Engineering Applications of Artificial Intelligence 131 (2024) 107816

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

Discrete artificial bee colony algorithm with fixed neighborhood search for traveling salesman problem

Xing Li, Shaoping Zhang, Peng Shao *

School of Computer and Information Engineering, Jiangxi Agricultural University, Nanchang, 330045, China

ARTICLE INFO

Keywords:
Intelligent algorithm
Artificial bee colony algorithm
Traveling salesman problem
Combinatorial optimization

ABSTRACT

For the artificial bee colony algorithm (ABC), it is easy to fall into local optimum and has lower convergence accuracy when solving the traveling salesman problem. For addressing this demerit further, a discrete artificial bee colony algorithm with fixed neighborhood search for traveling salesman problem (TSP), called DABC-FNS, is proposed. In DABC-FNS, the solution obtained by the discrete artificial bee colony algorithm is expressed by positive integer coding method. Meanwhile, the local enhancement strategy and the 2-opt strategy with fixed neighborhood search are introduced to improve the solution accuracy of the ABC algorithm. In order to verify the effectiveness of the DABC-FNS algorithm, more than 30 benchmark TSP instances are simulated by DABC-FNS algorithm and other state-of -the-art competitors. The experimental results show that the DABC-FNS algorithm has achieved better accuracy for most TSP instances, which also demonstrates that it can overcome the premature phenomenon and has certain advantages in solving the traveling salesman problem.

1. Introduction

The traveling salesman problem (TSP) is a typical combinatorial optimization problem and an important problem in the field of optimization. Solving TSP refers to finding the shortest path through all nodes, which minimizes the total travel cost. Therefore, such problems usually have the characteristic that the complexity of solving the problem increases exponentially with the expansion of the problem scale. As a classic *NP*-hard problem, the TSP problem has an extremely wide range of applications in daily life, such as path planning (Wolek et al., 2021; Bai, 2020; Bandeira et al., 2015), logistics distribution (Hernandez--Perez et al., 2018; Benavent et al., 2019), and resource scheduling (Aguirre Adrián et al., 2018; Deane and White, 1975; Aguirre and Papageorgiou, 2018). Accurate algorithms are gradually difficult to solve such problems, and the development of intelligent algorithms has opened up a new situation. Therefore, the utilizing of intelligent algorithms to solve combinatorial optimization problems has developed into the mainstream research direction, and it is meaningful to make the continuous metaheuristic algorithms which have good effect in the continuous search space handle the discrete problems (Crawford et al., 2017).

The intelligent algorithms are not limited by the nature of the problem, and have high efficiency and excellence performance. Among them, artificial bee colony algorithm (ABC) was proposed by foreign scholar Karaboga (2005) in 2005. Although it was originally used to solve continuous numerical optimization problems (Karaboga and Basturk, 2007, 2008), ABC is not only easy to implement but also has excellent algorithm efficiency. In recent years, ABC has become a well-known optimization technique. Besides the standard form, there are many outstanding improved versions (Zhu and Kwong, 2010; DING et al., 2009; Wang et al., 2014; Gao et al., 2013; Cui et al., 2018). For examples, Gao et al. (2018) constructed a strategy candidate pool to obtain high-quality candidate individuals by introducing three popular search strategies with different characteristics, and then the Parzen window method was used to estimate these candidate individuals to select the best individuals as descendants, achieving the effect of reducing the amount of computation. In addition, neighborhood mechanism was employed to balance the algorithm performance. In order to improve the foraging model of ABC, Li et al. (2016) introduced a new gene recombination operator (GRO) to compensate for the lack of interaction between leaders and combine the advantages of different dominant individuals. Zeng et al. (2022) proposed an efficient variant of the ABC algorithm based on adaptive search strategy and random grouping mechanism. By monitoring the success rates of current and previous iterations, an adaptive search strategy is designed and the search strategy is modified according to random groupings. Combining

* Corresponding author.
E-mail addresses: xingli05@126.com (X. Li), pshao@whu.edu.cn (P. Shao).

https://doi.org/10.1016/j.engappai.2023.107816
Received 26 April 2022; Received in revised form 2 November 2023; Accepted 26 December 2023
Available online 23 January 2024

the advantages of the sine and cosine algorithm (SCA) and the artificial bee colony (ABC), Brajević et al. (2022) proposed a new hybrid algorithm (HSCA) to solve the engineering design optimization problem, in which the SCA with improved search strategy and the ABC algorithm with two different search equations run alternately. Brajević et al. (Brajević, 2021) also used the useful information of the current optimal solution in the onlooker phase to modify the ABC search operator and improve its utilization tendency. In addition, the shuffle mutation operator is applied to the solutions created in the both bee phases to balance the ability of global exploration and local exploitation, as well as to improve the convergence speed. Generating test cases manually is labor-intensive, and the nature of ABC makes it a potential search algorithm that can be used to automate test set generators, as Sahin et al. (2021) proposed two multi-criterion combinatorial ABC algorithms that use variation and crossover operators to generate test cases.

Moreover, ABC algorithm has been modified to solve discrete combinatorial optimization problems (Jooda et al., 2021; Kaya et al., 2022) such as quadratic assignment problem, minimum spanning tree problem, knapsack problem (BK), workshop scheduling problem, vehicle routing problem (VRP) and traveling salesman problem (TSP). TSP problem is a classical combinatorial optimization problem and the model for many practical problems, but researchers have never stopped studying this problem. With the development of swarm intelligence algorithm as a powerful means to solve this problem, various algorithm variants gradually appear and constantly break through the obtained optimums. As one of the excellent swarm intelligence computing methods, the potential of the ABC in this problem is worthy of further exploration, which contributes to the development of diversified TSP processing methods and lays a foundation for application in daily production. For example, Kiran et al. (Kıran et al., 2013) found that discrete ABC is more promising for TSP, and 2-opt and 3 opt can enrich the solution of discrete algorithm to improve the quality of its solution. Aiming at the TSP problem, Zhong at el (Zhong et al., 2017). proposed a hybrid discrete ABC algorithm based on the threshold acceptance criterion, in which the employed bees and the onlooker bees used the threshold acceptance criterion to decide whether to accept the newly generated solution, and verified the advantages of the new solution to update the equation and the necessity of using the non-greedy acceptance strategy to maintain sufficient diversity. Choong at el (Choong et al., 2019). used a super-heuristic method, the Modified Choice Function (MCF), to automatically adjust the selection of neighborhood search heuristics adopted by both hired and bystanders to achieve adaptive control of the weight of their reinforcement and diversification components at different phases of the search process. In addition, by combining MCF with Lin-Kernighan (LK) local search strategy, the proposed MCF-ABC model has excellent performance in dealing with TSP. Khan et al. (Khan and Maiti, 2019) modified ABC to solve TSP through eight different solution updating rules and *K*-opt operation. Dong et al. (2019) used the generated neighborhood solution to improve ABC algorithm to solve an extended model of the multi-traveling salesman problem, which is the large-scale colored traveling salesman problem (CTSP). Karaboga and Gorkemli (2019) put forward a combined ABC algorithm called CABC and a fast CABC algorithm named qCABC with good TSP performance.

The research upsurge of ABC algorithm is not only the progress of thought and theory, but also brings substantial convenience to people's daily life. Tasgetiren et al. (2011) proposed a discrete ABC model of hybrid iterative greedy algorithm variant to find the arrangement of minimum total flow time, and applied it to batch flow shop scheduling problem. Tsai et al. (Tsai, 2014) integrated ABC and bee Algorithm (BA) to solve constrained optimization problems with equality or inequality constraints. Li and Pan et al. (Li and Pan, 2015) mixed ABC and tabu search (TS) to minimize the maximum completion time of the hybrid flow shop (HFS) scheduling problem with finite buffers, the proposed algorithm is tested on a large-scale instance set generated based on real production. Based on the artificial bee colony framework, Peng et al. (2021) came up with a hybrid artificial bee colony algorithm that merges several specialized strategies to address the maximal quasi-cluster problem, and abbreviated as HABC. When traffic is heavy, emergency vehicles cannot reach their destination on time. Patil and Ragha et al. (Patil and Ragha, 2020) proposed a routing scheme for vehicular ad hoc networks based on adaptive fuzzy message propagation and micro-artificial bee colony algorithm optimization. Zhao et al. (2020) combined an improved ABC algorithm with a social force model that simulates pedestrian movement, enabling people to dynamically plan paths based on real-time changes in evacuation in emergency situations. To aid in the diagnosis of eye diseases, Khomri et al. (2018) proposed an elite-guided multi-objective artificial bee colony algorithm as a method for retinal vessel segmentation. In the aspect of determining the parameters with the best performance, Santos et al. (2016) studied the effect of parameters in simulated annealing (SA) and discrete artificial bee colony on solving TSP, and the interaction of neighborhood structure with other parameters.

