




# A novel marine predators algorithm with adaptive update strategy

Tao Chen<sup>1</sup> · Yong Chen<sup>1</sup>  · Zhicheng He<sup>1</sup> · Eric Li<sup>2,3</sup> · Chenglin Zhang<sup>1</sup> · Yuanyi Huang<sup>4</sup>

Accepted: 16 October 2022 / Published online: 10 November 2022

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

## Abstract

The marine predators algorithm (MPA) is a metaheuristic algorithm for solving optimization problems. MPA divides the whole optimization process into three phases evenly, and each phase corresponds to a different search agent update strategy. Such a setup makes MPA inflexible when facing different optimization problems, which affects the optimization performance. In this paper, we propose a novel modified MPA hybridizing by Q-learning (QMPA), which applies reinforcement learning to the selection of update strategy, and selects the most appropriate position update strategy for search agents in different iteration stages and states. It can effectively compensate for the deficiency of MPA's adaptive ability when facing different optimization problems. The performance of QMPA is tested on classical benchmark functions, the CEC2014 test suite, and engineering problems. In the classical benchmark functions test, QMPA is compared with MPA in 10, 30, and 50 dimensions. QMPA performs better than MPA for seven of the ten functions when the dimension is 10 and 30. The results of dimension 50 show that QMPA outperforms MPA in 5 functions and is close to it in 4 functions. Then, comparing QMPA with algorithms such as grey wolf optimizer, particle swarm optimization, slime mould algorithm, sine cosine algorithm, reptile search algorithm, and aquila optimizer, the results show that QMPA has the best performance on 22 of the total 30 functions in the CEC2014 test suite. Finally, QMPA is tested on two commonly used real-world engineering problems and gives the most optimal results. In general, the adaptive update strategy proposed in this paper improves the optimization performance of the MPA algorithm in terms of convergence and stability.

**Keywords** Metaheuristic algorithm · Update strategy · Marine predators algorithm (MPA) · Reinforcement learning

---

✉ Zhicheng He  
hezicheng815@gmail.com

Extended author information available on the last page of the article

## 1 Introduction

In the 1960s, inspired by bionics, metaheuristic optimization algorithms, which take inspiration from random phenomena in nature and combine stochastic algorithms with local algorithms, emerged. Metaheuristic optimization algorithms can solve large-scale complex problems quickly and do not have any requirement on the objective function and are not limited to specific problems, and these make them one of the hot problems in the research of optimization problems. The core of the metaheuristic algorithm is exploration and exploitation, where exploration means exploring the entire search space. And exploitation means to use the information obtained from exploration to approach the optimal solution step by step. In general, a metaheuristic algorithm with good performance will usually maintain a balance between exploration and exploitation [1, 2].

The marine predators algorithm (MPA) is a nature-based metaheuristic algorithm proposed by Faramarzi et al. in 2020, with the main revelation being the widely adopted foraging strategies Levy and Brownian motion of marine predators [3]. It is proven to be an optimization algorithm with good performance. However, the whole optimization process is divided into three fixed phases according to the iteration number, and each phase has a specific location update formula. In each iteration phase, a specific location update formulas of search agents are defined. In other words, the position of the search agents will be updated in a fixed way according to the iterative process. Such an update strategy is not flexible and specific. When facing different optimization problems, such a solidified search agent position update strategy may affect the balance between exploration and exploitation in the optimization process, and the phenomenon of premature convergence may occur, thus affecting the optimization performance. According to the No Free Lunch Theorem [4], there does not exist any optimization algorithm that performs better than other optimization algorithms on all problems. Therefore, the search agent location update formula of the algorithm must be adaptively adjusted to the characteristics of the problem, as well as to the optimization situation. Maintaining a balance between exploration and exploitation during the optimization process and avoiding premature convergence will enable the algorithm to have better performance.

In this study, a novel modified Marine Predators Algorithm hybridizing by Q-learning (QMPA) is proposed, which adaptively adjusts the update strategies of different search agents at different iteration phases. This will avoid the imbalance between exploitation and exploration, effectively improving the ability to search globally and optimizing performance. The main contributions of this paper are as follows:

An adaptive control strategy based on reinforcement learning is proposed.

A novel MPA algorithm for search agent adaptive selection of location update method is proposed.

Based on 10 classical benchmark functions, QMPA and MPA are compared in three dimensions 10, 30, and 50.

Comparative experiments of QMPA and other algorithms are conducted on the CEC2014 test function set and two engineering problems.

The remaining part of the paper is structured as follows: A review of related work is given in Sect. 2, and Sect. 3 provides basic information about the MPA and the fundamentals of Q-learning. In Sect. 4, the proposed QMPA is illustrated in detail. Section 5 provides the experiments and the related results. Section 6 describes the threats to validity. The conclusion of the whole paper and recommendations for future related work are summarized in Sect. 7.

## 2 Related work

Since the proposed method is based on the adaptive control of the metaheuristic algorithm, this section will review both the metaheuristic algorithm and its adaptive control, respectively, to better highlight the existing research results and deficiencies.

Metaheuristic algorithms have developed rapidly in recent years and many algorithms have emerged. Among them, there are three main categories, which are swarm intelligence algorithms, physics-based and evolutionary algorithms. Algorithms such as genetic algorithm [5] and differential evolution algorithm [6] belong to evolutionary algorithms. Simulated annealing [7] and gravitational search [8] belong to the physics-based algorithms. The swarm intelligence algorithms include particle swarm optimization (PSO) [9], grey wolf optimizer (GWO) [10] and MPA [3], etc. Meanwhile, metaheuristic optimization algorithms have a wide range of applications in the fields of medical [11], mechanical design [12], parametric optimization [13] and image processing [14], and have good performance.

Although these metaheuristic algorithms may be able to solve relatively complex optimization problems in different scenarios, finding the global optimal solution is still a challenging task. This is because these algorithms usually converge prematurely and fall into local optimum due to the imbalance between exploitation and exploration [15]. The same problem exists for MPA. When MPA faces multimodal and nonlinear problems, it usually encounters with stagnation and falling into local optimum [16]. Literature [17] points out premature convergence is a problem for MPA, which divides the optimization iteration phase into three phases, namely, exploration phase, exploration to exploitation phase, and exploitation phase. It may lead to falling into a local optimum. In the scheduling optimization of charging and discharging, the problem of falling into local optimum of MPA is also pointed out in the literature [18]. Similarly, literatures [19–21] also illustrate the shortcomings of MPA such as lacking of population diversity and the tendency to converge prematurely in the last optimization stage.

Facing the problem that metaheuristic algorithms tend to converge too early and fall into local optimum, many scholars have proposed some solutions. For example, hybrid algorithms are constructed by hybridizing multiple methods, which can effectively improve the phenomenon of premature convergence. Emperor Penguins Colony algorithm combined with genetic operators has shown good performance in community detection in complex networks [22]. Integration of particle swarm algorithm with a genetic algorithm can solve the vehicle path optimization problem well [23]. Literature [24] combines ant colony algorithm with a genetic algorithm to better solve the delivery scheduling optimization problem. In the problem of determining optimal design parameters of model predictive control, the hybrid firefly–whale optimization algorithm

showed good performance [25]. In the literature [26], a hybrid algorithm combining particle swarm optimization algorithm and bat algorithm is proposed for solar PV microgrid capacity configuration optimization. To improve the performance of the MPA algorithm in the optimal design of automotive components, literature [27] hybridized the Nelder-Mead algorithm with MPA. Although the hybrid algorithm has good performance in a certain area, it is designed for a specific problem, which makes the application of the algorithm possess some limitations.

In order to make the metaheuristic optimization algorithm have good performance facing different complex problems, some scholars have improved the algorithm using adaptive control strategies. As studied by some scholars, there are two main improvement mechanisms: automatic adjustment and feedback adjustment. Automatic adjustment is the search for the optimal solution along with the search for parameters. In the literature [28], three parameters of the inertial weight and two cognition acceleration coefficients of PSO are updated together with the particles, improving the performance of the algorithm. In the literature [29], two fixed parameters in the differential evolution algorithm are iteratively searched in the specified range, enhancing the convergence velocity and its performance. However, automatic adjustment is a way of implicit parameter adjustment, which is not conducive to understanding and analysis. Feedback adjustment is used to guide parameter adjustment through the feedback on algorithm performance changes. An adaptive parameter control model is proposed in the literature [30]. The parameters of the particle swarm optimization algorithm are adjusted based on a linearly decreasing ideal velocity profile [31]. In the literature [32], the parameters of fruit fly optimization algorithm are controlled by feedback based on the state of the already generated solution. The literature [33] redefines a nonlinear step-factor parameter control strategy in order to improve the self-adaptability of MPA. The literature [34] set a strategy for the selection of exploration and exploitation phases of the MPA optimization process according to the iterative process, which has a certain improvement on the performance of the MPA algorithm. However, this strategy is limited by the fact that it is only related to the iterative process, and has some limitations on performance improvement. The literature [35] improves the exploration and exploitation capability of MPA by an adaptive update formula, which is based on the global optimal fitness. However, this adaptive update formula is only related to the current search state and cannot avoid falling into local optimum. Unlike automatic adjustment, feedback adjustment is performed by displaying parameters, which facilitates understanding and analysis, but is not conducive to controlling the direction and degree of adjustment.

As mentioned above, many metaheuristic optimization algorithms always suffer from premature convergence, and this is also true for MPA. Although there are many studies to improve MPA, they are mainly oriented to specific problems. In addition, there are limitations in studies targeting the combination of MPA and adaptive strategies. The performance of the MPA is limited when faced with different complex optimization problems, while the existing adaptive control strategies have some drawbacks. For MPA, the key to achieving good performance is to strike a balance between exploration and exploitation. On the one hand, the algorithm needs to search extensively for regions with exploitation prospects. On the other hand, the algorithm also needs to perform further in-depth exploitation within the regions with exploitation prospects searched in the earlier stage. It is known that reinforcement learning has a long-term perspective, and

it is widely used to guide the selection of actions based on the outcome of future trials [36–38]. In this study, reinforcement learning is combined with the MPA algorithm to adaptively adjust the update method of different search agents at different iteration stages by Q-learning. Specifically, the choice of update method is based on a Q-table that has been matured through multiple training cycles. The Q-table consists of two dimensions, including the action and the state. The action is the four update formulas defined in the standard MPA, and the state reflects the difference between the search agents and the optimum solution. Each search agent has a different Q-table at various iteration stages, and each search agent updates the Q-table by training it several times in each iteration. The Q-table updates are based on a reward function, which is set considering both different iteration stages and different search agents. The whole adaptive adjustment process is based on the feedback of reinforcement learning for future exploration, which is real-time, targeted and effective. At the same time, this adaptive adjustment mechanism is easy to understand and control. The approach proposed in this paper can enhance the balance between exploitation and exploration, avoid the phenomenon of premature convergence, and improve the performance in the face of different complex optimization problems.

### 3 Briefing on the MPA and Q-learning

For more effective discussion, the ideas of the MPA and Q-learning are firstly briefed.

#### 3.1 MPA

The marine predators algorithm is a metaheuristic optimization algorithm proposed based on the foraging law of marine predators [3]. It is mainly inspired by the predatory behaviors and foraging strategies of marine organisms such as sharks, sunfish, and swordfish, and solves the optimization problem by simulating the laws of these marine organisms in predation on their prey. Among them, the foraging strategies mainly include two categories of Levy flight and Brownian motion. In the optimization process, foraging strategies are selected according to the different movement speeds between predator and prey. The probability of encounter between predator and prey is maximized by weighing the choice of two foraging strategies, Levy flight, and Brownian motion, at different stages.

Like most other algorithms, the initial solution of MPA is uniformly distributed:

$$X_0 = X_{lb} + rd(X_{ub} - X_{lb}) \quad (1)$$

where  $rd$  is a random vector, uniformly distributed between 0 and 1,  $X_{lb}$  is the minimal boundary of the variables, and  $X_{ub}$  is the maximal boundary of the variables.

