Mathematical Biosciences and Engineering

MBE, 20(9): 16304–16329.
DOI: 10.3934/mbe.2023728
Received: 18 June 2023
Revised: 26 July 2023
Accepted: 04 August 2023
Published: 14 August 2023

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

Research article

Application of an improved whale optimization algorithm in time-optimal trajectory planning for manipulators

Juan Du, Jie Hou*, Heyang Wang and Zhi Chen

School of Mechanical Engineering, Taiyuan University of Science and Technology, Taiyaun 030024, China

* **Correspondence:** Email: s202112210641@stu.tyust.edu.cn.

Abstract: To address the issues of unstable, non-uniform and inefficient motion trajectories in traditional manipulator systems, this paper proposes an improved whale optimization algorithm for time-optimal trajectory planning. First, an inertia weight factor is introduced into the surrounding prey and bubble-net attack formulas of the whale optimization algorithm. The value is controlled using reinforcement learning techniques to enhance the global search capability of the algorithm. Additionally, the variable neighborhood search algorithm is incorporated to improve the local optimization capability. The proposed whale optimization algorithm is compared with several commonly used optimization algorithms, demonstrating its superior performance. Finally, the proposed whale optimization algorithm is employed for trajectory planning and is shown to be able to produce smooth and continuous manipulation trajectories and achieve higher work efficiency.

Keywords: trajectory planning; time-optimal; whale optimization algorithm; reinforcement learning; VNS algorithm

1. Introduction

Manipulators are multi-degree-of-freedom robots capable of autonomous operation and task execution. They have been utilized in fields including manufacturing, medical care and aerospace [1]. These manipulators operate autonomously and perform tasks efficiently. In manufacturing, manipulators streamline production, handle materials and ensure consistent quality. In medical care, they enable precise and minimally invasive surgeries, leading to faster recovery and improved

outcomes. Aerospace benefits from manipulators for assembling and maintaining components in challenging environments. However, as industrial level and job requirements continue to increase, the performance requirements for manipulators in various industries are becoming increasingly stringent. As a result of these requirements, several experts and scholars have dedicated a lot of time and effort to researching issues such as trajectory planning, path planning [2] and tracking control [3] of manipulators [4].

An important aspect of manipulator design is trajectory planning. It holds the key to minimizing operation time, reducing energy consumption and maximizing productivity. In manufacturing, optimized trajectory can streamline production processes and improve overall efficiency. In medical applications, precise trajectory planning allows for minimally invasive procedures with enhanced patient safety. Similarly, in aerospace, accurate trajectory planning ensures smooth and agile movements in challenging environments. It can be divided into multi-objective trajectory planning and single-objective trajectory planning. The planning of single-objective trajectory is mainly concerned with time, energy [5] and impact [6], while multi-objective trajectory planning combines multiple single-objective goals to meet different working environments [7,8]. Time-optimal trajectory planning is a crucial focus of current research due to its profound impact on manipulator performance. By enabling manipulators to complete tasks in the shortest possible time, this optimization technique significantly improves work efficiency, leading to enhanced productivity and reduced operational costs. With industries seeking streamlined processes and faster task execution, time-optimal trajectory planning plays a pivotal role in maximizing the potential of manipulators, making it a critical area of exploration and innovation in the field.

The paper [9] proposes an adaptive cuckoo algorithm, which has good convergence and convergence ability and combines with a quintic B-spline curve to obtain a smooth time-optimal trajectory. The paper [10] combines the original teaching-learning-based optimization algorithm with the variable neighborhood search (VNS) algorithm to improve escape ability from local optima and combines with a quintic B-spline curve to obtain time-optimal trajectory for the manipulator. The paper [11] proposes a local chaotic particle swarm optimization (PSO) algorithm, which solves the problem of early convergence into local optima in traditional particle swarm algorithm and combines with piecewise polynomial interpolation function to generate time-optimal trajectory. The paper [12] proposes an improved sparrow search algorithm, which uses tent chaotic mapping to optimize the generation of initial population, combines with an adaptive step factor to make the algorithm have good convergence effect and finally obtains a good operating trajectory.

In 2016, Mirjalili proposed a novel intelligent optimization algorithm known as whale optimization algorithm (WOA). Compared with other optimization algorithms such as the PSO, cuckoo search and genetic algorithm, the WOA has the advantages of fast convergence speed, simple algorithm and high convergence accuracy. These features make it an ideal choice for time-optimal trajectory planning in manipulators. The WOA exhibits rapid convergence, allowing the discovery of global optima within a limited number of iterations, thus reducing computation time. Additionally, its high accuracy ensures that planned trajectories closely approximate optimal solutions. In the context of time-optimal trajectory planning, precise trajectories are crucial for efficient manipulator motion. By improving the WOA, we can effectively address challenges in time-optimal trajectory planning, leading to improved motion efficiency and better alignment with industrial application requirements. The paper [13] proposed an improved whale optimization algorithm (IWOA), which designed dynamic inertia weights for two behaviors by improving the contraction-expansion mechanism and the spiral

updating mechanism, thus enhancing the search ability of the algorithm. However, it was observed that in later stages, the algorithm tended to get trapped in local optima. In paper [14], a multi-strategy whale optimization algorithm (MSWOA) was proposed, which incorporated adaptive weights, Lévy flight and evolutionary population dynamics to enhance the algorithm's search capability. However, it was found that the algorithm failed to converge to the global optimum in some test functions. The paper [15] proposed a modified whale optimization algorithm (MWOA) that employs probabilistic prey selection and adjusts the initialization of the population and the search strategy during the development phase to reduce the likelihood of getting trapped in local optima, thereby enhancing the algorithm's robustness. Nevertheless, the algorithm exhibits a relatively high time complexity while tackling optimization problems. Although all of these algorithms have achieved good results, they may not perform well in some target optimization problems.

Therefore, this study presents an enhanced version of the whale optimization algorithm (RLVWOA) that combines reinforcement learning and the VNS algorithms. First, an inertia weight is designed for the surrounding prey and bubble-net attack behavior of whales and the control weight value is optimized using the q-learning and SARSA algorithms to enable each generation of populations to obtain suitable inertia weight, thereby enhancing the global search capability of the algorithm. Then, combined with the VNS algorithm, the local search capability of the algorithm is improved through continuous neighborhood search. Compared to the standard WOA, the RLVWOA can adaptively control surrounding prey and bubble-net attack behaviors and with the assistance of VNS algorithm, it can effectively escape from local optima, thereby achieving robust search capabilities. Finally, the RLVWOA is used in conjunction with a quintic non-uniform B-spline (NURBS) curve to perform time-optimal trajectory planning for the manipulator and its feasibility is verified in MATLAB.

