strategy is adopted; the local path is optimized in blocks; the pheromone update mechanism is improved; and the path selection strategy is improved.	It can plan the optimal path with effective obstacle avoidance and little number of folds.	The running time of the algorithm has not been analyzed.
IDAACO [30]	Heuristic strategy with direction information; adaptive pseudo-random transmission strategy; improved local pheromone update mechanism and improved global pheromone update mechanism.	IDAACO has the advantage in terms of practicality and efficiency.	It is difficult to balance the relationship between the parameters and the calculation formula.
multi-factor improved ant colony algorithm [31]	Construct multi-factor heuristic function; distribute initial pheromone stepwise; update pheromone by classification; adopt max-min ant strategy and adaptively adjust pheromone volatile factor.	Good ability to adapt to the environment.	It has not been analyzed and compared with other excellent algorithms in a complex terrain barrier-free environment.
IACO [32]	The formula of transition probability is improved; the pheromone volatility factor of self-adaptive adjustment is designed; the variable neighborhood local search embedded by insertion operator and exchange operator is introduced.	The improved effect is obviously better than that of the traditional ant colony optimization.	The model does not take into account the dynamics and heterogeneity of customer needs.
IACO [33]	A pheromone update mechanism based on information entropy is designed, and variable neighborhood search is introduced.	Global and local search capabilities have been improved.	The running time of the algorithm is prolonged.

The other parts of the thesis are composed as follows: Section 2 is the relevant literature review; Section 3 introduces the basic concept of the traveling salesman problem; Section 4 introduces the basic ant colony optimization model; Section 5 introduces the principle and optimization mechanism of the proposed graph convolutional network improved ant colony optimization. In Section 6, the improved algorithm is simulated and tested on a wide range of benchmark examples, and compared with the traditional meta-heuristic algorithm to verify the improved effect of the algorithm. Section 7 applies some statistical tests to confirm that the observed differences between the original and improved versions are actually meaningful. Section 8 applies the improved algorithm to solve practical engineering problem to verify the practicality of the algorithm. Section 9 summarizes the overall research work and presents an outlook for the future.

2. Literature review

This part mainly focuses on the literature review on the solution of TSP and the improvement of ACO in recent years.

2.1. Traveling salesman problem solving

In order to obtain a satisfactory solution to the NP-hard traveling salesman problem, scholars have proposed lots of meta-heuristics to solve the TSP. Gunduz et al. [10] proposed a new discrete optimization method called DJAYA for the permutation-coded optimization problems, and studied the performance of the proposed algorithm on 14 different symmetric TSP data sets. Experimental results showed that the algorithm was an optional and competitive optimization algorithm. Panwar et al. [11] based on the meta-heuristic algorithm for solving continuous optimization problems: gray wolf optimizer (GWO), combined with the 2-opt algorithm, produced a novel discrete GWO algorithm for solving the symmetric traveling salesman problem. And the effectiveness of the proposed method was verified by experimental simulation. Kanna et al. [12] proposed a new hybrid algorithm by integrating two meta-heuristic algorithms with good performance: earthworm-based deer hunting optimization algorithm (DHOA). The proposed optimization algorithm showed better convergence performance and lower computational complexity when solving the TSP. Based on the rider optimization algorithm (ROA) and the spotted hyena optimizer algorithm (SHO), Krishna et al. [13] constructed a new hybrid algorithm: spotted hyena-based rider optimization (S-ROA). By solving the traveling salesman problem case, the good competitive performance of the constructed method was verified. Saji et al. [14] proposed a new discrete bat algorithm to solve the TSP. Lévy's flights and neutral crossover operator were introduced to enhance the ability of the algorithm to jump out the local optimum. The overall performance of the algorithm was therefore improved. There are also meta-heuristic algorithms such as discrete crow-inspired algorithms (DC) [15], discrete shuffled frog leaping algorithm (DSFLA) [16], differential evolution (DE) [17], discrete spider monkey optimization (DSMO) [18], etc. have also been proposed and applied to solve the TSP, and all have obtained good results. With the in-depth research and development of machine learning and graph neural networks, solving the TSP based on learning has also attracted the attention of scholars. Deudon et al. [19] proposed a neural combinatorial optimization framework to solve the TSP. In the framework, the city coordinates were used as input, and the neural network was trained by using reinforcement learning to predict the distribution of the city arrangement. Then the good performance of the proposed framework was verified by experimental simulation. Prates et al. [20] successfully applied graph neural networks to solve the TSP, proving that graph neural networks could learn the decision variables of solving the traveling salesman problem with little supervision. Kool et al. [21] proposed a model based on the attention layer, and trained the model by using REINFORCE with a simple baseline based on a deterministic greedy rollout. Finally, satisfactory results were obtained when solving problems such as orienteering problem (OP) and prize collecting TSP (PCTSP). Hu et al. [22] proposed a bidirectional graph neural network to solve the traveling salesman problem on any symmetric graph. The network used imitation learning to sequentially generate the next city to visit. At the same time, by designing a bidirectional message transfer layer, the graph was coded based on edge and partial solutions, so as to construct a near-optimum solution for traveling salesman problem on any symmetric graph. A summary of relevant literature reviews was shown in Table 1.

2.2. *Ant colony optimization improvement*

Ragmani et al. [23] proposed a new hybrid algorithm based on fuzzy logic and ACO to improve load leveling in cloud computing environments. The algorithm used Taguchi experimental design to obtain the optimum parameter value of the ant colony optimization, and defined a fuzzy module to assess pheromone value, thereby speeding up the computation of the algorithm. Aiming at the shortcomings of the ant colony optimization's slow convergence speed and prone to fall into the local optimum, Ebadinezhad et al. [24] proposed a dynamic evaporation ant colony optimization (DEACO) that dynamically adjusted the parameters of the ACO. It was used to solve the TSP example, which verified the good performance of the proposed algorithm in terms of convergence speed and search accuracy. Aiming at the shortcomings of low convergence accuracy and slow convergence speed in the ant colony optimization, Li et al. [25] proposed a pseudo-dynamic search ACO with an improved negative feedback mechanism. By introducing an angle and angle calculation rule in the pheromone transmission rule, it affected the probability of city selection and improved the ability of the algorithm to jump out of the local optimum. Then the algorithm updated the pheromone concentration on the optimum path and the worst path at the same time, and enhanced the weight of the pheromone concentration on the optimum path, thereby improving the convergence speed of the algorithm. Tuani et al. [26] proposed a heterogenous adaptive ant colony optimization (HAACO), which modified the transition probability formula in the ACO to introduce the heterogeneity of ACO. Then by introducing a set of parameter adaptation rules, the adaptiveness of α and β parameters in the ant colony optimization was achieved. At the same time, the 3-opt local search algorithm was integrated into the proposed algorithm to further improve the algorithm's search ability. Aiming at the problem that ACO has strong dependence on pheromone and is prone to fall into local optimum, Zheng et al. [27] proposed an improved hybrid genetic ant colony optimization. The optimal solution produced by the genetic algorithm was used as the initial information of the pheromone in the ant colony optimization to improve the algorithm's global search ability and rapid convergence. Li et al. [28] proposed an improved ant colony optimization to solve the path planning problem of unmanned cranes during hoisting, and improved the solution performance of the algorithm by improving the heuristic function, pheromone update mechanism, and path selection strategy. Aiming at the shortcomings of ant colony optimization to solve the path planning problem, such as falling into local optimum and slow convergence speed, Tang et al. [29] proposed an improved ant colony algorithm. They used a differentiated distribution strategy to improve the initial pheromone concentration on the path. Then, the ACO was improved by adopting local path block optimization strategy, introducing pheromone self-adjustment enhancement factor, introducing random state transition parameters, etc., so as to improve the convergence and stability of the algorithm. Liu et al. [30] proposed an improved dynamic adaptive ant colony optimization (IDAACO) to handle the problem of pipe routing design. IDAACO had designed four improved new mechanisms: heuristic strategy with directional information, adaptive pseudo-random transmission strategy, improved local pheromone update mechanism and improved global pheromone update mechanism. Then, the excellent performance of the improved algorithm in terms of practicability and high efficiency was verified through experimental simulation. Yang et al. [31] proposed a multi-factor improved ant colony optimization to solve the path planning problem of mobile robots. The performance of the algorithm was improved by constructing multi-factor heuristic functions, assigning initial pheromone stepwise, updating pheromone classification, adopting the maximum and minimum ant strategy and adaptively adjusting the pheromone volatility factor. He et al. [32] proposed an improved ant colony optimization (IACO) to solve the vehicle routing problem with soft time windows. Based on the basic ant colony optimization, they improved the transition probability formula, designed an adaptively adjusted pheromone volatility factor, and introduced the variable neighborhood local search embedded by the insertion operator and the exchange operator. They also set the conditions for starting and exiting the local search, and updated the current local optimal solution. Finally, the effectiveness of the improved algorithm was verified through experimental simulation. Wang et al. [33] proposed an improved ant colony algorithm to solve the green periodic vehicle routing problem with time windows. By designing a pheromone update mechanism based on information entropy in the algorithm and introducing variable neighborhood search, the algorithm's global and local search capabilities were improved. A summary of relevant literature reviews was shown in Table 2.

3. Traveling salesman problem

Imagine you are planning a travel route now. You start from the city where you live and travel through the cities you are interested in sequentially. You have passed through all these cities and only once. Finally, you are back to the city where you live, so how do you plan your route to minimize the distance you traveled? This is the typical traveling salesman problem (TSP). TSP includes symmetric traveling salesman problem (STSP) [34], asymmetric traveling salesman problem (ATSP) [35], pickup-and-delivery traveling salesman problem (PDTSP) [36], multiple traveling salesman problem (MTSP) [37], multi-objective traveling salesman problem (MOTSP) [38], etc. This article mainly takes the classic symmetric traveling salesman problem as the research objective, hereinafter referred to as the traveling salesman problem. It can be described as: in the given n cities $C = [C_1, C_2, ..., C_n]$, the distance $d_{ij}(i, j \in 1, 2, ..., n)$ between any two cities i and j is known, and $d_{ij} = d_{ji}$; find a closed path $\hat{C} = [\hat{C_1}, \hat{C_2}, ..., \hat{C_n}]$ that minimizes the objective function value of Eq (3.1) and satisfies the constraint condition of traversing each city once and finally returning to the starting point.

$$L(\hat{C}) = \sum_{i=1}^{n-1} d_{(\hat{C_i}, \hat{C_{i+1}})} + d_{(\hat{C_n}, \hat{C_1})} \tag{3.1}$$

According to the knowledge of graph theory in operational research, the quiddity of the traveling salesman problem is to seek a Hamilton loop with the smallest weight in a completely undirected graph with weights. Let $G = (V, E)$ be the weighted graph; $V = (1, 2, ..., n)$ is the set of nodes formed by all cities; E is the set of edges; the distance between each node $d_{ij} > 0(i, j \in V)$ is known, and $d_{ii} = 0, d_{ij} = d_{ji}$; let the expression of $x_{ij}(i, j \in V)$ be:

$$x_{ij} = \begin{cases} 1, & \text{if (i,j) is on the optimal closed path} \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

Then the TSP can be modeled as the following integer programming problem [10]:

$$\min \sum_{i} \sum_{j} d_{ij} x_{ij} \tag{3.3}$$

Subject to

$$\sum_{i \in V} x_{ij} = 1, \forall i \in V, i \neq j \tag{3.4}$$

$$\sum_{j \in V} x_{ij} = 1, \forall j \in V, i \neq j \tag{3.5}$$

$$\sum_{i,j \in S} x_{ij} \leq |S| - 1, 2 \leq |S| \leq n - 2, S \subset V \tag{3.6}$$

$$x_{ij} \in \{0, 1\}, \forall i, j \in V \tag{3.7}$$

where S is all non-empty subsets of the node set V, and $|S|$ represents the number of vertices contained in the set S. Equation (3.3) is the objective function, which means that the sum of distances is the smallest; Eqs (3.4)–(3.7) are all constraints; Eqs (3.4) and (3.5) indicate that each node in the weighted graph G is reached and left only once; Eq (3.6) constrains to eliminate the occurrence of subtour, and ensures that the optimal closed path is a single large loop that passes through all points.