After initialization, a matrix named Pre will be constructed and the predators' positions update will be based on this matrix. In addition, the best search agents are selected in each iteration to form the *Eli* matrix, which has the same dimensions as the Pre matrix, and the search agents in the *Eli* matrix is called the top predator.

$$Pre = \begin{bmatrix} X_{1,1} & X_{1,2} & \cdots & X_{1, Dim} \\ X_{2,1} & X_{2,2} & \cdots & X_{2, Dim} \\ \vdots & \vdots & \vdots & \vdots \\ X_{n,1} & X_{n,2} & X_{n,2} & X_{n, Dim} \end{bmatrix}_{n \times Dim} \tag{2}$$

where  $X_{i,j}$  presents the  $j$ th dimension of  $i$ th prey, Dim is the dimension of search agents, and  $n$  means the number of search agents.

Considering the speed between predator and prey, the search agents' search for superiority is divided into three stages. Each process corresponds to a different position update strategy.

Phase1: In the early phase of optimization, exploration is significant. Generally, the prey at this stage moves faster than the predator, and staying still is the better way for predators, and the position is updated in the following way.

$$\text{While } t \leq \frac{1}{3}t_{max}$$

$$\begin{aligned} \overrightarrow{step}_i &= \overrightarrow{M}_B \otimes \left( \overrightarrow{Eli}_i - \overrightarrow{M}_B \otimes \overrightarrow{Pre}_i \right), \quad i = 1, \dots, n \\ \overrightarrow{Pre}_i &= \overrightarrow{Pre}_i + C * \overrightarrow{rand} \otimes \overrightarrow{step}_i \end{aligned} \tag{3}$$

where  $\overrightarrow{M}_B$  is a vector representing the Brownian motion,  $C$  is a constant number,  $\otimes$  is entry-wise multiplications,  $\overrightarrow{rand}$  is a vector that is randomly and uniformly distributed from 0 to 1.

Phase2: During the intermediate phase of optimization, the speed of movement of prey and predator is very close, and at this time exploration and exploitation are equally important. Dividing the search agents into two parts, one for exploration and the other for exploitation, the detailed position update is shown as follows:

$$\text{While } \frac{1}{3}t_{max} < t < \frac{2}{3}t_{max}$$

For the first half of the search agents

$$\begin{aligned} \overrightarrow{step}_i &= \overrightarrow{M}_L \otimes \left( \overrightarrow{Eli}_i - \overrightarrow{M}_L \otimes \overrightarrow{Pre}_i \right), \quad i = 1, \dots, n/2 \\ \overrightarrow{Pre}_i &= \overrightarrow{Pre}_i + C * \overrightarrow{rand} \otimes \overrightarrow{step}_i \end{aligned} \tag{4}$$

where  $\overrightarrow{M}_L$  is a vector representing the Levy movements.

For another half of the search agents

$$\begin{aligned} \overrightarrow{step}_i &= \overrightarrow{M}_B \otimes \left( \overrightarrow{M}_B \otimes \overrightarrow{Eli}_i - \overrightarrow{Pre}_i \right), \quad i = n/2 + 1, \dots, n \\ \overrightarrow{Pre}_i &= \overrightarrow{Eli}_i + C * CF \otimes \overrightarrow{step}_i \end{aligned} \tag{5}$$

where  $CF$  is a parameter that is used in the optimization process to control the step size, and the formula of mathematics is shown as follows:

$$CF = \left(1 - t/t_{\max}\right)^{\left(2t/t_{\max}\right)} \tag{6}$$

Phase3: At the end of the optimization phase, prey is not as fast as predators. It is usually associated with high exploitation, and the position is updated in the following way.

$$\text{While } t \geq \frac{2}{3}t_{Max}$$

$$\begin{aligned} \overrightarrow{\text{step}}_i &= \overrightarrow{M}_L \otimes \left(\overrightarrow{M}_L \otimes \overrightarrow{\text{Eli}}_i - \overrightarrow{\text{Pre}}_i\right), i = 1, \dots, n \\ \overrightarrow{\text{Pre}}_i &= \overrightarrow{\text{Eli}}_i + C * CF \otimes \overrightarrow{\text{step}}_i \end{aligned} \tag{7}$$

In addition, some environmental changes may affect predator behavior. For example, sharks are distributed around the fish aggregating devices (FADs) most of the time [39]. The FADs are usually viewed as local optima and considering longer jumps can avoid falling into local optima. The mathematical expression for FADs is given as follows:

$$\overrightarrow{\text{Pre}}_i = \begin{cases} \overrightarrow{\text{Pre}}_i + CF \left[ \overrightarrow{X}_{lb} + \text{rand} \otimes \left( \overrightarrow{X}_{ub} - \overrightarrow{X}_{lb} \right) \right] \otimes \vec{U}, & \text{if } r \leq \text{FADs} \\ \overrightarrow{\text{Pre}}_i + [\text{FADs}(1 - r) + r] \left( \overrightarrow{\text{Pre}}_{r_1} - \overrightarrow{\text{Pre}}_{r_2} \right), & \text{if } r > \text{FADs} \end{cases} \tag{8}$$

where  $r$  is a number, randomly distributed from 0 to 1.  $\vec{U}$  is a vector containing only 0 and 1.  $r_1$  and  $r_2$  represent random indexes.

The pseudo-code of the MPA is given as follows:

**Algorithm 1** Pseudo-code of MPA**Input:** Search agents(Prey) population  $X_i(i = 1, 2, \dots, n)$ ,  $t_{max}$ **Output:** Optimal fitness value  $fit_{best}$ 

```

1: while  $t < t_{max}$  do
2:   Calculate the fitness value of prey
3:   Record the best value and accomplish memory saving
4:   if  $t \leq t_{max}/3$  then
5:     Update prey based on Eq.(4)
6:   else if  $t_{max}/3 < t < 2 * t_{max}/3$  then
7:     For the first half of the populations ( $i = 1, 2, \dots, n/2$ )
8:       Update prey based on Eq.(5)
9:     For the other half of the populations ( $i = n/2 + 1, \dots, n$ )
10:      Update prey based on Eq.(6)
11:   else if  $t \geq 2 * t_{max}/3$  then
12:     Update prey based on Eq.(8)
13:   end if
14:   Calculate the fitness value of prey and update the best value
15:   Accomplish memory saving
16:   Applying FADs effect and update based on Eq.(9)
17: end while
18: return The best fitness

```

The original algorithm divides the entire iterative cycle into three phases equally, and in the second phase, the search agents is fixed into two parts. Each phase adopts a corresponding update strategy. This fixed division is not conducive to a more efficient and targeted search agent search.

### 3.2 Q-learning

Reinforcement learning is a popular machine learning technique in recent years and has achieved remarkable results in many fields [36–38]. Q-learning is a value-based algorithm in RL, proposed by Watkins in 1989. The iterative formula for Q-learning is shown in Eq. (9).

$$Q(s_{m+1}, a_{m+1}) = (1 - \alpha)Q(s_m, a_m) + \alpha \left[ R(s_m, a_m) + \gamma \max_a Q(s_{m+1}, a) \right] \quad (9)$$

where  $\gamma$  represents the discount factor,  $\alpha$  denotes the learning rate,  $R(s_m, a_m)$  is the reward obtained by the agent for performing the action  $a_m$  in the state  $s_m$  and  $Q(s_m, a_m)$  is the cumulative reward earned by the agent at time  $m$ .

A Q value table is first created in Q-learning. The agent gets feedback through continuous interaction in the hell environment and forms a reward value for the agent's state-action pair. Through continuous iterative modification of the Q table, the positive reward action will be chosen by the search agent, and the probability of the corresponding action will increase. After several iterations, the optimal set of Actions is formed. Figure 1 demonstrates the model of Q-learning.



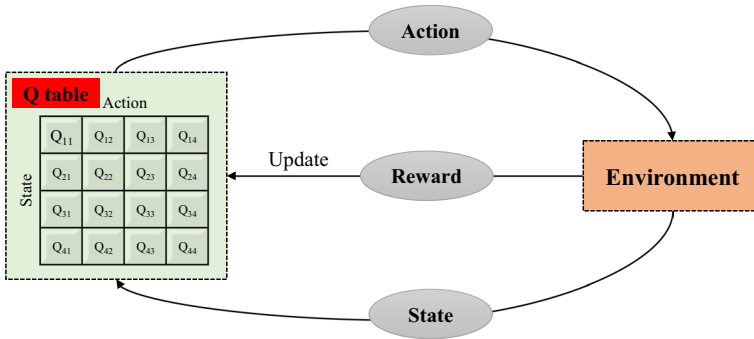


Fig. 1 The model of Q-learning

### 4 The proposed QMPA

QMPA is a new algorithm based on the standard MPA by Q-learning proposed in this paper. The algorithm mainly consists of four important components: action, state, reward, and Q table, as Fig. 2 shows, where the Q table is the core, which guides the search agent in which state to take which action, and its update is in turn based on the reward generated by the action taken by the search agent. This section will describe the components of QMPA in detail.

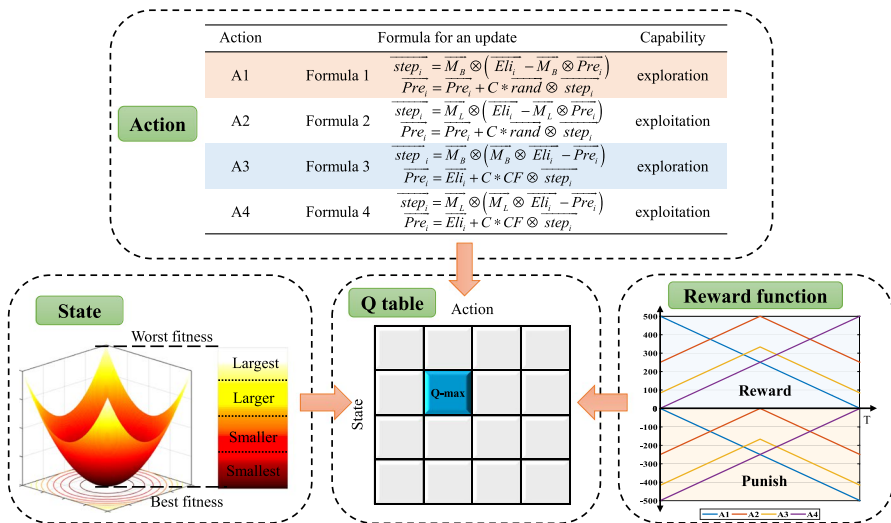


Fig. 2 The important elements in the QMPA

## 4.1 State and action

To distinguish the state that each search agent is in, this paper divides the whole search agent into four states according to the fitness, which are Largest, Larger, Smaller, and Smallest, as shown in Table 1, representing the relative performance concerning the global worst and the best, respectively. The global worst fitness and the best fitness are set as two extremes, and the difference between these two extremes is the average division into four parts, which are four different states. Each search agent determines the state based on the fitness. In the standard MPA, four update formulas are defined according to different stages, however, in this paper, these four update formulas are defined as four kinds of actions.

## 4.2 Q table

The next step is the design of the Q table, a two-dimensional table with rows representing state and action. When the search agent is judged to be in the state, it can select the corresponding row and then select the action that generates the largest Q value in this row.

## 4.3 Reward function

First, it is necessary to decide whether a search agent is rewarded or punished. If the fitness of a search agent after taking an action has improved compared with the previous one, it is rewarded and its corresponding Q value is increased. Conversely, if the fitness has decreased, it is punished and its corresponding Q value is decreased.

Determining the level of the reward or punishment will be the next step to be taken. In the standard MPA, it divides the entire iteration cycle into three parts, where the search agent is encouraged to do exploration during the first one-third of the iteration cycle, and all search agents take Eq. (3) for updating. In the middle third of the iteration cycle, half of the search agents are encouraged to exploit and updated using Eq. (4), and the other half of the search agents are updated using Eq. (5) to continue the exploration. In the final stage of the iteration, all search agents are encouraged to exploitation and updated using Eq. (7). Inspired by the standard MPA, this paper will set the reward and punishment intensity with the period of iteration and the different search agents, and the reward and punishment intensity formula are shown in Eq. (10).