The primary contribution of this study lies in the development of the RLVWOA algorithm, which innovatively integrates reinforcement learning algorithm and VNS algorithm. This integration leads to substantial performance improvements and presents an enhanced solution for the time-optimal trajectory planning problem in manipulators. The proposed enhancements significantly accelerate convergence and optimize the algorithm's capabilities, while mitigating the risk of getting trapped in local optima, thereby facilitating the discovery of more efficient trajectory paths. Consequently, this paper introduces a novel method for manipulator trajectory planning, leading to heightened work efficiency and smoother operations and exhibiting promising prospects for widespread application across various industries, encompassing manufacturing, medical care and aerospace.

The subsequent sections of this paper are organized as follows: Section 2 introduces the basic concepts of NURBS interpolation. Section 3 provides an overview of WOA, reinforcement learning and VNS algorithms. In Section 4, the proposed method for improving the WOA is described and a comparison between the RLVWOA and other commonly used single-objective algorithms is conducted on test functions. Section 5 focuses on the modeling of the PUMA560 robotic arm and compares the trajectory planning results obtained using the RLVWOA and traditional single-objective algorithms. The final section highlights the contribution of this study and suggests potential directions for future work.

2. Interpolation function

2.1. Basic concepts of NURBS curves

The NURBS interpolation is a widely used curve or surface fitting technique, which is also widely

used in the manipulator trajectory planning. Compared with traditional B-spline curves, NURBS curves have greater flexibility and accuracy and can better fit complex curve shapes. Based on the mathematical model of control points and nodes, it can generate smooth and continuous trajectories. By optimizing the weight of control points and the distribution of nodes, the optimal manipulator trajectory planning can be achieved, thereby improving the accuracy and efficiency of the manipulator. Using the NURBS interpolation for trajectory planning can help solve complex manipulator motion problems, while also improving the reliability and stability of the manipulator. A k-th NURBS curve can be expressed as a segmented rational polynomial function [16], as shown in Eq (1).

$$p(x)=\frac{\sum_{i=0}^{n}\omega_i d_i N_{i,k}(x)}{\sum_{i=0}^{n}\omega_i N_{i,k}(x)} \tag{1}$$

Where the weight factor of the NURBS curve is denoted by ω, d_i is the control vertex of the NURBS curve, k is the degree of the NURBS curve, x is the parameter of the NURBS curve and $N_{i,k}(x)$ is the basis function of the k-th NURBS curve. Here, $N_{i,k}(x)$ can be obtained by the De Boor-Koch formula from the node vector $X = [x_0, x_1, \cdots, x_{n+k}, x_{n+k+1}]$, as shown in Eqs (2) and (3) and $0/0$ is defined as 0 [17].

$$N_{i,k}(x)=\frac{x-x_i}{x_{i+k}-x_i}N_{i,k-1}(x)+\frac{x_{i+k+1}-x}{x_{i+k+1}-x_{i+1}}N_{i+1,k-1}(x) \tag{2}$$

$$N_{i,0}(x)=\begin{cases}1 & x_i \le x \le x_{i+1}\\ 0 & others\end{cases} \tag{3}$$

The NURBS interpolating curve is defined by the control points d_i $(i = 0, 1, 2, \cdots, n)$, where $n = m + k + 1$. The normalized knot vector has the form $x_0 = x_1 = \cdots = x_k = 0$, $x_{n+1} = x_{n+2} = \cdots = x_{n+k+1} = 1$ and the other knot values can be obtained by normalizing the time interval h_i between the path points by employing the chord length parameterization method [18], as shown in Eq (4):

$$x_i=\sum_{j=0}^{i-k-1}h_j \Big/ \sum_{j=0}^{m-1}h_j \qquad (i=k+1,k+2,\cdots,n) \tag{4}$$

2.2. The quintic NURBS curve matrix

The equation for calculating the derivative of a NURBS curve of degree k is expressed by Eq (5) [19]:

$$p^{(k)}(x)=\frac{A^{(k)}(x)-C_k^i\omega^{(i)}(x)p^{(k-i)}(x)}{\omega(x)} \tag{5}$$

Where $A(x)=\sum_{i=0}^{n}\omega_i d_i N_{i,k}(x)$, $\omega(x)=\sum_{i=0}^{n}\omega_i N_{i,k}(x)$.

According to Eq (6):

$$Q_m = \sum_{i=0}^{n+2} N_{i,k}(x_m) d_i \qquad i = 0,1,\cdots,n; m = 2,3,\cdots,n-1 \tag{6}$$

It can be derived that when solving for NURBS curves with $n + 1$ unknowns, four boundary conditions need to be added to ensure a unique solution to the equation system. Therefore, according to the actual motion conditions of the manipulator, the following four boundary conditions are added as shown in Eq (7):

$$\begin{cases} p(0)' = v_{start} \\ p(0)'' = a_{start} \\ p(x_n)' = v_{final} \\ p(x_n)'' = a_{final} \end{cases} \tag{7}$$

Where v and a represent the angular velocity and angular acceleration of the manipulator. Substituting Eq (7) into Eq (1), the matrix equation for solving all control points can be obtained as shown in Eq (8).

$$\begin{bmatrix} M_0 & & & \\ M_1 & \cdots & 0 & \\ M_2 & & & \\ & \ddots & & \\ & & M_{n-2} & \\ 0 & \cdots & M_{n-1} & \\ & & M_n & \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ \vdots \\ d_{n-2} \\ d_{n-1} \\ d_n \end{bmatrix} = \begin{bmatrix} Q_0 \\ Q1 \\ Q_2 \\ \vdots \\ Q_{n-2} \\ Q_{n-1} \\ Q_n \end{bmatrix} \tag{8}$$

Where $M_i = [N_{i,5(x)}, N_{i+1,5(x)}, N_{i+2,5(x)}, N_{i+3,5(x)}, N_{i+4,5(x)}]$, $i = 2, 3, \cdots, n–2$, x corresponds to the time associated with X, $M_1 = [-1\ 1\ 0 \cdots 0\ 0\ 0]$, $M_{n-1} = [0\ 0\ 0 \cdots 0\ -1\ 1]$, $M_0 = [1\ 0\ 0 \cdots 0\ 0\ 0]$, $M_n = [0\ 0\ 0 \cdots 0\ 0\ 1]$, $Q_1 = Q_{n-1} = 0$, all other variables are known data points.

The joint motion trajectory angle curves of the manipulator can be obtained using Eq (1). By using Eq (5) to solve the derivatives of the curve equation up to the third order, the angular velocity, angular acceleration and angular jerk curves for each joint can be acquired.

