TU Delft

Delft University of Technology

A Q-learning based multi-strategy integrated artificial bee colony algorithm with application in unmanned vehicle path planning

Ni, Xinrui; Hu, Wei; Fan, Qiaochu; Cui, Yibing; Qi, Chongkai

DOI
10.1016/j.eswa.2023.121303

Publication date
2024

Document Version
Final published version

Published in
Expert Systems with Applications

Citation (APA)
Ni, X., Hu, W., Fan, Q., Cui, Y., & Qi, C. (2024). A Q-learning based multi-strategy integrated artificial bee colony algorithm with application in unmanned vehicle path planning. *Expert Systems with Applications*, *236*, Article 121303. https://doi.org/10.1016/j.eswa.2023.121303

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.
For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.

Green Open Access added to TU Delft Institutional Repository

'You share, we take care!' - Taverne project

https://www.openaccess.nl/en/you-share-we-take-care

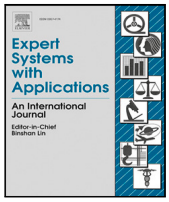
Otherwise as indicated in the copyright section: the publisher is the copyright holder of this work and the author uses the Dutch legislation to make this work public.

Expert Systems With Applications 236 (2024) 121303

Contents lists available at ScienceDirect

Expert Systems With Applications

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



A Q-learning based multi-strategy integrated artificial bee colony algorithm with application in unmanned vehicle path planning

Xinrui Ni [a], Wei Hu [b,*], Qiaochu Fan [c], Yibing Cui [d], Chongkai Qi [a]

[a] School of Traffic and Transportation, Beijing Jiaotong University, Beijing 100044, PR China
[b] School of Systems Science, Beijing Jiaotong University, Beijing 100044, PR China
[c] Delft Institute of Applied Mathematics, Delft University of Technology, 2628 CD, Delft, The Netherlands
[d] School of Mathematics and Statistics, Beijing Jiaotong University, Beijing 100044, PR China

ARTICLE INFO

Keywords:
Meta-heuristic algorithm
Artificial bee colony algorithm
Multi-strategy
Q-learning framework
Path planning

ABSTRACT

Artificial bee colony (ABC) is a prominent algorithm that offers great exploration capabilities among various meta-heuristic algorithms. However, its monotonous and one-dimensional search strategy limits its searching performance in the solving process. Thus, to address this issue, a Q-learning based multi-strategy integrated ABC algorithm (QMABC) is proposed. In the QMABC, multiple search strategies are proposed to utilize different individual experiences and search approaches for solution updates. Then, Q-learning is employed for strategy selection. In comparison to previous studies, this paper introduces more effective state and action configurations within the framework of Q-learning. To evaluate the performance of the QMABC, CEC 2017 benchmark functions are adopted to compare it to different meta-heuristic algorithms including ABC based and non-ABC based algorithms. Moreover, applications in path planning are implemented to further verify the effectiveness of the QMABC. Overall, it should be highlighted that the proposed QMABC demonstrates superiority in both numerical and practical experiments.

1. Introduction

Nowadays, multiple industries are transitioning towards intelligence and automation in the revolution brought about by computer technology. This shift has resulted in the emergence of increasingly complex operation research problems. These problems share troublesome features, including nonlinear, high-dimensional, nondifferentiable, etc., which lead to the inapplicability of traditional exact optimization algorithms and heuristic algorithms. In this case, meta-heuristic algorithms have become a hotspot for their excellent performance. Evolutionary algorithms constitute a significant category within the realm of meta-heuristic algorithms. In general, they are a combination of stochastic algorithms and local search. Meanwhile, they have the benefits of simple theory and high efficiency. Genetic algorithm (GA) (Holland, 1992), particle swarm optimization (PSO) (Kennedy & Eberhart, 1995), differential evolution (DE) (Storn & Price, 1997), artificial bee colony (ABC) (Karaboga & Basturk, 2007), and ant colony optimization (ACO) (Dorigo et al., 1996) are the paradigms of evolutionary algorithm.

Among the various algorithms, the ABC algorithm has good exploration ability and robustness. Thus, it is applied to a wide range of optimization problems, including workshop scheduling (Pan et al., 2011), vehicle path planning (Szeto et al., 2011), and market forecasting (Hsieh et al., 2011). However, the exploration and exploitation of the algorithm are always contradictory. At the same time, the search equation of ABC is simple and only one dimension of information is updated in each iteration, which results in a moderate exploitation ability and slow convergence speed (Liao et al., 2012). Therefore, lots of researchers have attempted to optimize it.

In the past few years, plenty of ABC variants have been modified to address its limitations. These variants have been created from different perspectives, including parameter adjustment, search equation improvement, algorithm scheme modification, and so on Cui et al. (2023). Nevertheless, a single search equation can hardly satisfy the requirements for effectiveness in every situation. Under the circumstances, some ABC variants address this problem by importing multiple strategies. For example, Chen et al. (2019) introduced three DE strategies, namely "rand/1/bin", "current-to-pbest/1/bin", and "current-to-rand/1", to the employed bee phase to replace the original strategy. Cao and Shi (2021) also added a DE strategy and an elite-guided strategy into the onlooker bee phase. Yong et al. (2021) applied three

* Corresponding author.
E-mail addresses: 19252014@bjtu.edu.cn (X. Ni), huwei@bjtu.edu.cn (W. Hu), q.fan-1@tudelft.nl (Q. Fan), yibing.cui@centralelille.fr (Y. Cui), 19252017@bjtu.edu.cn (C. Qi).

https://doi.org/10.1016/j.eswa.2023.121303
Received 27 March 2023; Received in revised form 28 July 2023; Accepted 22 August 2023
Available online 24 August 2023
0957-4174/© 2023 Elsevier Ltd. All rights reserved.

search strategies from DE, GA, and PSO to the employed bee phase. Brajevic et al. (2020) combined the search equations from the firefly algorithm and ABC, and made choices through a variance operator.

Although these modifications have made progress in the performance of the ABC algorithm, there are still several drawbacks. Firstly, although these works consider a variety of strategies, they are essentially homogeneous. The different strategies share the same decision mechanism for determining the number of dimensions to be updated and the similar form of search equation. Secondly, these strategy selection mechanisms are not robust. Due to the large randomness of the meta-heuristic algorithm, both the changes in fitness value and the number of successfully updated individuals will fluctuate dramatically in a single iteration. This can lead to bias in the strategy selection operators.

To alleviate the first issue, we choose strategies that vary significantly. The differences can be reflected in the following aspects: the number of dimensions to be updated can be merely one, entire, or stochastic; the guidance solution can be a randomly selected solution, global optima, or neighborhood optima; the base solution can move towards the target solution straightly or spirally, etc. By incorporating these strategies, they can meet different situations and complement each other when used together. To address the second issue, we focus on the Q-learning algorithm (Lee & Lee, 2021). As a typical RL algorithm, Q-learning can solve various problems through a decision-making agent learning from the interactions with its environment (He et al., 2023). The formation of the Q-table is influenced by the results of calculations over a period of time. In addition, the set of actions is relatively flexible. For example, the actions we set in this paper do not directly select a certain strategy but indirectly make decisions by changing the probability of selecting strategies in small steps. In this way, the strategy selection method through Q-learning has strong robustness because it is no longer sensitive to the results in a few iterations. Based on these considerations, we propose a novel ABC variant called Q-learning based multi-strategy integrated ABC (QMABC).

To verify the validity of the proposed QMABC algorithm, it is compared to the ABC variants and other improved meta-heuristic algorithms on the CEC 2017 benchmark functions in different dimensions. The experimental results illustrate that the QMABC algorithm is superior to the competitors in searching accuracy and convergence rate. Moreover, it is applied to the unmanned vehicle path planning. Compared to classical path planning methods, the proposed algorithm shows great superiority in the searching accuracy and execution time.

The main contributions of this paper can be summarized as follows:

(1) Several novel strategies are introduced into the algorithm, including one in the employed bee phase to enhance exploration capabilities and three in the onlooker bee phase to improve exploitation capabilities.
(2) Q-learning algorithm is utilized to adaptively determine the most suitable strategy for the current population. Different from the existing works (Wang et al., 2022c; Zhou & Zhao, 2023), in our Q-learning framework, the state setting focuses on the update status of the whole population instead of individuals. Strategies are indirectly selected based on probability values. It further enhances the search ability of the algorithm.
(3) Compared with ABC variants and other meta-heuristic algorithms, the proposed QMABC shows great superiority in terms of search accuracy, convergence speed, and robustness. Besides, the proposed QMABC algorithm also shows super performance compared with other path planners in both two-dimensional and three-dimensional path planning problems.

The remainder of this paper is organized as follows. Literature reviews and some relevant algorithms are presented in Section 2. Section 3 elaborates on the proposed QMABC algorithm in detail. In Section 4, numerical experiments are conducted to test the effectiveness of the proposed algorithm. Then, Section 5 applies the algorithm to concrete unmanned vehicle path planning problems. Finally, we give the conclusion in Section 6.

2. Preliminaries and related work

In this section, two basic algorithms (ABC and Q-learning) will be introduced, and overall reviews of relevant literature will be made.

2.1. The standard ABC algorithm

The ABC algorithm was first proposed by Karaboga and Basturk (2007), inspired by the behavior of bee colonies. In the bee colonies, the entire bees are divided into three types: employed bees, onlooker bees, and scout bees. The three types of bees cooperated in the process of honey gathering. Accordingly, the ABC algorithm imitates the behavior of three kinds of bees to achieve good optimization results.

2.1.1. Implement of ABC algorithm

There are four phases in the ABC algorithm, which are explained in detail in this section.

Initialization phase: ABC needs to generate a certain number of solutions as the initial population. We assume that the population size is SN and the dimension of one solution is D, then the form of solutions is $x_i = (x_{i,1}, x_{i,2}, \dots, x_{i,D})$, $i \in \{1, 2, \dots, SN\}$. For the jth dimension of the ith solution, ABC uses Eq. (1) to generate a feasible number.

$$x_{i,j} = x_j^{lower} + rand(0,1) \times (x_j^{upper} - x_j^{lower}), \tag{1}$$

where x_j^{lower} and x_j^{upper} are the lower bound and upper bound of the jth dimension. Similar to many EAs, the initial solutions of ABC are randomly generated with uniformly distributed probabilities within the range of feasible domains in each dimension.

Employed bee phase: ABC enters the loop and executes the three main phases in a continuous sequence. In this phase, each individual searches the entire solution space for a better candidate solution based on its corresponding food source. This step is primarily responsible for exploring the solution space. The search equation is shown below:

$$v_{i,j} = x_{i,j} + \phi \times (x_{i,j} - x_{k,j}), \tag{2}$$

where $k \neq i$ is a random number selected from $\{1, 2, \dots, SN\}$, ϕ is a random number with uniform distribution in the range of $[-1, 1]$, and $v_{i,j}$ is the jth dimension of the candidate solution.

Then the objective function value is used to evaluate the candidate solution and the original solution, and a better solution is chosen via the greedy choice strategy:

$$x_i^{new} = \begin{cases} x_i, & f(v_i) > f(x_i), \\ v_i, & otherwise. \end{cases} \tag{3}$$

It should be noted that ABC has a counter $trial_i$ to count the number of unsuccessful updates for solution i. If $f(v_i) \leq f(x_i)$, then v_i replaces x_i and $trial_i$ is set to 0; otherwise $trial_i$ is added by 1.

Onlooker bee phase: In the onlooker bee phase, new food sources are searched based on information about promising food sources. This phase mainly aims to achieve exploitation in the vicinity of high-quality solutions. To facilitate the calculation of the probability of each solution being selected, it is necessary to convert the objective function $f(x_i)$ into a fitness function fit_i through Eq. (4) as:

$$fit_i = \begin{cases} \dfrac{1}{1 + f_i}, & f_i \geq 0, \\ 1 + |f_i|, & otherwise, \end{cases} \tag{4}$$

where fit_i is always greater than 0 and inversely proportional to $f(x_i)$. A roulette wheel selection mechanism is used to choose base vector x_i with probability p_i as:

$$P_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n}. \tag{5}$$

It can be found that the larger the solution fitness value, the greater the probability of being selected. Then the process is the same as

that of the employed bee phase, including computing of the candidate solutions (Eq. (2)), making greedy choices (Eq. (3)) and updating the counter $trial_i$.

Scout bee phase: The scout bee phase aims to prevent the solution from falling into the local optimum by regenerating solutions that cannot be successfully updated for a certain number of iterations. If the counter $trial_i, i \in \{1, 2, \ldots, SN\}$ reaches the boundary conditions $trial_{limit}$, a novel solution will be generated by Eq. (1) and the $trial_i$ will be reset to 0.

2.1.2. Literature review of ABC algorithm

Currently, the mainstream optimization ideas of swarm intelligence algorithms can be categorized into three types (Das et al., 2016). The modification of the ABC algorithm is no exception (Karaboga et al., 2014). The first type is the optimization of the search equations, which adds or removes items or optimizes and adapts parameters in the search equations to reduce the number of manually defined parameters or obtain the balanced unity of the parameter values in different problem situations. The second is the combination of different algorithms to compensate for the shortcomings of a single algorithm so that it can achieve more preferable global and local search ability. The third is to carry on additional supplements or enhancements to the algorithm. In many cases, researchers will combine multiple types of ideas simultaneously to achieve better results in an optimization problem.

For the first type of thinking: Zhu and Kwong (2010) believed that the search equation of the traditional ABC algorithm moves a solution to another randomly selected solution, making it uncertain whether better solutions can be found. Although the algorithm has a stronger exploration ability, it will also greatly reduce the convergence speed. Therefore, the author proposed GABC algorithm, which combines the concept of the current global optimal solution in the PSO algorithm and adds the experience of the current global optimal solution into the search equation of ABC algorithm to guide the search for new solutions. This strategy improved the search efficiency and solving accuracy of ABC algorithm which was later used by many researchers. Xu et al. (2020) and Zhou et al. (2021) adopted the current global optimal solution-oriented strategy in their papers to guide the generation of new solutions. Based on the GABC algorithm, Zhao et al. (2022) proposed PAABC algorithm, which adjusted the combination of group experience by studying the coverage of the search equation and adaptively adjusted the search step size based on individual fitness, thus reducing the probability of invalid search and further improving the search efficiency and solving accuracy of the algorithm.

In addition to the search equation considering the current global optimal solution, Zhou et al. (2022) took into account the experience of the neighborhood optimal solution and used the current local optimal solution to guide exploitation, thereby, significantly improving the local search ability.

From another perspective, Akay and Karaboga (2012) found that the range of perturbation value significantly affects the exploration and exploitation ability of the algorithm. Therefore, their algorithm adaptively changed the scale of this item to use different disturbance values for different search stages.