4. Ant colony optimization

The ACO is inspired by the collaborative work of ants. By abstracting the problem to be studied as the problem of finding the optimal path, the node model is constructed. The construction process of the solution is the selection and movement process of ants between nodes. Solution elements that satisfy the solution conditions are continuously added to construct a complete feasible solution. Finally, it converges to the optimum solution of the problem under the effect of the pheromone positive feedback mechanism. The principle and mechanism of ant colony optimization make it a natural traveling salesman problem solving algorithm. This article also uses the traveling salesman problem as an instance to introduce the classic ant colony optimization model [39]:

In order to ensure the generality of the algorithm, the number of cities is set to n; the number of ants is set to m; the distance $d_{ij}(i, j \in 1, 2, ..., n)$ between any two cities i and j is known, and $d_{ij} = d_{ji}$, the Euclidean distance is used in the paper; the pheromone concentration on the path between city i and city j at time t is set to $\tau_{ij}(t)$. At the initial moment, the ACO sets the pheromone concentration on all paths to same value. Then let the ant choose the next city to visit according to a certain selection probability $p^k_{ij}(t)$, where $p^k_{ij}(t)$ is called the transition probability of the ant k at the time of t from the city i to the city j, which is the path selection process of the ant. The specific formula is expressed in Eq (4.1):

$$p_{ij}^{k}(t) = \begin{cases} \frac{[\tau_{ij}(t)]^{\alpha} \bullet [\eta_{ij}(t)]^{\beta}}{\sum_{s \in allow_k} [\tau_{is}(t)]^{\alpha} \bullet [\eta_{is}(t)]^{\beta}}, & j \in allow_k \\ 0, & j \notin allow_k \end{cases} \tag{4.1}$$

where $\eta_{ij}(t)$ is called the heuristic function, which represents expectation degree of the ants from the city i to the city j at time t, generally $\eta_{ij}(t) = \frac{1}{d_{ij}}$; $allow_k$ is the set of cities currently accessible by each ant k; α is a pheromone factor, which reflects the influence degree of pheromone when the ant chooses a path; β is a heuristic function factor, which reflects the influence degree of the heuristic function when the ant chooses a path. When all the ants complete a complete traversal, that is, after each ant independently completes its own feasible solution construction, the pheromone concentration on the inter-city path will increase while the ants secrete the pheromone, and decrease at the same time as the pheromone volatilizes. The specific update mechanism is as follows:

$$\tau_{ij}(t+1) = (1-\rho) \bullet \tau_{ij}(t) + \Delta\tau_{ij}, 0 < \rho < 1 \tag{4.2}$$

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \tag{4.3}$$

where ρ is the pheromone volatility factor, which reflects the degree of pheromone volatility, $\rho \in (0, 1)$; $\Delta\tau_{ij}$ represents the increased pheromone concentration that all ants secrete pheromone on the path between city i and city j in one cycle; $\Delta\tau_{ij}^{k}$ represents the increased pheromone concentration that a single ant k secretes pheromone on the path between city i and city j in one cycle, the Ant-cycle model [40] that can utilize global information is generally used to define $\Delta\tau_{ij}^{k}$:

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{L_k}, & \text{ant k goes from city i to city j} \\ 0, & \text{otherwise} \end{cases} \tag{4.4}$$

where Q is a pheromone constant, which denotes the total amount of pheromone secreted by each ant after completing a complete traversal; L_k denotes the length of the path taken by a single ant k after completing a complete traversal. Equation (4.2) simulates the ant's pheromone update process, including the volatility of pheromone and the superposition of pheromone on the path that the ant passes; Eq (4.3) is to superimpose new pheromone on all nodes path, and the value of $\Delta\tau_{ij}^{k}$ is calculated according to Eq (4.4).

5. Graph convolutional network improved ant colony optimization

The ACO has strong robustness, and the positive feedback mechanism of pheromone makes the ant colony optimization have a strong capacity to discover the optimum solution. But at the same time, the algorithm also has the following shortcomings [41]: in the initial stage of the solution, pheromone is lacking, and the solution speed is slow; in the later stage of the solution, under the influence of pheromone, it is easy to fall into the local optimum. Therefore, in view of the shortcomings of the ACO, this paper proposes a graph convolutional network improved ant colony optimization (GCNIACO). First, the graph convolutional network is introduced to generate a better solution, and the better solution is converted into the initial value of pheromone through the pheromone conversion strategy to improve the initial solution speed of the algorithm; the algorithm's capacity to jump out of the local optimum is then enhanced by dynamically adjusting the pheromone volatility factor and combining with the 3-opt algorithm.

5.1. Graph convolutional network

At the initial moment in the ant colony optimization, all paths have the same pheromone concentration, and the attraction to ants is also the same. As a result, the pheromone in the initial stage has poor guidance for the ants' choice of paths, and the convergence rate of the optimal solution is slower. In addition, it may search for irrelevant paths that produce a large number of non-optimal path components. The enhancement of pheromone on these paths interferes with the update of the overall pheromone of the algorithm, which is not conducive to exploring a better path. Therefore, in order to improve the search performance of the algorithm in the initial stage. The graph convolutional network is introduced to generate the better solution, and the better solution is converted into the initial value of the pheromone through the pheromone conversion strategy. Then, the difference in pheromone concentration between the better path and the poorer path at the initial stage is increased.

Convolutional neural network (CNN) requires regular data domains, such as 2-dimensional or 3-dimensional Euclidean grid images in computer vision (CV), 1-dimensional sequential text in natural language processing, and so on [42]. However, a lot of data is usually not in the domain of regular data, but in the domain of heterogeneous graphs. This requires the use of another common data structure, that is, a graph composed of vertices and edges. But it is also difficult to directly define operations such as convolution and pooling in the graph, which hinders the development of CNN, so these drives form the graph convolutional network (GCN). In 2019, Chaitanya et al. [43] introduced a new learning-based method to solve the traveling salesman problem, used a deep graph convolutional network to construct an efficient traveling salesman problem graph representation, and then output the optimal path in a non-autoregressive manner through a highly parallelized beam search. The main practice is: take a graph as input, extract synthetic features from the nodes and edges of the graph by stacking several graph convolutional layers, and train the graph convolution model to directly output the adjacency matrix corresponding to the path; then use post-hoc beam search technology to convert the adjacency matrix obtained from the model into a valid path. Among them, the adjacency matrix denotes the probability of the edge appearing in the path; the parameters of the graph convolution model are trained end-to-end by minimizing the cross-entropy loss through gradient descent.

In the construction of the graph convolutional layer, let x_i^l and e_{ij}^l respectively denote the node feature vector and edge feature vector at the graph convolutional layer l associated with node i and edge ij then the node feature and edge feature of the next layer are defined as Eqs (5.1) and (5.2):

$$x_i^{l+1} = x_i^l + ReLU(BN(W_1^l x_i^l + \sum_{j\sim i} \eta_{ij}^l \odot W_2^l x_j^l)) \;\; with \;\; \eta_{ij}^l = \frac{\sigma(e_{ij}^l)}{\sum_{j'\sim i} \sigma(e_{ij'}^l) + \varepsilon} \tag{5.1}$$

$$e_i^{l+1} = e_i^l + ReLU(BN(W_3^l e_{ij}^l + W_4^l x_i^l + W_5^l x_j^l)) \tag{5.2}$$

where $W \in \mathbb{R}^{h\times h}$ is the parameter that needed to train by the graph convolution model; h is the hidden dimension of each graph convolutional layer; σ is the sigmoid function; ε is a small value, which can ensure that the denominator of η_{ij}^l is not 0; $ReLU$ is rectified linear unit, which is used as an activation function in the graph convolutional network; BN stands for batch normalization. When $l = 0$, $x_i^{l=0}$ and $e_{ij}^{l=0}$ respectively represent the node feature and edge feature of the input layer, and their definitions are as shown in Eqs (5.3) and (5.4) respectively:

$$x_i^{l=0} = A_1 x_i + b_1 \tag{5.3}$$

$$e_{ij}^{l=0} = A_2 d_{ij} + b_2 \parallel A_3 \delta_{ij}^{k\text{-}NN} \tag{5.4}$$

$$\delta_{ij}^{k\text{-}NN} = \begin{cases} 1, & \text{if node i and node j are k nearest neighbors} \\ 2, & \text{if node i and node j are selfconnected} \\ 0, & \text{otherwise} \end{cases} \tag{5.5}$$

where x_i is the two-dimensional coordinate of the input graph, $x_i \in [0,1]^2$; $A_1 \in \mathbb{R}^{h\times 2}$, $A_2 \in \mathbb{R}^{\frac{h}{2}\times 1}$, $A_3 \in \mathbb{R}^{\frac{h}{2}\times 3}$, h is the hidden dimension of the graph convolutional layer, A_1, A_2, A_3, b_1, b_2 are the initial parameter values of the graph convolution model; $\parallel$ is the concatenation operator. Equation (5.3) embeds two-dimensional coordinate as h-dimensional feature vector; Eq (5.4) embeds the Euclidean distance d_{ij} of the edge as $\frac{h}{2}$-dimensional feature vector; the $\delta_{ij}^{k\text{-}NN}$ in Eq (5.5) is defined as the indicator function of the edge in the TSP, and the learning process of the model is accelerated by inputting k-nearest neighbor graph, usually $k = 20$.

The edge feature e_{ij}^L of the last layer of the graph convolution model is used to calculate the probability p_{ij}^{TSP} that the edge is connected in the TSP path. The p_{ij}^{TSP} can be regarded as a probabilistic heatmap of edge connections computed on an adjacency matrix, each $p_{ij}^{TSP} \in [0,1]^2$ and is computed with the multi-layer perceptron (MLP):

$$p_{ij}^{TSP} = MLP(e_{ij}^L) \tag{5.6}$$

After the graph convolution model outputs an adjacency matrix predicting edge occurrence probabilities, a probabilistic heatmap of edge connections is obtained. Using the beam search technique, starting from the first node, the probabilistic heatmap is explored by extending the b most likely edge connections between neighbor nodes. The first b local paths of each stage are then iteratively expanded until all nodes in the graph are visited. At the same time, during the search process, the nodes that have been visited before are shielded to ensure the validity of the path. The final predicted optimal path is the path with the shortest length among the b complete paths after the end of the beam search.

When improving the initial pheromone of the ACO, if the optimal solution predicted and generated by the graph convolutional network is directly converted into the initial pheromone, the information of the graph convolutional network solution may not be fully utilized because the number of selections is too small. Therefore, in order to fully and effectively utilize the information of graph convolutional network solutions, according to the literature [44], this paper designs the following pheromone conversion strategy. The main practice is: set the initial pheromone of the ant colony optimization as $\tau = \tau_c + \tau_{GCN}$. Among them, τ_c is the initial pheromone constant, and the value can be specifically set based on the actual situation, but please try to ensure that there is a large difference in pheromone concentration between the better path and the worse path after use, and this paper sets it as $\tau_c = 0.1$.

τ_{GCN} is generated by the conversion of the optimal solution of the graph convolutional network, which can be calculated by the following formulas:

$$\tau_{ij}^{GCN} = \frac{k_{ij}}{q} \tag{5.7}$$

$$\tau_{GCN} = \begin{bmatrix} \tau_{11}^{GCN} & \cdots & \tau_{1n}^{GCN} \\ \vdots & & \vdots \\ \tau_{n1}^{GCN} & \cdots & \tau_{nn}^{GCN} \end{bmatrix} \tag{5.8}$$

where τ_{ij}^{GCN} is the pheromone concentration on the path between nodes i and j; q is the first q short paths taken from the beam search of the graph convolutional network, according to the literature [44], the value of q is $q = 30$; k_{ij} is the number of times that every two city nodes (i, j) appear in q paths; n is the number of city nodes. In order to reduce the interference of pheromone reinforcement on poor paths, this paper sets a threshold for k_{ij}. When the number of occurrences of the two city nodes (i, j) in q paths is less than or equal to $10\%q$, $k_{ij} = 0$, otherwise the value of k_{ij} remains unchanged. The specific formula is shown in Eq (5.9):

$$k_{ij} = \begin{cases} k_{ij}, & \text{if } k_{ij} > 10\%q \\ 0, & \text{if } k_{ij} \leq 10\%q \end{cases} \tag{5.9}$$

5.2. *Dynamic pheromone volatility factor ρ*

The pheromone volatility factor ρ represents the volatility degree of pheromone in each iteration, which affects the global search ability and convergence speed of the ant colony optimization. When the value of ρ is too large, the pheromone on each path volatilizes faster, and the ant colony may search the path repeatedly, resulting in a decrease in the convergence speed; when the value of ρ is too small, the volatility speed of pheromone on each path is slow, and the accumulation of pheromone concentration on the path is too high, which may affect the randomness and global search ability of ant colony search, resulting in the algorithm falls into local optimum. Therefore, this paper proposes an improved method for dynamic pheromone volatility factor, as shown in Eq (5.10):

$$\rho = \begin{cases} \log(\frac{2\times NC\_max}{NC\_max+NC}), & \text{if } \log(\frac{2\times NC\_max}{NC\_max+NC}) > \rho_{min} \\ \rho_{min}, & \text{otherwise} \end{cases} \tag{5.10}$$

where NC_max is the total number of iterations of the algorithm; NC is the current iteration number of the algorithm; ρ_{min} is the minimum value of ρ, which is set to prevent ρ from being too small and reducing the algorithm convergence speed. And according to the literature [39], the best empirical result range of ρ is $0.1 \leq \rho \leq 0.99$, the paper sets the value of ρ_{min} to $\rho_{min} = 0.1$. The improved pheromone volatility factor change curve was shown in Figure 1.