3. Basic algorithms

3.1. Whale optimization algorithm

In 2016, Mirjalili et al. proposed the WOA, which is a recently developed metaheuristic search algorithm. The authors studied and analyzed the optimization ability of WOA from different perspectives such as structure and mathematical models. Experimental results showed that WOA not only has strong search ability and positive feedback, but also can achieve global optimization [20].

The most remarkable feature of a humpback whale is its sociality. Typically, a group of six or so humpback whales search for prey and confirm the target's position. Other groups of whales approach the prey through encircling contraction and spiral contraction and eventually succeed in eating the prey at the appropriate time. The algorithm consists of the following three stages:

(1) Surrounding prey

It is assumed that the optimal solution corresponds to the position of the target prey in the WOA. Each whale updates its relative position with respect to the target position using Eqs (9) and (10):

$$D = \left| C \times X^*(t) - X(t) \right| \tag{9}$$

$$X(t+1) = X(t) - A \times D \tag{10}$$

In these two equations, $X^*(t)$ represents the best position, $X(t)$ represents the present position and t represents the present iteration. A and C are adjustment factors, defined as:

$$A = 2a \cdot rand_1 - a \tag{11}$$

$$C = 2 \cdot rand_2 \tag{12}$$

where, $rand_1$ and $rand_2$ are random values uniformly distributed between 0 and 1 and a is a decreasing factor with a gradual reduction from 2 to 0, represented as:

$$a = 2 \times \frac{2t}{t_{\max}} \tag{13}$$

In the equation, t_{max} represents the maximum number of iterations.

(2) Bubble-net attack

In the WOA, the bubble-net attack is categorized into the contraction and encirclement mechanism and the spiral updating mechanism. The contraction and encirclement mechanism is the same as the formula for surrounding the prey, but with the range of A changed from $[-a,a]$ to $[-1,1]$. The spiral updating mechanism is represented by Eq (14):

$$X(t+1) = X^*(t) + D_q e^{bl} \cos\theta(2\pi l) \tag{14}$$

Here, l is a random number between –1 and 1. The constant b is used to represent the logarithmic spiral shape. D_q represents the distance between the whale and the prey, which is expressed by Eq (15).

$$D_q = \left| X^*(t) - X(t) \right| \tag{15}$$

Assuming that a whale chooses between the shrink-wrap and spiral update mechanisms with a probability of 50% during the hunting of a target prey, the position update is given by the Eq (16).

$$X(t+1) = \begin{cases} X(t) - A \cdot D & p < 0.5 \\ X^*(t) + D_q e^{bl} \cos\theta(2\pi l) & p \geq 0.5 \end{cases} \tag{16}$$

(3) Searching for prey

The whale decides to use the shrink and encircle mechanism or search for prey mechanism based on the size of parameter A. When $A \geq 1$, the whale cannot obtain the optimal position of the prey and therefore needs to randomly search for the target within its range, as expressed in Eqs (17) and (18).

$$D = \left| C \cdot X_m(t) - X(t) \right| \tag{17}$$

$$X(t+1)=X_m(t)-A\cdot D \tag{18}$$

3.2. Reinforcement learning algorithm

The reinforcement learning algorithm is proposed by Misky in 1954, which mainly consists of agent, environment, state, action and reward components [21].

Reinforcement learning is a type of machine learning algorithm inspired by biology that aims to learn through experimentation within the possible state-action pairs to find a mapping from states to actions that maximizes the cumulative reward [22]. In reinforcement learning, an agent interacts with its environment by exploring and making decisions based on the present state. The agent first explores and observes the current state S_t, then makes an action decision $action_t$ based on the perceived current state. The environment changes its state from S_t to S_{t+1} in response to the agent's action and returns a reward (or punishment) signal r_t to the agent. The agent adjusts its action decisions based on the reward feedback from the environment and trains itself to maximize current and future rewards. This process is called a Markov decision process. The basic principle is shown in the Figure 1.

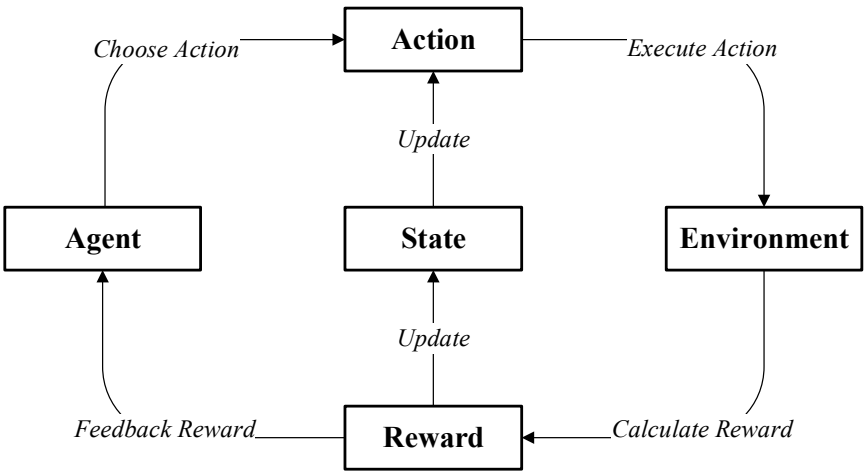


Figure 1. The basic principle of reinforcement learning.

Q-learning and SARSA are both value-based reinforcement learning algorithms. Their goal is to find the optimal policy by learning and optimizing the value function. Q-learning algorithm is an offline learning algorithm based on a greedy strategy, which learns the optimal value function by updating the state-action pairs. At each time step, the agent observes the current state and selects the next action based on the current policy function and value function. The agent then observes the next state S_{t+1} and receives the corresponding immediate reward r_t. On the other hand, SARSA algorithm is an online learning algorithm, which selects the next action and learns based on the current state and policy function. Therefore, SARSA's learning process is a continuous and constantly updated process, which can dynamically adapt to changes in the environment [23]. Specifically, the value function update formula for Q-learning and SARSA are as shown in Eqs (19) and (20):

$$Q(s,a)\leftarrow Q(s,a)+\alpha\left[r+\gamma \max_{a'}\left(Q(s',a')\right)-Q(s,a)\right] \tag{19}$$

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma Q(s',a') - Q(s,a)\right] \tag{20}$$

In these equations, *Q(s,a)* represents the value function of taking *action* in state *S*, α is the learning rate, γ is the discount factor, *r* is the immediate reward and $\max_{a'}$ is the operation of taking the maximum value among all possible *action'* in the next state *S'*.

3.3. Variable neighborhood search algorithm

The VNS algorithm is a heuristic optimization algorithm based on neighborhood search that can effectively solve many complex optimization problems. The original proposal of the algorithm can be attributed to Mladenovic and Hansen. It has gained extensive utilization in subsequent research endeavors [24]. The principle of the VNS algorithm is to search on different neighborhood structures and gradually approach the optimal solution by continuously expanding or reducing the neighborhood structure. During the search process, the VNS algorithm jumps out of local optimal solutions and seeks better solutions.