For the second type of thinking: The combination of ABC algorithm and DE algorithm is the most mainstream one. Gao et al. (2012) first introduced the mechanism of DE into the ABC algorithm, replacing the original search strategy with the strategy of DE. Such a combination can effectively improve the coverage degree of the algorithm's search space and significantly improve the algorithm's accuracy and convergence speed. Chen et al. (2019) further improved this combination by introducing three different DE search equations. Therefore, three search equations that perform differently were introduced at the same time. In practical application, the selection probability of the three strategies was adaptively allocated according to the actual optimization effect, and good results were achieved. Ustun et al. (2022) combined the DE in a more detailed way, and only adopted the crossover and mutation strategies in the onlooker bee phase to achieve better results.

In addition to combining with DE, Pan et al. (2017) proposed a compact ABC algorithm to combine the idea of the distribution estimation method, which used truncated normal distribution to replace the actual population, and finally converged to the optimal solution by updating the variance and mean. Banitalebi et al. (2015) also optimized the compact ABC algorithm to replace the truncated normal distribution with a segmented probability distribution, while increasing the perturbation of the probability distribution to increase randomness. This method can greatly save space and reduce the complexity of the algorithm, but it has the problem of low accuracy compared with the algorithm with the actual population.

Etminaniesfahani et al. (2022) considered the newer Fibonacci intelligent algorithm, embedded it into the onlooker bee phase of ABC, and selected either the original ABC strategy or the FA strategy according to a certain probability.

For the third type of thinking: Researchers also proposed additional supplements and strengthened the algorithm mechanism. Gao and Liu (2012) adopted the initial solution generation method based on opposition learning and a chaotic system, which effectively increased the diversity of initial solutions and enhanced the efficiency of the algorithm. Cui et al. (2017) adopted the fitness ranking rather than fitness to determine the selection probability. At the same time, adaptive parameters were used to adjust the selection probability of individuals assigned by ranking in different search periods, to balance the global and local search ability.

Liao et al. (2012) dynamically adjusted the probability of updating each dimension of the solution, which indirectly changed the number of updated dimensions. Cui et al. (2022) have also improved upon this mechanism by incorporating the idea of RL. Each time the search equation was used to update the solution, the number of updated dimensions was determined by the current situation and reward mechanism so that the program can automatically find the updated degree of the relatively suitable solution at different stages. It greatly improved the efficiency of the algorithm and the success rate of the updated solution. Aiming at the actual discretization problem, Lei and He (2022) developed three variants for the onlooker bee phase and introduced adaptive selection indexes for pattern selection.

Wang et al. (2022) introduced a significant change to the mechanism of the ABC algorithm, enabling each individual could continuously change according to the solution situation in the stage of three kinds of bees, and realized the dynamic balance of exploration and exploitation. This paper also created a new idea for the optimization of ABC algorithm. Zhou et al. (2023) incorporated fitness landscape analysis to assess the smoothness of the landscape of the solution space based on the population's diversity. If the landscape was smooth, a strategy more inclined towards exploitation was selected; conversely, if the landscape was relatively rugged, a strategy more inclined towards exploration was chosen.

2.2. The Q-learning algorithm

2.2.1. The basic idea of Q-learning algorithm

Since the 1980s, machine learning (ML) has attracted widespread interest in the field of artificial intelligence and it has been utilized in metaheuristics in the current year (Fister et al., 2016). The four main approaches in the field of ML are supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning (RL) (Sarker, 2021). Among them, RL is a kind of environment-driven approach that relies on reward or penalty and its objectives in order to take action to increase the reward or minimize the risk. With its unique advantages, RL is widely adopted in various fields, including the field of predictive analytics and intelligent decision-making, smart city and energy construction, traffic prediction and transportation, etc.

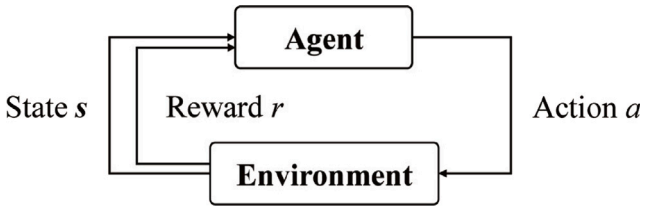


Fig. 1. Q-learning framework.

However, RL may not be the preferred approach for solving the basic or straightforward problems.

In the development of RL systems, Watkins and Dayan (1992) combined the theory of optimal control including the Bellman equation and Markov decision process together with temporal-difference learning to form a well-known Q-learning. Q-learning has three characteristics, which are model-free, off-policy, and bootstrapping. These give Q-learning plenty of benefits like stability, simplicity, rapidity, and adaptability to a larger state-space environment (Nguyen et al., 2020).

To be precise, the framework of Q-learning is displayed in Fig. 1. It involves four components of Q-table, state s, reward r, and action a. The agent recognizes state s_t from the environment and then takes a promising action a_t to execute based on the Q-table. After applying the action, the environment will be changed to state s_{t+1} and return the reward r to the agent. Then the Q-table will be updated through Eq. (6).

$$Q_{new}(s_t, a_t) = Q(s_t, a_t) + \alpha \times [r_{t+1} + \gamma \max_{a \in A} Q(s_{t+1}, a) - Q(s_t, a_t)], \tag{6}$$

where Q_{s_t,a_t} is the Q-value of taking action a_t at state s_t, α is the learning rate that controls the influence of past experiences and present reward, and γ is the discount rate. So far, a whole iteration has been completed. And it will continue until the terminal state is achieved. The pseudo code of Q-learning is shown as Algorithm 1.

Algorithm 1: Pseudo code of Q-learning algorithm

```
begin
    Initialize Q(s, a) arbitrarily;
    Choose a initial state s_1 randomly;
    while state s is not terminal do
        Choose action a_t from s_t using policy derived from Q-table;
        Take action a_t, obtain reward r_{t+1} and state s_{t+1};
        Update Q-table via Eq. (6)
    end while
end
```

2.2.2. Review of Q-learning based ABC

Currently, many researchers have incorporated RL into metaheuristic algorithms to enhance their accuracy, efficiency, and robustness. There are two main categories of ideas in the research of combining ABC with Q-learning. The first category focuses on utilizing Q-learning to adaptively adjust the parameters in the algorithm, aiming to achieve improved search effectiveness. The second category uses Q-learning to select the most suitable strategy from multiple search strategies by considering the corresponding search state.

In the first kind of idea, Long et al. (2022) defined states by considering the weighted average value of optimal fitness value, average fitness value, and population diversity within the algorithm population. Based on these states, the number of dimensions for updating is determined. Cui et al. (2022) defined states of Q-learning by considering whether the current search successfully updates the solution, and it was also adopted to determine the number of update dimensions.

In the second kind of idea, Wang et al. (2022c) initially designed multiple search strategies and employed Q-learning for strategy selection in ABC. The states and rewards of Q-learning were determined based on the updating status of individuals, while the actions involved the direct selection of a particular strategy. Similar approaches were proposed by Li et al. (2023) and Wang et al. (2022d), where the main ideas were analogous to the last study. However, these researches only considered the influence of individual update status when setting the states of Q-learning. In contrast, Jia et al. (2021) revealed that the overall performance of the population played a more crucial role in the adaptive adjustment process of RL. In this case, Zhou and Zhao (2023) introduced a distinct state and reward mechanism, which depended on whether the population had any improvement in one iteration. Nevertheless, its state setting remains simplistic, as it does not consider the variations in population update efficiency. Furthermore, within such kind of idea, the action setting remains simple, where all strategy selections are directly determined solely based on the values presented in the Q-table.

3. Proposed QMABC algorithm

As mentioned in Section 2, each of its three phases plays a different role. The employed bee phase is mainly used to explore the global solution space. The onlooker bee phase is mainly used to exploit local solution space. The scout bee phase is used for the regeneration of solutions trapped in local optimality. The effective coordination between these three phases allows ABC to achieve an excellent balance between exploration and exploitation. However, due to its single search equation and weak local search capability, we can further expand its global and local search abilities under the original ABC framework to achieve better solving precision and convergence speed of the algorithm.

In the remainder of this section, we will elaborate on several strategies and the framework of Q-learning for strategy choices.

3.1. Modified employed bee phase

Many researchers introduced the equation of DE into different algorithms as the unique search equation of DE allows the population to maintain a high degree of diversity and covers a large portion of the solution space (Bhattacharya & Chattopadhyay, 2010; Cui et al., 2022; Liu et al., 2010). Therefore, to further enhance the global search ability of the algorithm, we add an improved DE search equation into the employed bee phase.

In DE algorithm, the following two search equations are commonly used (Das et al., 2016):

$$DE/rand/1: v_{i,j} = x_{r1,j} + F \times (x_{r2,j} - x_{r3,j}), \tag{7a}$$

$$DE/current-to-best/1: v_{i,j} = x_{i,j} + F \times (x_{best,j} - x_{i,j}) + F \times (x_{r1,j} - x_{r2,j}). \tag{7b}$$

The $DE/rand/1$ takes x_{r1} as the base vector and adds a perturbation item related to two randomly selected vectors. Similarly, the base vector of $DE/current-to-best/1$ is the current solution itself and the mutated solution itself, respectively. Considering that our goal is to enhance the exploration in this phase, more individual participation can bring more mutation possibilities. It is more appropriate to use a stochastic solution as the base vector to improve the diversity of individuals (Ghosh et al., 2011; Mallipeddi et al., 2011). However, too many participants in the mutation process can easily lead to a large mutated value outside the domain. Thus, we propose a modified search equation by combining Eqs. (7a) and (7b) and further replace the current global best item with a random one. Moreover, the mutation coefficient F is substituted by a uniformly distributed random number to increase the stochasticity. Finally, the improved search equation is displayed in Eq. (8) as:

$$v_{i,j} = x_{r1} + \phi_{i,j} \times (x_{r2} - x_{r1}) + \varphi_{i,j} \times (x_{r3} - x_{r1}), \tag{8}$$

where $\phi_{i,j}$ and $\varphi_{i,j}$ are both uniform random numbers in the range of $[-1,1]$, $r1, r2, r3$ are random numbers in $\{1,2,\ldots,SN\}$. In Eq. (8), the first two items take x_{r1} as the base vector and perturb in the scope of $[2x_{r1}-x_{r2}, x_{r2}]$ when $x_{r1} \le x_{r2}$ or $[x_{r2}, 2x_{r1}-x_{r2}]$ when $x_{r1} > x_{r2}$. They give the search equation the ability to cover all the points in the range with equal probability. To further enhance the randomness of this search equation, we introduce an additional third item to enable the mutated point ulteriorly to move within the radius of the distance between x_{r3} and x_{r1} based on the original perturbance range. The enhanced randomness allows the search equation to perform stronger exploration capability.

$$x_{i,j} = \begin{cases} x_{i,j}, & rand_j > CR, \\ v_{i,j}, & otherwise. \end{cases} \tag{9}$$

We also take the number of updated dimensions into account as Eq. (9). For each dimension of the solution, whether it will mutate using the search equation Eq. (8) is fully based on the size relationship between a random number and crossover rate CR. It gives the search process more degree of freedom to enhance the exploration.

3.2. Three novel search strategies in onlooker bee phase

When it comes to the local search ability of the search equation, its impacts vary when the form and dimension of the problems are different. In fact, none of the search equations can always perform best in every function case. Instead, they only have excellent effects in certain situations. For this reason, more and more researchers have adopted multiple search equations in algorithm development to cope with different situations (Chen et al., 2019). Therefore, in the onlooker bee phase, we also aim to develop several powerful search equations to improve the local search ability of the algorithm and provide stability in diverse scenarios.

Strategy 1: Spiral approximation strategy

Among the many meta-heuristic algorithms, the whale optimization algorithm (WOA) and its variants (Abd El Aziz et al., 2017; Mirjalili & Lewis, 2016) have a distinctive strategy. In this strategy, movements from the base vector to the target vector are not in a straight line direction. Instead, the base vector takes the current global best solution as the target vector and moves towards it with a spiral trajectory. The search equation can be expressed as follow:

$$v_{i,j} = x_{gb,j} + |x_{gb,j} - x_{i,j}| \times e^{b \times r} \times cos(2\pi r), \tag{10}$$

where x_{gb} is the current global best solution, r is a uniformly distributed random number in the range of $[-1,1]$, and b is a constant parameter. When this strategy is activated, it updates every jth dimension for $j \in \{1,2,\ldots,SN\}$. Through Eq. (10), we can observe the attributes of this strategy. Firstly, the location update of the solution x_i is only related to the position of x_{gb} and the absolute distance between x_{gb} and x_i. Secondly, the approaching step size of the second item in Eq. (10) is determined by a composite function with exponential and cosine parts. So the approaching step size to the x_{gb} is more inclined to move in small steps instead of rapidly getting close to it. Because of these attributes, this strategy possesses great exploitation ability. Besides, owing to its full-dimensional update feature and spiral search mode, each individual has the possibility of falling on any side of the global best solution. Therefore, this search equation exhibits a higher capability for exploiting regions around the global best solution.

In this paper, we modify this strategy to make it more flexible in different search stages. Parameter b is turned into an adaptive parameter related to the function evaluations as:

$$b = 1 - \frac{FES}{FES_{max}}, \tag{11}$$

where FES is the number of objective function evaluations, FES_{max} is the maximum number of calls. After modification, parameter b will gradually decrease from 1 to 0 as FES increases. We display three cases with $b = 1, 0.5, 0.1$ in Fig. 2 to illustrate the effect of different parameter values.

We can see that the number of FES is small, and the value of parameter b is close to 1 in the early stage of algorithm execution. At this time, the possible positions of the updated solution are distributed around the center of the current global optimal solution. The distance between the new solution v_i and x_{gb} might be slightly greater than that between x_i and x_{gb}. With the progress of the loop, the value of b decreases, then the possible position of the new solution is gradually reduced to a smaller area around the global optimal solution. This mechanism allows the search equation to exploit in a larger area in the early period and a tendency to further exploit around x_{gb} in the later period. It nicely adjusts the choice of exploitation scope at different times.

Strategy 2: Global and neighborhood best guide strategy

In previous research, many ABC variants show that the elite individuals have been utilized to guide search (Xiang et al., 2015). For example, Zhu and Kwong (2010) proposed the global best guided ABC algorithm to take advantage of the valuable information from the superior individual. However, there is a risk that the best individual can be easily overutilized. To alleviate this problem, some neighborhood elite based ABC are proposed. Neighborhood guidance enhances the search in the local area and eliminates the probability of prematurity. Nevertheless, the global search ability is diminished in this way. Thus, by considering both the current global best individual and the neighborhood best individual, we make a fusion of such elite guidances and propose our strategy in Eq. (12) as follows:

$$v_{i,j} = x_{i,j} + c_1 \times \phi_{i,j} \times (x_{r1,j} - x_{i,j}) + c_2 \times \varphi_{i,j} \times (x_{gb,j} - x_{i,j}) + c_3 \times (x_{nb,j} - x_{i,j}), \tag{12}$$

where $x_{gb,j}$ is the jth dimension of the current global optimal solution, $x_{nb,j}$ is the jth dimension of the neighborhood optimal solution. In this paper, neighborhood represents an exact scope around the base solution. The identification of the neighborhood solutions depends on the size relationship between the distance of two solutions and the neighborhood radius based on Eq. (13). In Eq. (13), d donates the Euclidean distance defined as:

$$d_{i,j} = \sqrt{\sum_{n=1}^{D} (x_{i,n} - x_{j,n})}. \tag{13}$$

According to existing research, if the $d_{i,j}$ is smaller than the threshold value, then the two solutions can be regarded as neighborhood solutions to each other. The threshold value is usually set to 10 (Zhou et al., 2022).