The rest of this paper is organized as follows. The second chapter describes the principles of standard ABC and TSP, and the correspondence of variables when ABC is used to solve TSP. Then, in the next section, the motivation for the proposed algorithm is introduced, and the role of the improvement strategies is described in detail. The fourth chapter is mainly about the experimental results and statistical analysis. Finally, the conclusion is given in the fifth section.

2. Related work

2.1. Traveling salesman problem

The basic principle of traveling salesman problem is that a traveling salesman needs to go to some cities with the shortest distance. After starting from a city, he must traverse all cities once and only once on the way, and finally return to the starting point, that is, find the shortest route including all cities. TSP can be represented by a complete graph $G = (V, E)$, where $V = \{1, 2, 3, \ldots, n\}$ and E are the set of city nodes and the set of edges between two cities, respectively. The distance between cities is d_{ij}, $i, j \in V$, and $i \neq j$. The mathematical model of TSP can be expressed as the following Eq. (1)-Eq. (3).

$$min\ Z = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij} \tag{1}$$

$$s.t. \sum_{i=1}^{n} x_{ij} = 1 \tag{2}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \tag{3}$$

If the traveling salesman goes from the city i to the city j, then $x_{ij} = 1$, otherwise $x_{ij} = 0$. Let $V = (x_1, x_2, \ldots, x_n, x_1)$ be the complete traversal sequence of the traveling salesman, which gives the order of all the cities traversed by the traveling salesman. The above problem is transformed into determining a complete traversal sequence and finding the minimum value as shown in the following Eq. (4).

$$Z = \sum_{i=1}^{n-1} d_{x_i x_i} + 1 + d_{x_n x_1} \tag{4}$$

2.2. Artificial bee colony algorithm

The ABC algorithm process is inspired by the foraging behavior of bees, and bees are divided into employed bees, onlooker bees and scout bees according to different division of labor. In the whole process, bees promote population evolution through information sharing. The job of employed bees is to search for nectar sources, and there is a one-to-one relationship between employed bee and food source. After the employed bees have finished searching all food sources, they share information with the onlooker bees, such as the direction, distance, and abundance of the food source. The onlooker bees then judge the quality of the food

Table 1
Corresponding relationship between bee colony foraging behavior and TSP.

Bee colony foraging behavior	TSP
food source location	traversal sequence
fitness value of food source	traversal sequence length
optimal food source	shortest sequence path

sources according to the information obtained, and decide whether to follow the employed bees to start mining. When a food source is exhausted, the bees in that place will give up the nectar, they will change their roles as scout bee to find new food source to avoid the shortage of food sources at the same time. After finding a new food source, scout bees will turn into employed bees again. The corresponding relationship between bee colony foraging behavior and TSP is shown in Table 1.

The specific steps of the standard ABC algorithm are as follows.

(1) Initialization phase

Initialize the parameters, and randomly generate the initial food source positions (initial solutions). The initial solution X_i is generated according to Eq. (5).

$$x_{ij} = x_{minj} + rand \times \left(x_{maxj} - x_{minj}\right) \tag{5}$$

where $i = 1, 2, \ldots, SN$, and SN is the number of solutions, $j = 1, 2, \ldots, D$, D is the dimension of the solution (corresponding to the number of cities 'n' in the traveling salesman problem), *max* and *min* represent the upper and lower limits of the solution on the *j-th* dimensional search space, respectively. The function of *rand* is used to generate random numbers uniformly distributed between [0, 1].

(2) Employed bees phase

The employed bees search the neighborhood of all food sources at this phase. As shown in Eq. (6), a new solution V_i is searched in the neighborhood of X_i.

$$v_{ij} = x_{ij} + \varphi_{ij} \times \left(x_{ij} - x_{kj}\right) \tag{6}$$

where φ_{ij} is a normally distributed random number between [-1, 1], $k \in \{1, 2, \ldots, SN\}$ and $k \neq i$.

(3) Onlooker bees phase

The onlooker bees will further search for the selected better solutions through Eq. (6). After calculating the selection probability $P(X_i)$ of each feasible solution by Eq. (7), a better solution is chosen in the way of roulette.

$$P(X_i) = fit(X_i) \Big/ \sum_{i=1}^{SN} fit(X_i) \tag{7}$$

$$fit(X_i) = \begin{cases} 1/(1 + f(X_i)) & f(X_i) \geq 0 \\ 1 + |f(X_i)| & f(X_i) < 0 \end{cases} \tag{8}$$

where $f(X_i)$ is the objective function value corresponding to the solution X_i, $fit(X_i)$ is it's fitness value and the calculation equation is as in Eq. (8). When solving TSP, the fitness value $fit(X_i)$ of the feasible solution X_i is calculated as follows:

$$fit(X_i) = \sum_{i=1}^{n-1} d_{x_i x_{i+1}} + d_{x_n x_1} \tag{9}$$

(4) Scout bees phase

The converted scout bees randomly look for new food sources according to Eq. (5) at this phase. When a new nectar is found, the scout bee turn into employed bee to continue searching.

X_2	1	4	2	3	5
V_2	1	5	4	2	3

Fig. 1. Random single-point insert operation.

3. Proposed approach

3.1. Motivation

In summary, although ABC was originally designed to solve the continuous optimization problem, it is far more than that in terms of its application prospects. At the same time, the superiority of *K*-opt optimization technique in solving discrete combinatorial optimization problems is obvious to all. Since the *K*-opt algorithm has strong local optimization performance, and the ABC algorithm is famous for its excellent global optimization ability, we wondered whether could combine the two to achieve better optimization efficiency. After looking over the data, it is found that there are some intelligent algorithms that introduced the *K*-opt algorithm to optimize the performance of the algorithms when solving the combinatorial optimization problem. Among them, 2-opt and 3-opt techniques are the most common, but compared with other intelligent algorithms, there are few studies that took *K*-opt operator into ABC algorithm. Therefore, this paper is devoted to the improvement of the ABC algorithm by the 2-opt strategy and the research on the performance of this approach in solving TSP problems.

3.2. Designed strategies

The representation method of positive integer coding is applied to indicate the feasible solutions of TSP optimization problems, which are composed of city numbers represented by positive natural numbers that are not repeated. For example, the feasible solution X_1= (3, 1, 4, 5, 2) indicates that the traversal sequence of TSP is to visit the city numbered 3 first, then visit the remaining city numbers in the solution X_1 in turn, and finally return to the starting point city numbered 3.

3.2.1. Local search strategy

Considering that the update equations of solutions in ABC algorithm are continuous search, three kinds of random insertion operations are applied to replace the solution update method in the employed bee phase of standard ABC. The random insertion operations include random single-point insertion, random sequence insertion and random reverse sequence insertion, and more neighborhood operators can refer to the literature (Kıran et al., 2013). In the algorithm process, one of the above methods is randomly selected as the search operator each time.

The random single-point insertion operation adopted in paper is that randomly select the positions of the two cities in a solution, and insert the later city before the place of the forward one. Then, the front city and all the cities before the back city move back one. Meantime, the other cities remain unchanged. The random sequence insertion operation still randomly picks the positions of two cities, and determines the length of subsequence in range, then insert the cities from second position to meet length before the first city, and then shift all cities from the first one to subsequence by one. Finally, the third insertion operation is similar to the previous one, but flips the subsequence and performs the same operation as the random sequence insertion.

X_2	1	4	2	3	5
V_2	1	3	5	4	2

Fig. 2. Random sequence insertion operation.

X_2	1	4	2	3	5
V_2	1	5	3	4	2

Fig. 3. Random reverse sequence insertion operation.

Assuming that a food source individual $X_2 = (1, 4, 2, 3, 5)$, the new food source generated by the search is V_2. The search process of random single-point insertion operation, as shown in Fig. 1, if the 2nd and 5th positions are randomly selected, after the single-point insertion operation is performed, the city sequence $V_2 = (1, 5, 4, 2, 3)$ of the new food source. As shown in Fig. 2, if 2nd and 4th two positions are selected and the length of subsequence is 2 for the random sequence insertion operation, the solution sequence $V_2 = (1, 3, 5, 4, 2)$. If the random reverse sequence insertion operation selects the same city locations and subsequence length as the random sequence insertion operation, then the obtained sequence $V_2 = (1, 5, 3, 4, 2)$, and the specific operation is shown in Fig. 3.