**Table 1** The definition of state

Relative fitness	State
$\text{fitness} \geq 0.75\Delta F$	Largest
$0.5\Delta F \leq \text{fitness} < 0.75\Delta F$	Larger
$0.25\Delta F \leq \text{fitness} < 0.5\Delta F$	Smaller
$\text{fitness} < 0.25\Delta F$	Smallest

$$\begin{cases}
 R_{A1}(t) = 500 - \frac{500}{t_{\max}}t \\
 R_{A2}(t) = \begin{cases} 250 + \frac{500}{t_{\max}}t - \frac{167}{n}i, t \leq \frac{t_{\max}}{2} & (i = 1, 2, \dots, n) \\ 750 - \frac{500}{t_{\max}}t - \frac{167}{n}i, t > \frac{t_{\max}}{2} & (i = 1, 2, \dots, n) \end{cases} \\
 R_{A3}(t) = \begin{cases} 83 + \frac{500}{t_{\max}}t + \frac{167}{n}i, t \leq \frac{t_{\max}}{2} & (i = 1, 2, \dots, n) \\ 583 - \frac{500}{t_{\max}}t + \frac{167}{n}i, t > \frac{t_{\max}}{2} & (i = 1, 2, \dots, n) \end{cases} \\
 R_{A4}(t) = \frac{500}{t_{\max}}t \\
 P_{Ak}(t) = R_{Ak}(t) - 500 \quad (k = 1, 2, 3, 4)
 \end{cases} \tag{10}$$

where  $R_{Ak}$  denotes the reward value of the  $k$  th action,  $P_{Ak}$  represents the punishment value of the  $k$  th action, and  $n$  means the number of search agents.

As shown in Fig. 3, for rewards, actions A1 and A4 produce monotonically increasing and monotonically decreasing reward values, respectively, and are the same for all search agents. The reward values generated by actions A2 and A3 are a dashed line and are different for different search agents, varying in the blue shaded area. All  $n$  search agents are sorted sequentially, with the first search agent corresponding to the A2 action producing a reward value at the top of the blue area, and the A3 action producing a reward value at the bottom of the blue area. As the search agent number increases, the reward value created by the A2 action gradually moves downward, while the reward value created by the A3 action gradually moves upward. For the  $n/2$ nd search agent, the reward values generated

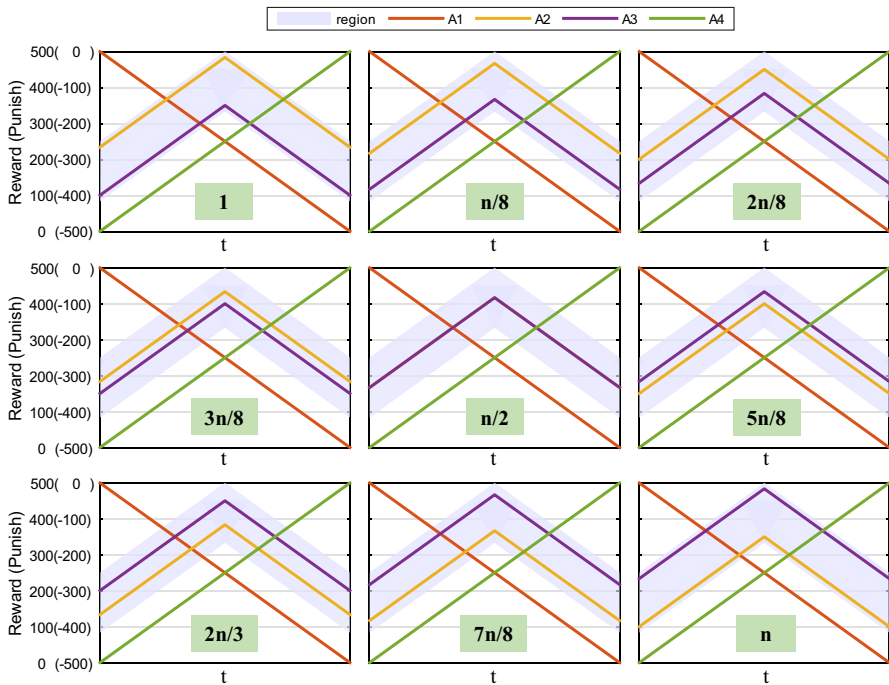


Fig. 3 Reward mechanism

by actions A2 and A3 are equal, and for the  $n^{\text{th}}$  search agent, the reward value generated by action A2 is located at the bottom of the blue area, while the reward value generated by action A3 is located at the top of the blue area. For the punishment, the punishment value is set similarly to the reward value, by subtracting 500 from the corresponding reward value.

In general, at the beginning of the iteration, action A1 yields the highest reward, indicating that the execution of action A1 is more encouraged at the beginning of the iteration. In the middle stage of the iteration, the reward values corresponding to actions A2 and A3 are shifted, with half of the search agents executing action A2 with a higher reward value and the other half executing action A3 with a higher reward value. In the final stages of the iteration, action A4 yields the strongest reward. The reason for setting such a reward mechanism is that in the early stage of optimization, the search agent needs more exploration ability to discover more potential regions. However, in the middle stage, exploration and exploitation are equally important in order to discover the optimal solution while jumping out of the local optimum. In the late stage of optimization, it needs more exploitation ability to find the optimal solution locally. Unusually, the reward mechanism set in this paper is dynamic, and it is related to the number of iterations and search agents. This flexible setting allows the algorithm to better maintain a balance between exploration and exploitation capabilities during the optimization process.

#### 4.4 The flowchart of QMPA

The flowchart of the QMPA is presented in Fig. 4, where  $t$  denotes the current number of iterations,  $t_{\text{max}}$  denotes the maximum number of iterations,  $epi$  is the maximum number of steps that Q-learning can explore in the future, and  $q$  denotes the number of steps that Q-learning currently explores. First, initialize the search agents,  $t_{\text{max}}$ ,  $n$ ,  $epi$ , then calculate the fitness. Next, the search agent position is updated, and the whole iteration cycle is divided into two parts. When running the first 80% of the iterative process, adaptive learning is performed by Q-learning to select the best position update formula

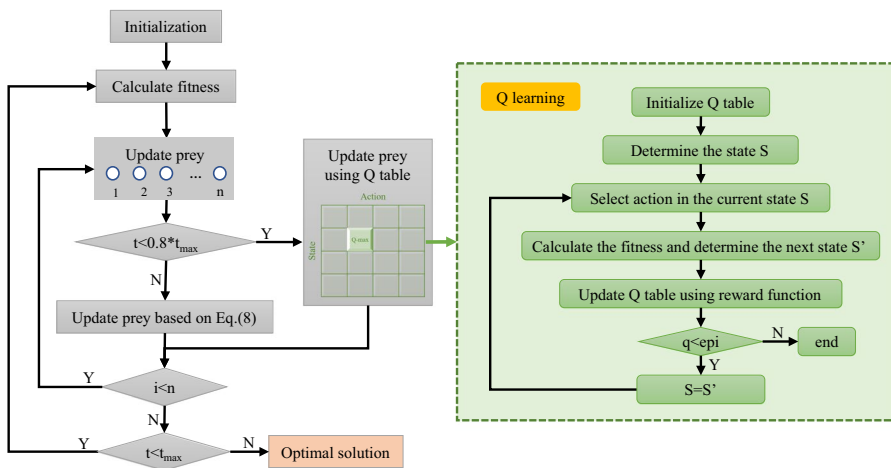


Fig. 4 The flowchart of the QMPA

for each search agent in each iteration. However, when the number of iterations is the last 20%, the position update is performed by Eq. (7). The reason for this setup is that later in the optimization phase, the global search needs to be stopped and a local search is performed when exploitation is very important. Therefore, it is very appropriate to use Eq. (7) to change the position of the search agent at the late stage of optimization. In Q-learning adaptive learning, firstly, the Q table is initialized. Then, the initial state  $S$  is determined according to the fitness of the search agent. Thirdly, the action is selected under the current state by the Q table. Fourthly, the fitness after performing the action is calculated and the next state  $S'$  is determined. Fifthly, the Q table is updated according to the reward function, and finally the current state is  $S'$  and the next cycle is continued. Until all the cycle counts are completed, a mature Q table is obtained, and then the Q table is used for position updating. The multiple cycles are designed to allow the search agent to explore *epi* steps forward, and a mature Q table can bring out future actions that maximize long-term benefits. Repeat the whole process for each search agent of each iteration to finish updating the position. Continue repeating the next iteration until the whole iteration process is completed. Algorithm 2 presents the pseudo-code of QMPA.

---

**Algorithm 2** Pseudo-code of QMPA
 

---

**Input:** Search agents(Prey) population  $X_i(i = 1, 2, \dots, n)$ ,  $t_{max}$ ,  $n$ , *epi*

**Output:** Optimal fitness value  $fit_{best}$

```

1: while  $t \leq t_{max}$  do
2:   Calculate the fitness value of prey
3:   Record the best value and accomplish memory saving
4:   if  $t == 1$  then
5:     while  $i \leq n$  do
6:       Randomly take the formula for updating prey
7:     end while
8:   else if  $t \leq 0.8 * t_{max}$  then
9:     while  $i \leq n$  do
10:      Initialize Q table
11:      Determine the state  $S$  in space
12:      while  $q \leq epi$  do
13:        Determine the action using Q table
14:        Calculate the fitness value and determine the state  $S'$  in space
15:        Update Q table using reward function
16:         $S = S'$ 
17:      end while
18:      Update prey using Q table
19:    end while
20:   else if  $t > 0.8 * t_{max}$  then
21:     while  $i \leq n$  do
22:       Update prey based on Eq.(8)
23:     end while
24:   end if
25:   Calculate the fitness value of prey and update the best value
26:   Accomplish memory saving
27:   Applying FADs effect and update based on Eq.(9)
28: end while
29: return The best fitness

```

---

## 5 Numerical experiment

To evaluate the proposed QMPA with the standard MPA as well as other algorithms, three different experiments were set up in this study. Experiment 1 was performed on 10 classical test functions, the second set of experiment was performed on the challenging CEC2014 test suite, and the third set of experiment was optimized for two engineering problems.

### 5.1 Experiment 1

In the first experiment, the behavior of QMPA and MPA in different dimensions is compared by commonly used classic test functions. These classic test functions include unimodal and multimodal functions and their properties are shown in Table 2. The unimodal functions are mainly designed to test the exploitation capability, while the exploration performance is experimented with by the multimodal functions. Figure 5 shows two-dimensional images of these functions. These functions will be tested in 10, 30, and 50 dimensions (Dim) using MPA respectively. Each test took 500 iterations, and the number of population was 25. To make the results more meaningful, each test function ran 31 times independently. The common parameter  $P = 0.5$  and  $FADs = 0.2$  in the algorithm, and the number of cycles(*epi*) in QMPA was set to 10.

Table 3 presents the results of tests for Dim = 10, and the comparison results for Dim = 30 and Dim = 50 are in Tables 4 and 5, and the optimal value among them is bolded. Table 3 demonstrates that the QMPA performs better than MPA for seven of the ten functions when the dimension is 10. However, the performance is close for the remaining three functions. The Wilcoxon signed-rank test was used to express more clearly the advantages and disadvantages of the algorithms. The “Test” column in Table 3 presents the results. “+” and “-” indicate that one algorithm is significantly more or less good than the other, respectively, and “≈” indicates that one algorithm is equal to the other. The test results show significant differences in the performance of these two algorithms; QMPA performs significantly better than MPA in seven functions and equally well in the remaining three functions.

The QMPA also has excellent performance in high dimensions. The results for dimensions 30 and 50 are presented in Tables 4 and 5, respectively. Table 4 demonstrates that the QMPA gives much better solutions compared with the MPA for seven functions and has similar performance to MPA for the other three functions. The performance of QMPA is worse than MPA in the TF3 function, better than MPA on five functions, and close to MPA in the remaining four functions in Table 5.