In Eq. (12), the orientation of the two elite solutions is apparent, which is more likely to trigger premature convergence of the solution process. Therefore, to slow down the convergence speed and strengthen the exploitation ability, only one dimension $j \in \{1,2,\ldots,SN\}$ is selected for an update each time.

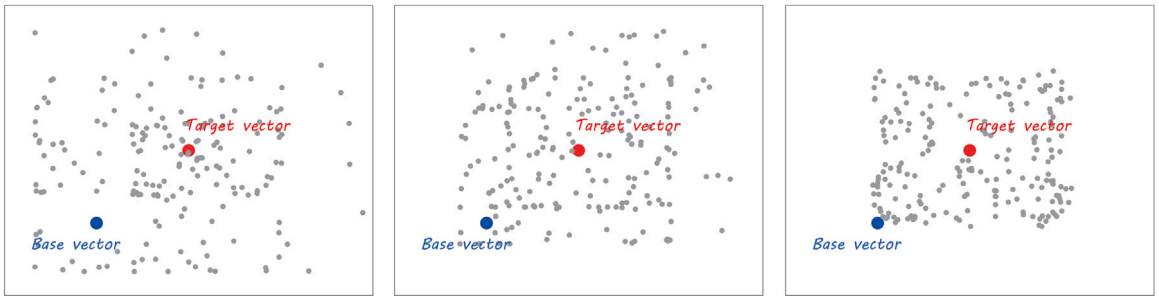
Besides, c_1, c_2, c_3 are three adaptive parameters to adjust search focus in different periods.

$$c_1 = \frac{f(x_{i,j}) - f(x_{r1,j})}{f(x_{i,j}) + f(x_{r1,j})}, \tag{14}$$

$$c_2 = 2 \times \frac{trial}{trial_{limit}}, \tag{15}$$

$$c_3 = 2 \times \left(1 - \frac{trial}{trial_{limit}}\right). \tag{16}$$

We believe that the fitness value of each solution can reflect its ability to guide search. In Eq. (14), c_1 determines the degree that controls how much to accept the experience of a randomly selected solution. When the difference between the fitness values of x_{r1} and x_i is large, it means that x_{r1} is much worse than x_i. At this time, the value of c_1 will be small, and the second item will guide the search away from



(a) b=1 (b) b=0.5 (c) b=0.1

Fig. 2. Location of points generated through strategy 1.

x_{r1}, and vice versa. c_2 and c_3 are two operators that determine which solution dominates the search direction between x_{gb} and x_{nb}. During the iteration, the value of c_2 increases, and the value of c_3 decreases as the unsuccessful update counter $trial_i$ increases. That is, x_{nb}'s experience counts more when there are fewer unsuccessful updates, and x_{gb}'s experience is considered more after a long period of unsuccessful updates. We adopt the operators setting as there may be better solutions near x_i that need to be discovered when the $trial_i$ is relatively small, and a premature move to x_{gb} will ignore the exploitation of neighborhood space. When the number of unsuccessful updates increases, it indicates that the neighborhood information has been fully developed, so further exploitation is made around the x_{gb}.

Strategy 3: Low dimensional dominance strategy

The strategies mentioned above are effective for high-dimensional problems but do not perform as well in low-dimensional problems. According to previous research and experimental results, the search equation $DE/rand/1$ has demonstrated exceptional performance in low-dimensional problems. Therefore, as a supplement to the above two strategies, we add the strategy shown in Eq. (17) in the onlooker bee phase as follows:

$$v_{i,j} = x_{i,j} + F \times (x_{r1,j} - x_{r2,j}). \tag{17}$$

This search equation is identical to the standard DE search equation, and the number of dimensions to update is also determined by the crossover rate CR through Eq. (9).

3.3. Q-learning framework for strategy selection

Although it is possible to introduce multiple search equations, it can be challenging to determine which one to use at a particular time. Q-learning, as one of the classical RL algorithms, can have a self-learning mechanism through the interaction between the reward and the environment. Therefore, we consider that the selection of the search equation can be entrusted to the Q-learning algorithm. In this way, a more appropriate strategy can be adopted based on the current calculation situation.

In this section, the method of using Q-learning to instruct strategy choice will be elaborated. To be specific, our algorithm introduces parameters p_1, p_2, p_3 that control the probability of each strategy being selected. Q-learning algorithm dynamically adjusts each probability during iteration. We do not use Q-learning directly to determine the chosen strategy for the following two reasons. Firstly, finding new solutions by search equations is a random process. Each strategy has the potential to find better solutions but they differ from each other for different probability of finding them. Secondly, some strategies may be more difficult to guide successful updates, but they are more effective once they succeed. For these reasons, it is necessary to allow for each strategy to be selected while avoiding too many ineffective updates.

Seven different actions can be taken to adjust the probability of choosing a strategy. Namely, at same, $p_1 + \mu$, $p_1 - \mu$, $p_2 + \mu$, $p_2 - \mu$, $p_3 + \mu$, $p_3 - \mu$, respectively. μ is the step size to adjust the probability. Meanwhile, the summation of the probability is 1, so the three parameters have only two degrees of freedom. Therefore, the seven actions can be reduced to five and the actions $p_3 + \mu$ and $p_3 - \mu$ can be deleted. Furthermore, to ensure that each strategy has a possibility of being chosen regardless of its effectiveness, two predetermined parameters lb and ub are set, where lb and ub are the lower bound and upper bound for the probability values p_i, p_j, p_k. That is, the probability values will be constrained in a range of $[lb, ub]$. It is important to note that as one of the parameters increases or decreases, the remaining two parameters decrease or increase in proportion to their values. For example, if $p_i, i \in \{1, 2, 3\}$ is added by μ, then the other two probability values $p_j, p_k, i \neq j \neq k$ can be calculated as:

$$p_j = p_j - \mu \times \frac{(p_j - lb)}{|p_j - lb| + |p_k - lb|}, \tag{18a}$$

$$p_k = p_k - \mu \times \frac{(p_k - lb)}{|p_k - lb| + |p_j - lb|}. \tag{18b}$$

Similarly, if the action being performed is $p_i - \mu$, the parameter lb in Eq. (18) will be replaced by the ub.

In the setting of the states, we consider the update situation of the population from a macro perspective. This is because judging by the individual's single success or failure is highly biased. Moreover, all individuals' updates depend on the experience of other individuals in the population so that the effectiveness of a strategy is more influenced by the population structure. To assess the population update situation under the current stage, we introduce a counter $Trial_i, i \in \{1, 2, \ldots, SN\}$ to record the number of unsuccessful updates during the onlooker bee phase. Similar to the counter $trial_i$ in the standard ABC, the counter $Trial_i$ is reset to zero when the ith solution undergoes a successful update, and it is incremented by one otherwise. The distinction lies in its activation solely during the onlooker bee phase. Thus, we set two states depending on Eq. (19) as:

$$n_s^G = SN - \sum_{i=1}^{SN} \max\left(Trial_i^G - Trial_i^{G-1}, 0\right), \tag{19}$$

where $Trial_i^G$ is the value of the unsuccessful update counter of the ith solution after iteration G.

Based on the above definition, the two states are defined as below:

s_1: $n_s^G \geq n_s^{G-1}$, the number of successfully updated solution is larger than or equal to that in the last iteration. Under the circumstances, the reward value is set as 1;

s_2: $n_s^G < n_s^{G-1}$, the number of successfully updated solution is less than that in the last iteration. Under the circumstances, reward value is set as -1.

To be precise, the structure of Q-table for strategy selection is shown in Table 1.

In Table 1, $Q(s_i, a_j)$ is the Q-value of taking action a_j at state s_i. Notice that, the population shares the same Q-table. In addition, many researchers adopt the $\epsilon - greedy$ mechanism for action selection in

Table 1
Q-table in QMABC.

		Action				
		a_1	a_2	a_3	a_4	a_5
state	s_1	$Q(s_1,a_1)$	$Q(s_1,a_2)$	$Q(s_1,a_3)$	$Q(s_1,a_4)$	$Q(s_1,a_5)$
	s_2	$Q(s_2,a_1)$	$Q(s_2,a_2)$	$Q(s_2,a_3)$	$Q(s_2,a_4)$	$Q(s_2,a_5)$

recent works, which allows the Q-learning method to balance the exploration and the exploitation well. However, the number of successful update maintains a certain degree of inherent randomness, so it is unnecessary to introduce the above mechanism. Thus, we follow the simple greedy selection to choose actions.

To summarize, the pseudo code is exhibited in Algorithm 2.

3.4. Complexity analysis

We adopt the method in Wang et al. (2022) to analyze the complexity of the proposed QMABC and compare it to the standard ABC. In ABC, the key factors that determine the algorithm complexity are population size SN, problem dimension D, and maximum iteration time I. For the initialization phase, the complexity depends on the number and dimension of individuals, so it is $O(SN \times D)$. For the employed bee phase, each bee will be updated in each iteration, so the complexity is $O(SN \times I)$. In the onlooker bee phase, except for the updates, the probabilities of being selected for each individual require calculations, so the complexity is $O(2 \times SN \times I)$. For the scout bee phase, the execution times are relatively few and the complexity can be omitted. Overall, the complexity of ABC is $O(SN \times D + 3 \times SN \times I)$.

QMABC has the same complexity in the initialization and scout bee phase as standard ABC. In the employed bee phase, the complexity becomes $O(CR \times D \times SN \times I)$, which is influenced by the crossover rate CR. In the onlooker bee phase of QMABC, the selection probability is canceled. Meanwhile, the strategies are related to the finding of global and neighborhood optimal solutions, and the update of multiple dimensions. Therefore, the complexity of the three strategies are $O(D \times SN \times I + SN \times I)$, $O(3 \times SN \times I)$, $O(CR \times D \times SN \times I)$, respectively. Therefore, the first strategy has the maximum complexity among the three strategies. In this case, we adopt the maximum value $O(D \times SN \times I + SN \times I)$ as the complexity in this process. In addition, the Q-learning framework includes calculating the state ($O(I)$), updating the Q table ($O(I)$), selecting the action ($O(I)$), and performing the action ($O(3 \times I)$), which has a total complexity of $O(6 \times I)$. Thus, the overall complexity of QMABC is $O(SN \times D + CR \times D \times SN \times I + D \times SN \times I + SN \times I + 6 \times I)$.

Although QMABC is more complex than ABC, it is acceptable due to two reasons. Firstly, in the actual computing task, the time cost of function evaluation accounts for a large proportion. On the contrary, the complexity of the algorithm itself has little impact. Secondly, the QMABC algorithm has a faster convergence speed. So it requires fewer iterations and function evaluation times in the solving process.

4. Numerical experiments

In this section, we conduct real-parameter optimization experiments to validate the effectiveness of the proposed algorithm. Twenty-nine CEC 2017 benchmark functions are employed for the QMABC algorithm compared to ABC variants and other novel meta-heuristic algorithms in different dimensions, respectively.

4.1. Benchmark function

The benchmark functions are obtained from 2017 IEEE Congress on Evolutionary Computation (CEC 2017) (Awad et al., 2016). They provide a relatively objective and impartial platform to evaluate the optimization performance of different meta-heuristic algorithms. For the rigor of the experiment, all the twenty-nine benchmark functions in CEC 2017 were taken into consideration.

For CEC 2017, it contains four types of benchmark functions, in which f_1, f_3 are 2 unimodal functions, $f_4 \sim f_{10}$ are 7 simple multimodal functions, $f_{11} \sim f_{20}$ are 10 hybrid functions, and $f_{21} \sim f_{30}$ are

Algorithm 2: Pseudo code of QMABC algorithm

Input: *Objective function F(x),* FES_{max}, SN, $trial_{limit}$, p_1, p_2, p_3, lb, ub
Output: *Feasible solution*

```
begin
    G=0;
    Generate SN candidate solutions using Eq. (1);
    Evaluate the initial population; Set trial_i = 0 for each solution;
    while FES < FES_max do
        for i = 1, 2, ..., SN do
            Randomly select r_1 ≠ r_2 ≠ r_3 from 1, 2, ..., SN;
            for j = 1, 2, ..., D do
                if rand ≤ CR then
                    Generate a new candidate solution v_i using Eq. (8);
                end if
            end for
            Evaluate the new solution v_i;
            if f(v_i) ≤ f(x_i) then
                x_i = v_i; trial_i = 0;
            else
                trial_i = trial_i + 1;
            end if
        end for
        n_s^G = 0;
        for i = 1, 2, ..., SN do
            Generate a random number RAND;
            if RAND ≤ p_1 then
                Generate a new candidate solution v_i using Eq. (10);
            else if RAND ≤ p_2 then
                Generate a new candidate solution v_i using Eq. (12);
            else
                Generate a new candidate solution v_i using Eq. (17);
            end if
            Evaluate the new solution v_i;
            if f(v_i) ≤ f(x_i) then
                x_i = v_i; trial_i = 0; Trial_i^G = 0;
            else
                trial_i = trial_i + 1; Trial_i^G = Trial_i^G + 1;
            end if
        end for
        FES = FES + 2 × SN;
        Calculate the value of n_s^G via Eq. (19);
        if n_s^G ≥ n_s^{G-1} then
            reward = 1;
            state = s_1;
        else
            reward = −1;
            state = s_2;
        end if
        Update Q-table via Eq. (6);
        Obtain p_1, p_2 and p_3 in the next iteration through Q-table and Eq. (18);
        for i = 1, 2, ..., SN do
            if trial_i > trial_limit then
                Generate a new candidate solution v_i using Eq. (1);
                Evaluate the new solution v_i;
                trial_i = 0; FES = FES + 1;
            end if
        end for
        G=G+1;
    end while
end
```

Table 2
Comparison algorithm selection of ABC variants.