3.2.2. Local search enhancement strategy

Although the random insertion operations have local search ability, it is also extremely blind, which will cause the search to fail to play a substantial role in the middle and later stage of the algorithm. In view of this deficiency, inspired by the literature (Xu et al., 2021; Qi et al., 2021), an enhanced local search strategy is adopted in this paper, which introduces the concept of neighborhood on the basis of the above insertion operations to reduce the blindness of search. Randomly select a city x_i in the path sequence, then find the previous city x_{i-1} and the next city x_{i+1}, compare $d(x_i, x_{i-1})$ and $d(x_i, x_{i+1})$, and take the larger number between them as the radius of the neighborhood. The distance between x_i and them is compared and the farther is taken as the radius of the neighborhood. The way city x_i generates a new feasible solution is that it is inserted into the previous locations of cities in the sequence that are within the adjacent radius other than cities x_{i-1} and x_{i+1}, which can connects two cities that may be closer together. In fact, this is still a selected point inserted into a location, which is no longer randomly generated but is tested for the best in sequence within the neighborhood determined by distance. Thus, this can also explain the generation of V_1, V_2, and V_3, which is more progressive than random insertion operations at some point.

For example, suppose that the feasible solution sequence is $X_1 = (1, 5, 3, 6, 4, 8, 2, 7)$, $x_i = 4$, $x_{i-1} = 6$ and $x_{i+1} = 8$, and $d(4,6) < d(4,8)$, only cities in 3rd, 6th and 2nd positions are in the neighborhood of x_i, then the new feasible solutions are $V_1 = (1, 5, 4, 3, 6, 8, 2, 7)$, $V_2 = (1, 5, 3, 4, 6, 8, 2, 7)$ and $V_3 = (1, 5, 3, 6, 8, 4, 2, 7)$, and update the current solution based on the best of them.

3.2.3. 2-Opt with fixed neighborhood search

2-opt is a path improvement algorithm, which is used to search for the potential better neighborhood solution of any solution in TSP. There may be better solutions than the original among these potentially feasible solutions, enabling shorter travel paths. Under the 2-opt method, all edges in a certain path sequence are searched to continuously update the current optimal solution. However, the move increases the time cost significantly while enhancing the ability of local search. Therefore, in order to balance the optimization effect and computational cost of the strategy, this paper adds the concept of fixed neighborhood search (*fns*) to the strategy. The specific implementation is that when searching for the next city of a certain city, the optimization is only carried out within the specified *fns*. As shown in Fig. 4, suppose the sequence $X_1 = (1, 2, 5, 4, 3, 6, 7, 8, 9, 10, 11, 12)$ is a feasible solution of TSP, $d(2,3) + d(5,6) < d(2,5) + d(3,6)$ and $fns = 3$. Then, for city 2, just search the subsequence 5, 4 and 3, whoes $fns = 3$, and finally a better solution $X_1' = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12)$ can be obtained.

3.2.4. Improvement strategy of scout bees phase

The phase of the standard ABC algorithm can maintain population diversity, which uses the same update equation as the initialization phase. This means that while it is possible to produce a solution with better quality, the final solution is usually of poor quality and requires more computation time to approach the optimal solution. In order to improve the effect of this phase, the information of the current optimal solution is combined with the enhancement strategy in Section 3.2.2 to speed up the convergence while maintaining a certain population diversity, which is beneficial to avoid getting stuck in local optimum.

Specifically, a new solution is generated by comparing the solution entering this phase with the current optimal solution, and finding out that the locations of the cities with the same number and the same location as the optimal solution remains unchanged, then the remaining other cities are randomly arranged. For example, suppose the feasible solution $X_2 = (1, 5, 3, 6, 4, 8, 2, 7)$ enters the scout bee phase, and the current optimal solution $X_{best} = (1, 7, 3, 6, 5, 8, 2, 4)$, then the same segment [1, *, 3, 6, *, 8, 2, *] is obtained through comparing X_2 and X_{best}, where * indicates that the city number of the location is unknown. Hence, a new solution $X_2' = (1, 4, 3, 6, 7, 8, 2, 5)$ is possible. Finally, the local enhancement strategy is performed on a random city in the new solution, and the generated best solution is used to replace the solution that has not been updated more than the preset limit times.

3.3. Design of DABC-FNS

In short, the idea of the algorithm proposed in paper is briefly summarized as follows. First, the solutions of the ABC algorithm are represented by the positive integer coding method. Considering the different roles of the bees, the search equation in the onlooker bee phase

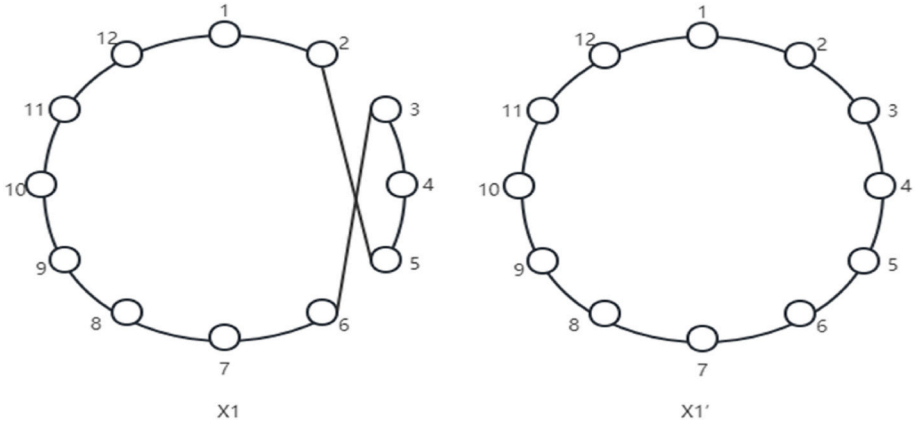

Fig. 4. Schematic diagram of fixed neighborhood 2-opt search.

Table 2
Description of the DABC-FNS algorithm.

Algorithm 1 Proposed Approach (DABC-FNS)

Begin
Randomly generate *SN* initial solutions X_i and compute objective function values of them;
Set *Cycle* =0 and $trial_i$=0;
while *Cycle*≤*MaxCycle* do
for *i*=1 to *SN* do
Generate a new solution V_i for X_i according to the section 3.2.1;
Calculate the fitness value of V_i according to the Eq. (9);
If $fit(V_i) \le fit(X_i)$
$X_i=V_i$;
$trial_i$=0;
else
$trial_i=trial_i+1$;
end if
end for
Calculate the $P(X_i)$ of every solution in the swarm according to the Eq. (7);
for *i*=1 to *SN* do
Choose the solution X_i whose $P(X_i)$ bigger than random number generated by rand function;
Generate a new solution V_i for X_i according to the section 3.2.2;
Calculate the fitness value of V_i according to the Eq. (9);
If *optr*>*rand*
Randomly generate a city position number from V_i;
Generate new solutions OV_i for V_i within the *fns* according to the section 3.2.3 and calculate the fitness value;
Replace V_i if there is a better solution among OV_i;
end if
If $fit(V_i) \le fit(X_i)$
$X_i=V_i$;
$trial_i$=0;
else
$trial_i=trial_i+1$;
end if
end for
if max{$trial_i$}> *limit*
Randomly generate a new solution for X_i according to the section 3.2.4;
end if
Cycle = *Cycle* +1;
if mod(*MaxCycle*,1000)==0
for *j*=1 to *n* do
Randomly swap the positions of two nodes in the current global optimal solution;
Reserve the best solution;
end for
end if
end while
output the best solution;
end

is further changed, and a local enhanced search strategy that is more in line with the duties of the onlooker bees is adopted. Second, before the end of the onlooker bees phase, a 2-opt strategy with fixed neighborhood search is introduced to eliminate most of the path intersections in feasible solutions. Then, in the scout bee phase, the local enhanced search strategy is combined again, which can not only maintain the diversity of the population, but also avoid reducing the convergence speed of the algorithm. Finally, the optimal solution generated every thousand generations is randomly exchanged with two elements, and the current optimal solution is tentatively optimized. The specific implementation of the complete algorithm is shown in Algorithm 1 in the following table. During the process, *Cycle* is the number of algorithm evaluations, the maximum number of *MaxCycle* evaluations is used as a termination condition, and *optr* and *fns* are the control parameters of the 2-opt strategy execution.

3.4. Complexity analysis of the designed algorithm

In this section, the computational complexity of the DABC-FNS algorithm is discussed according to the three main processes of the

Table 3
Optimal solutions for 30 TSP instances.

No	Instance	Optimal	No	Instance	Optimal
1	ulysses22	74	16	pr152	73 682
2	oliver30	420	17	d198	15 780
3	chn31	15 377	18	kroA200	29 368
4	att48	33 522	19	rand200	10 649
5	rand50	5553	20	tsp225	3916
6	eil51	426	21	a280	2579
7	berlin52	7542	22	pr299	48 191
8	eil76	538	23	rand400	14 722
9	pr76	108 159	24	fl417	11 861
10	rat99	1211	25	pr439	107 217
11	kroA100	21 282	26	ali535	202 339
12	rand100	7891	27	rat575	6773
13	pr107	44 303	28	rat783	8806
14	ch130	6110	29	fl1400	20 127
15	ch150	6528	30	u2319	234 256

Table 4
Experimental parameter settings.