The main idea of dynamic pheromone volatility factor proposed in the paper is: it is hoped that the pheromone volatility factor value will change with the number of iterations. In the initial stage of the algorithm iteration, the pheromone volatility factor value is set larger, the accumulation of pheromone concentration on the path is less, and the ants can search for more paths, so that the algorithm has a strong global search ability. After the algorithm iterates a certain number of times, the numerical setting of the pheromone volatility factor gradually decreases, so that the accumulation of pheromone concentration on the path gradually increases, which enhances the pulling and guiding effect of pheromone on the ant colony. The ants gradually converge to the path with high pheromone concentration, which speed up the algorithm convergence speed.

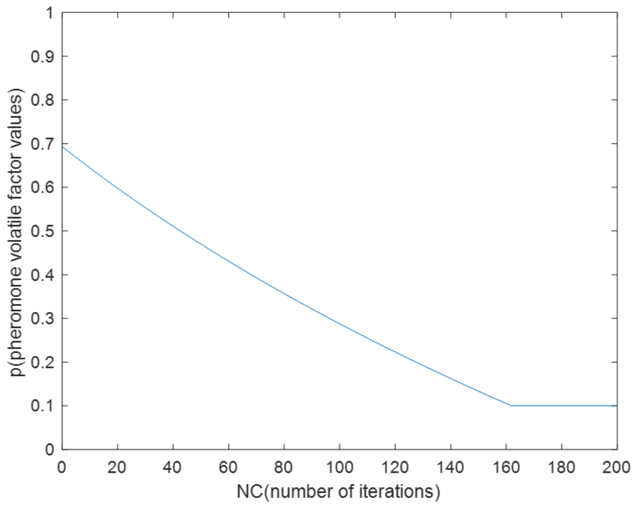

Figure 1. Dynamic pheromone volatility factor change curve.

5.3. 3-opt algorithm

When the ant colony optimization is searching for a feasible solution of the path, it may generate a suboptimal path with crossover phenomenon similar to Figure 2(a). The more paths intersect, the longer the path length, and the worse the quality of the obtained feasible solution. Therefore, this article uses the 3-opt algorithm [45] to optimize the ant colony optimization to further shorten the path length. That is, the de-crossing operation is performed on the part of the path that has a similar crossover phenomenon in Figure 2(a), and it is optimized into the optimal path in Figure 2(b). The specific operation steps are: starting from the first node $t = 1$, 5 points $c(t)$, $c(t+1)$, $c(t+2)$, $c(t+3)$, $c(t+4)$ are continuously selected each time; swap the positions of the three middle points in turn to get six kinds of sequential arrangements, and the calculated lengths are: $L_1 = d[c(t), c(t+1)] + d[c(t+1), c(t+2)] + d[c(t+2), c(t+3)] + d[c(t+3), c(t+4)]$; $L_2 = d[c(t), c(t+1)] + d[c(t+1), c(t+3)] + d[c(t+3), c(t+2)] + d[c(t+2), c(t+4)]$; $L_3 = d[c(t), c(t+2)] + d[c(t+2), c(t+1)] + d[c(t+1), c(t+3)] + d[c(t+3), c(t+4)]$; $L_4 = d[c(t), c(t+2)] + d[c(t+2), c(t+3)] + d[c(t+3), c(t+1)] + d[c(t+1), c(t+4)]$; $L_5 = d[c(t), c(t+3)] + d[c(t+3), c(t+2)] + d[c(t+2), c(t+1)] + d[c(t+1), c(t+4)]$; $L_6 = d[c(t), c(t+3)] + d[c(t+3), c(t+1)] + d[c(t+1), c(t+2)] + d[c(t+2), c(t+4)]$; then compare the size of the length $L_1, L_2, L_3, L_4, L_5, L_6$, and replace the original arrangement of $c(t)$ to $c(t+4)$ in the path with the arrangement corresponding to the minimum length; iterate $t = t+1$ in turn until $t = n$. In the TSP, it is important to note that: $c(n+1) = c(1)$, $c(n+2) = c(2)$, $c(n+3) = c(3)$, $c(n+4) = c(4)$.

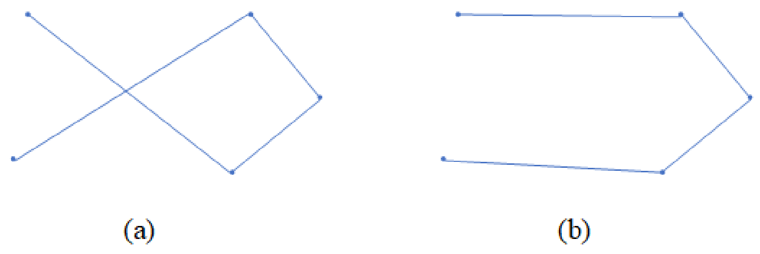

Figure 2. 3-opt algorithm de-crossing example.

5.4. GCNIACO flow

The main idea of graph convolutional network improved ant colony optimization is: the better solution generated by the graph convolutional network is converted into the initial pheromone of the ACO through the pheromone conversion strategy, and the difference of initial pheromone concentration on the best path and the suboptimal path is improved; then the pheromone volatility factor is improved to be dynamically adaptive, and the 3-opt algorithm is used for the de-crossing operation, which further optimizes the path length and enhances the algorithm's ability to jump out of the local optimum. The flowchart of graph convolutional network improved ant colony optimization was shown in Figure 3.

The implementation steps of the GCNIACO algorithm are as follows:

Step 1: initialize the coefficients related to the ant colony optimization and calculate the distance matrix between nodes;

Step 2: initialize the graph convolutional network and generate the better solution; then convert the better solution into the ant colony initial pheromone through the pheromone conversion strategy;

Step 3: randomly select the first city node to start for the ants; calculate the transition probability for each ant between city nodes according to Eq (4.1); select the next city node to visit by roulette, and record it in the path record table; repeat this process until all the ants have visited all city nodes;

Step 4: calculate the path length L_k that each ant passes through in the path record table;

Step 5: the 3-opt algorithm is introduced to perform the de-crossing operation on the best path in the path record table; if the optimized path length $L_{3\text{-}opt}$ is less than the best path $\min(L_k)$, use $L_{3\text{-}opt}$ to update $\min(L_k)$, and at the same time update the corresponding path node sequence in the path record table; if $L_{3\text{-}opt}$ is larger than $\min(L_k)$, directly jump to Step 6;

Step 6: record the best solution in the current iteration number;

Step 7: according to Eqs (4.2)–(4.4), update the pheromone on the path, wherein the pheromone volatility factor ρ is calculated according to Eq (5.10); then clear the path record table, and jump to Step 3 to continue the iteration;

Step 8: when the algorithm reaches the specified maximum number of iterations NC_max, stop the iteration and output the best solution.

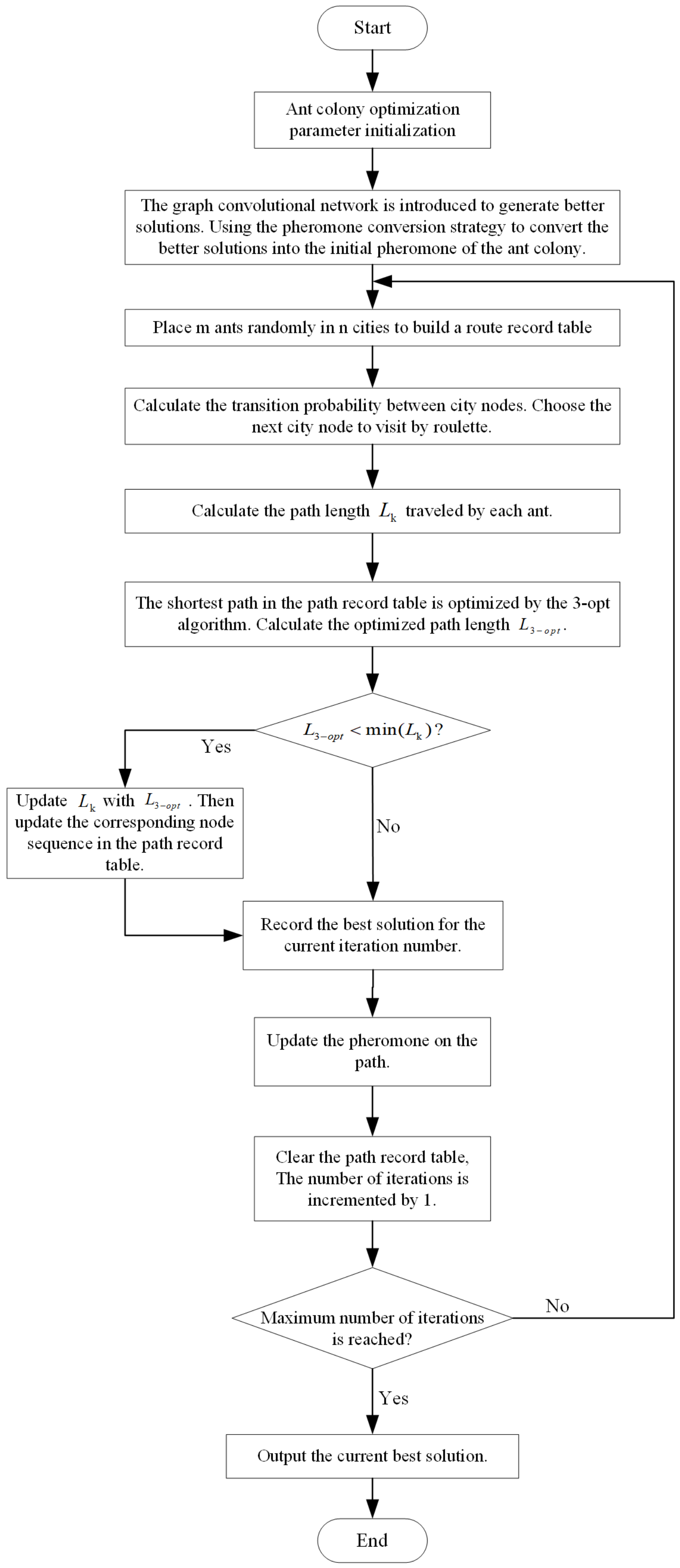
Start
Ant colony optimization parameter initialization
The graph convolutional network is introduced to generate better solutions. Using the pheromone conversion strategy to convert the better solutions into the initial pheromone of the ant colony.
Place m ants randomly in n cities to build a route record table
Calculate the transition probability between city nodes. Choose the next city node to visit by roulette.
Calculate the path length L_k traveled by each ant.
The shortest path in the path record table is optimized by the 3-opt algorithm. Calculate the optimized path length L_{3-opt}.
$L_{3-opt} < \min(L_k)$?
Yes
Update L_k with L_{3-opt}. Then update the corresponding node sequence in the path record table.
No
Record the best solution for the current iteration number.
Update the pheromone on the path.
Clear the path record table, The number of iterations is incremented by 1.
Maximum number of iterations is reached?
No
Yes
Output the current best solution.
End

Figure 3. Graph convolutional network improved ant colony optimization flowchart.