The main steps of the VNS algorithm are as follows:

Step 1. Initialization: Randomly generate an initial solution and set the initial neighborhood structure.

Step 2. Neighborhood structure: Generate new solutions by changing the current neighborhood structure. In each neighborhood structure, define a set of operations, such as insertion, deletion, exchange, etc., to generate new solutions.

Step 3. Neighborhood search: Search in the current neighborhood structure to find the best solution. If a better solution is found, go to Step 4. Otherwise, go to Step 5.

Step 4. Neighborhood expansion: Expand the neighborhood structure to better search for possible solutions.

Step 5. Neighborhood contraction: Contract the neighborhood structure to better search for possible solutions.

Step 6. Convergence check: Check if the algorithm has converged. If not, go back to Step 2. Otherwise, output the optimal solution.

The core idea of VNS algorithm is to continuously expand and contract the neighborhood structure to better search for possible solutions. In each neighborhood structure, a set of operations is defined and the best solution is selected based on greedy strategy.

4. Improved algorithm

The three behaviors of the WOA have a crucial impact on finding the optimal position, while the value of the inertia weight also plays a vital role in the optimization and search capability of the algorithm. The IWOA with dynamic inertia weight proposed in paper [13] introduces an inertia weight value in the surrounding prey and bubble-net attack behaviors, as shown in Eqs (21) and (22). Although this accelerates the convergence speed and improves the convergence capability of the algorithm, the inertia weight value is simply linearly decreased based on the current iteration, which may not be suitable for the current population. Therefore, this paper improves the IWOA algorithm by using reinforcement learning to optimize the control of the inertia weight value, making it more suitable for the current population and enhancing the convergence speed and optimization capability of the

algorithm. Additionally, the VNS algorithm is introduced to improve the local search capability of the algorithm and obtain better optimal solutions.

$$X(t+1)=\omega\times X(t)-A\times D \tag{21}$$

$$X(t+1)=\omega\times X^{*}(t)+D_q e^{bl}\cos\theta(2\pi l) \tag{22}$$

4.1. The design of the Q-table

The initial Q-table is a zero matrix of size $m \times n$, where m is the number of states and n is the number of actions. When the environment and actions change, the Q-table is updated according to Eqs (19) and (20), as shown in Eq (23).

$$Q(s,a)=\begin{array}{c} \\ S_1 \\ S_2 \\ \vdots \\ S_m \end{array}\begin{array}{c} \begin{array}{cccc} a_1 & a_2 & \cdots & a_n \end{array} \\ \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & \ddots & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \end{array} \tag{23}$$

According to the results proposed in [25], SARSA algorithm has faster convergence rate, while Q-learning has better overall performance. Moreover, [23] has verified that the combination of SARSA and Q-learning algorithms yields better convergence. The algorithm presented in this study utilizes both Q-learning and SARSA algorithms. However, it employs them at separate stages, as illustrated in Eq (24), where t_{max} represents the total number of iterations.

$$\begin{cases} SARSA & t \le t_{\max}/2 \\ Q-learning & t > t_{\max}/2 \end{cases} \tag{24}$$

4.2. The design of the states

To ensure that the WOA obtains better optimization capability and faster convergence speed with appropriate inertia weight values, the state design of the reinforcement learning algorithm needs to be considered. The design of the state should take into account the convergence, diversity and balance of the WOA. Therefore, the following aspects are taken into account in the design of the state:

$$C_t=\frac{\sum_{i=1}^{N} f(x_i^t)}{\sum_{i=1}^{N} f(x_i^1)} \tag{25}$$

$$D_t=\frac{\max f(x^t)}{\max f(x^1)} \tag{26}$$

$$B_t = \frac{\sum_{i=1}^{N} f\left(x_i^t\right)}{\sqrt{\frac{\sum_{i=1}^{N}\left(f\left(x_i^t\right) - \overline{f\left(x_i^t\right)}\right)}{N}}} \tag{27}$$

$$S_t = \omega_1 C_t + \omega_2 D_t + \omega_3 B_t \tag{28}$$

In this equation, t represents the iteration number of the algorithm, $f(x_i{}^t)$ represents the fitness function value of the i-th individual in the t-th iteration and C_t represents the ratio of the sum of fitness values of all individuals in the t-th iteration to that in the initial iteration, which reflects the convergence of the algorithm. D_t represents the ratio of the maximum fitness value of the t-th generation to that of the first generation, which reflects the diversity of the algorithm. B_t represents the ratio of the mean value to the standard deviation of each generation, which reflects the balance of the population in each generation. Equation (28) calculates the state value of each generation by weighted sum. Considering the importance of convergence and diversity of the algorithm, ω_1 and ω_2 are set to 0.35 and ω_3 is set to 0.3.

4.3. The design of the actions

Action refers to the agent's response, which is determined by the present state. With each successive population iteration, the agent selects suitable inertial weight values based on the environment. Larger values of ω may cause the algorithm to be trapped in a local optimal solution, while smaller values may affect the algorithm's global search ability. Therefore, ω is defined as 10 actions between (0–1), where the first action, a_1, generates a random number from (0.0–0.1) and the second action, a_2, generates a random number from (0.1–0.2) and so on. The detailed action values are shown in the Table 1.

Table 1. The table of actions.

Actions	ω ranges	Actions	ω ranges
a_1	$(0.0-0.1)$	a_6	$(0.5-0.6)$
a_2	$(0.1-0.2)$	a_6	$(0.6-0.7)$
a_3	$(0.2-0.3)$	a_7	$(0.7-0.8)$
a_4	$(0.3-0.4)$	a_9	$(0.8-0.9)$
a_5	$(0.4-0.5)$	a_{10}	$(0.9-1.0)$

4.4. The design of the rewards

The agent does not choose actions on its own, but selects the appropriate action based on the Q-table and the current state, in order to obtain more positive feedback. Designing a reward function as shown in Eq (29) can simultaneously take into account the convergence, diversity and balance of the algorithm, making the algorithm more capable of searching. The goal of this paper is to minimize the function value and the smaller state value, the better the performance of the algorithm. Therefore, when S_{t-1} is greater than S_t, the reward is positive, otherwise it is negative.

$$r = S_{t-1} - S_t \tag{29}$$

4.5. Action selection strategy

When the algorithm starts, the values in the Q-table are initialized to zero, which means the agent has no experience to rely on and must explore and learn by experience. By continuously investigating unknown environments, the agent gains more experience, it learns valuable knowledge to inform its actions. The *ε-greedy* strategy is a method that balances exploration and exploitation, as shown in Eq (30).

$$\pi(s_{t,}a_t) = \begin{cases} \max Q(s,a) & \varepsilon \geq k_{0-1} \\ a(Rand) & \varepsilon < k_{0-1} \end{cases} \tag{30}$$

Where ε represents the greedy rate and the value of $k_{0\text{-}1}$ is a randomly generated number within the range of 0 to 1. When $\varepsilon \geq k$, the agent chooses the action that maximizes the Q value, also known as the greedy strategy. When $\varepsilon < k$, exploration is performed and a random action is chosen.