SN	*MaxCycle*	*limit*	run times
40	30 000	500	10

employed bee phase, the onlooker bee phase and the scout bee phase. Assume f is the computational time of the objective function, and the computational time of ABC is $(SN \times f + SN \times f + f) = (f \times (2SN + 1))$, that is, the time complexity is $O(SN \times f)$. For the proposed DABC-FNS, its computational time is $(SN \times f + SN \times (u + optr \times fns) \times f + v \times f + 0.001n \times f) = (f \times SN \times (1 + u + 0.05n + v + 0.001n)) = (SN \times (0.05n + 0.001n) \times f) = (0.051n \times SN \times f)$, where SN is the total number of solutions, n denotes the instance scale, u and v represent the number of enhanced search in the phase of onlooker bee and scout bee, respectively, and the values of this two variables should be low as defined in Section 3.2.2, $0.001n \times f$ is generated by the search of lines 37 to 40 in Table 2. Therefore, the time complexity of the proposed DABC-FNS is $O(n \times SN \times f)$.

4. Experiment study

To validate the performance of the proposed DABC-FNS algorithm in handling TSP problems, 30 benchmark test instances with gradually increasing city sizes in TSPLIB instance library were used for experimental simulation. For the sake of fairness in subsequent experiments, the selected instances from TSPLIB are mostly typical benchmarks often tested in many related literatures. The experiment is divided into three parts, the first is the influence of parameters setting, the second is the actual effect of two main improvement strategies, and the last is the comparison experiment of the algorithm and variants. The number of cities in all instances is between 22 and 2319, and each instance is run for 10 times independently. The known optimal solutions of the test instances are shown in the optimal column in Table 3, which are also the optimum tour lengths of these TSP problems. During the test process of the DABC-FNS algorithm, the required parameters are set as exhibited in Table 4 below.

4.1. Effect of the fns

For the setting of two parameters *fns* and *optr* in DABC-FNS, the single variable principle was followed during the experiment. Since the added time cost of executing the 2-opt operator, the setting of the parameter *optr* directly affects the calculation time. Therefore, in order to reduce the time cost of enhancement strategy as much as possible on the basis of ensuring the quality of the solution, the value of *optr* is first set to a small value 0.1. After that, for the setting of the fixed neighborhood search parameter *fns*, it is adjusted through testing in the set $[0.1n, 0.2n, 0.3n, 0.4n, 0.5n]$, where the variable n in the set refers to the total number of cities of TSP instance. The results of this experiment are shown in Table 5 and in which the bold font is the better solutions

Table 5
Solution accuracy of DABC-FNS with different *fns* values.

No	$fns = 0.1n$		$fns = 0.2n$		$fns = 0.3n$		$fns = 0.4n$		$fns = 0.5n$	
	Best	BestErr/%	Best	BestErr/%	Best	BestErr/%	Best	BestErr/%	Best	BestErr/%
1	**72**	**−2.70E+00**	**72**	**−2.70E+00**	**72**	**−2.70E+00**	**72**	**−2.70E+00**	**72**	**−2.70E+00**
2	**420**	**0**	**420**	**0**	**420**	**0**	**420**	**0**	**420**	**0**
3	**15 377**	**0**	**15 377**	**0**	**15 377**	**0**	**15 377**	**0**	**15 377**	**0**
4	33 587	1.94E-01	**33 522**	**0**	**33 522**	**0**	**33 522**	**0**	**33 522**	**0**
5	**5553**	**0**	**5553**	**0**	**5553**	**0**	**5553**	**0**	**5553**	**0**
6	428	4.69E-01	427	2.35E-01	**426**	**0**	**426**	**0**	**426**	**0**
7	**7542**	**0**	**7542**	**0**	**7542**	**0**	**7542**	**0**	**7542**	**0**
8	550	2.23E+00	550	2.23E+00	543	9.29E-01	541	5.58E-01	**539**	**1.86E-01**
9	108 981	7.60E-01	108 644	4.48E-01	**108 159**	**0**	**108 159**	**0**	**108 159**	**0**
10	1257	3.80E+00	1220	7.43E-01	1228	1.40E+00	1212	8.26E-02	**1211**	**0**
11	21 647	1.72E+00	**21 282**	**0**	21 292	4.70E-02	**21 282**	**0**	**21 282**	**0**
12	8041	1.90E+00	8013	1.55E+00	7949	7.35E-01	8003	1.42E+00	**7941**	**6.34E-01**
13	44 566	5.94E-01	44 570	6.03E-01	44 522	4.94E-01	44 431	2.89E-01	**44 391**	**1.99E-01**
14	6430	5.24E+00	6258	2.42E+00	6217	1.75E+00	**6168**	**9.49E-01**	6173	1.03E+00
15	6735	3.17E+00	6726	3.03E+00	6659	2.01E+00	6615	1.33E+00	**6568**	**6.13E-01**
16	75 608	2.61E+00	74 278	8.09E-01	74 089	5.52E-01	**73 682**	**0**	**73 682**	**0**
17	16 356	3.65E+00	16 089	1.96E+00	16 063	1.79E+00	15 956	1.12E+00	**15 936**	**9.89E-01**
18	31 371	6.82E+00	30 872	5.12E+00	30 281	3.11E+00	30 108	2.52E+00	**29 698**	**1.12E+00**
19	11 311	6.22E+00	11 162	4.82E+00	11 079	4.04E+00	10 953	2.85E+00	**10 895**	**2.31E+00**
20	4229	7.99E+00	4131	5.49E+00	4079	4.16E+00	**4033**	**2.99E+00**	4067	3.86E+00
21	2882	1.17E+01	2704	4.85E+00	2721	5.51E+00	2727	5.74E+00	**2693**	**4.42E+00**
22	53 659	1.13E+01	51 815	7.52E+00	49 856	3.46E+00	49 826	3.39E+00	**49 796**	**3.33E+00**
23	16 036	8.93E+00	15 934	8.23E+00	15 741	6.92E+00	**15 517**	**5.40E+00**	15 530	5.49E+00
24	12 731	7.33E+00	12 546	5.78E+00	11 995	1.13E+00	**11 963**	**8.60E-01**	11 987	1.06E+00
25	121 325	1.32E+01	116 733	8.88E+00	113 022	5.41E+00	**110 432**	**3.00E+00**	112 098	4.55E+00
26	223 925	1.07E+01	211 100	4.33E+00	211 815	4.68E+00	**208 337**	**2.96E+00**	208 983	3.28E+00
27	7731	1.41E+01	7373	8.86E+00	**7129**	**5.26E+00**	7138	5.39E+00	7148	5.54E+00
28	10 061	1.43E+01	9696	1.01E+01	9437	7.17E+00	9448	7.29E+00	**9355**	**6.23E+00**
29	24 365	2.11E+01	21 898	8.80E+00	21 608	7.36E+00	21 311	5.88E+00	**20 940**	**4.04E+00**
30	265 612	1.34E+01	247 950	5.85E+00	245 010	4.59E+00	244 527	4.38E+00	**244 020**	**4.17E+00**