6. Experimental simulation

6.1. Generalization comparison of GCN models

When the graph convolutional network solves the TSP and generates a better solution, since the parameters of the graph convolutional network model are independent of the size of the instance city node, the model trained on the smaller instance graph can be used to solve any large instance [43]. In order to explore the influence of generalization effect of the graph convolution model in the graph convolutional network improved ant colony optimization on the holistic improvement of the algorithm, this paper conducts experimental simulations on the scale of the TSP with the city node size of 20, 50, and 100. Among them, the city coordinates of all traveling salesman problem instances are randomly and uniformly sampled in the two-dimensional unit square. The dataset and model hyperparameters of the graph convolutional network part in the GCNIACO algorithm are set by referring to [43]. The training data set corresponding to each problem scale is composed of 1,000,000 pairs problem instances and optimal path schemes, and validation data set is composed of 10,000 pairs problem instances and optimal path schemes. The optimal path scheme here is obtained using Concorde [46]. Each graph convolutional network model consists of 30 graph convolution layers and 3 layers of multi-layer perceptron, and the hidden dimension of each layer is $h = 300$. The beam width in beam search is fixed to $b = 1280$. During the training of the graph convolutional network, each training epoch randomly selects 10,000 subsets from the training set and divides them into 500 mini-batches for training. The Adam optimizer [47] with an initial learning rate of 0.001 is then used to minimize the cross-entropy loss for each mini-batch. The trained graph convolutional network model is validated on the validation set every five training epochs. If the validation loss does not reduce by at least 1% of the previous validation loss, then dynamically reduce the learning rate by dividing the learning rate by 1.01, thereby improving the learning ability of the model. Parameters of the ant colony optimization part in the GCNIACO algorithm are set in reference [48], the details were shown in Table 3. Then the GCNIACO algorithm uses different graph convolutional network model to solve the TSP 30 times for 20, 50 and 100 city nodes, and the results were demonstrated in Table 4.

Table 3. Related parameters setting table.

Parameter	Value
Number of ants m	$m = n$ (n is the number of cities)
Pheromone factor α	$\alpha = 1$
Heuristic function factor β	$\beta = 5$
Pheromone constant Q	$Q = 100$
The maximum number of iterations NC_max	$NC\_max = 200$

It can be obtained from Table 4 that the graph convolutional network model has a certain generalization ability, but there are also certain differences in the effect. On the traveling salesman problem with 20 nodes, the GCNIACO algorithm finds the same optimal solution for all three scale-trained graph convolutional network models. But the model trained with 100 nodes achieves better maximum value and average value than the other two models. On the traveling salesman problem with 50 nodes, the model trained with the corresponding 50 nodes is better than the other two models. However, compared

Table 4. 30 simulation results of GCNIACO algorithm using different graph convolutional network model.

Model	TSP20			TSP50			TSP100		
	Min	Max	Mean	Min	Max	Mean	Min	Max	Mean
TSP20 model	4.0024	4.0248	4.0032	6.0597	6.2367	6.1680	7.9006	8.1312	8.0093
TSP50 model	4.0024	4.0248	4.0032	6.0438	6.2284	6.1375	7.7763	8.1877	8.0006
TSP100 model	4.0024	4.0024	4.0024	6.0835	6.234	6.1602	7.8627	8.0831	7.97495

with the solution effect of the model trained by 20 nodes, the model trained by 100 nodes is better. On the traveling salesman problem with 100 nodes, although the model trained with 50 nodes can enable the GCNIACO algorithm to find a better minimum value, the model trained with 100 nodes can still maintain the advantage in the obtained maximum value and average value. Therefore, considering various factors, this paper chooses to use the graph convolutional network model trained on 100 nodes in the GCNIACO algorithm.

6.2. Validation of proposed improvement strategies

To verify the effectiveness of the improved strategies proposed in the study, the paper chooses to test it on the traveling salesman problem example with city node size of 50 and 100, where all coordinates are randomly and uniformly sampled in the two-dimensional unit square. Table 5 shows the relevant experimental results. Among them, ACO is the basic ant colony optimization, GCNIACO-1 is the improved algorithm for GCN optimization initialization, GCNIACO-2 is the improved algorithm with GCN and dynamic pheromone volatile factor optimization, and GCNIACO is the improved algorithm proposed in this paper. In order to ensure the validity and reliability of the results, each algorithm is run independently for 30 times, and the parameter settings are basically the same as those in Section 6.1, in which the default parameter is set to $\rho = 0.5$.

Table 5. Comparison of the results of different integration strategies.

Instance	Metric	Method			
		ACO	GCNIACO-1	GCNIACO-2	GCNIACO
TSP 50	Min	6.1705	6.0983	6.0807	6.0835
	Mean	6.2497	6.2093	6.2260	6.1602
	SD	0.0432	0.0546	0.0556	0.0378
TSP 100	Min	8.1672	8.0656	8.0229	7.8627
	Mean	8.2631	8.2081	8.2047	7.9750
	SD	0.0715	0.0541	0.0606	0.0545

It can be seen from Table 5 that the addition of the three improved strategies makes the improved ant colony optimization outperform the original algorithm in terms of best value and average value. Although these three strategies work at different stages, there is no doubt that they all have a positive impact on the results. Graph convolutional network can provide ”better” initial values for ACO’s pheromone memory. The adaptive dynamic adjustment of pheromone volatility factor can improve the global search ability and convergence speed [49]. The 3-opt algorithm can eliminate the intersections in the middle of the path and further optimize the path length.

6.3. Improvements on random TSP maps

By using GCNIACO and ACO to solve the TSP of the same city count, the improvement effect of the improved algorithm is analyzed. The size of the city is set to 20, 50, 100, and 200 respectively, and the specific city coordinates are randomly and uniformly sampled in the two-dimensional unit square. The parameter settings in GCNIACO are the same as in Section 6.1. The parameter settings in ACO are basically the same as those in GCNIACO, and the default parameter setting is $\rho = 0.5$. The results of 30 simulations are demonstrated in Table 6:

Table 6. Simulation results of GCNIACO and ACO.

TSP Data	TSP Optimal Solution	Simulation Best Solution		Simulation Worst Solution		Average Value		Time (s)		Deviation Rate %	
		ACO	GCNIACO	ACO	GCNIACO	ACO	GCNIACO	ACO	GCNIACO	ACO	GCNIACO
TSP20	4.0024	4.0024	4.0024	4.1389	4.0024	4.0570	4.0024	2.3213	10.4390	0	0
TSP50	6.0438	6.1705	6.0835	6.3367	6.234	6.2497	6.1602	29.8602	43.1829	2.0964	0.6569
TSP100	7.6685	8.1672	7.8627	8.4472	8.0831	8.2631	7.9750	261.2949	275.8855	6.5032	2.5324
TSP200	10.9946	12.0719	11.7334	12.3888	12.1438	12.2386	11.9566	2004.3610	2267.1775	9.7984	6.7197

Where the TSP optimal solution is the optimal path scheme obtained by using the Concorde solver; "Time" is the average running time of 30 simulations; the deviation rate indicates the degree of deviation between the best value obtained by the algorithm simulation and the optimal solution of TSP. The specific Eq of the deviation rate is shown in Eq (6.1):

$$\text{deviation rate} = \frac{\text{simulation best solution} - \text{TSP optimal solution}}{\text{TSP optimal solution}} \times 100\% \tag{6.1}$$

It can be learned from Table 6 that when the common parameter setting values are the same, the simulation best solution, worst solution and average value obtained by GCNIACO in solving traveling salesman problem are a lot better than ACO. However, the proposed graph convolutional network improved ant colony optimization also exhibits a no free lunch theorem effect [50], that is, when the algorithm improves the solving performance of one aspect of the problem, the solving performance on another hand is bound to decrease [51]. GCNIACO improves the ability to find the shortest path while making the program run time longer.

For the 20 city-scale traveling salesman problem, both GCNIACO and ACO can find the same optimal solution as the Concorde solver. But the worst solution and average value obtained by GCNIACO simulation are 0.1365 and 0.0546 less than ACO respectively. For the traveling salesman problem of 50, 100, and 200 cities, although neither GCNIACO nor ACO finds the same or smaller path than the optimal solution of TSP, GCNIACO still shows better performance than ACO. The best solution and average value obtained by GCNIACO for TSP50 simulation are respectively 0.087 and 0.0895 smaller than ACO, and the deviation rate is also 1.4395% smaller. The best solution and average value obtained by GCNIACO for TSP200 simulation are respectively 0.3385 and 0.282 smaller than ACO, and the deviation rate is also 3.0787% smaller. For the traveling salesman problem of 100 cities, the worst solution obtained by GCNIACO simulation is even 0.0841 less than the best solution of ACO simulation, and the deviation rate is 3.9708% smaller. Therefore, GCNIACO does have better solution performance than ACO. In order to more intuitively reflect the solution advantages of GCNIACO, the paper presents the optimization paths and optimization curves of two algorithms for solving TSP20, TSP50, TSP100 and TSP200, as shown in Figures 4–15 respectively.

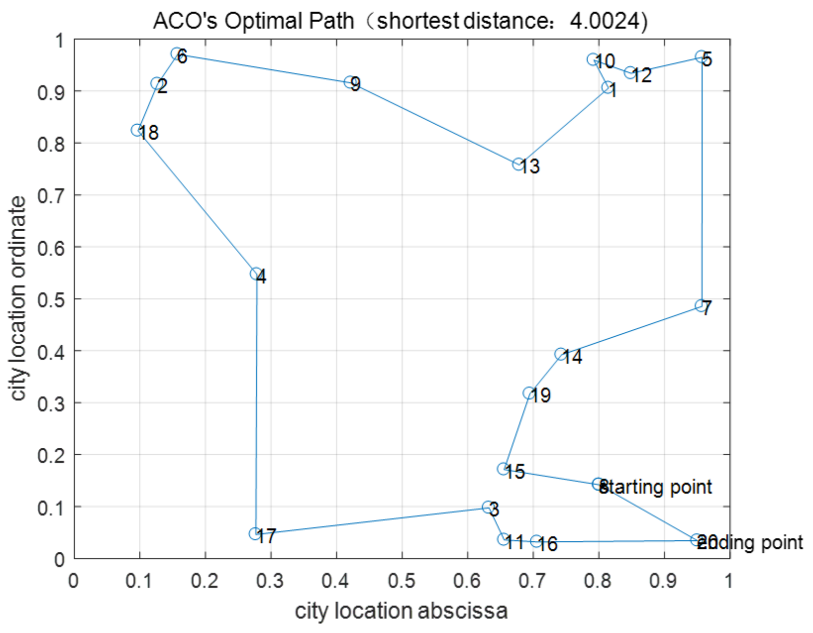

Figure 4. ACO's optimal path to solve TSP20.

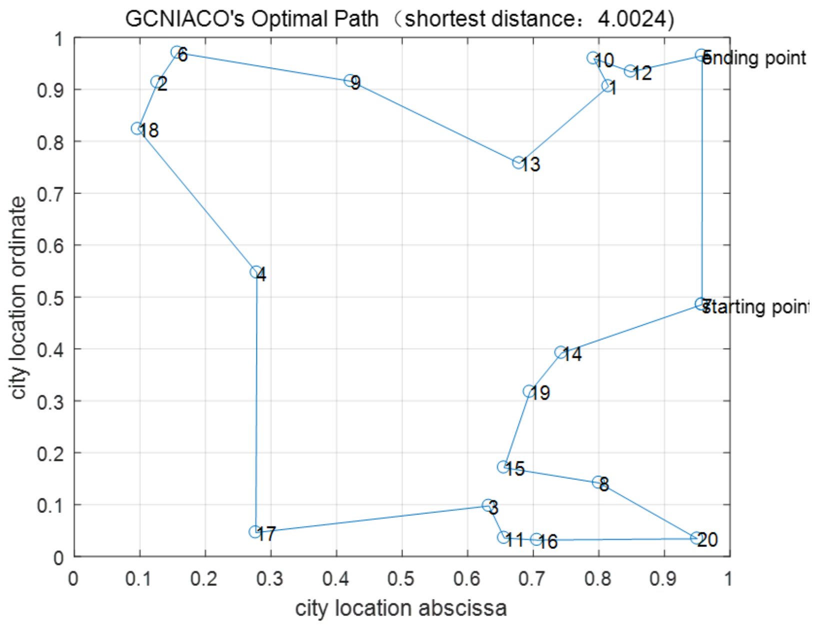

Figure 5. GCNIACO's optimal path to solve TSP20.