Algorithm	Source	Parameter setting
ABC	Karaboga and Basturk (2007)	$SN = 50, limit = SN * D$
PAABC	Zhao et al. (2022)	$SN = 50, limit = SN * D$
mABC	Ustun et al. (2022)	$SN = 50, limit = SN * D, F = 0.5, CR = 0.3$
BDLDABC	Wang et al. (2022)	$SN = 50, limit = SN * D, \alpha = 20, \beta = 0.4, \gamma = 0.01, \mu = 0.5$
ABC_RL	Cui et al. (2022)	$SN = 50, limit = SN * D, \alpha = 0.6, \gamma = 0.4, \epsilon = 0.3$
BABC	Zhu et al. (2023)	$SN = 50, limit = 50, hint = SN$
QMABC	This article	$SN = 50, limit = SN * D, \alpha = 0.2, F = 0.5, CR = 0.3, p_1 = 0.4, p_2 = p_3 = 0.3, lb = 0.1, ub = 0.7$

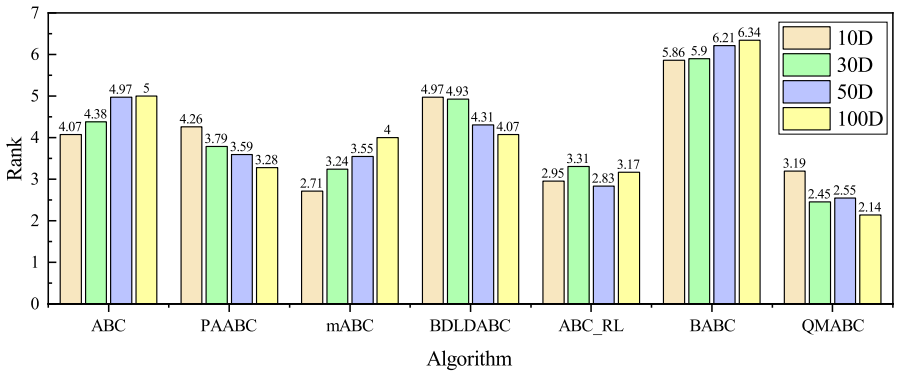


Fig. 3. Friedman test of comparison with ABC and ABC variants.

10 composition functions. Notice that the function f_2 has been deleted in the latest official code.

In the specification document of CEC 2017, the evaluation criteria are defined. For each benchmark function, the search range is $[-100, 100]$, and the optimal value is the corresponding serial number of function multiplied by 100. The terminal condition is the maximum number of function evaluations (FES_{max}), which is determined by the dimension of the problem. $FES_{max} = 10000 \times D$.

4.2. Comparison with ABC and ABC variants

In this section, the proposed QMABC is compared to the standard ABC and four latest ABC variants. Table 2 presents the selected algorithm, which includes the abbreviation, sources, and parameter settings of those algorithms. Since the parameters have a tremendous influence on the actual effect of the algorithms, the parameters of the comparison algorithms are taken from the corresponding paper to prevent the change of algorithm effects.

All algorithms were run 30 times independently on each of the 29 CEC 2017 benchmark functions while the mean values and standard deviations were recorded. We studied $D = 10$, 30, 50, and 100 respectively to verify the effectiveness of the algorithm under different problem dimensions. Table 3, S.Tables 1–3, and Fig. 3 show the comparison test results of QMABC algorithm with ABC and ABC variants. To show the results in a more rational way, we conduct two statistical measures: the Wilcoxon test and the Friedman test. The significant levels are both $\alpha = 0.05$ under the two tests.

The Wilcoxon test is used to compare two independent individuals, and here we use it to show the significance between QMABC and the competitors, respectively. The bottom row of the tables lists the summation of each case. Friedman test applies to comparisons between multiple samples, so we also put the six algorithms together for comparison. The average ranks are shown in Fig. 3.

According to the experimental results with $D = 10$ in S.Table 1, QMABC obtains the smallest objective function values in 7 benchmark functions $(f_3, f_5, f_7, f_8, f_{11}, f_{27}, f_{30})$ and the worst rankings in 2 functions (f_{21}, f_{28}). It inferiors to the mABC in more than half of the functions but outperforms the other four algorithms. In general, QMABC shows preferable results on the 10-dimensional problems. Especially when compared to standard ABC, it is better on 21 functions and slightly worse on only 7 functions. Nevertheless, QMABC still has room for improvement in the low-dimensional problem because it still has a gap compared to mABC.

Table 3 lists the experimental results with $D = 30$. QMABC has the best ranks in a total of 13 benchmark functions $(f_3, f_5, f_7, f_8, f_{14}, f_{16}, f_{17}, f_{20}, f_{22}, f_{25}, f_{27}, f_{29}, f_{30})$ and worst ranks in merely 2 functions (f_{10}, f_{28}). Through the results, we can find that QMABC surpasses all five comparison algorithms in the 30-dimension problems. Moreover, the number of first rankings has reached half of the total number of functions. The functions dominated by QMABC consist of multiple types, which illustrates that QMABC not only performs well in solving a single type of function but also shows comprehensive advantages.

In S.Table 2, the experimental results with $D = 50$ are presented. We can find that up to 12 functions $(f_1, f_3, f_5, f_8, f_9, f_{16}, f_{17}, f_{20}, f_{25}, f_{27}, f_{29}, f_{30})$ are dominated by the QMABC while the number of worst rankings (f_6) reduces to 1. QMABC surpasses the five algorithms of original ABC, PAABC, mABC, BDLDABC, ABC_RL, and BABC on 21, 21, 21, 19, 17, and 24 functions, respectively. While it fails to compete with them on 5, 6, 8, 6, 8, and 3 functions severally. These results indicate that QMABC has comprehensively outperformed its competitors in the 50-dimension problem.

In 100-dimensional problems, the results are displayed in S.Table 3. Based on the Wilcoxon test results, QMABC is the best algorithm on 13 functions $(f_3, f_5, f_8, f_9, f_{11}, f_{16}, f_{17}, f_{24}, f_{26}, f_{27}, f_{28}, f_{29}, f_{30})$ while the function f_6 is still a failing for QMABC. In the 100-dimensional problem, QMABC has the most obvious improvement over the original ABC, where QMABC has 25 functions performing better than the original ABC. Besides, the results point out that QMABC algorithm shows a better balance between exploitation and exploration abilities compared to other competitors during the search process.

In the Friedman test, a smaller ranking means better performance. According to the test results, we can find that QMABC always maintains a small ranking. QMABC is the third-best algorithm for the experiment with $D = 10$ and jumps to the top of the ranking for the experiments with $D = 30$, 50, and 100. Furthermore, the average ranking of QMABC in the test consistently decreases with the problem dimension increase.

Table 3
Comparison with ABC and ABC variants with D=30.

Benchmark	Statistic	ABC		PAABC		mABC		BDLDABC		ABC_RL		BABC		QMABC
f_1	Mean	3.79E+02	−	1.53E+04	−	1.30E−01	−	4.64E+03	+	2.69E+03	=	5.71E+03	+	2.50E+03
	Std.	2.72E+02		3.93E+04		2.57E−01		4.17E+03		3.70E+03		5.51E+03		2.62E+03
f_3	Mean	1.35E+05	+	1.22E+05	+	6.89E+04	+	3.42E+04	+	5.46E+04	+	1.00E+05	+	6.77E+03
	Std.	1.78E+04		1.93E+04		7.99E+03		7.16E+03		1.29E+04		1.76E+04		2.41E+03
f_4	Mean	3.60E+01	−	4.81E+01	−	7.28E+01	+	8.24E+01	+	7.79E+01	+	6.58E+01	+	5.97E+01
	Std.	2.62E+01		2.34E+01		1.59E+01		2.49E+01		1.65E+01		2.73E+01		2.62E+01
f_5	Mean	8.84E+01	+	3.87E+01	+	5.79E+01	+	5.61E+01	+	4.46E+01	+	1.19E+02	+	3.28E+01
	Std.	1.38E+01		6.38E+00		6.75E+00		1.56E+01		7.54E+00		1.69E+01		6.67E+00
f_6	Mean	1.30E−11	+	0.00E+00	−	0.00E+00	−	9.36E−04	+	0.00E+00	−	1.60E+00	+	2.77E−07
	Std.	1.53E−11		0.00E+00		0.00E+00		1.79E−03		2.08E−14		7.49E−01		9.10E−07
f_7	Mean	1.03E+02	+	6.29E+01	+	9.09E+01	+	9.98E+01	+	7.18E+01	+	1.70E+02	+	6.15E+01
	Std.	9.88E+00		4.99E+00		7.20E+00		2.15E+01		7.68E+00		1.94E+01		8.54E+00
f_8	Mean	9.58E+01	+	4.48E+01	+	6.80E+01	+	5.83E+01	+	5.16E+01	+	1.14E+02	+	3.50E+01
	Std.	1.18E+01		5.67E+00		8.11E+00		1.25E+01		7.60E+00		1.70E+01		7.61E+00
f_9	Mean	1.18E+03	+	2.90E+01	+	2.69E+02	+	8.74E+01	+	4.78E−01	=	3.36E+02	+	7.73E−01
	Std.	4.32E+02		1.71E+01		1.41E+02		6.12E+01		6.58E−01		2.93E+02		9.61E−01
f_{10}	Mean	2.28E+03	−	2.13E+03	−	2.27E+03	−	2.90E+03	−	2.15E+03	−	2.11E+03	−	2.73E+03
	Std.	2.71E+02		2.15E+02		1.97E+02		1.39E+03		3.09E+02		3.73E+02		2.69E+02
f_{11}	Mean	9.31E+02	+	7.87E+02	+	2.81E+01	−	8.73E+01	+	3.71E+01	−	1.03E+03	+	6.08E+01
	Std.	6.28E+02		4.91E+02		7.00E+00		3.27E+01		2.69E+01		6.23E+02		2.53E+01
f_{12}	Mean	9.51E+05	+	7.65E+05	+	5.88E+05	+	5.81E+04	−	2.08E+05	+	1.17E+06	+	1.06E+05
	Std.	3.21E+05		5.02E+05		2.55E+05		4.63E+04		2.35E+05		9.46E+05		8.72E+04
f_{13}	Mean	1.91E+04	+	1.22E+04	+	1.60E+04	+	2.06E+04	+	2.07E+04	+	1.83E+04	=	1.20E+04
	Std.	9.70E+03		5.52E+03		1.01E+04		1.85E+04		1.98E+04		2.22E+04		1.17E+04
f_{14}	Mean	1.24E+05	+	1.86E+05	+	1.48E+04	+	2.99E+04	+	2.65E+04	+	9.64E+04	+	6.93E+03
	Std.	7.20E+04		1.46E+05		8.87E+03		3.02E+04		2.75E+04		9.13E+04		8.78E+03
f_{15}	Mean	3.25E+03	+	4.10E+03	+	1.98E+03	=	8.48E+03	+	1.05E+04	+	8.40E+03	+	3.96E+03
	Std.	1.88E+03		2.96E+03		1.43E+03		8.66E+03		1.08E+04		8.33E+03		7.40E+03
f_{16}	Mean	6.34E+02	+	6.64E+02	+	5.14E+02	+	8.29E+02	+	4.65E+02	+	7.46E+02	+	3.67E+02
	Std.	1.40E+02		1.42E+02		1.24E+02		2.30E+02		1.46E+02		1.85E+02		1.58E+02
f_{17}	Mean	2.08E+02	+	1.71E+02	+	1.07E+02	+	2.30E+02	+	9.57E+01	+	3.40E+02	+	6.89E+01
	Std.	9.22E+01		8.61E+01		4.13E+01		1.55E+02		7.27E+01		1.26E+02		5.67E+01
f_{18}	Mean	2.81E+05	+	2.39E+05	+	2.07E+05	+	1.49E+05	=	2.05E+05	+	3.57E+05	+	1.53E+05
	Std.	1.27E+05		9.09E+04		1.03E+05		1.27E+05		1.47E+05		2.20E+05		8.89E+04
f_{19}	Mean	3.00E+03	=	6.76E+03	+	2.16E+03	−	1.18E+04	+	1.28E+04	+	8.68E+03	+	4.76E+03
	Std.	1.44E+03		4.38E+03		1.31E+03		1.48E+04		1.52E+04		8.66E+03		5.07E+03
f_{20}	Mean	2.59E+02	+	2.35E+02	+	1.48E+02	+	2.82E+02	+	1.31E+02	+	3.18E+02	+	9.12E+01
	Std.	8.52E+01		8.64E+01		6.60E+01		1.46E+02		1.02E+02		1.28E+02		5.60E+01
f_{21}	Mean	2.63E+02	+	2.36E+02	−	2.59E+02	−	2.63E+02	=	2.47E+02	−	3.00E+02	+	2.64E+02
	Std.	6.66E+01		2.59E+01		7.26E+00		1.67E+01		2.56E+01		5.87E+01		1.49E+01
f_{22}	Mean	6.38E+02	+	1.09E+02	+	2.36E+02	+	8.12E+02	+	3.40E+02	+	9.39E+02	+	1.00E+02
	Std.	1.05E+03		5.82E+00		5.27E+02		1.44E+03		7.22E+02		1.23E+03		7.37E−01
f_{23}	Mean	4.25E+02	+	3.99E+02	−	4.08E+02	=	4.13E+02	=	4.00E+02	−	4.46E+02	+	4.10E+02
	Std.	2.77E+01		7.00E+00		1.02E+01		1.84E+01		1.23E+01		4.78E+01		1.76E+01
f_{24}	Mean	4.87E+02	+	5.13E+02	+	4.91E+02	=	5.05E+02	+	4.97E+02	=	5.83E+02	+	4.95E+02
	Std.	1.92E+02		1.72E+01		1.14E+01		2.07E+01		1.32E+01		4.93E+01		2.70E+01
f_{25}	Mean	3.84E+02	+	3.87E+02	+	3.87E+02	+	3.89E+02	+	3.87E+02	+	3.88E+02	+	3.78E+02
	Std.	7.60E−01		5.75E−01		1.08E+00		3.50E+00		9.03E−01		2.04E+00		1.08E+00
f_{26}	Mean	3.60E+02	−	1.38E+03	+	1.59E+03	+	1.77E+03	+	1.31E+03	+	3.16E+02	−	1.40E+03
	Std.	4.22E+02		5.54E+02		4.11E+02		2.01E+02		5.78E+02		2.04E+02		1.53E+02
f_{27}	Mean	5.15E+02	+	5.16E+02	+	5.07E+02	+	5.18E+02	+	5.03E+02	+	5.22E+02	+	5.00E+02
	Std.	4.17E+00		4.69E+00		3.65E+00		1.00E+01		5.15E+00		7.01E+00		1.57E−04
f_{28}	Mean	4.00E+02	−	4.17E+02	−	4.16E+02	−	4.08E+02	−	4.10E+02	−	4.31E+02	−	4.95E+02
	Std.	3.86E+00		1.71E+01		5.00E+00		1.90E+01		1.11E+01		1.50E+01		1.69E+01
f_{29}	Mean	6.46E+02	+	5.76E+02	+	5.98E+02	+	7.56E+02	+	4.91E+02	+	7.03E+02	+	4.50E+02
	Std.	7.94E+01		6.12E+01		7.68E+01		1.75E+02		5.63E+01		1.07E+02		8.50E+01
f_{30}	Mean	1.11E+04	+	8.56E+03	+	6.28E+03	+	5.68E+03	+	6.41E+03	+	1.17E+04	+	2.46E+03
	Std.	2.21E+03		3.44E+03		1.51E+03		3.35E+03		3.75E+03		4.88E+03		2.50E+03
Wilcoxon	+/=/−	23/1/5		22/0/7		19/3/7		23/3/3		20/3/6		25/1/3		

1"+","−", and "=" symbolize that QMABC is better than, worse than, or similar to its competitor, respectively.
2 The significance value for Wilcoxon test is $\alpha = 0.05$.