Figures 6, 7, 8 show the convergence curves of MPA and QMPA from low to high dimensions, respectively. The convergence curves in the figures are obtained as the average of 31 runs, which makes the results more credible. From the graphs, we can find that the QMPA has a good representation in most of the functions. Specifically, from Fig. 6, it can be found that MPA appears premature over TF1, TF2, TF3, TF4, and TF7. And in TF5, TF6, and TF8, the lack of exploitation capacity of MPA in the later stage is exposed. Similarly, in Figs. 7 and 8, MPA appears to converge

prematurely over TF1, TF2, TF4, and TF8. And the lack of exploitation capability is reflected in TF5, TF6, and TF9. In contrast, QMPA behaves differently, both in low and high dimensions. On the one hand, QMPA can effectively jump away from the local optimum, and in addition, it has a better exploitation ability in the later stage of optimization.

The unimodal functions provide a good assessment of the algorithm’s ability to exploitation because it has a single optimal solution within the globe. From the results, the QMPA has an overall dominance over the unimodal functions. In contrast, the multimodal functions are different and have lots of local optimums that grow exponentially with the number of dimensions. For assessing the exploratory ability of an algorithm, the multimodal function is very suitable. The results show that the QMPA outperforms MPA on some multimodal functions. That is, the QMPA is superior to the MPA in terms of both exploitation ability and exploration ability. This is due to the adaptive adjustment of the search agent update method, which allows the search agent in different states to take the most suitable update method. When faced with different problems, QMPA can effectively maintain the balance between exploitation and exploration in the optimization process and avoid the phenomenon of premature aging.

**Table 2** Mathematical formulation and properties of classic test functions

Function	Domain	Global opt
$TF_1(x) = \sum_{i=1}^d x_i^2$	$[-100,100]$	0
$TF_2(x) = \sum_{i=1}^d  x_i  + \prod_{i=1}^d  x_i $	$[-100,100]$	0
$TF_3(x) = \sum_{i=1}^d \left(\sum_{j=1}^d x_j\right)^2$	$[-100,100]$	0
$TF_4(x) = \text{Max}\{ x_i , 1 \leq i \leq d\}$	$[-100,100]$	0
$TF_5(x) = \sum_{i=1}^{d-1} [100(x_{i+1} - x^2)^2 + (x_i - 1)^2]$	$[-30,30]$	0
$TF_6(x) = \sum_{i=1}^d -x_i \sin\left(\sqrt{ x_i }\right)$	$[-500,500]$	-418.98*Dim
$TF_7(x) = \sum_{i=1}^d [x_i^2 - 10 \cos(2\pi x_i) + 10]$	$[-5.12,5.12]$	0
$TF_8(x) = -20 \exp\left(-0.2 \exp\sqrt{\frac{1}{d} \sum_{i=1}^d x_i^2}\right) - \exp\left(\frac{1}{d} \sum_{i=1}^d \cos(2\pi x_i) + 20 + e\right)$	$[-32,32]$	0
$TF_9(x) = \frac{\pi}{d} \left\{ 10 \sin(\pi y_1) + \sum_{i=1}^{d-1} (y_i - 1)^2 [1 - 10 \sin^2(\pi y_{i+1})] + (y_d - 1)^2 + \sum_{i=1}^d u(x_i, 10, 100, 4) \right\}$	$[-50,50]$	0
$TF_{10}(x) = 0.1 \left\{ \sin^2(3\pi x_1) + \sum_{i=1}^{d-1} (x_i - 1)^2 [1 + \sin^2(3\pi x_i + 1)] + (x_d - 1)^2 [1 + \sin^2(2\pi x_d)] + \sum_{i=1}^d u(x_i, 5, 100, 4) \right\}$	$[-50,50]$	0

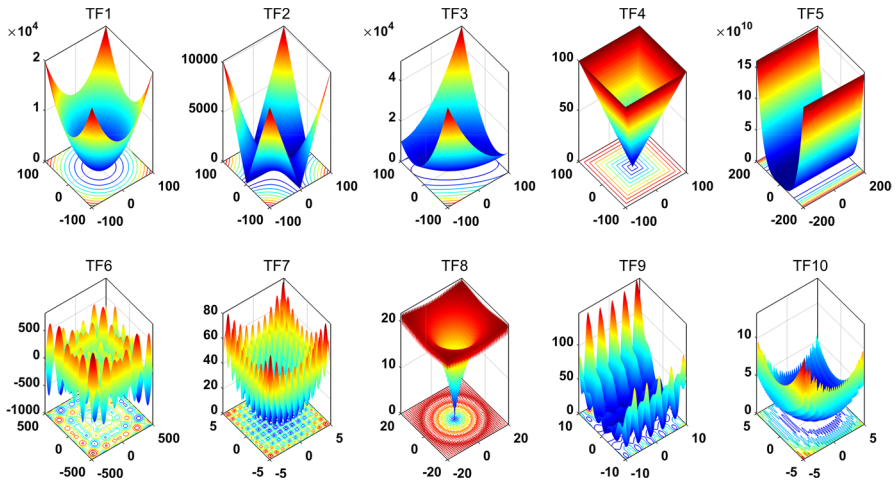


Fig. 5 Two-dimensional view of classic test functions

Table 3 Comparison of MPA and QMPA on classic test functions ( Dim = 10)

Function		Best	Median	Worst	Mean	Std	Test
TF1	MPA	1.83E-32	2.15E-30	1.04E-28	8.51E-30	1.93E-29	+ <sup>1</sup>
	QMPA	9.70E-63	2.61E-58	4.27E-53	<b>1.43E-54</b>	7.80E-54	
TF2	MPA	2.61E-18	3.12E-17	2.69E-16	7.49E-17	8.11E-17	+
	QMPA	5.96E-37	1.16E-35	6.11E-34	<b>5.31E-35</b>	1.21E-34	
TF3	MPA	2.18E-17	4.12E-15	3.06E-13	4.31E-14	7.54E-14	+
	QMPA	1.81E-27	2.29E-23	2.51E-19	<b>1.22E-20</b>	4.84E-20	
TF4	MPA	1.03E-13	1.05E-12	3.76E-12	1.47E-12	1.14E-12	+
	QMPA	2.06E-24	1.30E-22	5.71E-21	<b>7.28E-22</b>	1.47E-21	
TF5	MPA	0.9890	2.4513	3.5689	2.4196	0.6042	+
	QMPA	0.9407	1.7051	3.9201	<b>1.8288</b>	0.7004	
TF6	MPA	-3952.95	-3547.53	-3239.29	-3574.96	213.78	+
	QMPA	-4071.39	-3596.88	-3242.32	<b>-3637.22</b>	220.36	
TF7	MPA	0	0	1.27E-11	4.23E-13	2.31E-12	+
	QMPA	0	0	5.97E-13	<b>1.99E-14</b>	1.09E-13	
TF8	MPA	8.88E-16	4.44E-15	7.99E-15	4.80E-15	1.43E-15	≈
	QMPA	8.88E-16	4.44E-15	4.44E-15	<b>2.78E-15</b>	1.80E-15	
TF9	MPA	2.03E-12	9.66E-12	2.98E-11	<b>1.20E-11</b>	7.69E-12	≈
	QMPA	1.45E-12	1.09E-11	5.05E-11	1.29E-11	1.23E-11	
TF10	MPA	4.49E-12	5.37E-11	2.53E-10	<b>6.46E-11</b>	5.42E-11	≈
	QMPA	8.47E-12	4.76E-11	1.94E-10	6.47E-11	4.68E-11	

<sup>1</sup>QMPA-MPA Wilcoxon signed-rank test result at a significance level of  $\alpha < 0.05$ .



**Table 4** Comparison of MPA and QMPA on classic test functions (Dim=30)

Function		Best	Median	Worst	Mean	Std	Test
TF1	MPA	6.31E-25	2.13E-23	4.20E-22	7.41E-23	1.04E-22	+ <sup>1</sup>
	QMPA	1.25E-44	2.46E-42	2.29E-41	<b>4.98E-42</b>	6.39E-42	
TF2	MPA	4.53E-15	1.49E-13	8.62E-13	2.06E-13	2.13E-13	+
	QMPA	3.41E-27	3.20E-26	5.53E-25	<b>7.56E-26</b>	1.26E-25	
TF3	MPA	2.49E-08	5.15E-05	2.29E-03	<b>2.27E-04</b>	4.92E-04	≈
	QMPA	1.01E-07	3.59E-05	5.03E-03	4.05E-04	1.12E-03	
TF4	MPA	1.05E-09	3.22E-09	1.40E-08	3.45E-09	2.36E-09	+
	QMPA	4.35E-16	7.54E-15	6.66E-14	<b>1.63E-14</b>	1.97E-14	
TF5	MPA	24.3928	25.5313	26.8944	25.5850	0.5023	+
	QMPA	23.7037	24.8658	26.0461	<b>24.8691</b>	0.5832	
TF6	MPA	-9782.23	-8636.68	-7747.74	-8655.26	538.10	+
	QMPA	-9679.58	-8752.98	-7660.91	<b>-8690.10</b>	508.45	
TF7	MPA	0	0	0	0	0	≈
	QMPA	0	0	0	0	0	
TF8	MPA	2.99E-13	1.33E-12	2.68E-12	1.32E-12	5.95E-13	+
	QMPA	8.88E-16	4.44E-15	4.64E-15	<b>4.44E-15</b>	6.49E-16	
TF9	MPA	2.70E-09	1.39E-07	3.57E-03	2.14E-07	7.68E-04	+
	QMPA	4.64E-10	3.40E-07	3.69E-04	<b>5.57E-09</b>	7.42E-04	
TF10	MPA	1.09E-07	1.10E-02	1.21E-01	9.78E-02	3.47E-02	≈
	QMPA	8.18E-08	2.97E-02	4.39E-01	<b>1.21E-02</b>	1.09E-01	

<sup>1</sup>QMPA-MPA Wilcoxon signed-rank test result at a significance level of  $\alpha < 0.05$

## 5.2 Experiment 2

In the second experiment, the more challenging CEC2014 test suite [40] will be selected to assess the performance of the QMPA. The CEC2014 test suite contains a total of 30 test functions in four categories, including simple functions, such as unimodal and simple multimodal functions, and also more complex functions, such as hybrid and composition functions.

In this experiment, we will not only prove that the QMPA is superior to the MPA but also better than some commonly used optimization algorithms, such as GWO, PSO, Slime Mould Algorithm(SMA) [41], Sine Cosine Algorithm (SCA) [42], Repetile Search Algorithm(RSA) [43], and Aquila Optimizer(AO) [44]. The parameters of QMPA and MPA in this experiment are the same as those in the first experiment. The detailed parameter settings of GWO, PSO, SMA, SCA, RSA, and AO are shown in Table 6.

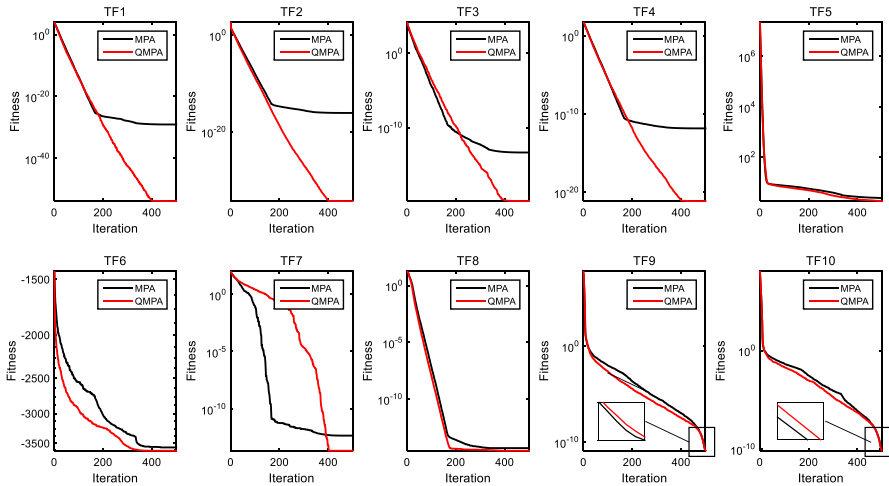
To make the comparative analysis meaningful, all algorithms are compared in the same environment. The number of population was 25, the dimension was 10, and the iteration time was set to 500. To avoid randomness, the mean of the results of 31 independent runs is taken as the comparison result.