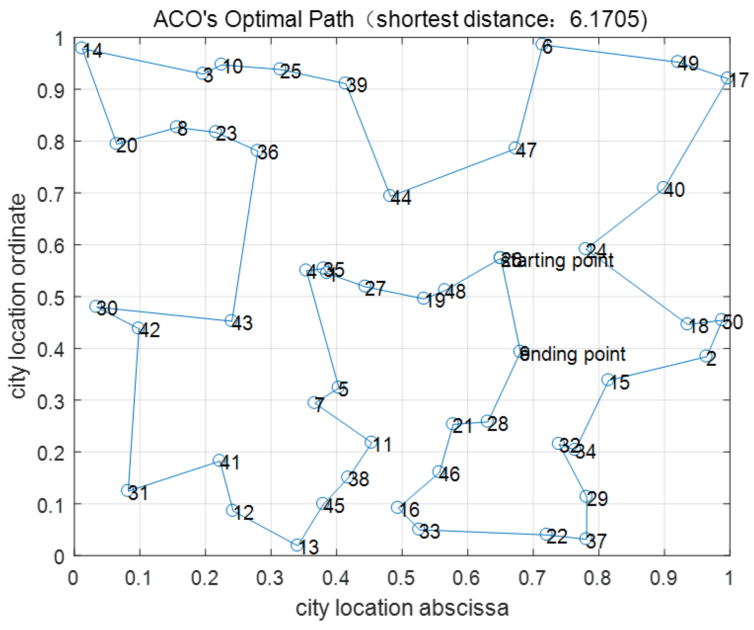

Figure 6. ACO's best path to solve TSP50.

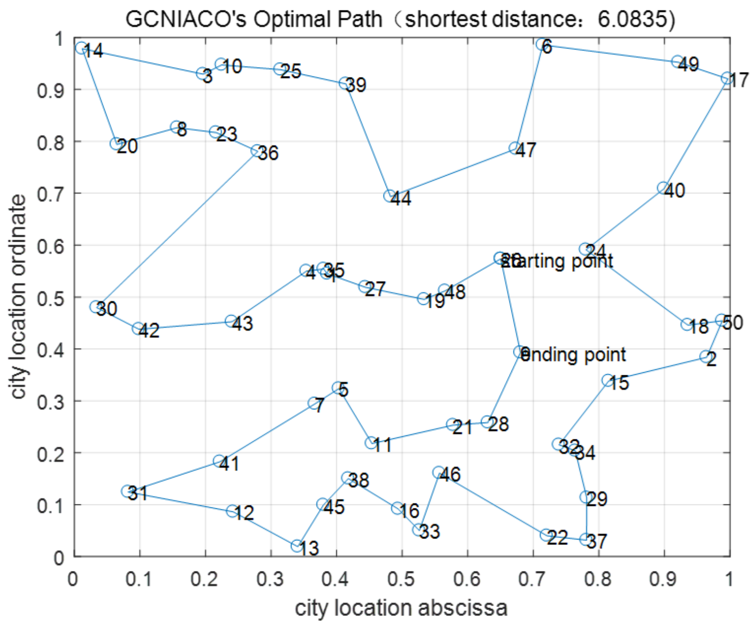

Figure 7. GCNIACO's best path to solve TSP50.

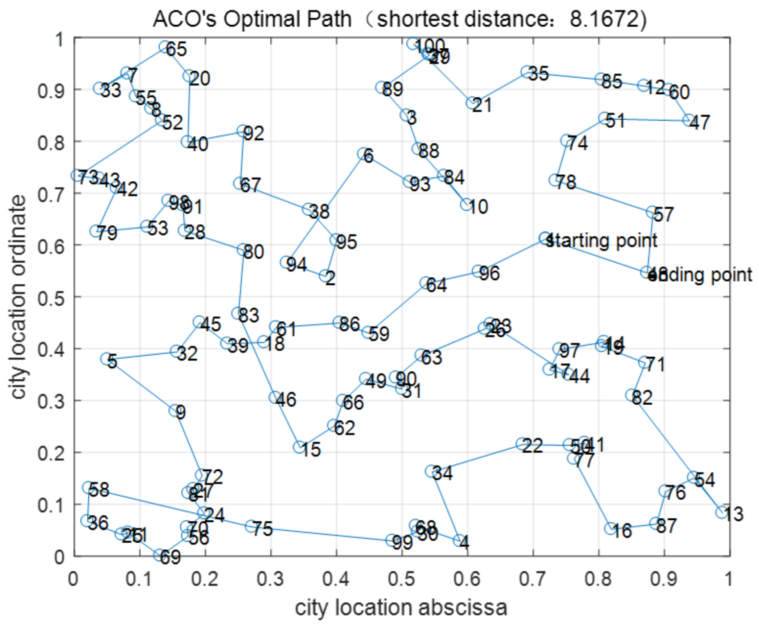

Figure 8. ACO's best path to solve TSP100.

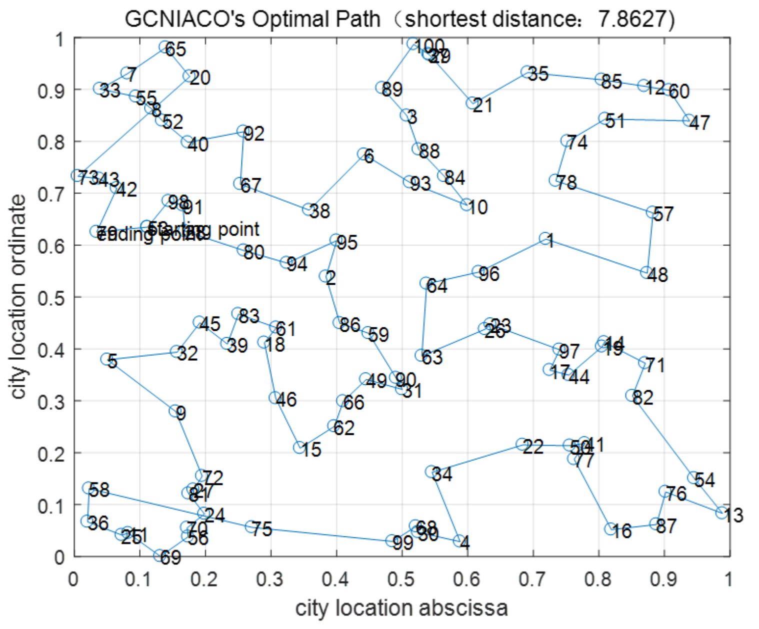

Figure 9. GCNIACO's best path to solve TSP100.

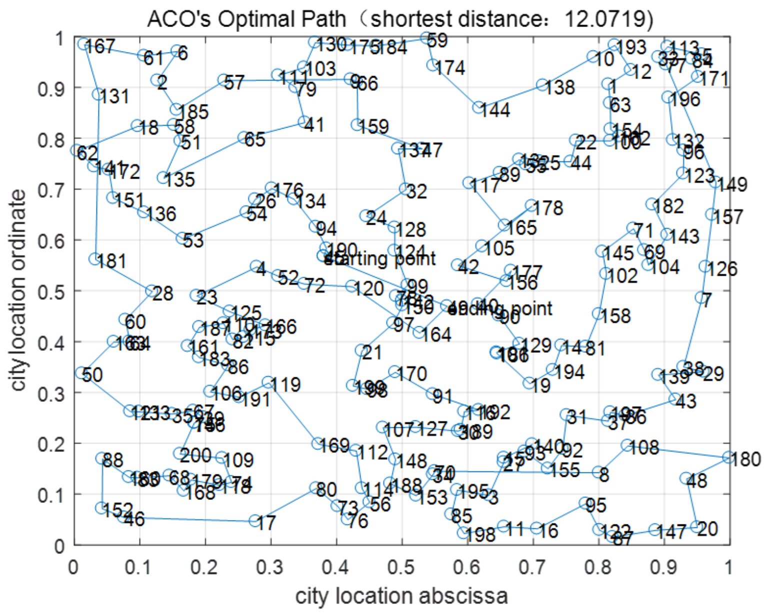

Figure 10. ACO's best path to solve TSP200.

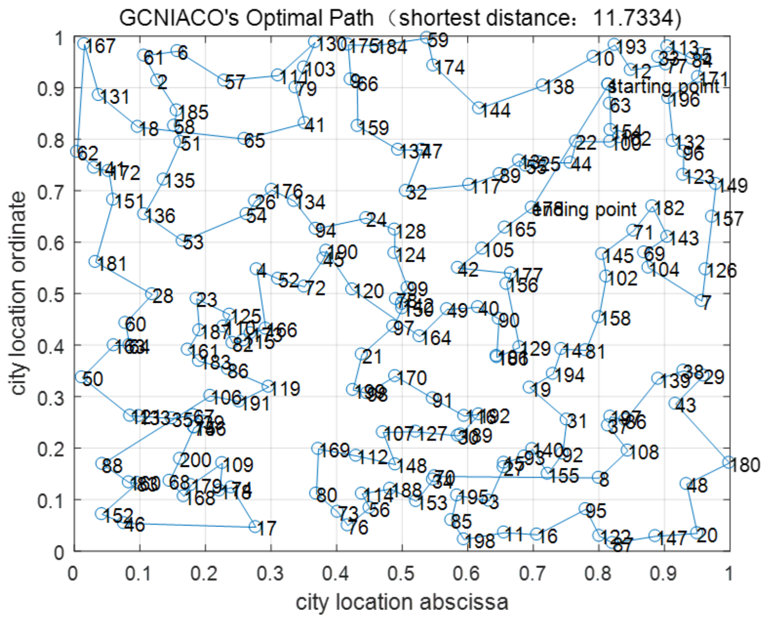

Figure 11. GCNIACO's best path to solve TSP200.

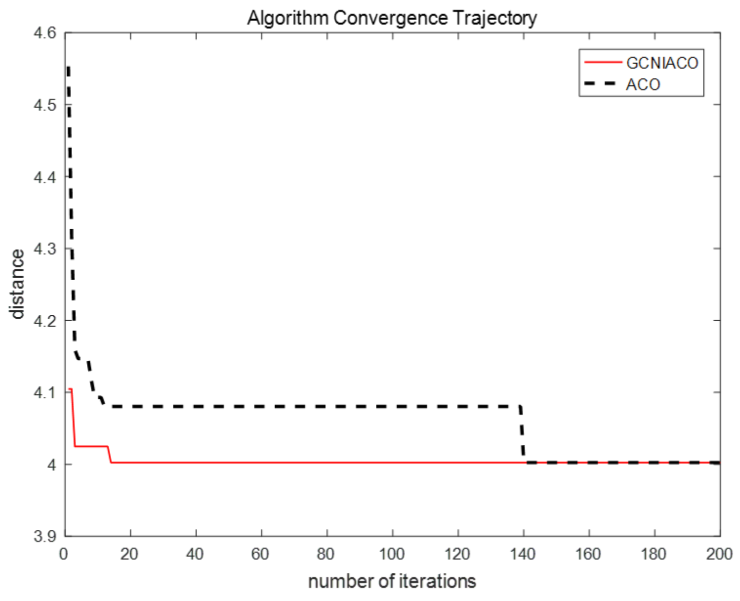

Figure 12. The optimization curve of TSP20.