4.6. The design of neighborhoods

The objective of this paper is to minimize the optimization problem. Therefore, the design of the VNS aims to expedite the discovery of the global minimum by exploring various neighborhoods. The three neighborhoods are designed as follows:

1) Randomly choose a variable and reduce its value through a certain amount.
2) Randomly choose a variable and multiply it through a generated number within the range of 0 to 1.
3) Randomly select two variables and swap their positions.

4.7. The algorithm processes

The combination of reinforcement learning algorithm, the VNS algorithm and the WOA requires considering reward, state, action and action selection strategy. The WOA is treated as the environment and the state S is calculated based on Eq (28) and at each iteration, S_t is updated to S_{t+1}. The learning component comprises the agent and the reward r. The entire procedure can be divided into four sequential steps. To begin with, the agent obtains the environment state S_t for the t-th iteration, then chooses action based on the Eq (30) and adjusts ω value. The WOA will iterate using the updated ω. After completing one iteration, the environment state will transition from S_t to S_{t+1}. Lastly, the reward r is calculated based on the Eq (29) and the Q-table value is updated by Eq (19) or Eq (20). After t iterations, the agent will select optimal ω based on prior exploration experience for the current state. The algorithm flowchart of the RLVWOA is shown in Figure 2.

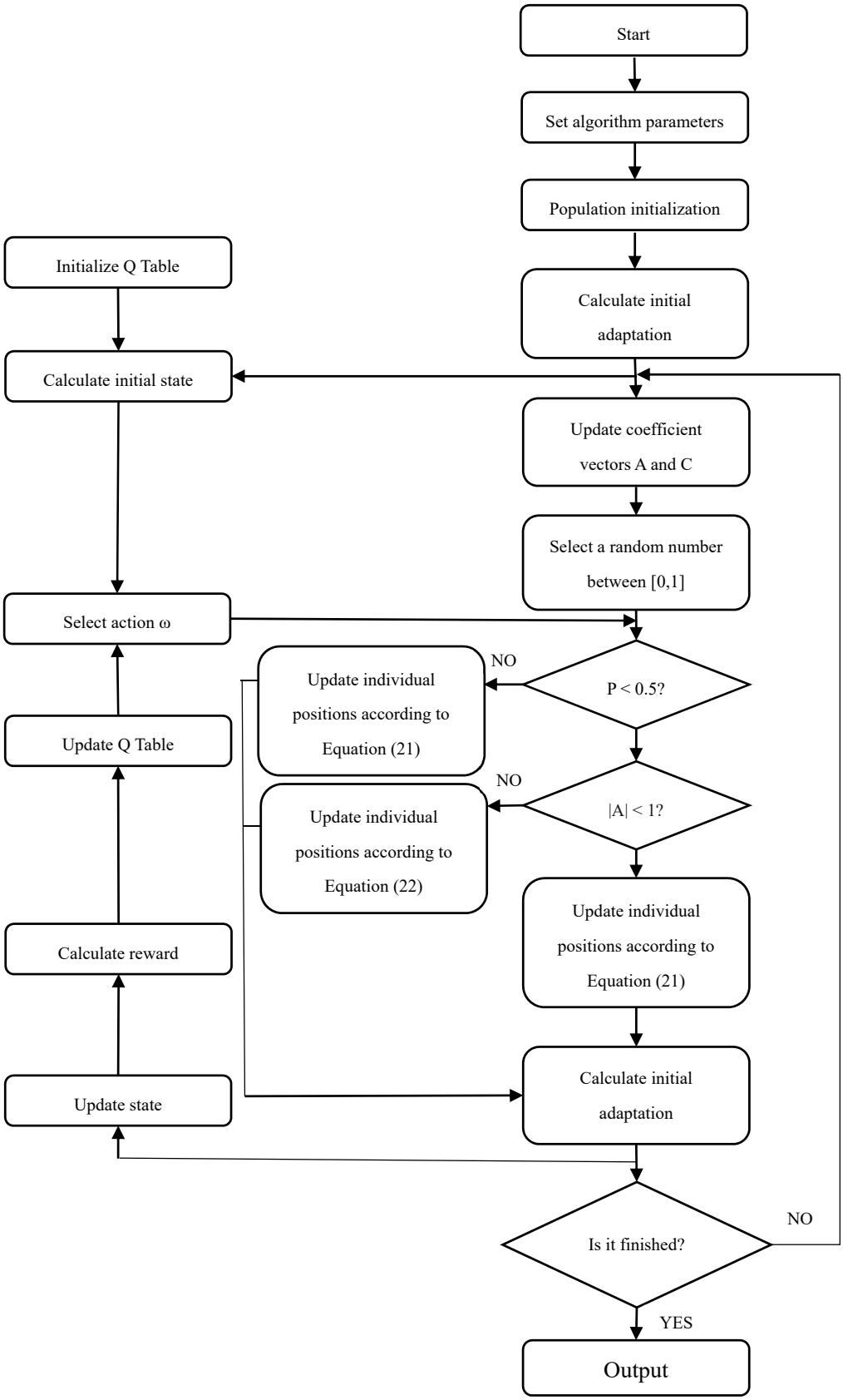


Figure 2. The flowchart of RLVWOA.

4.8. Comparative validation

To verify the feasibility of the RLVWOA, twenty standard benchmark functions were selected for testing [26], as shown in Table 2 and compared with the reptile search algorithm (RSA) [27], snake optimization (SO) [28], WOA, IWOA, MSWOA and MWOA. To ensure the fairness of the experiment, using the same computer, the population number of all algorithms $N = 30$, dimension $D = 30$, number of iterations $t_{max} = 300$ and other parameter settings for each algorithm are shown in Table 3.

Table 2. The table of testing functions.