Table 7 presents the results of the comparison of QMPA with the competing algorithms, which includes the standard deviation (Std) and mean of the results of

**Table 5** Comparison of MPA and QMPA on classic test functions (Dim=50)

Function		Best	Median	Worst	Mean	Std	Test
TF1	MPA	2.14E-22	2.59E-21	1.49E-20	3.65E-21	3.84E-21	+ <sup>1</sup>
	QMPA	3.93E-41	1.41E-39	1.81E-37	<b>1.02E-38</b>	3.29E-38	
TF2	MPA	2.33E-14	1.05E-12	6.97E-12	1.74E-12	1.63E-12	+
	QMPA	3.01E-25	2.56E-24	3.45E-23	<b>4.50E-24</b>	6.54E-24	
TF3	MPA	1.37E-04	1.67E-02	3.12E-01	<b>4.51E-02</b>	6.30E-02	-
	QMPA	6.88E-03	3.89E-01	4.98E+00	7.46E-01	1.14E+00	
TF4	MPA	8.86E-09	2.37E-08	7.04E-08	2.69E-08	1.44E-08	+
	QMPA	8.31E-14	3.04E-13	2.41E-11	<b>1.30E-12</b>	4.35E-12	
TF5	MPA	45.0027	46.5590	48.5399	46.6735	0.8834	≈
	QMPA	44.2538	45.8795	48.5540	<b>46.0664</b>	1.3086	
TF6	MPA	-14,883.39	-13,569.68	-12,263.02	-13,550.38	689.12	+
	QMPA	-15,632.46	-13,705.70	-12,314.58	<b>-13,739.91</b>	797.36	
TF7	MPA	0	0	0	0	0	≈
	QMPA	0	0	0	0	0	
TF8	MPA	6.58E-13	5.88E-12	1.62E-11	6.95E-12	4.28E-12	+
	QMPA	4.44E-15	4.44E-15	4.44E-15	<b>4.44E-15</b>	0	
TF9	MPA	1.15E-03	8.59E-03	1.81E-02	8.94E-03	4.87E-03	≈
	QMPA	1.90E-03	8.28E-03	1.81E-02	<b>8.63E-03</b>	4.62E-03	
TF10	MPA	0.0501	0.5586	1.4554	<b>0.5812</b>	0.2471	≈
	QMPA	0.2158	0.6644	3.1691	0.8240	0.6017	

<sup>1</sup>QMPA-MPA Wilcoxon signed-rank test result at a significance level of  $\alpha < 0.05$ .



**Fig. 6** Convergence Graphics (Dim = 10)

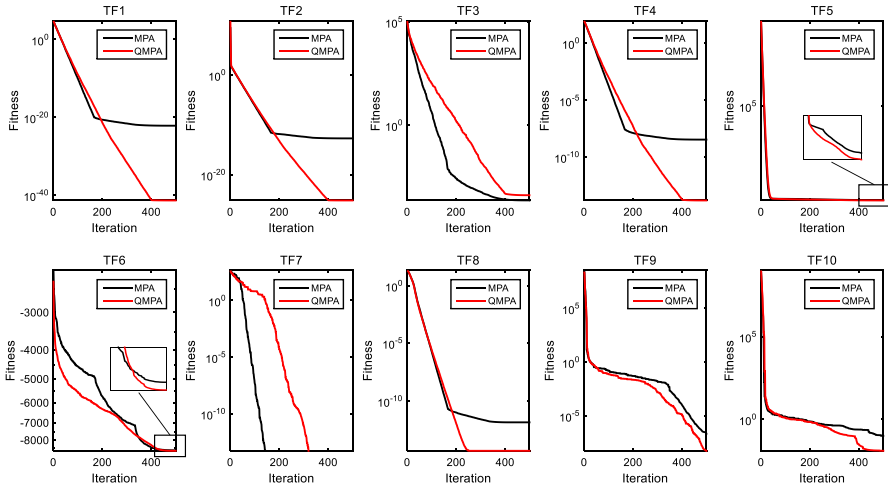


Fig. 7 Convergence Graphics (Dim = 30)

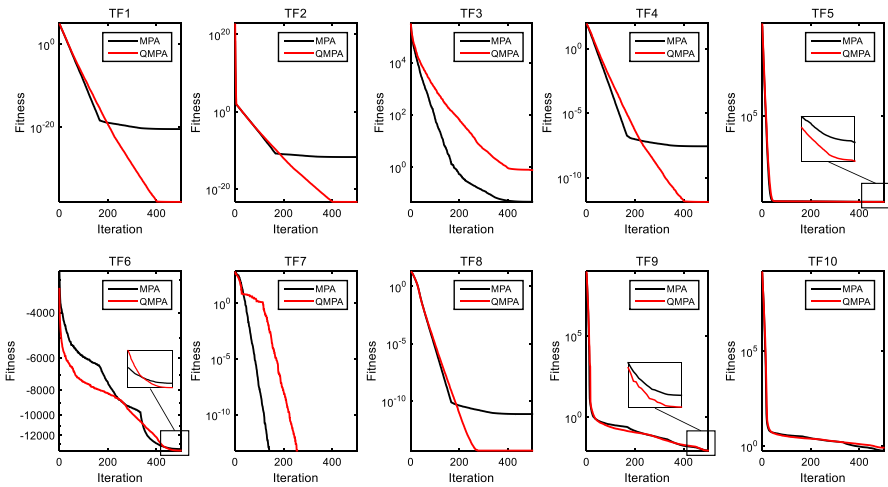


Fig. 8 Convergence Graphics (Dim = 50)

31 runs. Furthermore, to clearly demonstrate the excellent performance of QMPA, pairwise statistical tests are performed for QMPA with MPA, GWO, PSO, SMA, SCA, RSA, and AO. The results of 31 runs of each algorithm were subjected to the Wilcoxon signed-rank test with statistical significance  $\alpha = 0.05$ , and are shown in Table 8.

Table 7 shows the test results, and the last row counts the performance of each algorithm in achieving the optimal value on the CEC2014 test suite. For unimodal functions (F1-F3), have only one optimal solution, making them the best candidates for assessing algorithm exploitation capabilities. The mean of the QMPA received

1st place over the F1 and F2 functions, and for F3, QMPA tied with MPA for 1st place. The QMPA also achieved two 1st places and one 2nd place in the comparison of standard deviations. It reflects that QMPA has good exploitation ability and robustness.

The second class is simple multimodal functions (F4-F16) with two or more local optimal solutions. Therefore, it is very appropriate to use such functions to evaluate the ability of QMPA and other algorithms to move away from local optima and exploration. From the results, QMPA is an excellent algorithm and achieves the 1st place in the mean for 12 of the 13 tested functions.

For the last two categories, hybrid functions (F17-F22) have a lot of local optima, as do the composition functions (F23-F30), and test functions with such characteristics are useful for assessing the ability of an algorithm to escape from local minima. Among the hybrid functions, QMPA obtains the best average values of F17, F18, F19, F20, and F21, and only performs worse than MPA in F22. In the composite function, the QMPA has an ordinary performance, with the best average value obtained only on F26 and F30.

Based on these results, the QMPA proposed in this paper obtained the best mean value on 22 functions and the best standard deviation value on 16 functions in the CEC2014 test suite. In contrast, the original algorithm MPA obtained only 7 best values on the mean performance. In other words, QMPA obtained the best overall performance and outperformed the other algorithms, even standard MPA, in balancing exploration and exploitation capabilities and getting rid of local optimizations.

Furthermore, this chapter presents the convergence analysis of QMPA with MPA, GWO, PSO, SMA, SCA, RSA, and AO for some of the tested functions based on the average of 31 independent runs, and the results are demonstrated in Fig. 9. QMPA achieves stable behavior for all functions, indicating that the algorithms are convergent. QMPA exhibits two advantages over other algorithms. One of them is that it does not converge prematurely. From Fig. 9, it can be found that RSA has obvious premature phenomenon, which appears in most test functions, while QMPA has a completely different performance. The second is the ability to ensure a balance between exploration and exploitation throughout the optimization process. For the whole optimization process, a strong exploration capability is required in the early stage, while the later stage of optimization requires a higher algorithm exploitation capability. Obviously, the fitness of QMPA decreases faster in the early stage, and

**Table 6** Parameter settings of GWO, PSO, SMA, SCA, RSA, and AO

Algorithm	Parameter settings
GWO	$a = [0, 2][0, 2]$
PSO	$c_1 = 2; c_2 = 2; v_{\max} = 6$
SMA	$z = 0.03$
SCA	$A = 2$
RSA	$\alpha = 0.1, \beta = 0.1$
AO	$\alpha = 0.1, \delta = 0.1$

Table 7 Results of the CEC2014 test suite

Functions		MPA	QMPA	GWO	PSO	SMA	SCA	RSA	AO
Unimodal functions	F1	Mean	<b>100.15</b>	1.04E+07	6.64E+04	4.10E+05	1.25E+07	1.53E+08	1.11E+07
		Std	<b>0.1882</b>	9.30E+06	7.97E+04	7.12E+05	6.83E+06	1.07E+08	7.48E+06
F2	Mean	211.68	<b>209.60</b>	5.92E+07	2349.9767	6368.2878	1.18E+09	7.35E+09	1.11E+06
	Std	56.8470	<b>49.9980</b>	1.62E+08	3469.0788	3798.7185	4.30E+08	1.59E+09	8.09E+05
F3	Mean	<b>300.00</b>	<b>300.00</b>	1.12E+04	4273.1232	5575.1604	1.21E+04	9970.25	6538.09
	Std	0.0004	<b>0.0002</b>	8505.80	4551.85	5308.02	6701.66	2722.71	2264.76
Simple multimodal functions	F4	Mean	420.66	434.44	<b>404.42</b>	425.05	491.21	1680.19	437.43
	Std	16.0089	14.5492	17.7819	<b>1.8263</b>	15.8000	34.8906	580.53	12.64
F5	Mean	520.01	<b>519.44</b>	520.54	520.36	520.21	520.51	520.46	520.39
	Std	<b>0.0236</b>	3.2336	0.0749	0.1676	0.1461	0.0825	0.0829	0.1151
F6	Mean	602.59	<b>602.32</b>	602.99	603.98	604.17	607.70	610.17	605.48
	Std	1.0573	1.5420	1.4942	1.8321	1.9652	1.3479	<b>0.7701</b>	1.1475
F7	Mean	700.21	<b>700.08</b>	701.81	700.15	700.45	713.75	805.11	700.96
	Std	0.1303	<b>0.0438</b>	1.1163	0.0701	0.1254	4.1654	25.8506	0.1587
F8	Mean	809.63	<b>803.63</b>	813.63	829.85	806.23	847.88	878.81	827.35
	Std	3.2938	<b>1.5950</b>	6.5192	10.7467	3.1561	7.6938	7.7912	9.6713
F9	Mean	916.43	<b>915.08</b>	917.87	927.09	923.66	951.77	962.75	928.46
	Std	<b>6.0841</b>	6.7437	8.8698	10.0443	8.0554	6.8375	7.3764	9.0762
F10	Mean	1218.91	<b>1189.62</b>	1388.71	1684.64	1213.96	2150.15	2139.26	1501.98
	Std	114.26	<b>101.02</b>	242.95	249.38	123.79	194.32	211.17	191.59
F11	Mean	1628.14	<b>1486.28</b>	1634.86	2060.20	1876.66	2637.39	2567.92	2011.22
	Std	235.62	251.67	284.46	355.80	400.97	<b>168.61</b>	225.64	285.85
F12	Mean	1200.13	<b>1200.04</b>	1201.24	1200.17	1200.24	1201.65	1201.28	1200.64
	Std	0.0978	<b>0.0423</b>	0.6789	0.1411	0.1011	0.2912	0.2624	0.3563