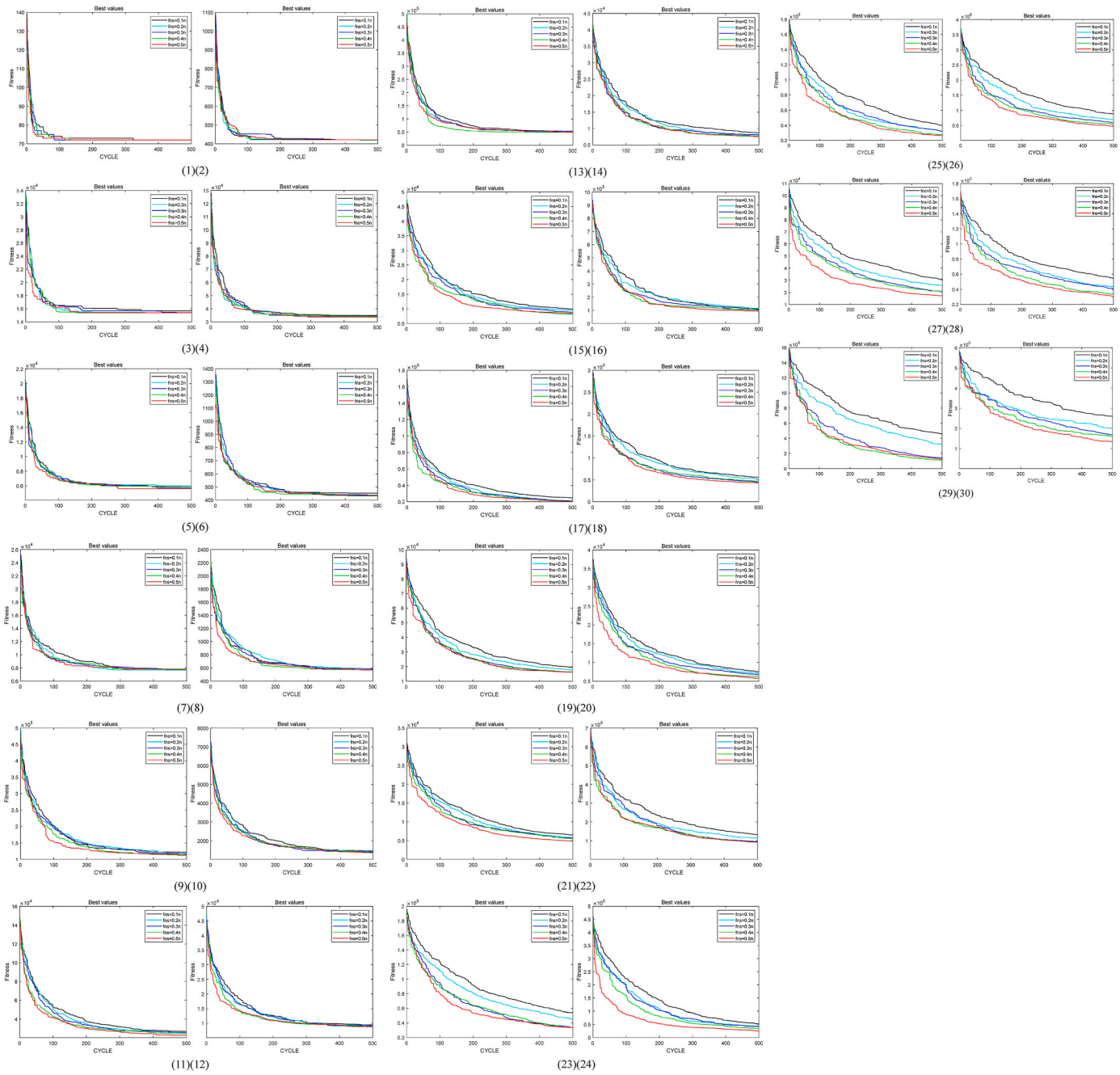

Fig. 5. Solving curves for DABC-FNS with different *fns* values.

among competitors, and the Best column represents the optimal solution obtained by the algorithm, and the BestErr is the best solution error rate, calculated according to Eq. (10).

$$BestErr = (Best - Optimal)/Optimal \times 100\% \tag{10}$$

According to the data in Table 5, it can be seen that compared with other *fns* values, DABC-FNS with $fns = 0.5n$ can get the optimal solutions of 10 TSP instances numbered 2 to 7 and 9 to 11 and 16 among test instances, and there are 23 instances have the higher accuracy. Among the 23 instances, there are 10 instances with fewer cities that the DABC-FNS at other *fns* values can also obtain solutions with the same accuracy. Here, for the 5 instances of ulysses22, oliver30, chn31, rand50 and berlin52, all the values of *fns* can get the optimal solution of these instances. Moreover, the obtained final solution of instance ulysses22 lower than its known optimal solution, but this does not exclude the error in the calculation process. Therefore, for the small-scale TSP problem, the value of *fns* in DABC-FNS just slightly affects the performance, and generally can find excellent feasible solutions. However, the algorithm with $fns = 0.5n$ also has certain shortcomings. For example, from the results of the three instances of rand100 and rand400 and rat783, the error rates between the obtained solutions and the optimal solutions are relatively high, reaching five to six percent. This does not mean that DABC-FNS with $fns = 0.5n$ is difficult to use for solving larger-scale problems. Referring to the solutions of numbered 29 and 30 instances, which with thousands of cities, it is evident that DABC-FNS still has potential in dealing with large-scale TSP.

To intuitively analyze the impact of different *fns* values on the performance of the DABC-FNS, the first 500 generations of experimental data during the test were intercepted, and the solution curves of all test instances were drawn, as displayed in Fig. 5. In the graph, the ordinate

Table 6
Results of different *fns* values in Friedman test.

fns values	Mean ranking
0.1*n*	4.62
0.2*n*	3.63
0.3*n*	2.88
0.4*n*	2.10
0.5*n*	**1.77**

Fitness represents the obtained travel path length of the feasible solution, and the horizontal axis *CYCLE* represents the iteration times of the test. According to the trend of each solution curve in the figure, it can be seen that for most instances, the DABC-FNS algorithm has faster convergence speed and higher convergence accuracy when its *fns* value equal to 0.5*n*. In addition, it can be found that the solution effect for a few instances is also competitive when the value of *fns* is 0.4*n*. For example, for the instance ch130, the speed and accuracy of the obtained solution are good, even surpassing the efficiency of $fns = 0.5n$, which shows that the value of *fns* in different examples is not as large as possible.

From a statistical point of view, Table 6 shows the Friedman test results of the solving performance of DABC-FNS with different *fns* values. The ranking results manifest a trend that the performance of the algorithm gradually improves with the increase of the *fns*, but it seems to be the opposite of the above conclusion that not the larger of *fns* is the better. The reason is that Friedman test studies the distribution of experimental data from the overall level, it may not be the optimal scheme for any individual. However, it is also a good method to balance the overall situation. The statistical results show that the best ranking of Friedman test is $fns = 0.5n$, which also supports the above experimental analysis results.

4.2. Optimization strategies effect analysis

In order to analyze the effectiveness of the improved strategies in the proposed algorithm, this part experiment independently tests the improved performance of the two main strategies on all benchmark instances, and compares them with the complete DABC-FNS algorithm. The first algorithm named DABC-2opt, completely delete the enhancement strategy in Section 3.2.2 in DABC-FNS. This means that part of the search effect will be lost in the phase of scout bees, but still different from the way of the solution generate in the initialization phase. The second is to strike out the 2-opt strategy with fixed neighborhood search, and referred to as the EDABC algorithm for short. The data obtained from the comparison experiment is exhibited in Table 7.

From Table 7 and it can be known that the strategies designed in Sections 3.2.2 to 3.2.4 have a remarkable improvement effect on ABC algorithm. After the introduction of these strategies, the solution performance of the DABC-FNS algorithm is improved on all test instances, so the combination of the two main strategies has a positive impact. Comparing with part experimental results in Section 4.1, it can be realized that the proposed algorithm has improved the solution accuracy of ten instances, including instances numbered 15, 19, 20, 21, 23, 24, 25, 27, 29 and 30, which indicates that DABC-FNS has the potential to obtain feasible solution closer to the optimal solution. The solving curves of the DABC-2opt, EDABC and DABC-FNS three algorithms are shown in Fig. 6. Based on the curve trends in the graph, compared with the other two algorithms, DABC-FNS has absolute advantages in terms of solving speed and accuracy. By analyzing the data in Table 7, except that the three algorithms all get the same solutions in six instances numbered 1 to 5 and 7, it is found that the DABC-2opt algorithm has higher solution accuracy in 12 instances of different scales numbered 8 and 10 to 19 and 24, while the EDABC algorithm has higher solution accuracy on the other 11 instances except instance 5.

On the basis of the above different algorithms, the three kinds of convergence curves in the process of solving is given in Fig. 6. From the curve trends, it is clear that the DABC-FNS algorithm does have higher solution accuracy and faster convergence speed in evidence. Comparing the convergence curves of DABC-2opt and EDABC, we can see that for most instances, the local enhancement strategy has a better solving

Table 7
Comparison of optimization effects of different strategies in DABC-FNS.