Function types	Test functions	Dimension	Range	Optimal value
Unimodal test functions	$F_1(x)=\sum_1^n x_i^2$	—	$[-100,100]$	0
	$F_2(x)=\sum_{i=1}^n \lvert x_i\rvert+\prod_{i=1}^n \lvert x_i\rvert$	—	$[-10,10]$	0
	$F_3(x)=\sum_{i=1}^n\left(\sum_{i=1}^n \lvert x_i\rvert\right)^2$	—	$[-100,100]$	0
	$F_4(x)=\max_i\{\lvert x_i\rvert,-1<i<n\}$	—	$[-100,100]$	0
	$F_5(x)=\sum_{i=1}^n\left[100\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)^2\right]$	—	$[-30,30]$	0
	$F_6(x)=\sum_{i=1}^n\left(\lvert x_i+0.5\rvert\right)^2$	—	$[-100,100]$	0
	$F_7(x)=\sum_{i=1}^n ix_1^4+random[0,1)$	—	$[-1.28,1.28]$	0
Multimodal test functions	$F_8(x)=\sum_{i=1}^n\left[x_i^2-10\cos\left(2\pi x_i\right)+10\right]$	—	$[-5.12,5.12]$	0
	$F_9(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^n x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^n\cos(2\pi x_i)\right)+20+e$	—	$[-32,32]$	0
	$F_{10}(x)=\frac{1}{4000}\sum_{i=1}^n x_i^2-\prod_{i=1}^n\cos\left(x_i/\sqrt{i}\right)+1$	—	$[-600,600]$	0
	$F_{11}(x)=\frac{\pi}{n}\left\{10\sin\left(\pi y_1\right)+\sum_{i=1}^{n-1}\left(y_i-1\right)^2\left[1+10\sin^2\left(\pi y_i+1\right)\right]+\left(y_n+1\right)^2\right\}$ $+\sum_{i=1}^n u\left(x_i,10,100,4\right)$	—	$[-50,50]$	0
	$y_i=1+\frac{x_i+1}{4},u\left(x_i,a,k,m\right)=\begin{cases}k\left(x_i-a\right)^m & x_i>a\\ 0 & -a\le x_i\le a\\ k\left(-x_i-a\right)^m & x_i<-a\end{cases}$			
	$F_{12}(x)=0.1\left\{\sin^2\left(3\pi x_1\right)+\left(x_i-1\right)^2\left[1+\sin^2\left(2\pi x_n\right)\right]\right\}+\sum_{i=1}^n u\left(x_i,5,100,4\right)$	—	$[-50,50]$	0

Continued on next page

Function types	Test functions	Dimension	Range	Optimal value
Fixed-dimension test functions	$F_{13}(x)=\left(\frac{1}{500}+\sum_{j=1}^{25}\left(j+\sum_{i=1}^{2}\left(x_i+a_{ij}\right)^6\right)^{-1}\right)^{-1}$	2	$[-65,65]$	0.998
	$F_{14}=\sum_{i=1}^{11}\left[a_i-\frac{x_1\left(b^2+b_ix_2\right)}{b_i^2+b_ix_3+x_4}\right]^2$	4	$[-5,5]$	0.0003
	$F_{15}(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	2	$[-5,5]$	-1.0316
	$F_{16}(x)=\left[1+\left(x_1+x_2+1\right)^2\left(19-14x_1+3x^2-14x_2+6x_1x_2+3x_2^2\right)\right]\times$ $\left[30+\left(2x_1-3x_2\right)^2\times\left(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2\right)\right]$	2	$[-2,2]$	3
	$F_{17}(x)=\sum_{i=1}^{4}c_i\exp\left(-\sum_{j=1}^{3}a_{i\ ij}\left(x_j-p_{ij}\right)^2\right)$	3	$[0,1]$	-3.86
	$F_{18}(x)=\sum_{i=1}^{4}c_i\exp\left(-\sum_{j=1}^{6}a_{i\ ij}\left(x_j-p_{ij}\right)^2\right)$	6	$[0,1]$	-3.32
	$F_{19}(x)=-\sum_{i=1}^{5}\left[\left(X-a_i\right)\left(X-a_i\right)^T+c_i\right]^{-1}$	4	$[0,10]$	-10.1532
	$F_{20}(x)=-\sum_{i=1}^{10}\left[\left(X-a_i\right)\left(X-a_i\right)^T+c_i\right]^{-1}$	4	$[0,10]$	-10.5363

Table 3. The parameter settings for each algorithm.

Algorithms	Parameters
RSA	$e_1 = 0.1$, $e_2 = 0.005$
SO	$c_1 = 0.5$, $c_2 = 0.05$, $c_3 = 2$, Q = 0.25, Temp = 0.6
WOA	—
IWOA	—
MSWOA	y = 4, z = 0.152
MWOA	$CF_1 = 2.5$, $CF_2 = 1.5$
RLVWOA	$\varepsilon = 0.6$, $\alpha = 0.06$, $\gamma = 0.85$

Each testing function is run 30 times using each algorithm separately. The comparative results are shown in Table 4 and the time it takes for each algorithm is shown in Table 5. The highlighted section denotes the algorithms that achieved the highest performance for each testing function.

Table 4. The comparative results of testing functions.

Test functions	Statistical value	RSA	SO	WOA	IWOA	MSWOA	MWOA	RLVWOA
F1	Optimal value	0.0000E+00	1.0594E-57	1.6733E-50	2.8101E-230	4.5054E-96	4.5421E-269	0.0000E+00
	Worst value	0.0000E+00	6.7824E-53	3.5706E-42	5.9480E-195	2.4253E-88	8.4466E-263	0.0000E+00
	Mean value	**0.0000E+00**	4.4717E-54	1.7604E-43	2.1211E-196	2.5366E-89	7.4316E-264	**0.0000E+00**
	Ranking	1	6	7	4	5	3	1
F2	Optimal value	0.0000E+00	6.7589E-24	6.1016E-35	4.1921E-118	1.4205E-50	7.8666E-139	0.0000E+00
	Worst value	0.0000E+00	1.1900E-20	4.9976E-29	1.2704E-99	6.4475E-48	2.2239E-135	0.0000E+00
	Mean value	**0.0000E+00**	1.5326E-21	3.8624E-30	5.9705E-101	9.1358E-49	2.7283E-136	**0.0000E+00**
	Ranking	1	7	6	4	5	3	1
F3	Optimal value	0.0000E+00	1.5654E-40	3.0269E+04	0.0000E+00	8.2776E-86	1.9469E-232	0.0000E+00
	Worst value	0.0000E+00	1.4045E-29	1.3224E+05	1.6452E-183	1.7716E-81	1.3811E-223	0.0000E+00
	Mean value	**0.0000E+00**	4.9159E-31	6.6117E+04	5.4841E-185	1.3149E-82	5.2984E-225	**0.0000E+00**
	Ranking	1	6	7	4	5	3	1
F4	Optimal value	0.0000E+00	3.7028E-25	1.0335E+01	1.6630E-112	6.5323E-44	2.9568E-121	0.0000E+00
	Worst value	0.0000E+00	2.4937E-22	8.7564E+01	5.4242E-88	1.7391E-42	3.1038E-119	0.0000E+00
	Mean value	**0.0000E+00**	6.6951E-23	5.7219E+01	1.8081E-89	5.5798E-43	1.0369E-119	**0.0000E+00**
	Ranking	1	6	7	4	5	3	1
F5	Optimal value	9.9963E-30	7.2091E-02	2.7728E+01	2.6948E+01	6.5759E-03	2.8496E+01	2.0736E+01
	Worst value	9.0000E+00	2.8977E+01	2.8784E+01	2.7906E+01	2.8705E+01	2.8809E+01	2.3765E+01
	Mean value	**9.1487E-01**	2.3869E+01	2.8424E+01	2.7481E+01	1.1391E+00	2.8723E+01	2.2540E+01
	Ranking	1	4	6	5	2	7	3
F6	Optimal value	7.6409E-01	4.6424E-03	2.2133E-01	7.5193E-02	8.9967E-05	6.1511E-01	1.1161E-04
	Worst value	2.5000E+00	7.3794E+00	1.4996E+00	2.8020E-01	5.8065E-03	3.3575E+00	3.1061E-04