Table 7 (continued)

Functions	MPA	QMPA	GWO	PSO	SMA	SCA	RSA	AO
F13	Mean	<b>1300.14</b>	1300.25	1300.37	1300.38	1300.79	1303.92	1300.51
	Std	0.0588	<b>0.0530</b>	0.0769	0.1483	0.1425	0.1400	0.9969
F14	Mean	1400.25	<b>1400.24</b>	1400.41	1400.35	1401.50	1411.63	1400.44
	Std	<b>0.0630</b>	0.1229	0.2016	0.1149	0.1905	0.8031	4.2317
F15	Mean	1501.35	<b>1500.77</b>	1502.53	1501.24	1510.88	5586.49	1504.98
	Std	0.4070	<b>0.3031</b>	1.0637	0.4079	0.5967	6.7633	2.2659
F16	Mean	1602.86	<b>1602.52</b>	1602.81	1603.10	1603.65	1603.85	1603.11
	Std	0.3276	0.3866	0.4661	0.3892	0.3383	0.2100	<b>0.1691</b>
F17	Mean	1767.85	<b>1756.48</b>	1.16E+05	3903.62	2.55E+04	5.67E+04	5.01E+05
	Std	50.2314	<b>45.5651</b>	1.95E+05	1998.74	3.28E+04	5.76E+04	8.41E+04
F18	Mean	1925.40	<b>1802.56</b>	1.16E+04	8036.18	1.27E+04	6.25E+04	1.81E+05
	Std	38.1489	<b>1.1385</b>	7244.36	5431.02	1.21E+04	8.10E+04	4.13E+05
F19	Average	1902.19	<b>1901.44</b>	1903.11	1903.11	1902.34	1906.40	1919.31
	Std	0.4927	<b>0.4852</b>	0.9809	1.0315	0.5683	1.1832	11.4046
F20	Average	2091.90	<b>2001.77</b>	8968.02	6682.75	6696.73	9369.56	2.69E+04
	Std	47.9167	<b>0.9187</b>	7403.60	6386.41	8283.37	4565.82	6.97E+04
F21	Average	2559.96	<b>2104.70</b>	10.627.88	6886.48	7467.35	1.89E+04	1.83E+06
	Std	205.05	<b>6.7356</b>	5440.07	4417.96	7979.64	1.27E+04	2.91E+06
F22	Average	<b>2227.35</b>	2229.75	2300.96	2338.67	2241.95	2280.86	2395.36
	Std	<b>6.1731</b>	45.3847	60.5594	52.6908	53.6816	26.8073	54.0638

Table 7 (continued)

Functions	MPA	QMPA	GWO	PSO	SMA	SCA	RSA	AO
Composition functions	Average	2625.29	2629.46	2635.88	2629.46	2651.25	2500	2500
	Std	23.2522	<b>0.0000</b>	5.1001	<b>0.0000</b>	8.7158	<b>0</b>	<b>0</b>
F24	Average	<b>2519.38</b>	2534.21	2536.17	2572.66	2559.68	2599.7	2586.67
	Std	5.1374	31.2718	28.0634	27.9726	8.0549	<b>1.3498</b>	28.1825
F25	Average	<b>2655.60</b>	2676.98	2690.60	2695.94	2698.48	2699.99	2696.61
	Std	24.6032	28.7613	22.5630	12.6361	11.6534	<b>0.0042</b>	10.3061
F26	Average	2700.13	<b>2700.11</b>	2703.42	2703.46	2700.81	2703.88	2700.52
	Std	<b>0.0441</b>	0.0550	17.9313	17.9232	0.0762	1.0071	0.1691
F27	Average	<b>2756.36</b>	3007.25	3022.28	3099.76	2900.00	2900	2864.08
	Std	134.63	140.69	128.78	119.71	<b>0.0000</b>	<b>0</b>	74.6118
F28	Average	3174.07	3250.93	3293.91	3748.37	3297.28	<b>3000</b>	<b>3000</b>
	Std	5.8926	74.3537	114.51	293.75	<b>0.0000</b>	<b>0</b>	4.15E-13
F29	Average	3381.41	3.68E+05	3.82E+05	3.33E+03	2.26E+05	<b>3100</b>	5103.56
	Std	140.48	7.58E+05	8.99E+05	356.7899	5.87E+05	2.13E+04	<b>0</b>
F30	Average	3665.13	3524.76	4599.07	5212.55	3734.30	<b>3200</b>	4508.42
	Std	137.97	75.3946	644.74	450.69	267.84	691.55	<b>0</b>
The number of optimal	Average	5	21	0	1	2	4	2
	Std	6	14	0	2	3	9	1

**Table 8** Wilcoxon Signed-Rank Test Results for QMPA and Other Algorithms on the CEC2014 Test Suite

	QMPA/MFA		QMPA/GWO		QMPA/PSO		QMPA/SMA		QMPA/SCA		QMPA/RSA		QMPA/AO		
	T+	T-	win	T+	T-	win	T+	T-	win	T+	T-	win	T+	T-	win
F1	0	496	+	0	496	+	0	496	+	0	496	+	0	496	+
F2	8	488	+	0	496	+	1	495	+	0	496	+	0	496	+
F3	0	496	+	0	496	+	0	496	+	0	496	+	0	496	+
F4	236	260	=	89	407	+	460	36	-	186	310	=	0	496	+
F5	223	273	=	0	496	+	0	496	+	3	493	+	0	496	+
F6	217	279	=	161	335	+	49	447	+	62	434	+	1	495	+
F7	28	468	+	0	496	+	41	455	+	1	495	+	0	496	+
F8	3	493	+	8	488	+	0	496	+	40	456	+	0	496	+
F9	195	301	=	208	288	=	25	471	+	58	438	+	0	496	+
F10	206	290	=	67	429	+	1	495	+	199	297	=	0	496	+
F11	142	354	+	177	319	=	15	481	+	42	454	+	0	496	+
F12	31	465	+	0	496	+	40	456	+	1	495	+	0	496	+
F13	57	439	+	4	492	+	3	493	+	1	495	+	0	496	+
F14	192	304	=	82	414	+	78	418	+	111	385	+	0	496	+
F15	31	465	+	0	496	+	34	462	+	33	463	+	0	496	+
F16	91	405	+	120	376	+	34	462	+	49	447	+	0	496	+
F17	0	496	+	0	496	+	0	496	+	0	496	+	0	496	+
F18	0	496	+	0	496	+	0	496	+	0	496	+	0	496	+
F19	34	462	+	0	496	+	16	480	+	37	459	+	0	496	+
F20	0	496	+	0	496	+	0	496	+	0	496	+	0	496	+
F21	0	496	+	0	496	+	0	496	+	0	496	+	0	496	+
F22	118	378	+	19	477	+	10	486	+	112	384	+	57	439	+
F23	31	465	+	0	496	+	496	0	-	496	0	-	496	0	-
F24	340	156	+	181	315	=	40	456	+	113	383	+	121	375	+
													15	481	+
													31	465	+



Table 8 (continued)

	QMPA/MFA		QMPA/GWO		QMPA/PSO		QMPA/SMA		QMPA/SCA		QMPA/RSA		QMPA/AO		
	T+	T- win	T+	T- win	T+	T- win	T+	T- win	T+	T- win	T+	T- win	T+	T- win	
F25	400	96 -	123	373	86	410	100	396	71	425	53	443	82	414	+
F26	191	305 =	30	466	40	456	0	496	0	496	0	496	0	496	+
F27	470	26 -	226	270	79	417	416	80	120	376	416	80	436	60	-
F28	451	45 -	146	350	2	494	496	0	135	361	496	0	496	0	+
F29	171	325 =	108.5	387.5	193	303	163	333	171	325	423	73	172	324	=
F30	62	434	0	496	0	496	22	474	0	496	496	0	0	496	+
Total	19/8/3		26/4/0		27/1/2		25/2/3		29/1/0		25/0/5		27/1/2		

the convergence value is better in the later stage on F3, F5, F6, F8, F10, F12, F13, and F16. In other words, QMPA is able to control the balance between exploration and exploitation at different stages of optimization, and also this is the guarantee of having good optimization performance.

For further analysis of the test results, some boxplots are given in Fig. 10. The boxplot is drawn using the results of each independent run of each algorithm and shows the distribution of the results of 31 independent runs well. Compared with MPA and other competing algorithms, QMPA is narrower and more to the left in the distribution of boxplot for most functions. This also reflects that the QMPA possesses stronger robustness and better optimization performance. It is further proved that the adaptive control strategy of search agent position updating method proposed in this paper helps QMPA to have stable and excellent performance when facing different complex optimization problems.

In addition, for further comparative analysis, paired statistical tests were performed for QMPA and MPA, GWO, PSO, SMA, SCA, RSA, and AO, respectively. The results of 31 runs of each algorithm were subjected to the Wilcoxon sign-rank test for statistical significance  $\alpha=0.05$ . The null hypothesis of this test is that "No differences are found between QMPA and the other algorithms in terms of the median number of optimal solutions obtained with the same test function."

The statistical test results are illustrated in Table 8. Where "+" indicates that QMPA outperforms the comparison algorithm at the 95% significance level, while "-" is the opposite, indicating that QMPA is worse than the comparison algorithm. "=" means the results of QMPA are not significantly different from those of the comparison algorithm. The last row summarizes the performance of QMPA and other algorithms in the "+", "=", and "-" cases. It is obvious that QMPA performs much well than other optimization algorithms. Compared with the standard MPA, the QMPA performs better on 19 functions, and they are comparable on 8 functions.

### 5.3 Experiment 3

In this section, two commonly used real-world engineering problems are chosen to perform performance tests of the QMPA. There are some equations or inequality constraints in their optimization models, and in this experiment, the constrained problem will be transformed into an unconstrained one using a simple penalty function method. The constraints and optimization model can be found in [45]. The maximum number of iterations was set as 500, with a total of 25 populations and 31 independent runs.

#### 5.3.1 Tension/compression spring design

Finding the minimum weight is the optimization objective of this engineering problem. The three-dimensional physical drawing is shown in Fig. 11, and the optimization variables involved are mean coil diameter ( $D$ ), the number of active coils ( $N$ ), and wire diameter ( $d$ ).