It indicates that QMABC has good performance in all dimensions and absolute preponderance in high-dimensional problems.

Fig. 4 exhibits the convergence performance of compared ABC variants, in which we select several representative functions (f_3, f_8, f_{27}, f_{30}) to demonstrate the characteristics of QMABC. We can find that the QMABC has a faster convergence speed on f_3 and is more precise on f_8. Moreover, it maintains both of these superiorities on f_{27}, f_{30}. The functions above cover many types, ranging from simple to complex, so the capability of QMABC is well-validated. It is important to note that QMABC does not show the best performance on all functions, but performs better in most cases.

4.3. Comparison with other meta-heuristic algorithms

To further investigate the performance of the QMABC, we selected the standard DE and four latest meta-heuristic algorithms to participate in the comparison experiment. The abbreviation, sources, and parameter settings of those algorithms are displayed in Table 4. The reason why we choose the original DE algorithm to participate in the

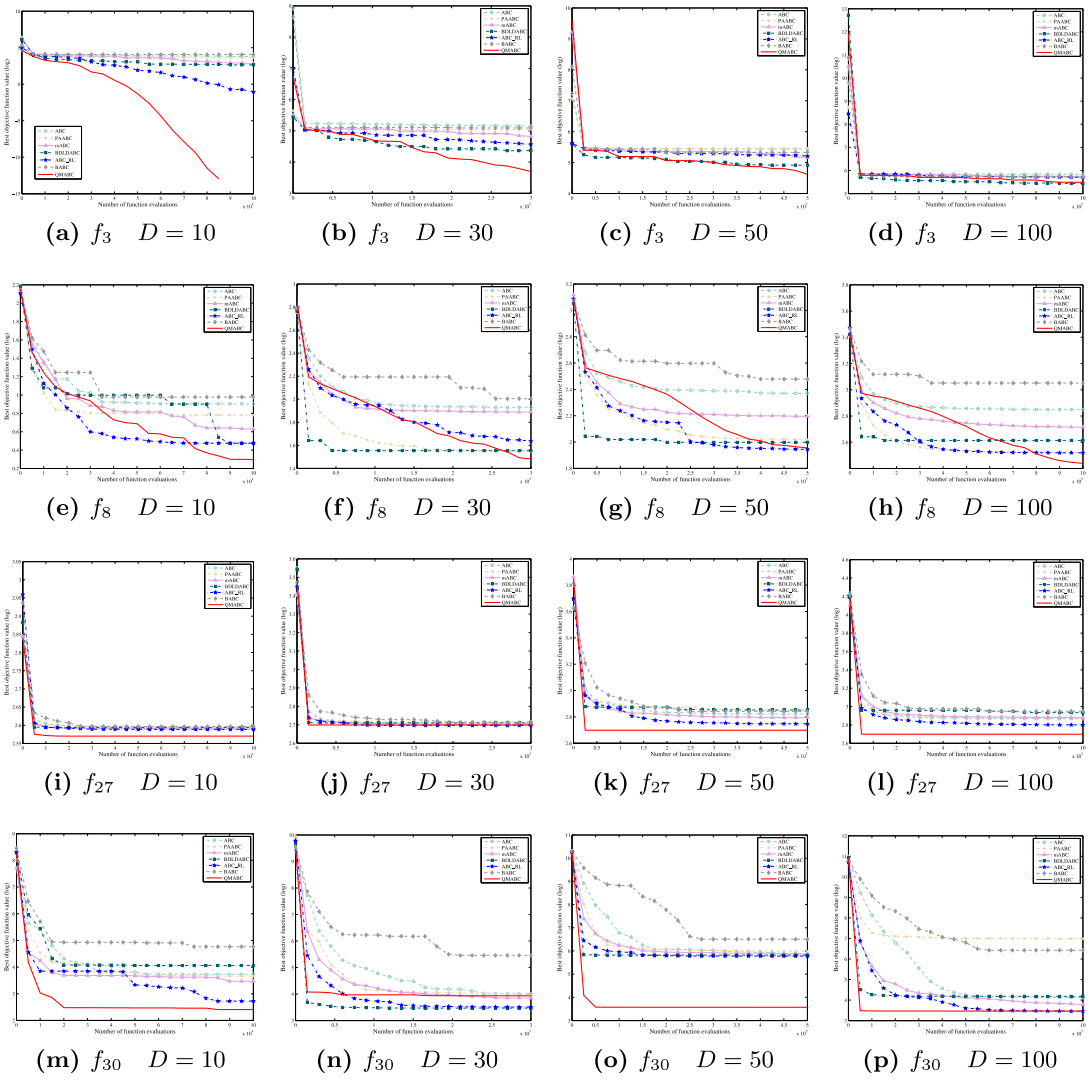


Fig. 4. The convergence performance of compared ABC variants.

Table 4
Comparison algorithm selection of other meta-heuristic algorithms.

Algorithm	Source	Parameter setting
DE	Storn and Price (1997)	$NP = 50, F = 0.5, CR = 0.3, DE/rand/1$
LJA	Iacca et al. (2021)	$NP = 50, \beta = 1.8$
ACWOA	Tang et al. (2022)	$NP = 50, p = 0.5, limit_A = 0.5, b = 1$
APDPSO	Liu et al. (2022)	$NP = 50, \omega = 0.729, c = 1.49$
CUSDE	Zou et al. (2022)	$NP = 50, c = 40, F = 0.5, CR = 0.9$
QMABC	This article	$SN = 50, limit = SN * D, \alpha = 0.2, F = 0.5, CR = 0.3, p_1 = 0.4, p_2 = p_3 = 0.3, lb = 0.1, ub = 0.7$

comparison here is that the fast convergence and local development ability of DE are incomparable to most of the variants. DE can be compared with QMABC here as a representative of the low dimensional dominance algorithm.

Similar to the comparison with ABC variants, these meta-heuristic algorithms are run 30 times independently on each of the 29 CEC 2017 benchmark functions with the mean values and standard deviations output. Two statistical methods, the Wilcoxon test and the Friedman test, are also applied to test the significance of the comparison. The difference is that the dimension of the test changes to $D = 10$, 30, and 50. The experimental results are exhibited in Table 5, S.Tables 4–5, and Fig. 5.

On the basis of the experimental results, QMABC completely surpasses LJA and APDPSO on total 29 functions with $D = 10, 30, 50$. Furthermore, QMABC is superior to DE, ACWOA, and CUSDE on nearly two-thirds functions.

According to the Friedman test results, QMABC performs the best average ranking in all experiments with $D = 10$, 30, and 50. In addition, similar to the trend of comparing to the ABC variants, QMABC also illustrates a high-dimensional superiority over other meta-heuristic algorithms.

Fig. 6 displays the convergence performance of QMABC and compared meta-heuristic algorithms on three typical functions including

Table 5
Comparison with other meta-heuristic algorithms with D=10.

Benchmark	Statistic	DE		LJA		ACWOA		APDPSO		CUSDE		QMABC
f_1	Mean	8.11E+02	−	1.71E+08	+	2.81E+03	+	1.18E+08	+	0.00E+00	−	1.39E+03
	Std.	1.86E+03		6.46E+07		3.02E+03		1.86E+08		2.59E−15		1.69E+03
f_3	Mean	7.60E−02	+	2.48E+03	+	1.69E−01	+	3.29E+03	+	0.00E+00	=	0.00E+00
	Std.	4.09E−01		6.84E+02		4.28E−01		1.83E+03		0.00E+00		1.47E−14
f_4	Mean	6.53E+00	+	9.01E+00	+	2.63E+00	−	1.84E+01	+	3.15E−09	−	3.59E+00
	Std.	1.05E+00		3.38E+00		1.86E+00		1.88E+01		1.69E−08		9.07E−01
f_5	Mean	1.13E+01	+	3.88E+01	+	7.30E+00	+	2.56E+01	+	2.13E+01	+	2.77E+00
	Std.	2.67E+00		4.86E+00		1.48E+00		7.83E+00		5.48E+00		1.05E+00
f_6	Mean	0.00E+00	=	6.50E+00	+	1.87E−05	+	3.17E+00	+	3.08E−03	+	0.00E+00
	Std.	0.00E+00		1.24E+00		9.99E−06		4.00E+00		4.27E−03		0.00E+00
f_7	Mean	2.39E+01	+	5.02E+01	+	1.71E+01	+	6.69E+01	+	3.36E+01	+	1.25E+01
	Std.	1.98E+00		6.35E+00		2.56E+00		1.45E+01		4.97E+00		8.21E−01
f_8	Mean	1.17E+01	+	3.65E+01	+	7.46E+00	+	2.69E+01	+	2.07E+01	+	2.72E+00
	Std.	2.86E+00		6.13E+00		2.13E+00		7.39E+00		5.00E+00		1.05E+00
f_9	Mean	3.15E−06	+	1.75E+01	+	3.70E−08	+	1.14E+02	+	1.51E−02	+	0.00E+00
	Std.	1.68E−05		6.32E+00		5.41E−08		1.01E+02		8.16E−02		0.00E+00
f_{10}	Mean	4.14E+02	+	9.77E+02	+	2.38E+02	+	1.25E+03	+	1.56E+03	+	1.53E+02
	Std.	2.06E+02		2.13E+02		1.02E+02		3.60E+02		1.96E+02		1.20E+02
f_{11}	Mean	2.09E+00	+	5.55E+01	+	2.74E+00	+	1.09E+02	+	3.79E+00	+	1.24E+00
	Std.	1.01E+00		2.01E+01		1.30E+00		6.91E+01		1.85E+00		1.23E+00
f_{12}	Mean	3.61E+03	−	3.81E+06	+	1.63E+04	+	2.07E+06	+	8.25E+01	−	7.59E+03
	Std.	5.89E+03		3.44E+06		1.41E+04		1.63E+06		5.35E+01		5.70E+03
f_{13}	Mean	2.75E+02	−	1.08E+04	+	2.43E+03	+	1.14E+04	+	8.98E+00	−	2.13E+03
	Std.	8.56E+02		1.07E+04		3.55E+03		5.10E+03		3.20E+00		3.16E+03
f_{14}	Mean	2.07E+00	−	8.40E+01	+	1.41E+01	+	1.00E+02	+	2.26E+01	+	1.66E+00
	Std.	6.30E+00		2.71E+01		8.96E+00		3.54E+01		1.13E+00		3.91E+00
f_{15}	Mean	4.99E−01	−	4.50E+02	+	4.87E+00	+	1.86E+03	+	1.65E+00	+	1.46E+00
	Std.	4.78E−01		1.77E+02		2.79E+00		2.25E+03		8.37E−01		8.39E−01
f_{16}	Mean	1.78E+00	−	7.51E+01	+	1.99E+00	+	1.42E+02	+	4.73E+01	+	5.21E+00
	Std.	3.32E+00		1.71E+01		1.87E+00		8.93E+01		2.41E+01		2.15E+01
f_{17}	Mean	3.16E−01	−	7.33E+01	+	2.59E+00	+	7.74E+01	+	4.07E+01	+	2.40E+00
	Std.	4.18E−01		1.14E+01		2.17E+00		2.68E+01		6.83E+00		5.53E+00
f_{18}	Mean	3.96E+01	−	4.41E+04	+	3.10E+03	+	1.95E+04	+	2.04E+01	+	1.10E+01
	Std.	1.71E+02		7.49E+03		3.75E+03		1.27E+04		2.17E−01		2.15E+01
f_{19}	Mean	4.99E+00	−	4.39E+02	+	1.92E+01	+	1.05E+04	+	1.56E+00	+	3.15E−01
	Std.	1.85E+01		2.94E+02		6.72E+01		6.81E+03		3.52E−01		4.46E−01
f_{20}	Mean	9.37E−02	−	6.13E+01	+	5.13E−01	+	7.03E+01	+	2.95E+01	+	3.86E−01
	Std.	1.64E−01		1.32E+01		6.16E−01		4.60E+01		5.55E+00		4.51E−01
f_{21}	Mean	1.89E+02	+	2.20E+02	+	1.01E+02	−	2.32E+02	+	1.16E+02	−	1.71E+02
	Std.	4.87E+01		4.09E+01		1.12E+00		1.97E+01		2.92E+01		4.64E+01
f_{22}	Mean	1.00E+02	−	1.19E+02	+	8.60E+01	=	1.22E+02	+	1.00E+02	+	9.29E+01
	Std.	1.45E−01		3.82E+00		2.93E+01		1.58E+01		9.40E+00		2.33E+01
f_{23}	Mean	3.08E+02	+	3.41E+02	+	3.10E+02	+	3.35E+02	+	3.28E+02	+	3.06E+02
	Std.	3.04E+00		4.97E+00		2.93E+00		1.27E+01		6.55E+00		1.55E+00
f_{24}	Mean	3.45E+02	+	3.54E+02	+	2.27E+02	−	3.64E+02	+	2.72E+02	=	3.21E+02
	Std.	3.83E+00		4.98E+01		1.19E+02		9.84E+00		9.09E+01		5.14E+01
f_{25}	Mean	4.27E+02	=	4.47E+02	+	3.88E+02	−	4.55E+02	+	3.98E+02	−	4.25E+02
	Std.	2.22E+01		1.47E+01		5.35E+01		2.01E+01		1.88E−01		2.33E+01
f_{26}	Mean	3.31E+02	+	4.15E+02	+	2.89E+02	+	7.51E+02	+	4.33E+02	+	2.90E+02
	Std.	3.74E+01		1.38E+01		3.41E+01		4.29E+02		1.17E+02		9.65E+01
f_{27}	Mean	3.91E+02	+	3.97E+02	+	3.89E+02	+	4.04E+02	+	4.46E+02	+	3.73E+02
	Std.	1.75E+00		3.03E+00		3.37E−01		9.05E+00		4.57E+01		1.76E+00
f_{28}	Mean	5.52E+02	+	5.72E+02	+	3.73E+02	−	7.66E+02	+	4.84E+02	+	4.49E+02
	Std.	8.64E+01		1.26E+02		9.35E+01		1.54E+02		1.04E+01		6.73E+01
f_{29}	Mean	2.58E+02	+	2.85E+02	+	2.40E+02	−	3.40E+02	+	2.53E+02	+	2.49E+02
	Std.	7.02E+00		3.05E+01		1.55E+01		3.84E+01		1.13E+01		7.86E+00
f_{30}	Mean	5.69E+04	+	2.57E+05	+	3.89E+03	+	1.38E+06	+	2.02E+02	−	3.75E+02
	Std.	2.03E+05		3.19E+05		5.69E+03		1.07E+06		5.13E−01		5.91E+02
Wilcoxon	+/=/−	16/2/11		29/0/0		22/1/6		29/0/0		20/2/7		

1 "+","−", and "=" symbolize that QMABC is better than, worse than, or similar to its competitor, respectively.
2 The significance value for Wilcoxon test is $\alpha = 0.05$.

f_7, f_{16}, f_{27}. According to these convergence curves, the faster convergence speed and more precise solution are apparent for QMABC.