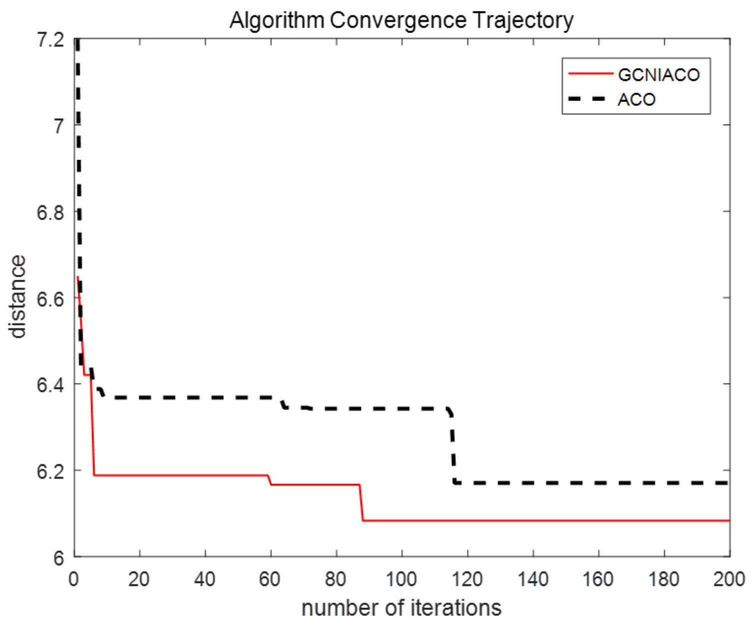

Figure 13. The optimization curve of TSP50.

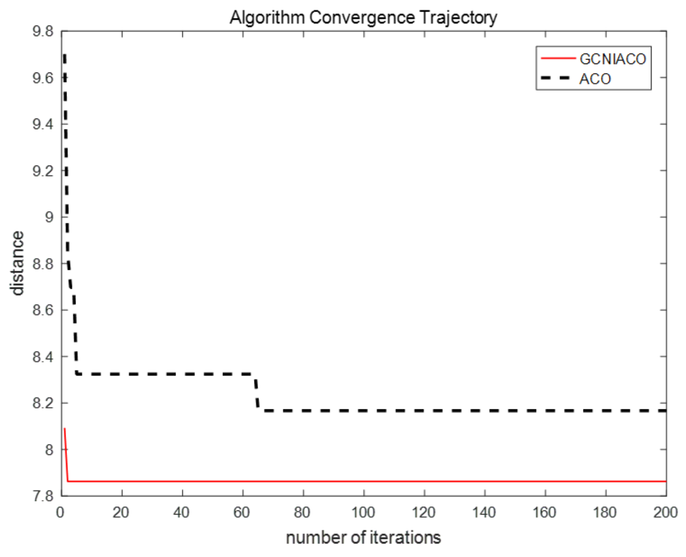

Figure 14. The optimization curve of TSP100.

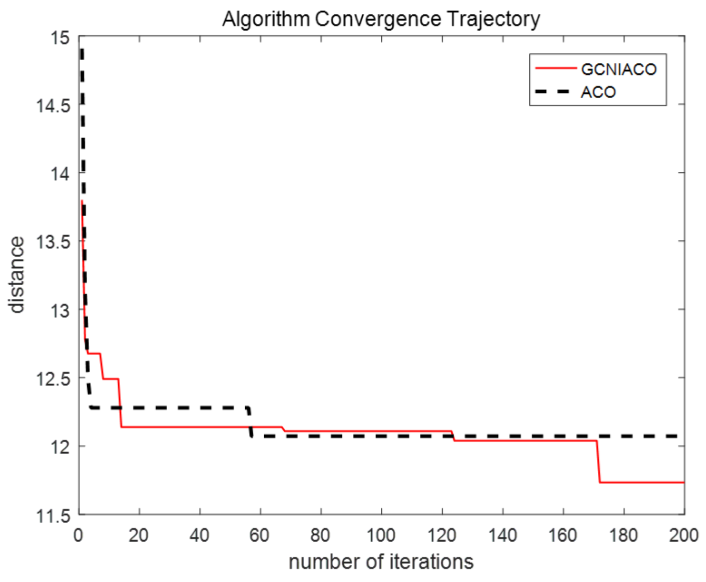

Figure 15. The optimization curve of TSP200.

The simulation results of GCNIACO, genetic algorithm (GA) and simulated annealing algorithm (SA) on the random TSP maps are shown in Table 7. The parameters in GA are set according to the literature [53]; the parameters in SA are set according to the literature [54]. Each algorithm is simulated and run 30 times.

Table 7. Simulation results of GCNIACO, GA and SA on random TSP maps.

TSP Data	GCNIACO		GA		SA	
	Min	Mean	Min	Mean	Min	Mean
TSP20	4.0024	4.0024	4.0024	4.0024	4.0024	4.0137
TSP50	6.0835	6.1602	6.0895	6.1907	6.2888	6.9454
TSP100	7.8627	7.9750	7.8618	8.2427	9.7904	10.6192
TSP200	11.7334	11.9566	13.2943	14.2418	17.4146	19.4999

As can be seen from the above table, GCNIACO exhibits better performance than GA and SA on all four random TSP maps. For the 20 city-scale traveling salesman problem, GCNIACO and GA have similar solving performance, but outperform SA in terms of mean. For the traveling salesman problem of 50 city-scale and 200 city-scale, GCNIACO outperforms GA and SA in terms of both min and mean. For the 100 city-scale traveling salesman problem, the solution performance of GCNIACO is better than SA. Although GCNIACO fails to find a best value smaller than GA in 30 simulation experiments, it still outperforms GA in terms of mean.

6.4. Improvements on the classic TSP dataset

The coordinates of the city nodes used in the above experimental simulations are all generated by random sampling. The following selects and uses five classic data sets (lin105, ch130, ch150, kroA200, pr264) in the TSPLIB standard library [52] to conduct GCNIACO simulation tests. Then it is compared and analyzed with the optimization results of ant colony optimization (ACO), genetic algorithm (GA) and simulated annealing algorithm (SA). During simulation, the parameter settings in GCNIACO are the same as in Section 6.1; the parameter settings in ACO are basically the same as those in GCNIACO, and the default parameter is set to $\rho = 0.5$; the parameters in GA are set according to literature [53]; the parameters in SA are set according to literature [54]. The results of 30 simulations are demonstrated in Table 8:

Table 8. 30 simulation results of TSP classic data set.

TSP Dataset	ACO		GA		SA		GCNIACO	
	Min	Mean	Min	Mean	Min	Mean	Min	Mean
lin105	14830.6866	15073.5580	14786.4110	15512.8010	18556.4131	21340.0438	**14514.1671**	14659.4311
ch130	6324.6628	6443.1325	6558.5385	6813.1315	8084.2883	9048.3802	**6283.4869**	6301.0280
ch150	6764.5732	6818.0849	7180.8975	7654.0091	9161.5215	10357.5391	**6623.6822**	6669.7449
kroA200	31292.036	31653.9286	36321.2295	38525.2332	49946.3911	55515.0597	**30460.4066**	30811.9496
pr264	52968.3627	53615.1862	61883.9618	65786.1819	143192.5697	178424.9334	**52420.9296**	53253.7852

According to Table 8, the graph convolutional network improved ant colony optimization proposed in the thesis shows good solution performance on the five classic TSP datasets. In 30 simulation tests, the best path shorter than ACO [48], GA [53] and SA [54] can be found. The average value of 30 simulation solutions is also better than the above algorithm, which verifies the improvement effect of the algorithm.

7. Statistical analysis

To confirm that the observed differences between the original and improved versions are actually meaningful, this section uses two statistical analysis methods to evaluate the differences between the GCNIACO, ACO, GA, and SA. Referring to literature [55] and literature [11], the paper first chooses to use Friedman test among nonparametric tests to verify whether there is a significant difference between the algorithms. The Friedman test is to rank each algorithm, and the algorithm with better performance has a smaller ranking value, that is, the best performing algorithm is ranked 1, the second best is ranked 2, and so on. In the event of a tie, the average ranking is calculated [56]. Under the null hypothesis, this hypothesis states that all algorithms behave similarly. When there is a significant difference between the algorithms, the null hypothesis is rejected. The paper uses SPSS software to perform Friedman test on Tables 6–8, and the rank average of each algorithm is shown in Table 9. Among them, the calculated Friedman statistic for 3 degrees of freedom (according to the χ^2 distribution) is 22.0112. And within the 95% confidence interval, the critical value of the χ^2 distribution is 7.81. 22.0112 is greater than 7.81, and the observed p-value is 6.5E-5 (< 0.05), from which it can be seen that GCNIACO is the best algorithm with the smallest rank among the compared algorithms.

In addition, post hoc analysis using Holm test is also chosen to evaluate the significance of GCNIACO's better statistical performance. GCNIACO is the control algorithm, and the null hypothesis holds that these algorithms are equivalent. Tables 10 shows the unadjusted and adjusted p-values obtained using the Holm test. Since SPSS software is not convenient for scientific notation here, it is uniformly accurate to three decimal places, which does not affect the analysis of results. Analysis of the data shows that all p-values are less than 0.05, so the null hypothesis is rejected and it can be considered that GCNIACO outperforms ACO, GA and SA at the 95% confidence level.

Table 9. Friedman nonparametric test.

Algorithms	Average Ranks
GCNIACO	1.06
ACO	2.44
GA	2.61
SA	3.89

Table 10. Holm test.

Algorithms	Unadjusted p-value	Adjusted p-value
ACO	0.022	0.022
GA	0.011	0.022
SA	0.000	0.000

8. Engineering case

The above content introduces the proposed graph convolutional network improved ant colony optimization, and verifies the effectiveness of the improved algorithm on some TSP datasets. In this section, the graph convolutional network improved ant colony optimization is applied to solve the vehicle routing problem (VRP) in engineering applications. And the solution result will be compared with the basic ant colony optimization to verify the practicality of the improved algorithm.

8.1. Vehicle routing problem

The vehicle routing problem (VRP) [57] was proposed by Dantzig and Ramser in 1959, and it is still at the forefront of combinatorial optimization research. In order to facilitate the establishment of the model, it is assumed that there is only one distribution center and the location coordinates are known; all distribution vehicles start from the distribution center and return to the distribution center in turn after delivery is completed. The rated load of the distribution vehicle is known, the demand at the demand point is known, and each demand point has one and only one vehicle responsible for distribution [58]. According to the above assumptions and reference [59], a model aiming at the lowest distribution cost is established:
Minimize:

$$Z = c_0 m' + \sum_{i=0}^{L}\sum_{j=0}^{L}\sum_{k=1}^{m'} c d_{ij} x_{ijk} + c_1 \sum_{k=1}^{m'} (\max(\sum_{i=1}^{L}(g_i y_{ki} - Q), 0)) \tag{8.1}$$

Subject to

$$\sum_{i=1}^{L} g_i y_{ki} \le Q,\ k \in [1, m'] \tag{8.2}$$

$$\sum_{k=1}^{m'} y_{ki} = 1,\ i \in [0, L] \tag{8.3}$$

$$\sum_{i=0}^{L}\sum_{k=1}^{m'} x_{ijk} = 1,\ j \in [0, L] \tag{8.4}$$

$$\sum_{j=0}^{L}\sum_{k=1}^{m'} x_{ijk} = 1,\ i \in [0, L] \tag{8.5}$$

$$\sum_{j=0}^{L}\sum_{k=1}^{m'} x_{0jk} = \sum_{i=0}^{L}\sum_{k=1}^{m'} x_{i0k} \tag{8.6}$$

where Z represents the distribution cost; c_0 is the unit cost of the vehicle; m' is the count of vehicles; L is the count of demand points; c is the unit cost of the distance traveled by the vehicle; $d_{ij}(i, j = 1, 2, ..., L)$ is the distance between the demand point i and the demand point j; c_1 is the overload penalty coefficient; g_i is the demand of the demand point i; Q is the rated load of the vehicle; $k(k = 1, 2, ..., m')$ is the vehicle number. The distribution center number is 0, the demand points number are $1, 2, ...,$ and the variables x_{ijk} and y_{ki} are respectively defined as:

$$x_{ijk} = \begin{cases} 1, & \text{vehicle k travels from point i to point j} \\ 0, & \text{otherwise} \end{cases} \tag{8.7}$$

$$y_{ki} = \begin{cases} 1, & \text{the delivery task of demand point i is completed by vehicle k} \\ 0, & \text{otherwise} \end{cases} \tag{8.8}$$

Equation (8.1) is the objective function of the model. Constraint (8.2) indicates that each vehicle load does not exceed the rated load. Equations (8.3)–(8.5) indicate that only one vehicle is allowed to make one delivery at each demand point. Equation (8.6) indicates that the vehicles all start from the distribution center and finally return to the distribution center.