This problem has been studied in many literatures using some optimization algorithms, such as CA [46], GSA [8], BFOA [47], and PFA [48]. Table 9 presents the

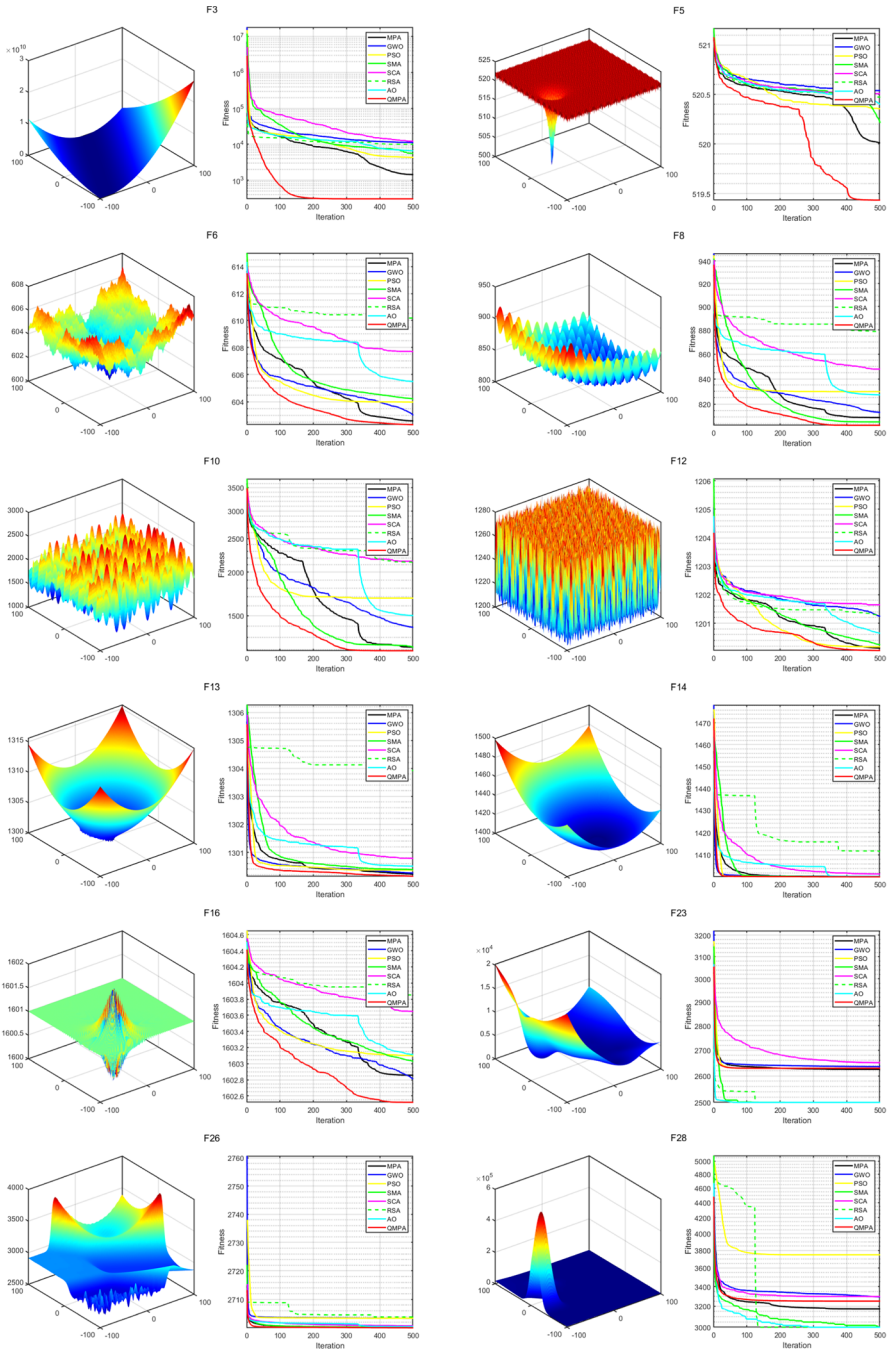


Fig. 9 Convergence curves of QMPA and other algorithms over CEC2014 functions

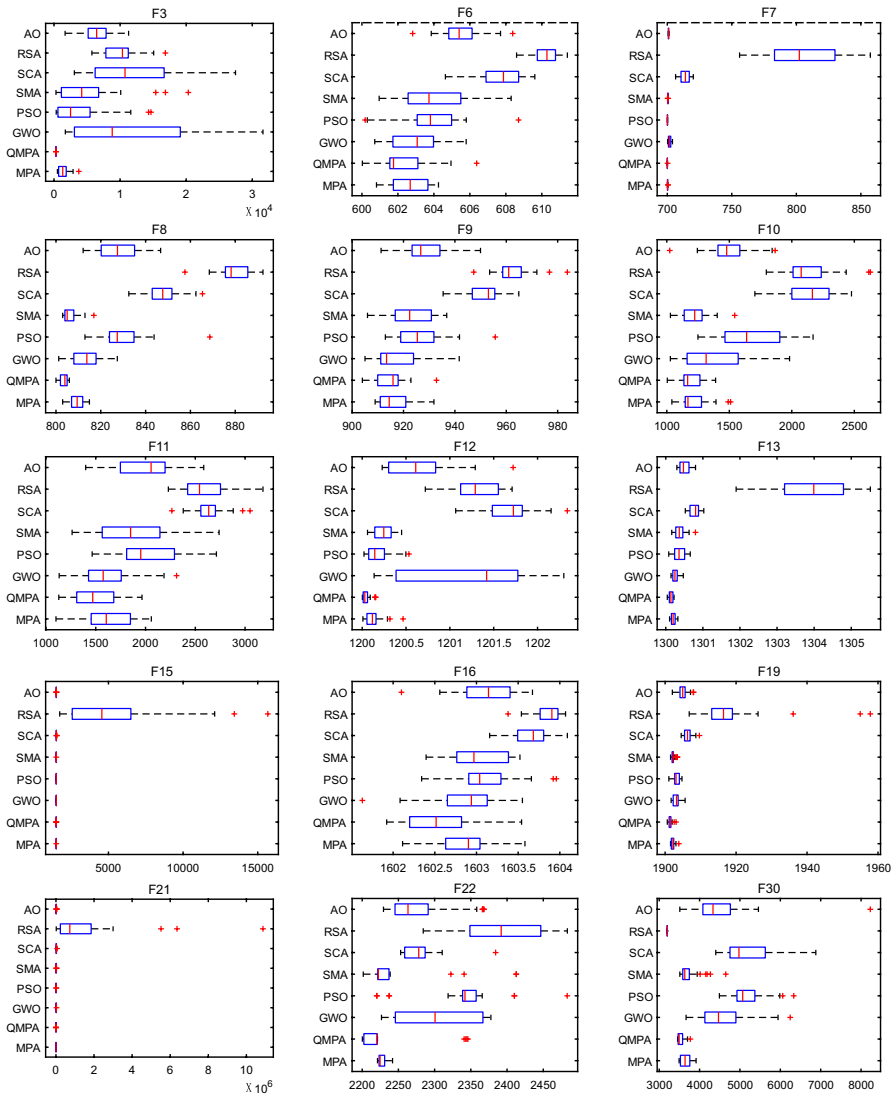


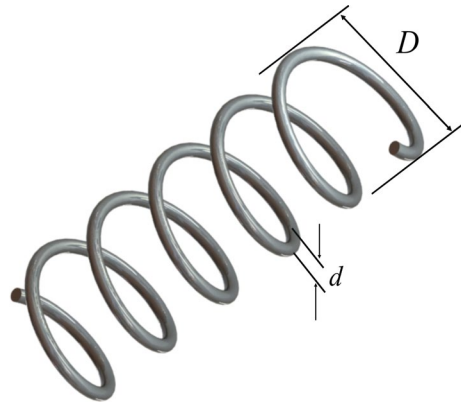
Fig. 10 Boxplots of QMPA and competitor algorithms

optimization results using different algorithms. The optimal results of the PFA, MPA, and QMPA are very close, but the QMPA results are slightly better.

### 5.3.2 Welded beam design

The target of the optimization of this optimization problem is to obtain the lowest manufacturing cost of the welded beam. As it is shown in Fig. 12, this optimization

**Fig. 11** Description of the tension/compression spring design example



**Table 9** The best results were obtained

Algorithm	$d$	$D$	$N$	$f$
CA [46]	0.050000	0.317395	14.031795	0.012721
GSA [8]	0.050276	0.323680	13.525410	0.012702
BFOA [47]	0.051825	0.359935	11.107103	0.012671
PFA [48]	0.051726	0.357629	11.235724	0.012665
MPA	0.051570	0.353881	11.457236	0.0126655
QMPA	0.051769	0.358659	11.176065	0.0126653

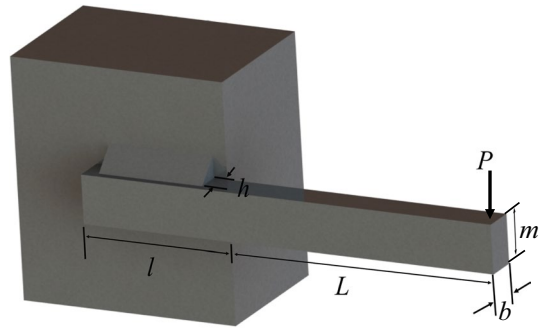
model has four optimization variables, that are the thickness of weld ( $h$ ), length of the attached part of the bar ( $l$ ), the height of the bar ( $m$ ), and thickness of the bar ( $b$ ).

The optimization results of this issue were compared and analyzed by selecting the results of other literature, including SA [49], HS [50], HAS-GA [51], and GSA [8], as shown in Table 10. It is clear that the QMPA gives the most optimal results, which has clearly validated the excellent performance of the QMPA again.

## 6 Threats to validity

Threats to validity are generally classified as threats to internal validity and threats to external validity. In this paper, the threats to internal validity mainly come from the implementation of the QMPA and the comparison algorithm used in the comparison tests. For the implementation of the QMPA, programming based on the source code of MPA, along with testing and analysis in a variety of simple problems to ensure the correctness of the algorithm code; For the implementation of the comparison algorithms, the open source code provided by the original author of the corresponding algorithm is used, and the best parameters proposed in the original article are followed for experiments. Throughout the process, all the codes are carefully checked to ensure error-free. The threats to external validity mainly come from the test functions and engineering optimization problems used in this paper. To reduce

**Fig. 12** Description of the welded beam design example



**Table 10** The best results were obtained

Algorithm	$h$	$l$	$m$	$b$	$f$
SA [49]	0.2471	6.1451	8.2399	0.2497	2.4426
HS [50]	0.2442	6.2231	8.2915	0.2400	2.3807
HSA-GA [51]	0.2231	1.5815	12.8468	0.2245	2.2500
GSA [8]	0.182129	3.856979	10.000	0.202376	1.87995
MPA	0.244384	2.862740	8.291469	0.244369	1.832607
QMPA	0.244356	2.862617	8.291472	0.244368	1.832581

the threats to external validity, the test functions and engineering optimization problems selected in this paper are widely used in many papers and competitions, and their correctness and reasonableness have a certain degree of confidence. Also, the corresponding scripts are carefully checked for correctness.

## 7 Conclusion

A novel modified marine predators algorithm hybridizing by Q-learning (QMPA) is proposed to improve the ability of MPA to find the global optimal solution by adaptively adjusting the search agent position update strategy in different iteration stages and states. The adaptive update strategy selection of the MPA is finally achieved with the help of Q-learning by processes of determining states and actions, designing Q tables, and choosing reward functions. The statistical results of different experiments demonstrate that the QMPA has a better optimization capability compared with the standard MPA and other optimization algorithms.

In this paper, only two engineering problems are studied to test the proposed algorithm, which can be employed in different fields to solve complex engineering real-world problems in future work. Also, it is anticipated the proposed adaptive control strategy can be applied to other metaheuristic optimization problems.

**Acknowledgements** The work is supported from National Key R&D Program of China (Grant No. 2020YFA0710904-03), Hunan Distinguished Youths Funds of China (Grant No. 2021JJ10016), National Natural Science Foundation of China (Grant No. U20A20285), National Natural Science Foundation of China (Grant No. 52172357), National Key R&D Program of China (Grant No. 2019YFB1706504), Hunan Innovative Province Construction Project (Grant No. 2020GK4010), Hunan Natural Science Foundation of China (General Program: 2020JJ4196), and Guangxi Specially Appointed Experts Funding.

**Data availability** The data supporting the findings of this study can be obtained from the corresponding author upon reasonable request.