4.4. Effectiveness analysis

To validate the effectiveness of the proposed Q-learning-based strategy selection framework, we compare it with two more selection approaches under the same search strategies. In the first approach, the three search strategies for the onlooker bee phase are randomly selected during each update, aiming to validate the effectiveness of the proposed method in comparison to the absence of any strategy selection mechanism. The first approach is referred to as QMABC$_{rand}$. In the second approach, we utilize Q-learning to directly select one strategy in the onlooker bee phase, rather than the indirect method of adjusting the probability of each strategy being selected. Specifically, in this approach, the state setting, rewards, update equation of the Q-table, and parameters are identical to those described in Section 3. The difference lies in the action setting, which consists of three options, corresponding to the selection of the respective strategy. And this

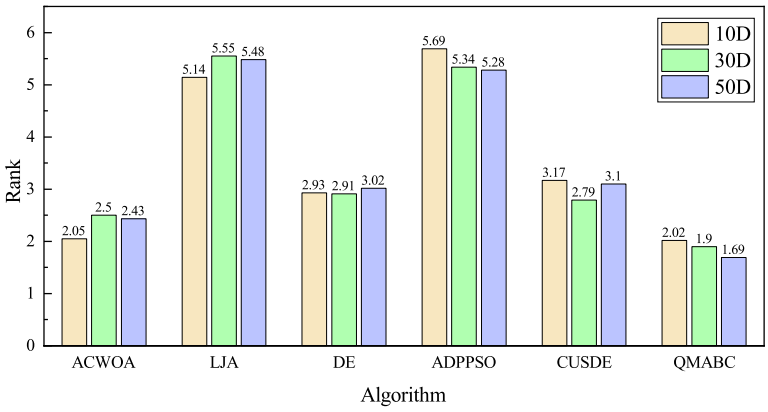


Fig. 5. Friedman test of comparison with other meta-heuristic algorithms.

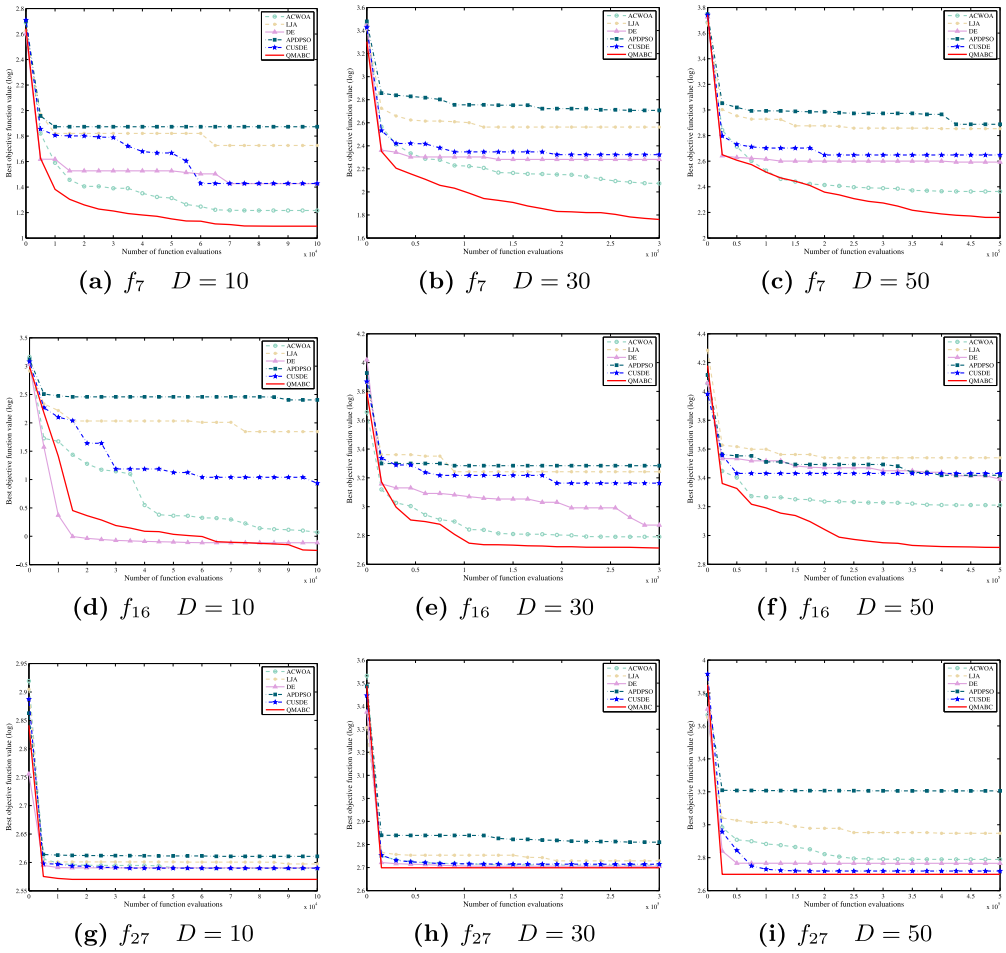


Fig. 6. The convergence performance of compared meta-heuristic algorithms.

approach is referred to as QMABC$_{direct}$. Besides, the proposed algorithm in Section 3 is referred to as QMABC.

We conduct a comprehensive comparison of the three approaches using the CEC 2017 benchmark function. The final evaluation encompasses pairwise comparison results and an overall ranking for each approach. The results are shown in Table 6.

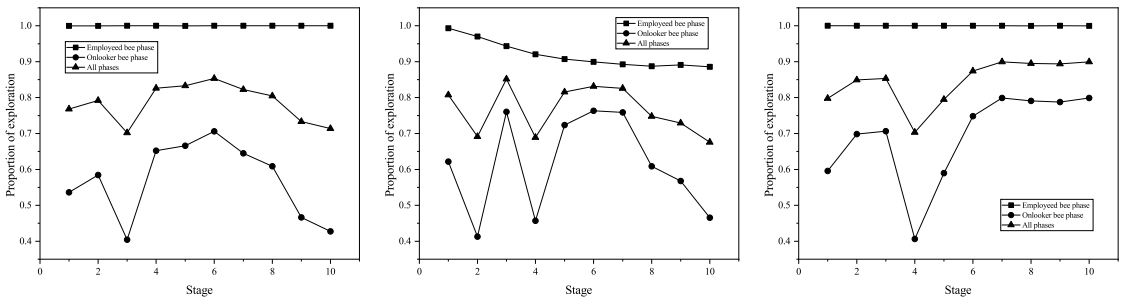
The results from Table 6 reveal that the proposed QMABC consistently maintains the best overall ranking and demonstrates advantages in pairwise comparisons across the three distinct problem dimensions. These results indicate that, compared to the direct strategy selection through Q-learning (QMABC$_{direct}$) or random strategy selection (QMABC$_{rand}$), adjusting the probabilities of each strategy being selected through Q-learning proves to be a more effective approach.

Furthermore, The differences among the approaches are relatively small when the problem dimension is low (30D), whereas significant variations emerge as the problem dimension increases (50D,100D). It indicates that when the problem complexity is low and each strategy has a sufficient number of executions, the effectiveness of the proposed Q-learning framework is limited. However, when confronted with higher-dimensional and more complex problems, the limited number of iterations does not allow each strategy to be fully utilized. In such a situation, the proposed Q-learning strategy selection framework

Table 6
Effectiveness comparison of different selection method.

Dimension	Evaluation	QMABC$_{rand}$	QMABC$_{direct}$	QMABC
30D	+/ = /−	8/18/3	18/10/1	
	Average rank	2.03	2.00	1.97
50D	+/ = /−	10/15/4	16/11/2	
	Average rank	1.89	2.56	1.55
100D	+/ = /−	14/11/4	17/11/1	
	Average rank	1.86	2.59	1.55

1 "+","−", and "=" symbolize that QMABC is better than, worse than, or similar to its competitor, respectively. The significance value for the test is $\alpha = 0.05$.



(a) f_5 (b) f_{13} (c) f_{24}

Fig. 7. The exploration and exploitation ability of QMABC.

in this paper effectively allocates the utilization rate of each strategy, yielding superior performance.

4.5. Exploration and exploitation ability analysis

In meta-heuristic algorithms, exploration and exploitation abilities are two critical factors that influence algorithm performance. Currently, the similarity to the closest neighbor (SCN) is a commonly used indicator to differentiate between exploration and exploitation. The mathematic definition of SCN is as follows Brajevic and Stanimirovic (2018):

$$SCN(x_{new}, population) = \min\left\{\|x_{new} - x_i\|^2 \,|x_i \in population\right\}, \tag{20}$$

where x_{new} is the new solution generated by the search equation, $population$ includes all the individuals. In other words, the classification of an update as exploration or exploitation can be determined by measuring the closest distance between the newly generated individual and all individuals in the original population. The specific method for this classification is shown in Eq. (21):

$$exploration: SCN(x_{new}, population) > \epsilon, \tag{21a}$$

$$exploitation: SCN(x_{new}, population) \leq \epsilon, \tag{21b}$$

where ϵ is the threshold value to distinguish exploration and exploitation. The value of ϵ is determined by the domain of the variable, and the calculation is as follows:

$$\epsilon = \frac{\left(x^{upper} - x^{lower}\right)}{100}. \tag{22}$$

Based on the approach above, we investigate the exploration and exploitation abilities of two bee phases and search stages in the proposed QMABC algorithm during the solving process. We still use the CEC 2017 benchmark function to test the performance, with a problem dimension of 30. The algorithm parameters are set to the same values as described earlier. A total of ten search stages are set, where each of the stages includes one-tenth of the maximum iterations. The proportion of exploration behavior is calculated as the average value within each stage.

As shown in Fig. 7, there is a noticeable difference in the proportions of exploration between the two bee phases. In the employed bee phase, the proportion of exploration behavior approaches 1, indicating that the search equation in Eq. (8) exhibits strong exploration capabilities across all search stages. In the onlooker bee phase, the proportion of exploration behavior consistently remains lower than that of the employed bee phase, and it can reach lower levels. This indicates that the three strategies in the onlooker bee phase ensure the algorithm's exploitation capability and dynamically adjust the balance between exploration and exploitation based on the search situation.

5. Application in unmanned vehicle path planning

With the advancement of informatization, intelligence, and automation in many industries typically automated driving, solving optimization problems in related fields has become a crucial challenge for development. Among them, the path planning of unmanned vehicles has become a crucial proposition in the field of automated driving. Path planning problem of unmanned vehicles focuses on finding the best path for a vehicle to navigate through complex geographic environments. The complexity of such an optimization problem is reflected in how to formulate interactions between the path and the environment with obstacles. As for this problem, there are three mainstream modeling methods: global path planning (Song et al., 2020), search-based path planning (Guo et al., 2022), and sample-based path planning (Wang, Jia, et al., 2022). Since meta-heuristic algorithms are more suitable for the global path planning method, we will utilize and explain it below.

In previous research, three methods are mainly used in environment modeling of global path planning, which are the grid method (Song et al., 2021), the topology method (Pattnaik et al., 2022), and the free space method (Xu et al., 2020). The grid method divides the working space of the vehicle into many grid-like units. The occupancy or free of a grid is generally represented by a number 1 or 0. The defects of this method lie in the high complexity of model solving and the trouble of generating and updating solutions. The topology method divides the working environment into many small regions according to some features of the topology structure, and then establishes a network by connecting relationships between small regions. The drawback of this method is the lack of accuracy. The free space method uses geometric shapes to represent the area of obstacles and a certain number of absolutely free points connected in turn to form a path. It is relatively

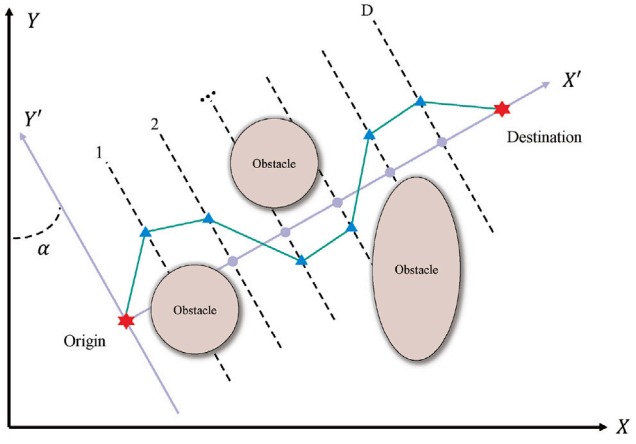


Fig. 8. Environment modeling for path planning.

superior because it takes into account the accuracy of the solution and the complexity of the environment modeling.

Thus, we utilize the free space method as the environment modeling method in this section and focus on the performance of the proposed algorithm applied to this specific problem.

5.1. 2D path planning

5.1.1. Environment modeling

For two-dimensional path planning, the environment can be regarded as a top view of a three-dimensional scene. An environment typically contains the following elements: the origin and destination of the path, as well as obstacles. In the general free space method, we describe the environment in the coordinate system XOY. However, this modeling method will result in two degrees of freedom in X and Y directions for each waypoint, which greatly increases the burden of computation. As shown in Fig. 8, we can therefore implement coordinate transformation to eliminate one degree of freedom. The new coordinate system takes the path origin as the coordinate origin, the direction of the path destination as the positive direction of the x-axis, and the vertical direction of the line connecting the origin and the destination as the direction of the y-axis. The three elements constitute a new coordinate system $X'O'Y'$.

There is an exact transformation relationship between the points under the new coordinates and the original coordinates, as shown in Eq. (23):

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos\ \alpha & -\sin\ \alpha \\ \sin\ \alpha & \cos\ \alpha \end{bmatrix} \left(\begin{bmatrix} x' \\ y' \end{bmatrix} - \begin{bmatrix} x'_o \\ y'_o \end{bmatrix} \right) + \begin{bmatrix} x_o \\ y_o \end{bmatrix}, \tag{23}$$

where (x', y') is the point under the new coordinate system and (x, y) is the point under the original coordinate system. α is the rotation angle between two coordinates. The rotation angle can be obtained in the following way:

$$\alpha = \arctan\left(\frac{y_d - y_o}{x_d - x_o} \right). \tag{24}$$

We assume that a complete path comprises two endpoints and D nodes. The two endpoints cannot be moved, and the horizontal position of each node in the new coordinate system is successively the $(D+1)$ equinox of the line between the origin and the destination. Their abscissa in the new coordinate system can be expressed as:

$$x'_i = i \times \frac{|x'_d - x'_o|}{D+1}. \tag{25}$$

In this way, the node can only move longitudinally under the new coordinate system. The degree of the freedom of the solution space is reduced to the number of nodes D. The solution space can be described as $S = \{y'_1, y'_2, \ldots, y'_D\}$. Notably, the larger the value of D, the higher the precision of the path planner. Therefore, careful consideration should be given when setting D values to balance model complexity and accuracy requirements.