No	Instance	DABC-2opt		EDABC		DABC-FNS	
		Best	BestErr/%	Best	BestErr/%	Best	BestErr/%
1	ulysses22	**72**	**−2.70E+00**	**72**	**−2.70E+00**	**72**	**−2.70E+00**
2	oliver30	**420**	**0**	**420**	**0**	**420**	**0**
3	chn31	**15 377**	**0**	**15 377**	**0**	**15 377**	**0**
4	att48	**33 522**	**0**	**33 522**	**0**	**33 522**	**0**
5	rand50	**5553**	**0**	**5553**	**0**	**5553**	**0**
6	eil51	428	4.69E-01	428	4.69E-01	**426**	**0**
7	berlin52	**7542**	**0**	**7542**	**0**	**7542**	**0**
8	eil76	555	3.16E+00	556	3.35E+00	**541**	**5.58E-01**
9	pr76	109 587	1.32E+00	109 331	1.08E+00	**108 159**	**0**
10	rat99	1255	3.63E+00	1258	3.88E+00	**1218**	**5.78E-01**
11	kroA100	21 518	1.11E+00	21 729	2.10E+00	**21 282**	**0**
12	rand100	8050	2.01E+00	8144	3.21E+00	**7956**	**8.24E-01**
13	pr107	45 038	1.66E+00	45 205	2.04E+00	**44 391**	**1.99E-01**
14	ch130	6384	4.48E+00	6452	5.60E+00	**6180**	**1.15E+00**
15	ch150	6941	6.33E+00	7027	7.64E+00	**6577**	**7.51E-01**
16	pr152	75 733	2.78E+00	78 198	6.13E+00	**73 682**	**0**
17	d198	16 493	4.52E+00	16 561	4.95E+00	**15 954**	**1.10E+00**
18	kroA200	32 108	9.33E+00	32 709	1.14E+01	**29 832**	**1.58E+00**
19	rand200	11 671	9.60E+00	11 877	1.15E+01	**10 878**	**2.15E+00**
20	tsp225	4293	9.63E+00	4280	9.30E+00	**4031**	**2.94E+00**
21	a280	3092	1.99E+01	2936	1.38E+01	**2691**	**4.34E+00**
22	pr299	57 550	1.94E+01	54 291	1.27E+01	**49 154**	**2.00E+00**
23	rand400	20 062	3.63E+01	17 544	1.92E+01	**15 468**	**5.07E+00**
24	fl417	13 910	1.73E+01	14 032	1.83E+01	**11 974**	**9.53E-01**
25	pr439	141 420	3.19E+01	129 031	2.03E+01	**110 287**	**2.86E+00**
26	ali535	270 250	3.36E+01	241 469	1.93E+01	**210 053**	**3.81E+00**
27	rat575	10 419	5.38E+01	8284	2.23E+01	**7142**	**5.45E+00**
28	rat783	16 025	8.20E+01	11 168	2.68E+01	**9442**	**7.22E+00**
29	fl1400	36 865	8.32E+01	26 910	3.37E+01	**20 690**	**2.80E+00**
30	u2319	578 281	1.47E+02	298 928	2.76E+01	**243 328**	**3.87E+00**

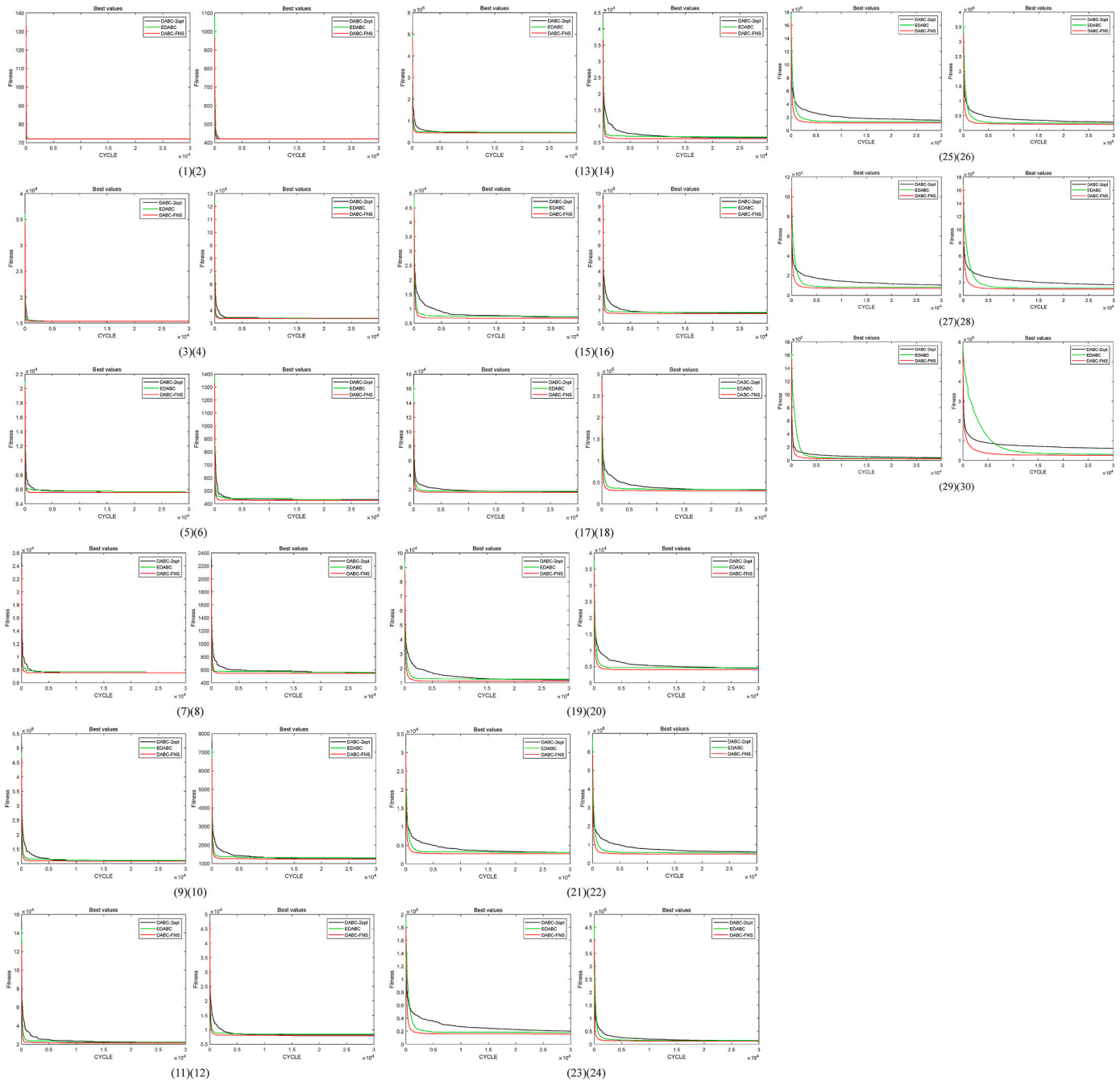

Fig. 6. Comparison of convergence curves of three algorithms.

Table 8
Friedman test results of three different algorithms.

Algorithms	Mean ranking
DABC-2opt	2.38
EDABC	2.42
DABC-FNS	**1.20**

speed, while the 2-opt strategy with *fns* is slightly inferior. As for the accuracy of solution, it seems to be the same. To demonstrate the correctness of this argument, the Friedman test also is performed on the solutions of the three algorithms, and the test result is shown in Table 8. Combined with the statistical results in the table, it can be seen that the 2-opt strategy with *fns* is not worse than the local enhancement strategy, and the average ranking of feasible solutions of the DABC-2opt algorithm is even a bit better than that of the EDABC algorithm. This means that under the control of the adaptive parameter *optr*, the 2-opt strategy with *fns* still works in the middle and late stage of the algorithm. Undoubtedly, the performance of the DABC-FNS algorithm that combines the two strategies ranks first, and statistical experiments show that its algorithm efficiency is significantly better than the other two single-strategy algorithms.

4.3. Algorithm effect comparison

After the first two experiments of parameter adjustment and main strategy effect test, the parameter setting and working mechanism of the entire algorithm are clarified. Now, the results of the solutions obtained by DABC-FNS algorithm are compared with the other two ABC variants

Table 9
Comparison of solving performance between DABC-FNS and two other ABC variants.

No	Instance	Optimal	Combined2 ABC		Combined2 2-OPT		DABC-FNS	
			Best	Mean	Best	Mean	Best	Mean
1	ulysses22	74	**72**	**72**	**72**	**72**	**72**	**72**
2	oliver30	420	421	426.8	**420**	420.5	**420**	**420**
3	chn31	15 377	15 379	15 505	15 379	15570.6	**15 377**	**15 377**
4	att48	33 522	33 952	34 466	33 772	34108.6	**33 522**	**33 522**
5	rand50	5553	5651	5765.6	5680	5767.4	**5553**	**5553**
6	eil51	426	440	445.2	437	441	**426**	**426.3**
7	berlin52	7542	7740	7873	**7542**	7743.4	**7542**	**7542**
8	eil76	538	557	568.2	556	561.8	**539**	**543.6**
9	pr76	108 159	109 297	112 004	110 061	112 533	**108 159**	**108 315**
10	rat99	1211	1275	1290.4	1273	1283.6	**1211**	**1229.4**
11	kroA100	21 282	21 736	22284.2	21 833	22032.8	**21 282**	**21332.4**
12	rand100	7891	8178	8357.2	8098	8302.4	**7941**	**8003.3**
13	pr107	44 303	45 367	46018.2	45 007	45 637	**44 391**	**44550.5**
14	ch130	6110	6453	6561.2	6460	6530.6	**6173**	**6227**
15	ch150	6528	7046	7135.6	7069	7154	**6568**	**6681.1**
16	pr152	73 682	76 949	77962.4	75 332	77341.4	**73 682**	**74054.1**
17	d198	15 780	16 770	16839.4	16 527	16767.6	**15 936**	**16000.2**
18	kroA200	29 368	32 980	33198.4	32 415	33130.4	**29 698**	**30324.9**
19	rand200	10 649	11 945	12 127	11 762	12002.6	**10 878**	**11038.6**
20	tsp225	3916	4479	4524	4407	4516.6	**4031**	**4080**
21	a280	2579	3233	3332	3248	3301	**2691**	**2734.5**
22	pr299	48 191	60 477	62066.2	61 957	62734.3	**49 154**	**50107.2**
23	rand400	14 722	21 350	22041.8	22 097	22 233	**15 468**	**15524.9**
24	fl417	11 861	17 247	17565.2	15 987	16557.8	**11 974**	**12055.6**
25	pr439	107 217	164 958	168 595	167 515	168 347	**110 287**	**113 966**
26	ali535	202 339	301 377	315 724	307 340	315 837	**208 983**	**211 459**
27	rat575	6773	11 915	12069.8	11 928	12032.8	**7142**	**7224.2**
28	rat783	8806	20 032	20207.8	19 640	20168.2	**9355**	**9446.1**
29	fl1400	20 127	88 574	90013.6	85 121	87327.2	**20 690**	**21310.5**
30	u2319	234 256	1 130 270	1 139 190	1 136 650	1 140 220	**243 328**	**244 575**