## Declarations

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

## References

1. Dokeroglu T, Sevinc E, Kucukyilmaz T, Cosar A (2019) A survey on new generation metaheuristic algorithms. *Comput Ind Eng*. <https://doi.org/10.1016/j.cie.2019.106040>
2. Agrawal P, Abutarboush HF, Ganesh T, Mohamed AW (2021) Metaheuristic algorithms on feature selection: a survey of one decade of research (2009–2019). *IEEE Access* 9:26766–26791. <https://doi.org/10.1109/ACCESS.2021.3056407>
3. Faramarzi A, Heidarinejad M, Mirjalili S, Gandomi AH (2020) Marine predators algorithm: a nature-inspired metaheuristic. *Expert Syst Appl*. <https://doi.org/10.1016/j.eswa.2020.113377>
4. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. *IEEE Trans Evol Comput* 1(1):67–82. <https://doi.org/10.1109/4235.585893>
5. Li Y, Zhao L, Zhou S. Review of genetic algorithm. *Materials science and engineering*, PTS 1-22011. p 365–7
6. Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *J Global Optim* 11(4):341–359. <https://doi.org/10.1023/A:1008202821328>
7. Hwang C-R (1988) Simulated annealing: theory and applications. *Acta Appl Math* 12(1):108–111. <https://doi.org/10.1007/BF00047572>
8. Rashedi E, Nezamabadi-Pour H, Saryazdi S (2009) GSA: a gravitational search algorithm. *Inf Sci* 179(13):2232–2248. <https://doi.org/10.1016/j.ins.2009.03.004>
9. Poli R, Kennedy J, Blackwell T (2007) Particle swarm optimization. *Swarm Intell* 1(1):33–57. <https://doi.org/10.1007/s11721-007-0002-0>
10. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. *Adv Eng Softw* 69:46–61. <https://doi.org/10.1016/j.advengsoft.2013.12.007>
11. Al-qaness MAA, Ewees AA, Fan H, Abualigah L, Abd Elaziz M (2020) Marine predators algorithm for forecasting confirmed cases of COVID-19 in Italy, USA, Iran and Korea. *Int J Environ Res Public Health*. <https://doi.org/10.3390/ijerph17103520>
12. Hu G, Zhu X, Wei G, Chang C-T (2021) An improved marine predators algorithm for shape optimization of developable Ball surfaces. *Eng Appl Artif Intell*. <https://doi.org/10.1016/j.engappai.2021.104417>
13. Soliman MA, Hasanien HM, Alkhalayli A (2020) Marine predators algorithm for parameters identification of triple-diode photovoltaic models. *IEEE Access* 8:155832–155842. <https://doi.org/10.1109/access.2020.3019244>
14. Xing Z, He Y (2021) Many-objective multilevel thresholding image segmentation for infrared images of power equipment with boost marine predators algorithm. *Appl Soft Comput*. <https://doi.org/10.1016/j.asoc.2021.107905>
15. Li X-L, Serra R, Olivier J (2022) A multi-component PSO algorithm with leader learning mechanism for structural damage detection. *Appl Soft Comput*. <https://doi.org/10.1016/j.asoc.2021.108315>

16. Khan NH, Jamal R, Ebeed M, Kamel S, Zeinoddini-Meymand H, Zawbaa HM (2022) Adopting scenario-based approach to solve optimal reactive power dispatch problem with integration of wind and solar energy using improved marine predator algorithm. *Ain Shams Eng J*. <https://doi.org/10.1016/j.asej.2022.101726>
17. Houssein EH, Emam MM, Ali AA (2022) An optimized deep learning architecture for breast cancer diagnosis based on improved marine predators algorithm. *Neural Comput Appl*. <https://doi.org/10.1007/s00521-022-07445-5>
18. Sowmya R, Sankaranarayanan V (2022) Optimal vehicle-to-grid and grid-to-vehicle scheduling strategy with uncertainty management using improved marine predator algorithm. *Comput Electr Eng*. <https://doi.org/10.1016/j.compeleceng.2022.107949>
19. Sun C-J, Gao F (2021) A tent marine predators algorithm with estimation distribution algorithm and gaussian random walk for continuous optimization problems. *Comput Intell Neurosci*. <https://doi.org/10.1155/2021/7695596>
20. Hassan MH, Yousri D, Kamel S, Rahmann C (2022) A modified Marine predators algorithm for solving single- and multi-objective combined economic emission dispatch problems. *Comput Ind Eng*. <https://doi.org/10.1016/j.cie.2021.107906>
21. Yousri D, Abd Elaziz M, Oliva D, Abraham A, Alotaibi MA, Hossain MA (2022) Fractional-order comprehensive learning marine predators algorithm for global optimization and feature selection. *Knowl Based Syst*. <https://doi.org/10.1016/j.knosys.2021.107603>
22. Harifi S, Mohammadzadeh J, Khaliliani M, Ebrahimnejad S (2021) Hybrid-EPC: an emperor penguins colony algorithm with crossover and mutation operators and its application in community detection. *Progr Artif Intell* 10(2):181–193. <https://doi.org/10.1007/s13748-021-00231-9>
23. Kangah JK, Appati JK, Darkwah KF, Soli MAT (2021) Implementation of an H-PSOGA optimization model for vehicle routing problem. *Int J Appl Metaheuristic Comput* 12(3):148–162. <https://doi.org/10.4018/IJAMC.2021070106>
24. Sheng W, Shao Q, Tong H, Peng J (2021) Scheduling optimization on takeout delivery based on hybrid meta-heuristic algorithm. In: 2021 13TH International Conference on Advanced Computational Intelligence (ICACI)2021. p 372–7
25. Cimen ME, Yalcin Y (2022) A novel hybrid firefly-whale optimization algorithm and its application to optimization of MPC parameters. *Soft Comput* 26(4):1845–1872. <https://doi.org/10.1007/s00500-021-06441-6>
26. Almadhor A, Rauf HT, Khan MA, Kadry S, Nam Y (2021) A hybrid algorithm (BAPSO) for capacity configuration optimization in a distributed solar PV based microgrid. *Energy Rep* 7:7906–7912. <https://doi.org/10.1016/j.egy.2021.01.034>
27. Panagant N, Yildiz M, Pholdee N, Yildiz AR, Bureerat S, Sait SM (2021) A novel hybrid marine predators-Nelder-Mead optimization algorithm for the optimal design of engineering problems. *Mater Test* 63(5):453–457. <https://doi.org/10.1515/mt-2020-0077>
28. Li S-F, Cheng C-Y (2017) Particle swarm optimization with fitness adjustment parameters. *Comput Ind Eng* 113:831–841. <https://doi.org/10.1016/j.cie.2017.06.006>
29. Shen X, Zou D, Zhang X. A self-adaptive differential evolution with dynamic selecting mutation strategy. 2017 International Conference on Vision, Image and Signal Processing (ICVISIP) 2017. p 5–10
30. Aleti A, Moser I (2016) A systematic literature review of adaptive parameter control methods for evolutionary algorithms. *ACM Comput Surv*. <https://doi.org/10.1145/2996355>
31. Iwasaki N, Yasuda K, Ueno G (2006) Dynamic parameter tuning of particle swarm optimization. *IEEJ Trans Electr Electr Eng* 1(4):353–363. <https://doi.org/10.1002/tee.20078>
32. Sang H-Y, Pan Q-K, Duan P-Y (2019) Self-adaptive fruit fly optimizer for global optimization. *Nat Comput* 18(4):785–813. <https://doi.org/10.1007/s11047-016-9604-z>
33. Fan Q, Huang H, Chen Q, Yao L, Yang K, Huang D (2022) A modified self-adaptive marine predators algorithm: framework and engineering applications. *Eng Comput* 38(4):3269–3294. <https://doi.org/10.1007/s00366-021-01319-5>
34. Ramezani M, Bahmanyar D, Razmjooy N (2021) A new improved model of marine predator algorithm for optimization problems. *Arabian J Sci Eng* 46(9):8803–8826. <https://doi.org/10.1007/s13369-021-05688-3>
35. Owoola EO, Xia K, Ogunjo S, Mukase S, Mohamed A (2022) Advanced marine predator algorithm for circular antenna array pattern synthesis. *Sensors*. <https://doi.org/10.3390/s22155779>




36. Qi C, Zhu Y, Song C, Cao J, Xiao F, Zhang X et al (2021) Self-supervised reinforcement learning-based energy management for a hybrid electric vehicle. *J Power Sour*. <https://doi.org/10.1016/j.jpowsour.2021.230584>
37. Shamsi M, Kenari AR, Aghamohammadi R (2021) Reinforcement learning for traffic light control with emphasis on emergency vehicles. *J Supercomput*. <https://doi.org/10.1007/s11227-021-04068-w>
38. Cai SK, Han DZ, Li D, Zheng ZB, Crespi N (2022) An reinforcement learning-based speech censorship chatbot system. *J Supercomput*. <https://doi.org/10.1007/s11227-021-04251-z>
39. Filmlalter JD, Dagorn L, Cowley PD, Taquet M (2011) First descriptions of the behavior of silky sharks, *Carcharhinus falciformis*, around drifting fish aggregating devices in the Indian Ocean. *Bull Mar Sci* 87(3):325–337. <https://doi.org/10.5343/bms.2010.1057>
40. Liang J, Qu B, Suganthan P (2013) Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization
41. Li S, Chen H, Wang M, Heidari AA, Mirjalili S (2020) Slime mould algorithm: a new method for stochastic optimization. *Future Gener Comput Syst Int J Escience* 111:300–323. <https://doi.org/10.1016/j.future.2020.03.055>
42. Mirjalili S (2016) SCA: a sine cosine algorithm for solving optimization problems. *Knowl-Based Syst* 96:120–133. <https://doi.org/10.1016/j.knosys.2015.12.022>
43. Abualigah L, Abd Elaziz M, Sumari P, Geem ZW, Gandomi AH (2022) Reptile Search Algorithm (RSA): a nature-inspired meta-heuristic optimizer. *Expert Syst Appl*. <https://doi.org/10.1016/j.eswa.2021.116158>
44. Abualigah L, Younsri D, Abd Elaziz M, Ewees AA, Al-qaness MAA, Gandomi AH (2021) Aquila optimizer: a novel meta-heuristic optimization algorithm. *Comput Ind Eng*. <https://doi.org/10.1016/j.cie.2021.107250>
45. Abualigah L, Elaziz MA, Sumari P, Geem ZW, Gandomi AH (2022) Reptile Search Algorithm (RSA): a nature-inspired meta-heuristic optimizer. *Expert Syst Appl* 191:116158. <https://doi.org/10.1016/j.eswa.2021.116158>
46. Coello Coello CA, Becerra RL (2004) Efficient evolutionary optimization through the use of a cultural algorithm. *Eng Optim* 36(2):219–236. <https://doi.org/10.1080/03052150410001647966>
47. Passino KM (2002) Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Syst Mag* 22(3):52–67. <https://doi.org/10.1109/MCS.2002.1004010>
48. Yapici H, Cetinkaya N (2019) A new meta-heuristic optimizer: Pathfinder algorithm. *Appl Soft Comput* 78:545–568. <https://doi.org/10.1016/j.asoc.2019.03.012>
49. Atiqullah MM, Rao SS (2000) Simulated annealing and parallel processing: an implementation for constrained global design optimization. *Eng Optim* 32(5):659–685. <https://doi.org/10.1080/03052150008941317>
50. Lee KS, Geem ZW (2005) A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. *Comput Methods Appl Mech Eng* 194(36):3902–3933. <https://doi.org/10.1016/j.cma.2004.09.007>
51. Hwang S-F, He R-S (2006) A hybrid real-parameter genetic algorithm for function optimization. *Adv Eng Inform* 20(1):7–21. <https://doi.org/10.1016/j.aei.2005.09.001>

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

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

## Authors and Affiliations

Tao Chen<sup>1</sup> · Yong Chen<sup>1</sup>  · Zhicheng He<sup>1</sup> · Eric Li<sup>2,3</sup> · Chenglin Zhang<sup>1</sup> · Yuanyi Huang<sup>4</sup>

- <sup>1</sup> State Key Laboratory of Advanced Design and Manufacturing for Vehicle Body, Hunan University, Changsha 410082, People's Republic of China
- <sup>2</sup> School of Computing, Engineering & Digital Technologies, Teesside University, Campus Heart, Southfield Rd, Middlesbrough TS1 3BX, UK
- <sup>3</sup> School of Energy and Power Engineering, Jiangsu University of Science and Technology, Zhenjiang 212003, People's Republic of China
- <sup>4</sup> SAIC GM Wuling Automobile Co., Ltd, Liuzhou 545007, People's Republic of China