8.2. Solving steps

Step 1: initialize the coefficients related to the ant colony optimization and calculate the distance matrix between nodes;

Step 2: initialize the graph convolutional network and generate the better solution that meets the vehicle load requirements; then convert the better solution into the ant colony initial pheromone through the pheromone conversion strategy;

Step 3: put m ants in the distribution center, each ant selects the next demand point according to Eq (4.1), and records it in the path record table; then judge whether the vehicle reaches the rated load, if so, return to the distribution center, and insert the serial number of the distribution center into the path record table; repeat this process until all the ants have visited all demand points;

Step 4: calculate the distribution cost Z_k corresponding to each ant in the path record table;

Step 5: the 3-opt algorithm is introduced to perform de-crossing operation on the best path in the path record table, and at the same time, it is judged whether the de-crossing operation meets the vehicle load requirements, if not, the de-crossing operation is cancelled; if the optimized distribution cost $Z_{3\text{-}opt}$ is less than the best cost $\min(Z_k)$, use $Z_{3\text{-}opt}$ to update $\min(Z_k)$, and at the same time update the corresponding path node sequence in the path record table, if $Z_{3\text{-}opt}$ is larger than $\min(Z_k)$, directly jump to Step 6;

Step 6: record the best solution in the current iteration number;

Step 7: according to Eqs (4.2)–(4.4), update the pheromone on the path, wherein the pheromone volatility factor ρ is calculated according to Eq (5.10); then clear the path record table, and jump to Step 3 to continue the iteration;

Step 8: when the algorithm reaches the specified maximum number of iterations NC_max, stop the iteration and output the best solution.

8.3. Simulation solution

The case data such as distribution center coordinates, demand point coordinates, and demand point demand used in this paper are all from the R101 case in http://web.cba.neu.edu/ msolomon/r101.htm. It is set that the distribution center needs to deliver to 100 demand points. The rated load of the distribution vehicle is set to 500. In the simulation, let $c_0 = 0$, $c = 1$, $c_1 = \infty$, then the distribution cost is only related to the distance traveled by the vehicle. The parameter settings of GCNIACO and ACO are basically the same as those in 6.1, and the default parameter is set to $\rho = 0.5$. In order to ensure the validity and reliability of the results, each algorithm is independently run 30 times during the simulation, and the solution results are shown in Table 11:

Table 11. VRP solution results.

		Methods	
		ACO	GCNIACO
Best		740.0967	725.9472
Mean		753.9658	743.6432
Worst		767.8407	755.6621
Count of Vehicles		3	3
Best Running Route	Vehicle 1	0-53-58-40-21-73-72-74-22-41-75-56-23-67-39-25-55-4-54-80-68-77-3-79-33-81-9-51-20-30-70-31-88-0	0-53-58-40-21-73-72-74-22-75-56-39-4-54-80-68-77-3-79-33-81-9-51-20-30-70-31-88-7-48-82-0
	Vehicle 2	0-27-28-26-12-76-50-1-69-52-18-60-83-84-5-99-96-94-95-97-92-59-93-85-91-100-37-98-61-16-44-14-42-87-0	0-27-69-1-50-76-12-26-28-89-6-94-95-97-92-59-96-99-93-85-91-100-37-98-61-16-86-38-44-14-43-42-87-0
	Vehicle 3	0-89-6-13-2-57-15-43-38-86-17-45-8-46-47-36-49-64-11-19-7-82-48-62-10-90-63-32-66-71-65-35-34-78-29-24-0	0-13-2-57-15-41-23-67-25-55-24-29-78-34-35-71-65-66-32-90-63-10-62-19-11-64-49-36-47-46-8-45-17-84-5-60-83-18-52-0

where 0 is the number of the distribution center, and $1, 2, ...$ are the number of demand points.

As can be seen from the above table, compared with ACO, GCNIACO can find the best value with lower distribution cost when solving VRP, and also outperforms ACO in terms of average and worst value. The best path diagrams and optimization curves in Figures 16–18 more intuitively demonstrate the solution advantages of GCNIACO.

9. Conclusions

In this study, a graph convolutional network improved ant colony optimization is proposed. By introducing graph convolutional network, dynamically adjusting pheromone volatile factor and introducing 3-opt algorithm, the deficiencies of ACO in solving the TSP are made up. The study combines graph convolutional network to improve ant colony optimization, which provides an idea for the combination of machine learning and swarm intelligence. At the same time, the generalized graph convolutional network model is selected, and the corresponding graph convolutional network does not need to be trained for each specific city node scale, which reduces operating costs and saves computing resources to a certain extent. And it also makes the proposed algorithm have certain scalability. The simulation test results of the graph convolutional network improved ant colony optimization on TSP datasets and VRP engineering application example verify the excellent performance of the improved algorithm in seeking the optimum solution. GCNIACO is a metaheuristic algorithm based on swarm intelligence. It can be applied to optimization problems in various fields, such as biological sequence alignment, digital microfluidic biochip contamination fault removal, molecular spectrum wavelength selection, molecular docking, epistasis detection of genome-wide association, etc. Moreover, GCNIACO, as an improved algorithm of basic ant colony optimization, may obtain relatively better results when solving optimization problems.

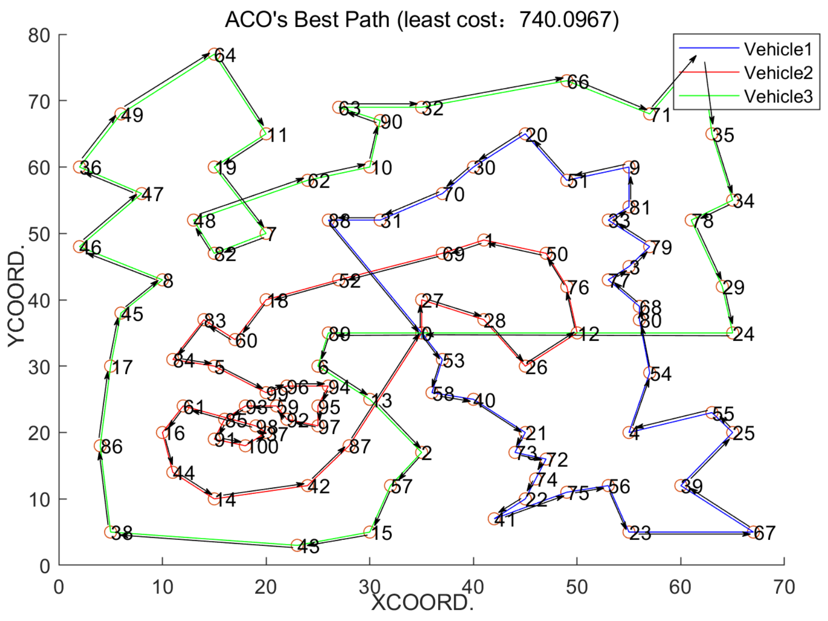

Figure 16. The running route of ACO to solve the VRP.

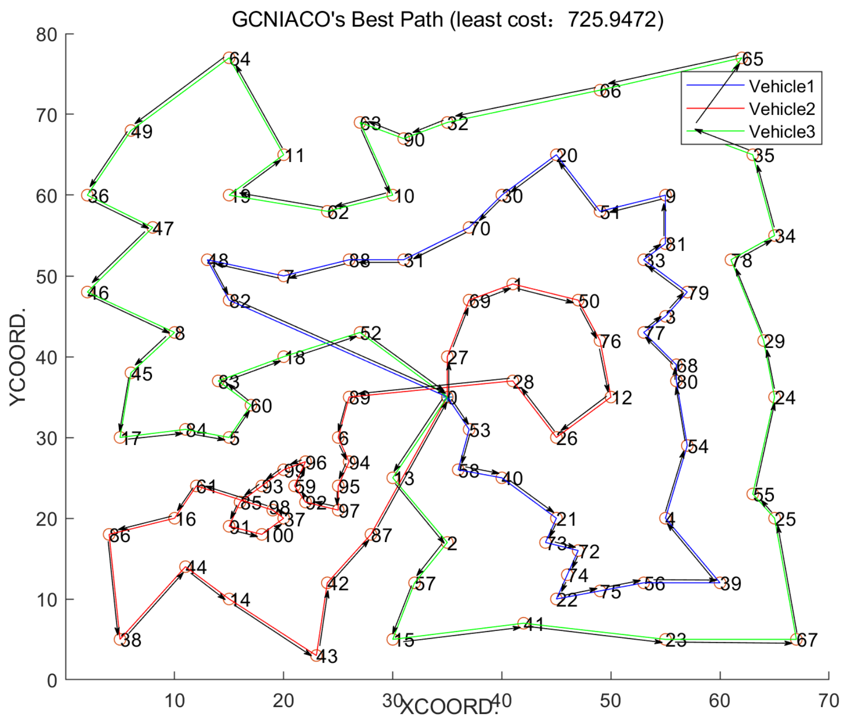

Figure 17. The running route of GCNIACO to solve the VRP.

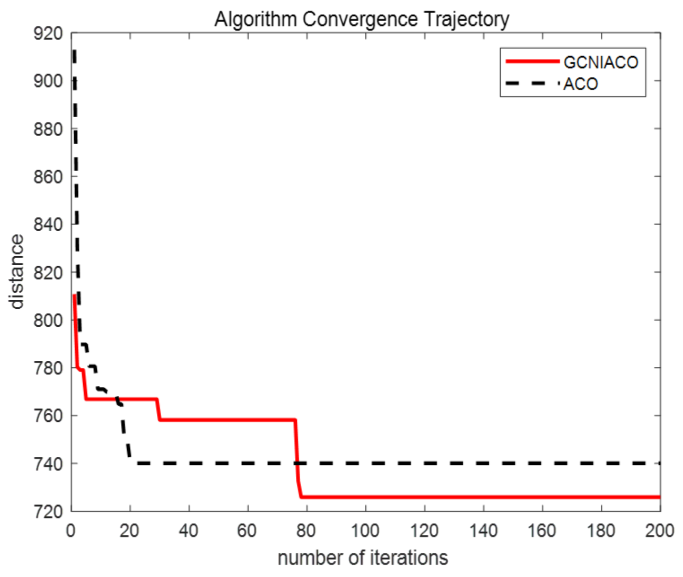

Figure 18. The optimization curve diagram of minimum distribution cost.

In the study, only the pheromone volatility factor in the ant colony optimization is adaptively improved. Therefore, future work can also consider improving other parameters of the ant colony optimization into an adaptive dynamic adjustment method. In addition to combining with graph convolutional network, combining ant colony optimization with other recently proposed algorithms is also one of the future study directions. Applying the improved algorithm to solve practical issues in real life, such as logistics distribution center location problem, robot path planning problem, etc., is also a direction worth considering for future study.

Acknowledgments

The study is subsidized by Tianjin Natural Science Foundation (No. 20JCYBJC00320), National Natural Science Foundation of China (No. 61401307) and Tianjin Postgraduate Research Innovation Project (No. 2021YJSS277).

Conflict of interest

The authors declare there is no conflict of interest.

References

1. R. Laborda, Optimal combination of currency strategies, *North Am. J. Econ. Finance*, **43** (2018), 129–140. https://doi.org/10.1016/j.najef.2017.10.010
2. S. Singh, K. C. Tiwari, Exploring the optimal combination of image fusion and classification techniques, *Remote Sens. Appl.: Soc. Environ.*, **24** (2021), 100642. https://doi.org/10.1016/j.rsase.2021.100642

3. J. H. Kim, I. Park, S. P. Chung, H. Y. Kim, I. K. Min, S. J. Kim, et al., Optimal combination of clinical examinations for neurologic prognostication of out-of-hospital cardiac arrest patients, *Resuscitation*, **155** (2020), 91–99. https://doi.org/10.1016/j.resuscitation.2020.07.014

4. X. L. Qin, Z. X. Liu, L. Tian, The optimal combination between selling mode and logistics service strategy in an e-commerce market, *Eur. J. Oper. Res.*, **289** (2021), 639–651. https://doi.org/10.1016/j.ejor.2020.07.029

5. G. H. David, A. A. Antonio, E. Molina, A Combinatorial model to optimize air traffic flow management problems, *Comput. Oper. Res.*, **112** (2019), 104768. https://doi.org/10.1016/j.cor.2019.104768

6. T. András, S. S. Maricruz, L. Végvári, G. P. Szijjártó, J. L. Margitfalvi, A. Trunschke, et al., Combinatorial optimization and synthesis of multiple promoted MoVNbTe catalysts for oxidation of propane to acrylic acid, *Catal. Today*, **363** (2021), 45–54. https://doi.org/10.1016/j.cattod.2019.03.047

7. S. Bharati, P. Podder, M. R. H. Mondal, Hybrid deep learning for detecting lung diseases from X-ray images, *Inf. Med. Unlocked*, **20** (2020), 100391. https://doi.org/10.1016/j.imu.2020.100391

8. G. Laporte, The traveling salesman problem: an overview of exact and approximate algorithms, *Eur. J. Oper. Res.*, **59** (1992), 231–247. https://doi.org/10.1016/0377-2217(92)90138-Y

9. A. Colorni, M. Dorigo, V. Maniezzo, Distributed optimization by ant colonies, in *Proceedings of the first European conference on artificial life*, Elsevier Publishing, (1991), 134–142.