Continued on next page

Test functions	Statistical value	RSA	SO	WOA	IWOA	MSWOA	MWOA	RLVWOA
	Mean value	2.1743E+00	4.4086E+00	7.9930E-01	1.4466E-01	1.7009E-03	1.3112E+00	**1.8950E-04**
	Ranking	6	7	4	3	2	5	1
F7	Optimal value	1.0110E-05	6.2095E-05	1.4119E-04	7.0811E-06	1.7239E-07	3.7601E-06	2.2875E-06
	Worst value	6.2927E-04	1.1254E-03	1.5898E-02	6.5371E-04	1.0118E-03	4.5397E-04	7.1596E-04
	Mean value	1.5921E-04	4.5255E-04	6.0256E-03	1.8576E-04	2.4335E-04	**1.0148E-04**	1.0593E-04
	Ranking	3	6	7	4	5	1	2
F8	Optimal value	0.0000E+00	7.2065E-08	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
	Worst value	0.0000E+00	5.3511E+01	5.6843E-14	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
	Mean value	**0.0000E+00**	1.1389E+01	3.7896E-15	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	Ranking	1	7	6	1	1	1	1
F9	Optimal value	4.4409E-16	3.9968E-15	4.4409E-16	4.4409E-16	4.4409E-16	4.4409E-16	4.4409E-16
	Worst value	4.4409E-16	3.9968E-15	1.4655E-14	4.4409E-16	4.4409E-16	4.4409E-16	4.4409E-16
	Mean value	**4.4409E-16**	3.9968E-15	5.5363E-15	**4.4409E-16**	**4.4409E-16**	**4.4409E-16**	**4.4409E-16**
	Ranking	1	6	7	1	1	1	1
F10	Optimal value	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
	Worst value	0.0000E+00	0.0000E+00	1.1102E-16	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
	Mean value	**0.0000E+00**	**0.0000E+00**	3.7007E-18	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**	**0.0000E+00**
	Ranking	1	1	7	1	1	1	1
F11	Optimal value	2.2137E-01	3.6574E-05	1.2078E-02	2.8349E-03	1.7022E-05	2.0941E-02	3.1591E-06
	Worst value	2.6507E+00	1.6389E+00	1.0512E-01	1.4901E-02	8.5385E-04	2.4330E-01	9.2504E-06
	Mean value	8.9943E-01	2.6277E-01	3.4739E-02	8.4345E-03	2.5276E-04	9.2786E-02	**5.9142E-06**
	Ranking	7	6	4	3	2	5	1
F12	Optimal value	2.2126E-32	1.8618E-04	1.4283E-01	5.6834E-02	4.2010E-05	3.6647E-01	4.5057E-05

Continued on next page

Test functions	Statistical value	RSA	SO	WOA	IWOA	MSWOA	MWOA	RLVWOA
	Worst value	4.8668E-31	2.9991E+00	1.6724E+00	4.2908E-01	5.9192E-02	1.2855E+00	1.3373E-04
	Mean value	**1.4798E-31**	1.1562E+00	7.3644E-01	2.0608E-01	1.1832E-02	7.1408E-01	8.3747E-05
	Ranking	1	7	6	4	3	5	2
F13	Optimal value	1.9928E+00	9.9800E-01	9.9800E-01	9.9800E-01	9.9800E-01	9.9801E-01	9.9800E-01
	Worst value	1.2671E+01	5.9288E+00	1.0763E+01	1.0763E+01	5.9288E+00	1.2671E+01	9.9800E-01
	Mean value	4.5384E+00	1.2678E+00	4.1024E+00	2.4429E+00	2.0692E+00	8.3489E+00	**9.9800E-01**
	Ranking	6	2	5	4	3	7	1
F14	Optimal value	4.8810E-04	3.0861E-04	3.2189E-04	3.1045E-04	3.1720E-04	3.6499E-04	3.0749E-04
	Worst value	7.8796E-03	1.6236E-03	1.7046E-02	7.2231E-04	2.2520E-03	1.7160E-03	3.0767E-04
	Mean value	2.8231E-03	6.6713E-04	1.2266E-03	4.2590E-04	8.5203E-04	7.0167E-04	**3.0755E-04**
	Ranking	7	3	6	2	5	4	1
F15	Optimal value	-1.0316E+00	-1.0316E+00	-1.0316E+00	-1.0316E+00	-1.0316E+00	-1.0316E+00	-1.0316E+00
	Worst value	-1.0284E+00	-1.0316E+00	-1.0316E+00	-1.0305E+00	-1.0159E+00	-8.6893E-01	-1.0316E+00
	Mean value	-1.0308E+00	**-1.0316E+00**	**-1.0316E+00**	-1.0314E+00	-1.0309E+00	-9.8871E-01	**-1.0316E+00**
	Ranking	6	1	1	4	5	7	1
F16	Optimal value	3.0000E+00	3.0000E+00	3.0000E+00	3.0000E+00	3.0000E+00	3.0033E+00	3.0000E+00
	Worst value	3.0038E+01	3.0000E+00	3.0023E+00	3.0171E+00	3.0304E+01	3.5457E+01	3.0000E+00
	Mean value	3.9019E+00	**3.0000E+00**	3.0002E+00	3.0026E+00	4.0003E+00	1.0559E+01	**3.0000E+00**
	Ranking	5	1	3	4	6	7	1
F17	Optimal value	3.8469E+00	3.8628E+00	3.8628E+00	3.8628E+00	3.8627E+00	3.8589E+00	3.8628E+00