for solving the TSP in paper (Kıran et al., 2013). The two variants are Combined2 ABC and Combined2 2-OPT, respectively. The connection between them is that the second is based on the result of Combined2 ABC, and then the 2-opt operator is introduced to optimize the solution. Although this literature also analyzes other neighborhood operators, there are two main reasons for choosing these two variants for comparison. First of all, the results of the two algorithms are basically better solutions. Secondly, the insertion operations used in the employed bee phase of DABC-FNS is the same as that of Combined2 ABC, and also involves the application of 2-opt technology. Therefore, it is comparatively significant. In order to make a fair comparison, the two algorithms of comparison basically adopt their article settings in the experiment, and ensure that the experimental parameter settings of DABC-FNS are equal to or less than them.

The exact experimental data are shown in Table 9. Because the average of the obtained solution is more representative of the algorithm performance, the Mean column is also added as a research indicator. And the mean error rate (MeanErr) can be calculated according to Eq. (11), which is similar to the calculation method of the best solution error rate BestErr.

$$MeanErr = (Mean - Optimal)/Optimal \times 100\% \tag{11}$$

Comparing the data in the above Table 9 and it can be seen that the overall performance of the DABC-FNS algorithm is superior to the competitors in all instances. In addition, the ABC variant adding 2-opt does perform better in most instances than Combined2 ABC, but both of them appear to be incapable of dealing with larger-scale problems. Since the literature (Kıran et al., 2013) only gives the solution results of a few TSP test instances, the data in table obtained by the algorithms reproduced through the idea of the paper is affected by various factors and there is a little error compared with the original data, but taking this into account, DABC-FNS is still far better than the competitors in solving TSP.

However, the development of such algorithms has made a qualitative leap in recent years, so the results of the proposed algorithm for solving TSP problems are compared with the two algorithms in (Karaboga and Gorkemli, 2019) (namely CABC and qABC). The parameters in the experiment are set according to the data in (Karaboga and Gorkemli, 2019), as shown in Table 10, and the test instances are consistent with it. Table 11 shows the experimental results of the proposed algorithm and the comparison algorithm, including two evaluation indexes, the best value and the average value, in which the obtained best solution of the column is shown in bold.

Table 10
Experimental parameter settings.

SN	*MaxCycle*	*limit*	run times
20	800 000	$SN \times n$	20

According to the results of the three algorithms, their actual performance is analyzed statistically by Friedman test and Wilcoxon tests. According to the results of the three algorithms, their actual performance is analyzed statistically by Friedman test and Wilcoxon test. Through observing the data in Table 12, we can see that the optimal value obtained by the DABC-FNS algorithm has a poor ranking, with the average value ranking higher than qABC but lower than CABC. It shows that the ability of this algorithm to deal with TSP problems needs to be improved, but the stability of the algorithm is relatively good. However, Wilcoxon test results also show that there is no significant difference between the three, that is, they have similar ability to solve TSP, and they all have competitive solving performance.

Finally, some experimental results in this paper are roughly compared with the data of more recent excellent papers (Choong et al., 2019; Khan and Maiti, 2019) in Table 13, which are also researches on the ABC algorithm involving the swap operators and *K*-opt algorithm to solve the TSP problems.

Based on the results in Table 13 and it is not difficult to find that the accuracy of the three in the best values is similar, and they are almost the

Table 11
Comparison of the solving performance of three ABCs.

Instance	Optimal	CABC		qABC		DABC-FNS	
		Best	Mean	Best	Mean	Best	Mean
berlin52	7542	**7542**	**7542**	**7542**	**7542**	**7542**	**7542**
kroA100	21 282	**21 282**	21 291	**21 282**	21288.9	**21 282**	**21 282**
pr144	58 537	58 590	58641.6	58 590	58624.2	**58 537**	**58 537**
ch150	6528	**6549**	**6561.2**	6553	6563.6	6544	6580.2
kroB150	26 130	**26 186**	26 338	26 251	26 344	26 233	**26307.4**
pr152	73 682	**73 682**	73 792	**73 682**	73903.5	**73 682**	**73 682**
rat 195	2323	**2343**	2355.4	2347	**2353.2**	2380	2390.4
d198	15 780	**15 825**	**15854.4**	15 845	15875.7	15 838	15884.6
kroA200	29 368	29 467	**29517.5**	**29 461**	29526.6	29 579	29802.4
ts225	126 643	**126 643**	126751.7	**126 643**	**126 643**	**126 643**	**126 643**
pr226	80 369	80 872	81 133	80 876	81057.7	**80 463**	**80630.6**
pr299	48 191	48 640	48743.1	**48 513**	**48 723**	49 032	49367.8
lin 318	42 029	**42 756**	43037.2	42 829	**43037.1**	43 304	43531.2
pcb442	50 778	**51 309**	**51539.1**	51 403	51542.3	52 663	52 774
fl1577	22 249	**22 665**	22813.7	22 677	**22793.7**	23 755	23 852

Table 12
Statistical results.

Friedman test			Wilcoxon test		
Algorithm	Best ranking	Mean ranking	DABC-FNS vs.	Best *P*	Mean *P*
CABC	**1.63**	**1.83**	CABC	0.091	0.331
qABC	2.17	2.17	qABC	0.182	0.041
DABC-FNS	2.20	2.00			

optimal solutions. Specifically, the solution of DABC-FNS is better on kroB150, and the MCF-ABC achieves the highest quality on kroA200. Note that according to the published paper (Choong et al., 2019), MCF-ABC has also achieved good results in a large number of TSP examples, and its setting of some experimental parameters is lower. As for DABC-FNS, more iteration numbers are needed to get optimal results which may be due to the influence of algorithm strategies design, which is also the direction of future improvement in this paper. Compared with the other two, it is insufficient for DABC-FNS that the accuracy of the mean value is poorer, hence the stability of the algorithm performance still has room to be improved. Since the ABC variant ABC+3-opt described above is only tested on a small number of small-scale problems, it is not known whether its ability to solve larger-scale problems will remain undiminished. However, compared with solving small-scale problems, the performance of the DABC-FNS algorithm does not rapidly degrade when dealing with large-scale problems, which will also be one of the focuses of follow-up research.

5. Conclusion

Artificial bee colony algorithm is a model that simulates bees foraging based on swarm intelligence. When dealing with discrete combinatorial optimization problem, it is necessary to adapt the search mechanism of solutions in different phases. To solve the traveling salesman problem, a local enhancement search strategy is designed to modified the search methods of onlooker bees and scout bees in this paper, and a 2-opt strategy with fixed neighborhood search is introduced to eliminate the intersection in the travel, then a discrete ABC Algorithm with fixed neighborhood search called DABC-FNS is proposed. Based on the excellent global optimization ability of the ABC algorithm, integrated with the local enhanced search and 2-opt strategy that can improve the local optimization capability. It is helpful for DABC-FNS to jump out of the local optimum and find the global optimum. The instance study shows that the solution accuracy and convergence speed of the improved algorithm are promoted, and DABC-FNS is competitive in performance improvement. The proposed DABC-FNS algorithm has achieved good results for most instances when applied to solving TSP problems, while compared with several better ABC algorithms, there is still room such as the optimization of algorithm strategies for improvement in some instances, which is also the direction of our further research.

Compliance with ethical standards conflict of interest

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

CRediT authorship contribution statement

Xing Li: Conceptualization, Writing – review & editing, Methodology. **Shaoping Zhang:** Formal analysis, and, Data curation, Funding acquisition. **Peng Shao:** Supervision, Methodology, Funding acquisition, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence

Table 13
Comparison of solving performance of DABC-FNS with ABC+3-opt and MCF-ABC.