10. M. Gunduz, M. Aslan, DJAYA: A discrete Jaya algorithm for solving traveling salesman problem, *Appl. Soft Comput.*, **105** (2021), 107275. https://doi.org/10.1016/j.asoc.2021.107275

11. K. Panwar, K. Deep, Discrete Grey Wolf Optimizer for symmetric travelling salesman problem, *Appl. Soft Comput.*, **105** (2021), 107298. https://doi.org/10.1016/j.asoc.2021.107298

12. S. K. R. Kanna, K. Sivakumar, N. Lingaraj, Development of Deer Hunting linked Earthworm Optimization Algorithm for solving large scale Traveling Salesman Problem, *Knowledge-Based Syst.*, **227** (2021), 107199. https://doi.org/10.1016/j.knosys.2021.107199

13. M. M. Krishna, N. Panda, S. K. Majhi, Solving traveling salesman problem using hybridization of rider optimization and spotted hyena optimization algorithm, *Expert Syst. Appl.*, **183** (2021), 115353. https://doi.org/10.1016/j.eswa.2021.115353

14. Y. Saji, M. Barkatou, A discrete bat algorithm based on Lévy flights for Euclidean traveling salesman problem, *Expert Syst. Appl.*, **172** (2021), 114639. https://doi.org/10.1016/j.eswa.2021.114639

15. G. H. Al-Gaphari, R. Al-Amry, A. S. Al-Nuzaili, Discrete crow-inspired algorithms for traveling salesman problem, *Eng. Appl. Artif. Intell.*, **97** (2021), 104006. https://doi.org/10.1016/j.engappai.2020.104006

16. Y. Huang, X. N. Shen, X. You, A discrete shuffled frog-leaping algorithm based on heuristic information for traveling salesman problem, *Appl. Soft Comput.*, **102** (2021), 107085. https://doi.org/10.1016/j.asoc.2021.107085

17. I. M. Ali, D. Essam, K. Kasmarik, A novel design of differential evolution for solving discrete traveling salesman problems, *Swarm Evol. Comput.*, **52** (2020), 100607. https://doi.org/10.1016/j.swevo.2019.100607

18. M. A. H. Akhand, S. I. Ayon, S. A. Shahriyar, N. Siddique, H. Adeli, Discrete spider monkey optimization for travelling salesman problem, *Appl. Soft Comput.*, **86** (2020), 105887. https://doi.org/10.1016/j.asoc.2019.105887

19. M. Deudon, P. Cournut, A. Lacoste, Y. Adulyasak, L. M. Rousseau, Learning heuristics for the TSP by policy gradient, in *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, Springer, Cham, **10848** (2018), 170–181. https://doi.org/10.1007/978-3-319-93031-2_12

20. M. O. R. Prates, P. H. C. Avelar, H. Lemos, L. Lamb, M. Vardi, Learning to solve np-complete problems: A graph neural network for decision tsp, in *Proceedings of the AAAI Conference on Artificial Intelligence*, **33** (2019), 4731–4738.

21. W. Kool, H. V. Hoof, M. Welling, Attention, learn to solve routing problems!, preprint, arXiv: 1803.08475.

22. Y. J. Hu, Z. Zhang, Y. Yao, X. P. Huyan, X. S. Zhou, W. S. Lee, A bidirectional graph neural network for traveling salesman problems on arbitrary symmetric graphs, *Eng. Appl. Artif. Intell.*, **97** (2021), 104061. https://doi.org/10.1016/j.engappai.2020.104061

23. A. Ragmani, A. Elomri, N. Abghour, K. Moussaid, M. Rida, An improved hybrid fuzzy-ant colony algorithm applied to load balancing in cloud computing environment, *Procedia Comput. Sci.*, **151** (2019), 519–526. https://doi.org/10.1016/j.procs.2019.04.070

24. S. Ebadinezhad, DEACO: adopting dynamic evaporation strategy to enhance ACO algorithm for the traveling salesman problem, *Eng. Appl. Artif. Intell.*, **92** (2020), 103649. https://doi.org/10.1016/j.engappai.2020.103649

25. J. Li, Y. Xia, B. Li, Z. G. Zeng, A pseudo-dynamic search ant colony optimization algorithm with improved negative feedback mechanism, *Cognit. Syst. Res.*, **62** (2020), 1–9. https://doi.org/10.1016/j.cogsys.2020.03.001

26. A. F. Tuani, E. Keedwell, M. Collett, Heterogenous Adaptive Ant Colony Optimization with 3-opt local search for the Travelling Salesman Problem, *Appl. Soft Comput.*, **97** (2020), 106720. https://doi.org/10.1016/j.asoc.2020.106720

27. J. Y. Zheng, X. Q. Cheng, J. J. Fu, Application research of improved ant colony algorithm in TSP, *Comput. Simul.*, **38** (2021), 126–130+167.

28. M. L. Li, Q. Z. Li, Path Planning of Unmanned Crane Based on Improved Ant Colony Algorithm, *Comput. Simul.*, **38** (2021), 172–176+226.

29. X. H. Tang, S. J. Xin, Improved ant colony algorithm for mobile robot path planning, *Comput. Eng. Appl.*, in press.

30. C. Liu, L. Wu, X. D. Huang, W. S. Xiao, Improved dynamic adaptive ant colony optimization algorithm to solve pipe routing design, *Knowledge-Based Syst.*, **237** (2022), 107846. https://doi.org/10.1016/j.knosys.2021.107846

31. L. W. Yang, L. X. Fu, N. Guo, Z. Yang, H. Q. Guo, X. Y. Xu, Path planning with multi-factor improved ant colony algorithm, *Comput. Integr. Manuf. Syst.*, in press.

32. M. L. He, Z. X. Wei, X. H. Wu, Y. T. Peng, An improved ant colony optimization algorithm for vehicle routing problem with soft time windows, *Comput. Integr. Manuf. Syst.*, in press.

33. S. B. Wang, R. Hu, B. Qian, M. Y. Liu, Improved Ant Colony Optimization for Solving Green Periodic Vehicle Routing Problem, *Control Eng. China*, in press. https://doi.org/10.14107/j.cnki.kzgc.20200581

34. A. C. Cinar, S. Korkmaz, M. S. Kiran, A discrete tree-seed algorithm for solving symmetric traveling salesman problem, *Eng. Sci. Technol. Int. J.*, **23** (2020), 879–890. https://doi.org/10.1016/j.jestch.2019.11.005

35. G. Campuzano, C. Obreque, M. M. Aguayo, Accelerating the Miller–Tucker–Zemlin model for the asymmetric traveling salesman problem, *Expert Syst. Appl.*, **148** (2020), 113229. https://doi.org/10.1016/j.eswa.2020.113229

36. H. P. Hipólito, S. G. Juan-José, A Branch-and-cut algorithm for the split-demand one-commodity pickup-and-delivery travelling salesman problem, *Eur. J. Oper. Res.*, **297** (2022), 467–483. https://doi.org/10.1016/j.ejor.2021.05.040

37. O. Cheikhrouhou, I. Khoufi, A comprehensive survey on the multiple traveling salesman problem: Applications, approaches and taxonomy, *Comput. Sci. Rev.*, **40** (2021), 100369. https://doi.org/10.1016/j.cosrev.2021.100369

38. M. Cornu, T. Cazenave, D. Vanderpooten, Perturbed decomposition algorithm applied to the multi-objective traveling salesman problem, *Comput. Oper. Res.*, **79** (2017), 314–330. https://doi.org/10.1016/j.cor.2016.04.025

39. H. B. Duan, *The Principle and Application of Ant Colony Algorithm*, Science Press, Beijing, 2005.

40. L. Ma, G. Zhu, A. B. Ning, *Ant Colony Optimization Algorithm*, Science Press, Beijing, 2007.

41. J. W. Zhuo, B. W. Li, Y. S. Wei, J. Qin, *Application of MATLAB in Mathematical Modeling*, Beihang University Press, Beijing, 2014.

42. X. Bresson, T. Laurent, Residual gated graph convNets, preprint, arXiv: 1711.07553. https://doi.org/10.48550/arXiv.1711.07553

43. K. J. Chaitanya, T. Laurent, X. Bresson, An efficient graph convolutional network technique for the travelling salesman problem, preprint, arXiv: 1906.01227. https://doi.org/10.48550/arXiv.1906.01227

44. Z. Yang, H. Zhou, L. Q. Zhu, W. Li, Chemical reaction ant colony optimization algorithm, *Appl. Res. Comput.*, **31** (2014), 2925–2927+2946.

45. X. H. Zhong, On the approximation ratio of the 3-Opt algorithm for the (1,2)-TSP, *Oper. Res. Lett.*, **49** (2021), 515–521. https://doi.org/10.1016/j.orl.2021.05.012

46. D. L. Applegate, R. E. Bixby, V. Chvatal, W. J. Cook, *The Traveling Salesman Problem: A Computational Study*, Princeton university press, Princeton, 2006.

47. D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, preprint, arXiv: 1412.6980. https://doi.org/10.48550/arXiv.1412.6980

48. Z. W. Ye, Z. B. Zheng, Configuration of Parameters α, β, ρ, in ant algorithm, *Geomatics Inf. Sci. Wuhan Univ.*, **29** (2004), 597–601.

49. T. R. Zhang, B. K. Wu, F. Q. Zhou, Research on improved ant colony algorithm for robot global path planning, *Comput. Eng. Appl.*, **58** (2022), 282–291.

50. D. H. Wolpert, W. G. Macready, No free lunch theorems for optimization, *IEEE Trans. Evol. Comput.*, **1** (1997), 67–82. https://doi.org/10.1109/4235.585893

51. S. Bharati, P. Podder, M. R. H. Mondal, N. Gandhi, Optimized NASNet for Diagnosis of COVID-19 from Lung CT Images, *Intell. Syst. Design Appl.*, **1351** (2021), 647–656. https://doi.org/10.1007/978-3-030-71187-0_59

52. TSPLIB, Available from: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/ .

53. S. W. Yu, *MATLAB Optimization Algorithm Case Analysis and Application*, Tsinghua University Press, Beijing, 2014.

54. L. Yu, F. Shi, H. Wang, F. Hu, *30 Case Studies of MATLAB Intelligent Algorithms*, 2nd edition, Beijing University of Aeronautics and Astronautics Press, Beijing, 2015.

55. J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, *Swarm Evol. Comput.*, **1** (2011), 3–18. https://doi.org/10.1016/j.swevo.2011.02.002

56. A. Khamparia, S. Bharati, P. Podder, D. Gupta, A. Khanna, T. K. Phung, et al., Diagnosis of breast cancer based on modern mammography using hybrid transfer learning, *Multidimension. Syst. Signal Process.*, **32** (2021), 747–765. https://doi.org/10.1007/s11045-020-00756-7

57. G. B. Dantzig, J. H. Ramser, The truck dispatching problem, *Manage. Sci.*, **6** (1959), 80–91. https://doi.org/10.1287/mnsc.6.1.80

58. Z. F. Wang, H. L. Du, S. F. An, C. J. Zhang, An improved ant colony algorithm based on vehicle routing problem, *J. Huaqiao Univ. (Nat. Sci.)*, **34** (2013), 36–39.

59. T. Fei, L. Y. Zhang, Y. S. Sun, Solution of vehicle routing optimization problem based on DNA-ant colony algorithm, *Comput. Eng.*, **40** (2014), 206–213.