5.1.2. Path planning modeling

Path planning modeling can be flexibly adjusted to the actual problem. Common factors include path length, steering angle, distance from obstacles, etc. In this paper, we only take the path length into account. In this case, the total path length can be represented by a summation of path length between every two adjacent points as:

$$C_d = d_{Origin,1} + \sum_{j=1}^{D} d_{j,j+1} + d_{D,Destination}, \tag{26}$$

where we use the Euclidean distance as the distance between two points. Therefore, it is calculated as:

$$d_{j,j+1} = \sqrt{(x_{j+1} - x_j)^2 + (y_{j+1} - y_j)^2}. \tag{27}$$

Besides, in order to facilitate the execution of the meta-heuristic algorithms to solve constrained optimization problems, constraints are generally converted into penalties. The penalty is a binary value function, where the function value is zero when the constraints are satisfied and an extremely large number if not. In this case, the constraint ensures that the path does not pass through any obstacle. The corresponding penalty is displayed as:

$$penalty = \begin{cases} 0, & collision - free, \\ M, & otherwise. \end{cases} \tag{28}$$

To be precise, the collision-free path is judged via two parts. Firstly, all nodes and endpoints must be positioned outside of any obstacles in the environment. Secondly, the lines between two adjacent nodes cannot go through the obstacles. To achieve this goal, we can divide the lines into n equal parts and extract the position of multisection points. And then those multisection points are utilized to verify if the corresponding section of the path goes through the obstacle. Similar to the value of D, the larger n is, the more accurate the judgment of path collision will be. Therefore, the value of n should be determined according to the actual geometry of the obstacles. In the following simulation, the value of n is 2.

Finally, the constrained optimization problem is simplified to a problem with merely one single objective function in Eq. (29):

$$Minimize \quad f_{cost} = \omega_1 \times C_d + penalty. \tag{29}$$

5.1.3. Experimental results

In this section, we compare the performance of the proposed QMABC algorithm with four typical path planning algorithms in three different environments. The four algorithms are PRM, RRT, BRRT, and A* algorithm.

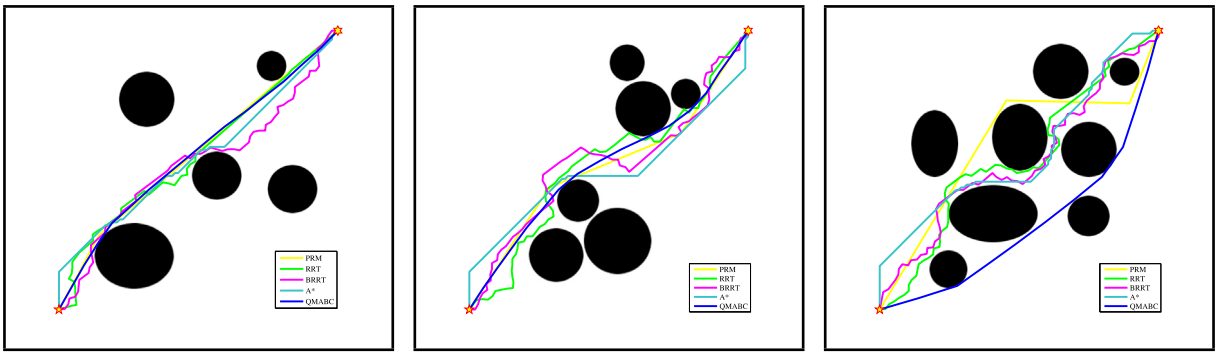
In the experiments, considering the scale of environments, the number of sampling points in PRM is defined as 50. The step length of RRT and BRRT are set as 10. As for the QMABC, the parameter settings are the same as in the algorithm description with $SN = 50$, $D = 10$, and the determination condition is set as $FES_{max} = 2000 \times D$. The ω_1 is set as 1 and the penalty is set as 1000.

The simulations are conducted with MATLAB R2013b on Intel(R) Core(TM) i5-8265U with 1.80 GHz CPU and 8.00 GB memory. Each algorithm is executed independently 10 times. The average path length and running time for each algorithm are shown in Table 7. In addition, we also display the path of the best simulation results in Fig. 9 aiming to present the upper limit of capability for every typical path planner.

As can be seen in Table 7 and Fig. 9, both the average path length and the execution time for QMABC are less than other typical

Table 7
Experimental results and comparison between other path planners.

Environment	Attribute	PRM	RRT	BRRT	A*	QMABC
Map 1	Path length	693.462	751.560	807.872	719.828	684.876
	Running time (s)	3.926	2.885	2.747	33.483	2.825
Map 2	Path length	752.248	826.420	858.109	752.046	693.117
	Running time (s)	3.898	3.735	2.979	57.526	2.845
Map 3	Path length	767.549	822.918	847.833	769.619	723.804
	Running time (s)	3.809	12.381	3.806	70.078	3.026



(a) Map 1 **(b)** Map 2 **(c)** Map 3

Fig. 9. The best path for different algorithms in different 2D environments.

algorithms, regardless of the complexity of the environment. In particular, the solving speed of the QMABC is more stable in different environments, rather than significantly slowing down with the increase of environmental complexity like A* and RRT. Moreover, the path generated by QMABC is straighter and smoother, aligning better with the practical requirement of unmanned vehicles. To sum up, QMABC is an excellent method in path planning problems with short path length, rapidity, and smoothness.

5.2. 3D path planning

In this section, we consider three-dimensional path planning problems with the objective of minimizing the path length.

5.2.1. Environment modeling

In the 3D environment modeling, the biggest difference with the 2D environment modeling is that the environments have elevation change and the planned paths need to strictly fit the terrain. In this case, the free space method in the 2D case is no longer applicable. Because if the distance between the adjacent linearly connected nodes is relatively far, The path formed by the two nodes cannot fit the terrain, which will cause distortion. Therefore, we propose a two stage three-dimensional modeling method to address the problem. The first stage of the method refers to the free space method to set a certain number of primary nodes. In the second stage, we add sub-nodes adaptively between adjacent nodes to address the issue above. The proposed methods are as follows:

Firstly, it is necessary to rasterize the environment map. The rasterized map can be expressed as a $n \times m$ matrix, where the rows of the matrix represent the nth grid region in the x-axis direction, and the columns of the matrix represent the mth grid region in the y-axis direction. Each position of the matrix has a specific value representing the elevation in the grid area.

Secondly, the 3D environment map can be set in an XYZ coordinate system. The form of a path can be expressed as:

$$Path = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ \cdots & \cdots & \cdots \\ x_n & y_n & z_n \end{bmatrix}. \tag{30}$$

In the first stage, the x-axis is divided into n segments on average according to the grid size, and the corresponding coordinates of path nodes in the x-axis direction are determined. For each path node, which can move freely in the plane perpendicular to the x-axis, there are two dimensions of freedom. The solution space can be expressed as $S = \{\pi_1, \pi_2, \ldots, \pi_n\}$, where $\pi_i, i \in \{1, 2, \ldots, n\}$ is the plane corresponding to each x-axis equal division point, as Fig. 10 shows. At the same time, due to the one-to-one correspondence between terrain z_i and y-axis position y_i in a map, namely $z_i = f(x_i, y_i)$, the degree of freedom of each path node can be reduced to 1.

In the second stage, the primary purpose is to reasonably adjust the path elevation between adjacent nodes to achieve the matching between the path and terrain. The method is to add corresponding sub-nodes adaptively according to the grid size and path length between adjacent nodes. The position of each sub-node in the y-axis direction falls on the m equal division points. The elevation of sub-nodes is determined according to the actual terrain. The process is shown in Fig. 11.

In this case, the number of sub-nodes can be determined according to Eq. (31):

$$n_{sub} = floor\left(\frac{|y_k - y_{k-1}|}{L_{k,k-1}}\right). \tag{31}$$

5.2.2. Path planning modeling

In the 3D path planning modeling, this paper still takes minimizing the total path length as the objective and further considers the constraints of collision avoidance and climbing ability to model the

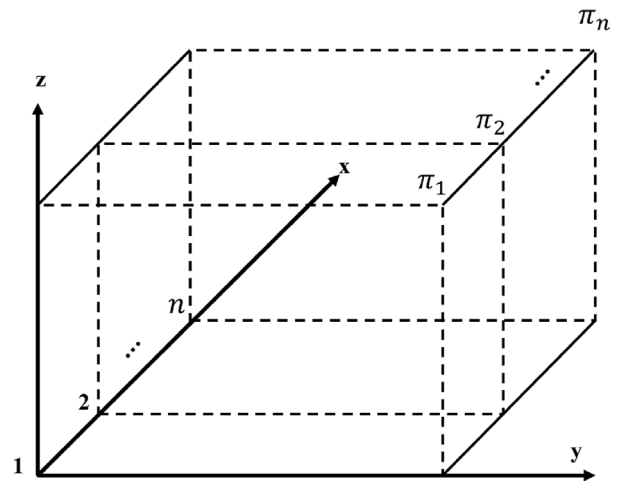


Fig. 10. First stage of 3D environment modeling.

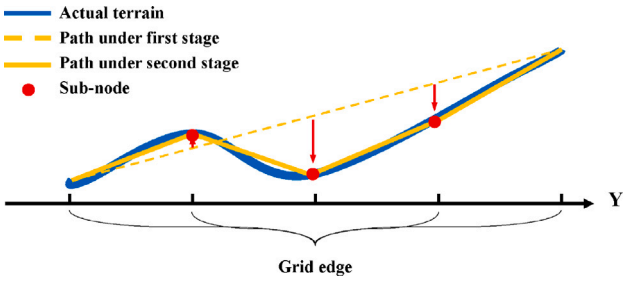


Fig. 11. Second stage of 3D environment modeling.

problem. The path length is the sum of the distance between adjacent path nodes. Therefore, the total path length can be expressed as:

$$L = \sum_{i=1}^{n-1} \sum_{j=1}^{n_{sub}^i} \left(d_{i,1}^i + d_{j,j+1}^i + d_{n_{sub}^i,i+1}^i \right), \tag{32}$$

where $d_{j,j+1}^i$ is the distance between the jth sub-node and the $(j+1)$th sub-node between the ith node and the $(i+1)$th node, and $d_{i,1}^i, d_{n_{sub}^i,i+1}^i$ respectively represent the distance between the sub-node and the adjacent primary node. It should be noted that the distance $d_{j,j+1}^i$ in this subsection is different from the previous definition, and the distance in three-dimensional space is shown as Eq. (33):

$$d_{j,j+1} = \sqrt{(x_{j+1} - x_j)^2 + (y_{j+1} - y_j)^2 + (z_{j+1} - z_j)^2}. \tag{33}$$

Constraints on collision and vehicle climbing ability can be converted into penalties and added to the objective function according to the above ideas. The transformation of collision avoidance constraint is as follows:

$$penalty_c = \begin{cases} 0, & collision - free, \\ M_c, & otherwise. \end{cases} \tag{34}$$

The constraint of vehicle climbing ability is similar to the collision avoidance constraint. That is, when the climbing angle is beyond the acceptable value, the penalty is an extremely large value; otherwise, it is 0. The penalty is shown as:

$$penalty_s = \begin{cases} 0, & \theta_{min} \le \theta \le \theta_{max}, \\ M_s, & otherwise, \end{cases} \tag{35}$$

where the calculation of angle θ is as follows:

$$\theta = \arctan\left(\frac{z_{j+1} - zj}{\sqrt{(x_{j+1} - x_j)^2 + (y_{j+1} - y_j)^2}} \right). \tag{36}$$

To sum up, the final problem modeling can be integrated into a single minimization objective function, which can be expressed as:

$$Minimize \quad F_{cost} = \omega \times \sum_{i=1}^{n-1} \sum_{j=1}^{n_{sub}^i} \left(d_{i,1}^i + d_{j,j+1}^i + d_{n_{sub}^i,i+1}^i \right) + Penalty_c + Penalty_s. \tag{37}$$

5.2.3. Experimental results

Similar to the experiments in 2D path planning, we compare the performance of the proposed QMABC algorithm with several meta-heuristic algorithms in the built environment. The model parameters are set to $\omega = 1, M_c = 1000, M_s = 1000$, and the algorithm parameters are the same as those described in the algorithm description. The experimental equipment environmental parameters are the same as the 2D problem. Each algorithm is executed 20 times independently.

The experimental results are the average path length and the shortest path length to show the robustness and upper limit of capability of different algorithms respectively. In addition, the execution time is no longer recorded because the generation of sub-nodes depends on the initial solution, which will have a great impact on the calculation time, resulting in a lack of universality. The results are shown in Table 8 and Fig. 12. In Fig. 12, the color represents the terrain elevation, and the red cylinder areas represent the obstacle zones, which means the unmanned vehicles cannot pass through this area. Since there are many algorithms implemented in the experiments, to avoid confusion about the paths, we only put four comparison algorithms and the proposed QMABC in the figures.

According to the solving results, the algorithm proposed in this paper achieves the best performance in both the best value and the average value within a certain number of iterations. However, compared with BDLDABC and CUSDE, its advantages are not obvious, so there is still room for improvement.

6. Conclusions

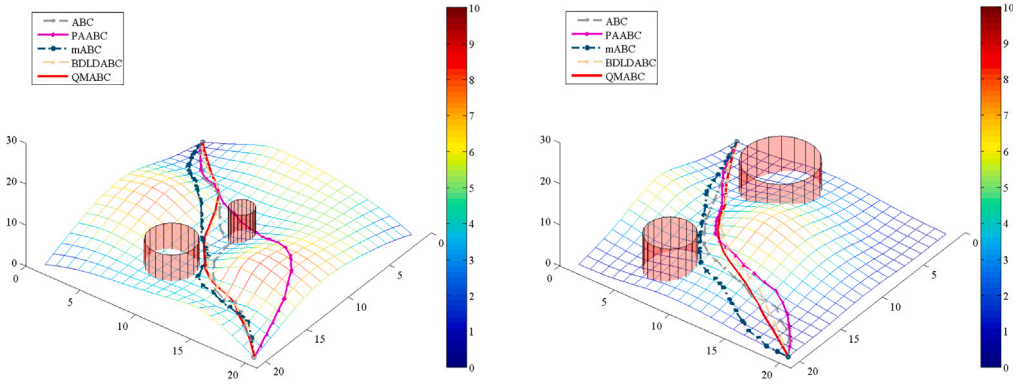
This paper proposes a novel multi-strategy based ABC algorithm integrated with a Q-learning framework for strategies selection named QMABC. To compensate for the limited exploitation ability of a single search equation, three distinct but effective strategies are added to the onlooker bee phase. Meanwhile, Q-learning is adopted to adjust the probability of each strategy being chosen via learning the historical population updating experience in the onlooker bee phase. In addition, a modified DE search equation replaces the original one in the employed bee phase to improve the exploration ability and increase the diversity of the population.