Instance	optimal	ABC+3-opt (Khan and Maiti, 2019)		MCF-ABC (Choong et al., 2019)		DABC-FNS	
		Best	Mean	Best	Mean	Best	Mean
eil51	426	427	427.01	**426**	**426**	**426**	426.3
berlin52	7542	**7542**	**7542**	**7542**	**7542**	**7542**	**7542**
eil76	538	**538**	538.15	**538**	**538**	539	543.6
rat99	1211	**1211**	1211.5	-	-	**1211**	1229.4
kroA100	21 282	**21 282**	21287.19	**21 282**	**21 282**	**21 282**	21332.4
kroB150	26 130	-	-	26 524	26 524	**26 235**	**26 452**
pr152	73 682	**73 682**	73691.64	**73 682**	**73 682**	**73 682**	74054.1
kroA200	29 368	29 450	29 469	**29 368**	**29 368**	29 698	30324.9

the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

This work is funded by the National Natural Science Foundation of China (No.71863018 and 71403112), and Jiangxi Provincial Social Sciences Planning Project (21GL12) and Science and Technology Plan Projects of Jiangxi Provincial Education Department (No.GJJ200424).

References

Aguirre Adrián, M., Songsong, L., Papageorgiou, L.G., 2018. Optimisation approaches for supply chain planning and scheduling under demand uncertainty. Chem. Eng. Res. Des. 138, 341–357.
Aguirre, A.M., Papageorgiou, L.G., 2018. Medium-term optimization-based approach for the integration of production planning, scheduling and maintenance. Comput. Chem. Eng. 116 (AUG.4), 191–211.
Bai, Q.C., 2020. Improved particle swarm optimization-based path planning of mobile sink in wireless sensor networks. Chin. J. Sensors Actuators 33 (5), 733–737.
Bandeira, T.W., Pereira Coutinho, W., Brito, A., et al., 2015. Analysis of path planning algorithms based on travelling salesman problem embedded in UAVs. In: Brazilian Symposium on Computing Systems Engineering, IEEE, pp. 70–75.
Benavent, E., Landete, M., Jose Salazar-Gonzalez, J., et al., 2019. The probabilistic pickup-and-delivery travelling salesman problem. Expert Syst. Appl. [J] 121 (MAY), 313–323.
Brajević, I., 2021. A shuffle-based artificial bee colony algorithm for solving integer programming and minimax problems. Mathematics 9 (11).
Brajević, I., Stanimirović, P.S., Li, S., et al., 2022. Hybrid sine cosine algorithm for solving engineering optimization problems. Mathematics 10 (23).
Choong, S.S., Wong, L.P., Lim, C.P., 2019. An artificial bee colony algorithm with a Modified Choice Function for the traveling salesman problem. Swarm Evol. Comput. 44, 622–635.
Crawford, B., Soto, R., Astorga, G., et al., 2017. Putting continuous metaheuristics to work in binary search spaces. Complexity 1–19.
Cui, L., Li, G., Luo, Y., et al., 2018. An enhanced artificial bee colony algorithm with dual-population framework. Swarm Evol. Comput. 43, 184–206.
Deane, R.H., White, E.R., 1975. Balancing workloads and minimizing set-up costs in the parallel processing shop. J. Oper. Res. Soc. 26 (1), 45–53.
DING, Hai-jun, Qing-xian, F.E.N.G., 2009. Artificial bee colony algorithm based on Boltzmann selection policy. Comput. Eng. Appl. 45 (31), 53–55.
Dong, X., Lin, Q., Xu, M., et al., 2019. Artificial bee colony algorithm with generating neighbourhood solution for large scale coloured traveling salesman problem. Instit. Eng. Technol. 13 (10), 1483–1491.
Gao, W., Liu, S., Huang, L., 2013. A novel artificial bee colony algorithm based on modified search equation and orthogonal learning. IEEE Trans. Cybern. 43 (3), 1011–1024.
Gao, W., Wei, Z., Luo, Y., et al., 2018. Artificial bee colony algorithm based on Parzen window method. Appl. Soft Comput. 74.
Hernandez-Perez, H., Jose Salazar-Gonzalez, J., Santos-Hernandez, B., 2018. Heuristic algorithm for the split-demand one-commodity pickup-and-delivery travelling salesman problem. Comput. Oper. Res. 97, 1–17.
Jooda, J.O., Makinde, B.O., Odeniyi, O.A., et al., 2021. A review on hybrid artificial bee colony for feature selection. Glob. J. Adv. Res. 8, 170–177.
Karaboga, D., 2005. An Idea Based on Honey Bee Swarm for Numerical Optimization, Technical Report - TR06.
Karaboga, D., Basturk, B., 2007. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) Algorithm. J. Global Optim. 39 (3), 459–471.
Karaboga, D., Basturk, B., 2008. On the performance of artificial bee colony (ABC) algorithm. Appl. Soft Comput. 8, 687–697.
Karaboga, D., Gorkemli, B., 2019. Solving traveling salesman problem by using combinatorial artificial bee colony algorithms. Int. J. Artif. Intell. Tool. 28 (1).
Kaya, E., Gorkemli, B., Akay, B., et al., 2022. A review on the studies employing artificial bee colony algorithm to solve combinatorial optimization problems. Eng. Appl. Artif. Intell. 115.
Khan, I., Maiti, M.K., 2019. A swap sequence based artificial bee colony algorithm for traveling salesman problem. Swarm Evol. Comput. 44, 428–438.
Khomri, B., Christodoulidis, A., Djerou, L., et al., 2018. Retinal blood vessel segmentation using the elite-guided multi-objective artificial bee colony algorithm. IET Image Process. 12 (12), 2163–2171.
Kıran, M.S., İşcan, H., Gündüz, M., 2013. The analysis of discrete artificial bee colony algorithm with neighborhood operator on traveling salesman problem. Neural Comput. Appl. 23, 9–21.
Li, J.Q., Pan, Q.K., 2015. Solving the large-scale hybrid flow shop scheduling problem with limited buffers by a hybrid artificial bee colony algorithm. Inf. Sci. 316, 487–502.
Li, G., Cui, L., Fu, X., et al., 2016. Artificial bee colony algorithm with gene recombination for numerical function optimization. Appl. Soft Comput. 52, 146–159.
Patil, S.D., Ragha, L., 2020. An adaptive fuzzy based message dissemination and micro-artificial bee colony algorithm optimized routing scheme for vehicular ad hoc network. IET Commun. 14 (6).
Peng, B., Wu, L., Wang, Y., et al., 2021. Solving maximum quasi-clique problem by a hybrid artificial bee colony approach. Inf. Sci. 578, 214–235.
Qi, Y.H., Cai, Y.G., Huang, G.W., et al., 2021. Discrete fireworks algorithm with fixed radius nearest-neighbor search 3-opt for travelling salesman problem. Appl. Res. Comput. 38 (6), 1642–1647.
Sahin, O., Akay, B., Karaboga, D., 2021. Archive-based multi-criteria Artificial Bee Colony algorithm for whole test suite generation. Eng. Sci. Technol., Int. J. 24 (3), 806–817.
Santos, A.S., Madureira, A.M., Varela, M., 2016. Study on the impact of the NS in the performance of meta-heuristics in the TSP. In: IEEE International Conference on Systems.
Tasgetiren, M.F., Pan, Q.K., Suganthan, P.N., et al., 2011. A discrete artificial bee colony algorithm for the total flowtime minimization in permutation flow shops. Inf. Sci. 181 (16), 3459–3475.
Tsai, H.C., 2014. Integrating the artificial bee colony and bees algorithm to face constrained optimization problems. Inf. Sci. 258 (3), 80–93.
Wang, H., Wu, Z., Rahnamayan, S., et al., 2014. Multi-strategy ensemble artificial bee colony algorithm. Inf. Sci. 279, 587–603.
Wolek, A., Mcmahon, J., Dzikowicz, B.R., et al., 2021. The orbiting dubins traveling salesman problem: planning inspection tours for a minehunting AUV. Aut. Robots 45 (3), 31–49.
Xu, W.H., Zhang, G.R., Wei, C.X., et al., 2021. Imperialist competitive algorithm based on adaptive inheritance strategy to solve traveling salesman problem. Appl. Res. Comput. 38 (11), 3349–3353.
Zeng, T., Wang, W., Wang, H., et al., 2022. Artificial bee colony based on adaptive search strategy and random grouping mechanism. Expert Syst. Appl. 192.
Zhao, Y., Liu, H., Gao, K., 2020. An evacuation simulation method based on an improved artificial bee colony algorithm and a social force model. Appl. Intell. 51, 100–123.
Zhong, Y., Lin, J., Wang, L., et al., 2017. Hybrid discrete artificial bee colony algorithm with threshold acceptance criterion for traveling salesman problem. Inf. Sci. 421, 70–84.
Zhu, G., Kwong, S., 2010. Gbest-guided artificial bee colony algorithm for numerical function optimization. Appl. Math. Comput. 217 (7), 3166–3173.