Both numerical and practical experiments are performed to demonstrate the performance of the proposed QMABC. In the numerical experiments, the comparison between QMABC and ABC variants, as well as other meta-heuristic algorithms are conducted on 29 CEC 2017 benchmark functions. The comparison results illustrate the outstanding accuracy and robustness of the proposed algorithm. In the practical experiments, both 2D and 3D path planning are taken as the optimization problems with the test of solving accuracy. The experimental results in path planning also show that QMABC is superior to four well-recognized path planning methods and other meta-heuristic algorithms.

Although QMABC has achieved good results, there is still room for further improvement. For example, more premium strategies can be introduced into QMABC for better results, which requires a lot of

Table 8
Experimental results and comparison between different algorithms in 3D environments.

Algorithm	Attribute	Map 1	Map 2
ABC	Mean length	33.6289	31.3661
	Shortest length	32.6398	30.8461
PAABC	Mean length	36.3252	31.5022
	Shortest length	35.3541	30.9233
mABC	Mean length	33.6818	30.8905
	Shortest length	32.4546	30.5141
BDLDABC	Mean length	33.697	31.487
	Shortest length	31.2304	30.2509
ABC_RL	Mean length	33.7287	31.0012
	Shortest length	32.3125	30.6231
DE	Mean length	34.8564	31.4614
	Shortest length	31.1356	30.2474
LJA	Mean length	35.2879	30.7804
	Shortest length	31.9123	29.911
ACWOA	Mean length	35.115	30.8407
	Shortest length	34.8759	30.7628
APDPSO	Mean length	38.0627	35.8649
	Shortest length	36.1127	33.2666
CUSDE	Mean length	34.119	30.179
	Shortest length	31.1624	29.8768
QMABC	Mean length	33.5649	30.3501
	Shortest length	31.0353	29.863



(a) Map 1 (b) Map 2

Fig. 12. The best path for different algorithms in different 3D environments.

work. Furthermore, QMABC can be extended to accommodate discrete or multi-objective problems.

CRediT authorship contribution statement

Xinrui Ni: Conceptualization, Methodology, Software, Data curation, Visualization, Writing – original draft. **Wei Hu:** Conceptualization, Methodology, Supervision, Investigation, Validation, Writing – reviewing & editing. **Qiaochu Fan:** Methodology, Supervision, Writing – reviewing & editing. **Yibing Cui:** Conceptualization, Software, Data curation. **Chongkai Qi:** Software, Data curation.

Declaration of competing 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.

Data availability

Data will be made available on request.

Acknowledgments

This work was supported by the Fundamental Research Funds for the Central Universities, China under Grant 2023JBMC042 and the National Natural Science Foundation of China under Grant 72288101.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.eswa.2023.121303.

References

Abd El Aziz, M., Ewees, A. A., & Hassanien, A. E. (2017). Whale optimization algorithm and moth-Flame optimization for multilevel thresholding image segmentation. *Expert Systems with Applications*, *83*, 242–256.

Akay, B., & Karaboga, D. (2012). A modified Artificial Bee Colony algorithm for real-parameter optimization. *Information Sciences*, *192*, 120–142.

Awad, N. H., Ali, M. Z., Suganthan, P. N., Liang, J. J., & Qu, B. Y. (2016). *Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective real-parameter numerical optimization*: *Technical report*.

Banitalebi, A., Aziz, M. I. A., Bahar, A., & Aziz, Z. A. (2015). Enhanced compact artificial bee colony. *Information Sciences*, *298*, 491–511.

Bhattacharya, A., & Chattopadhyay, P. K. (2010). Hybrid differential evolution with biogeography-based optimization for solution of economic load dispatch. *IEEE Transactions on Power Systems*, *25*(4), 1955–1964.

Brajevic, I., & Stanimirovic, P. (2018). An improved chaotic firefly algorithm for global numerical optimization. *International Journal of Computational Intelligence Systems*, *12*(1), 131–148.

Brajevic, I., Stanimirovic, P. S., Li, S., & Cao, X. (2020). A hybrid firefly and multi-strategy artificial bee colony algorithm. *International Journal of Computational Intelligence Systems*, *13*(1), 810–821.

Cao, Y., & Shi, H. (2021). An adaptive multi-strategy artificial bee colony algorithm for integrated process planning and scheduling. *IEEE ACCESS*, *9*, 65622–65637.

Chen, X., Tianfield, H., & Li, K. (2019). Self-adaptive differential artificial bee colony algorithm for global optimization problems. *Swarm and Evolutionary Computation*, *45*, 70–91.

Cui, Y., Hu, W., & Rahmani, A. (2022). A reinforcement learning based artificial bee colony algorithm with application in robot path planning. *Expert Systems with Applications*, *203*.

Cui, Y., Hu, W., & Rahmani, A. (2023). Fractional-order artificial bee colony algorithm with application in robot path planning. *European Journal of Operational Research*, *306*(1), 47–64.

Cui, L., Li, G., Wang, X., Lin, Q., Chen, J., Lu, N., & Lu, J. (2017). A ranking-based adaptive artificial bee colony algorithm for global numerical optimization. *Information Sciences*, *417*, 169–185.

Das, S., Mullick, S. S., & Suganthan, P. N. (2016). Recent advances in differential evolution – An updated survey. *Swarm and Evolutionary Computation*, *27*, 1–30.

Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics*, *26*(1), 29–41.

Etminaniesfahani, A., Gu, H., & Salehipour, A. (2022). ABFIA: A hybrid algorithm based on artificial bee colony and Fibonacci indicator algorithm. *Journal of Computational Science*, *61*.

Fister, I., Suganthan, P. N., Fister, I., Jr., Kamal, S. M., Al-Marzouki, F. M., Perc, M., & Strnad, D. (2016). Artificial neural network regression as a local search heuristic for ensemble strategies in differential evolution. *Nonlinear Dynamics*, *84*(2), 895–914.

Gao, W.-f., & Liu, S.-y. (2012). A modified artificial bee colony algorithm. *Computers & Operations Research*, *39*(3), 687–697.

Gao, W., Liu, S., & Huang, L. (2012). A global best artificial bee colony algorithm for global optimization. *Journal of Computational and Applied Mathematics*, *236*(11), 2741–2753.

Ghosh, A., Das, S., Chowdhury, A., & Gini, R. (2011). An improved differential evolution algorithm with fitness-based adaptation of the control parameters. *Information Sciences*, *181*(18), 3749–3765.

Guo, Y., Yao, D., Li, B., He, Z., Gao, H., & Li, L. (2022). Trajectory planning for an autonomous vehicle in spatially constrained environments. *IEEE Transactions on Intelligent Transportation Systems*, *23*(10), 18326–18336.

He, J., Liu, X., Duan, Q., Chan, W. K. V., & Qi, M. (2023). Reinforcement learning for multi-item retrieval in the puzzle-based storage system. *European Journal of Operational Research*, *305*(2), 820–837.

Holland, J. H. (1992). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. The MIT Press.

Hsieh, T.-J., Hsiao, H.-F., & Yeh, W.-C. (2011). Forecasting stock markets using wavelet transforms and recurrent neural networks: An integrated system based on artificial bee colony algorithm. *Applied Soft Computing*, *11*(2), 2510–2525.

Iacca, G., dos Santos Junior, V. C., & de Melo, V. V. (2021). An improved Jaya optimization algorithm with Levy flight. *Expert Systems with Applications*, *165*.

Jia, D., Guo, H., Song, Z., Shi, L., Deng, X., Perc, M., & Wang, Z. (2021). Local and global stimuli in reinforcement learning. *New Journal of Physics*, *23*(8).

Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, *39*(3), 459–471.

Karaboga, D., Gorkemli, B., Ozturk, C., & Karaboga, N. (2014). A comprehensive survey: Artificial bee colony (ABC) algorithm and applications. *Artificial Intelligence Review*, *42*(1), 21–57.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95 - International conference on neural networks, vol. 4* (pp. 1942–1948).

Lee, H.-R., & Lee, T. (2021). Multi-agent reinforcement learning algorithm to solve a partially-observable multi-agent problem in disaster response. *European Journal of Operational Research*, *291*(1), 296–308.

Lei, D., & He, S. (2022). An adaptive artificial bee colony for unrelated parallel machine schedulingwith additional resource and maintenance. *Expert Systems with Applications*, *205*.

Li, H., Gao, K., Duan, P.-Y., Li, J.-Q., & Zhang, L. (2023). An improved artificial bee colony algorithm with Q-learning for solving permutation flow-shop scheduling problems. *IEEE Transactions on Systems Man Cybernetics-Systems*, *53*(5), 2684–2693.

Liao, X., Zhou, J., Zhang, R., & Zhang, Y. (2012). An adaptive artificial bee colony algorithm for long-term economic dispatch in cascaded hydropower systems. *International Journal of Electrical Power & Energy Systems*, *43*(1), 1340–1345.

Liu, H., Cai, Z., & Wang, Y. (2010). Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. *Applied Soft Computing*, *10*(2), 629–640.

Liu, Q., Li, J., Ren, H., & Pang, W. (2022). All particles driving particle swarm optimization: Superior particles pulling plus inferior particles pushing. *Knowledge-Based Systems*, *249*.

Long, X., Zhang, J., Qi, X., Xu, W., Jin, T., & Zhou, K. (2022). A self-learning artificial bee colony algorithm based on reinforcement learning for a flexible job-shop scheduling problem. *Concurrency and Computation-Practice & Experience*, *34*(4).

Mallipeddi, R., Suganthan, P. N., Pan, Q. K., & Tasgetiren, M. F. (2011). Differential evolution algorithm with ensemble of parameters and mutation strategies. *Applied Soft Computing*, *11*(2), 1679–1696.

Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software*, *95*, 51–67.

Nguyen, T. T., Nguyen, N. D., & Nahavandi, S. (2020). Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE Transactions on Cybernetics*, *50*(9), 3826–3839.

Pan, T.-S., Dao, T.-K., Pan, J.-S., & Nguyen, T.-T. (2017). An unmanned aerial vehicle optimal route planning based on compact artificial bee colony. In *Advances in intelligent information hiding and Multimedia signal processing, vol 2, vol. 64* (pp. 361–369).

Pan, Q.-K., Tasgetiren, M. F., Suganthan, P. N., & Chua, T. J. (2011). A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem. *Information Sciences*, *181*(12), 2455–2468.

Pattnaik, S. K., Mishra, D., & Panda, S. (2022). A comparative study of meta-heuristics for local path planning of a mobile robot. *Engineering Optimization*, *54*(1), 134–152.

Sarker, I. H. (2021). Machine learning: Algorithms, real–world applications and research directions. *SN Computer Science*, *2*(3), 160.

Song, P.-C., Pan, J.-S., & Chu, S.-C. (2020). A parallel compact cuckoo search algorithm for three-dimensional path planning. *Applied Soft Computing*, *94*.

Song, B., Wang, Z., & Zou, L. (2021). An improved PSO algorithm for smooth path planning of mobile robots using continuous high-degree Bezier curve. *Applied Soft Computing*, *100*.

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

Szeto, W. Y., Wu, Y., & Ho, S. C. (2011). An artificial bee colony algorithm for the capacitated vehicle routing problem. *European Journal of Operational Research*, *215*(1), 126–135.

Tang, C., Sun, W., Xue, M., Zhang, X., Tang, H., & Wu, W. (2022). A hybrid whale optimization algorithm with artificial bee colony. *Soft Computing*, *26*(5), 2075–2097.

Ustun, D., Toktas, A., Erkan, U., & Akdagli, A. (2022). Modified artificial bee colony algorithm with differential evolution to enhance precision and convergence performance. *Expert Systems with Applications*, *198*.

Wang, J., Jia, X., Zhang, T., Ma, N., & Meng, M. Q.-H. (2022). Deep neural network enhanced sampling-based path planning in 3D space. *IEEE Transactions on Automation Science and Engineering*, *19*(4), 3434–3443.

Wang, Y., Jiao, J., Liu, J., & Xiao, R. (2022). A labor division artificial bee colony algorithm based on behavioral development. *Information Sciences*, *606*, 152–172.

Wang, J., Lei, D., & Cai, J. (2022). An adaptive artificial bee colony with reinforcement learning for distributed three-stage assembly scheduling with maintenance. *Applied Soft Computing*, *117*.

Wang, J., Lei, D., & Li, M. (2022). A Q-learning-based artificial bee colony algorithm for distributed three-stage assembly scheduling with factory eligibility and setup times. *Machines*, *10*(8).

Watkins, C., & Dayan, P. (1992). Q-learning. *Machine Learning*, *8*(3–4), 279–292.

Xiang, Y., Zhou, Y., & Liu, H. (2015). An elitism based multi-objective artificial bee colony algorithm. *European Journal of Operational Research*, *245*(1), 168–193.

Xu, F., Li, H., Pun, C.-M., Hu, H., Li, Y., Song, Y., & Gao, H. (2020). A new global best guided artificial bee colony algorithm with application in robot path planning. *Applied Soft Computing*, *88*.

Yong, Z., Chun-lin, H., Xian-fang, S., & Xiao-yan, S. (2021). A multi-strategy integrated multi-objective artificial bee colony for unsupervised band selection of hyperspectral images. *Swarm and Evolutionary Computation*, *60*.

Zhao, M., Song, X., & Xing, S. (2022). Improved artificial bee colony algorithm with adaptive parameter for numerical optimization. *Applied Artificial Intelligence*, *36*(1).

Zhou, X., Lu, J., Huang, J., Zhong, M., & Wang, M. (2021). Enhancing artificial bee colony algorithm with multi-elite guidance. *Information Sciences*, *543*, 242–258.

Zhou, X., Song, J., Wu, S., & Wang, M. (2023). Artificial bee colony algorithm based on online fitness landscape analysis. *Information Sciences*, *619*, 603–629.

Zhou, X., Wu, Y., Zhong, M., & Wang, M. (2022). Artificial bee colony algorithm based on adaptive neighborhood topologies. *Information Sciences*, *610*, 1078–1101.

Zhou, B., & Zhao, Z. (2023). An adaptive artificial bee colony algorithm enhanced by Deep Q-Learning for milk-run vehicle scheduling problem based on supply hub. *Knowledge-Based Systems*, *264*.

Zhu, G., & Kwong, S. (2010). Gbest-guided artificial bee colony algorithm for numerical function optimization. *Applied Mathematics and Computation*, *217*(7), 3166–3173.

Zhu, S., Pun, C.-M., Zhu, H., Li, S., Huang, X., & Gao, H. (2023). An artificial bee colony algorithm with a balance strategy for wireless sensor network. *Applied Soft Computing*.

Zou, L., Pan, Z., Gao, Z., & Gao, J. (2022). Improving the search accuracy of differential evolution by using the number of consecutive unsuccessful updates. *Knowledge-Based Systems*, *250*.