Continued on next page

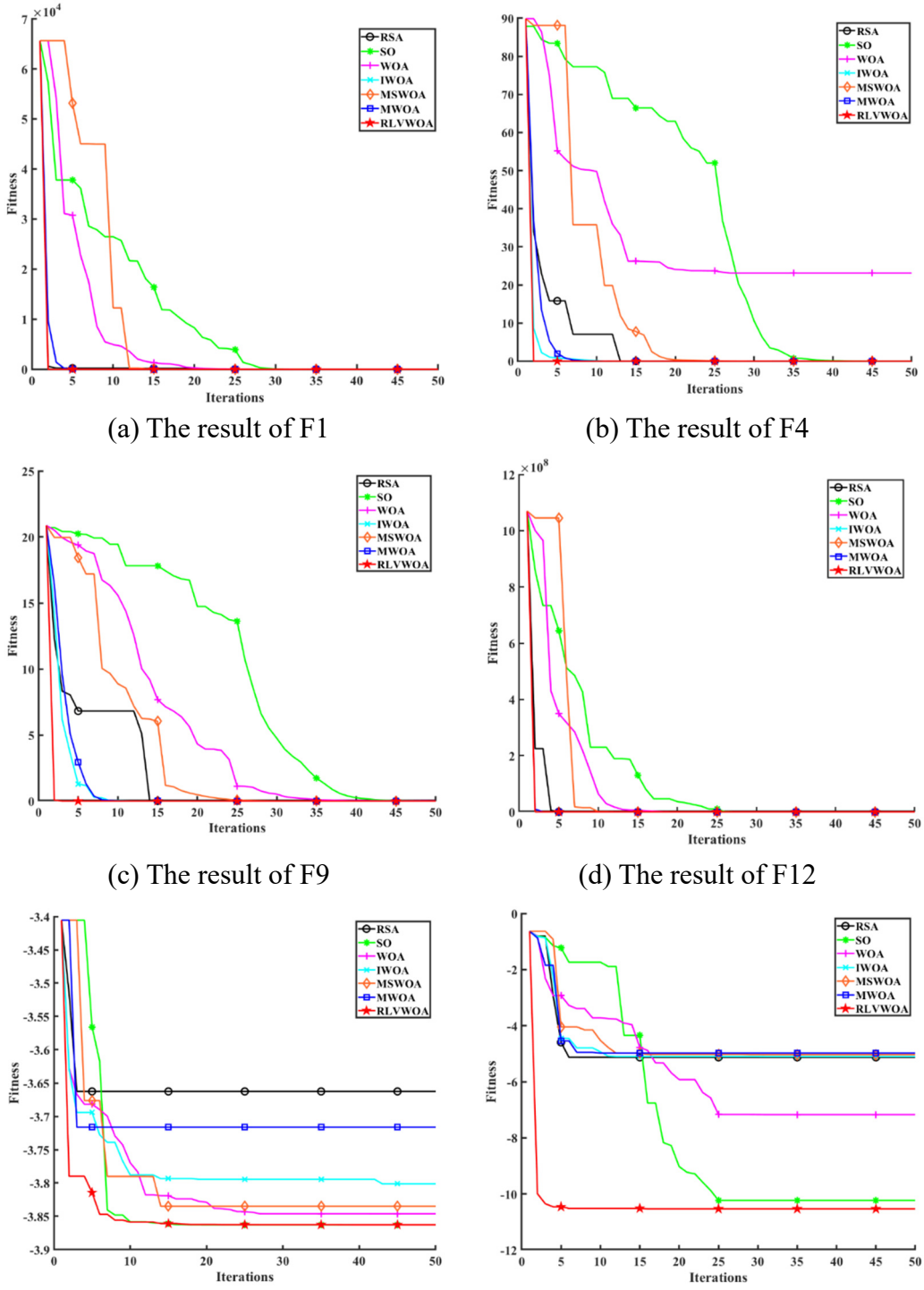
Test functions	Statistical value	RSA	SO	WOA	IWOA	MSWOA	MWOA	RLVWOA
	Worst value	-3.5478E+00	-3.8628E+00	-3.7247E+00	-3.8137E+00	-3.8557E+00	-3.6536E+00	-3.8628E+00
	Mean value	-3.7333E+00	**-3.8628E+00**	-3.8442E+00	-3.8547E+00	-3.8601E+00	-3.7973E+00	**-3.8628E+00**
	Ranking	7	1	5	4	3	6	1
F18	Optimal value	-3.0478E+00	-3.3220E+00	-3.3211E+00	-3.3209E+00	-3.1360E+00	-3.1751E+00	-3.3220E+00
	Worst value	-1.8405E+00	-3.2031E+00	-2.6337E+00	-3.0740E+00	-2.9754E+00	-2.2517E+00	-3.3218E+00
	Mean value	-2.6054E+00	-3.3101E+00	-3.1984E+00	-3.2688E+00	-3.0962E+00	-2.8468E+00	**-3.3219E+00**
	Ranking	7	2	4	3	5	6	1
F19	Optimal value	-5.0552E+00	-1.01532E+01	-1.01524E+01	-1.01413E+01	-1.01138E+01	-9.2944E+00	-1.01532E+01
	Worst value	-5.0552E+00	-5.31481E+00	-5.04992E+00	-5.03683E+00	-2.61764E+00	-3.9793E+00	-1.01526E+01
	Mean value	-5.0552E+00	-9.77132E+00	-8.03439E+00	-6.67967E+00	-7.17146E+00	-5.1171E+00	**-1.01531E+01**
	Ranking	7	2	3	5	4	6	1
F20	Optimal value	-5.5591E+00	-1.05364E+01	-1.05345E+01	-1.04907E+01	-1.05289E+01	-7.1999E+00	-1.05363E+01
	Worst value	-3.7343E+00	-6.54959E+00	-1.67406E+00	-5.11187E+00	-1.78908E+00	-3.3409E+00	-1.05360E+01
	Mean value	-5.0964E+00	-1.02217E+01	-5.79879E+00	-6.34792E+00	-6.78931E+00	-4.6944E+00	**-1.05363E+01**
	Ranking	6	2	5	4	3	7	1
Average Ranking	—	3.8	4.1	5.3	3.45	3.55	4.4	1.2

Table 5. Running time of each algorithm.

Test functions	Time (s)						
	RSA	SO	WOA	IWOA	MSWOA	MWOA	RLVWOA
F1	9.818	1.752	2.041	3.072	4.564	3.546	14.695
F2	9.979	1.794	2.019	3.082	4.819	3.658	14.800
F3	10.546	3.332	3.490	7.229	7.320	5.454	20.511
F4	9.971	1.764	1.980	4.029	5.785	3.511	13.810
F5	10.075	1.924	2.319	3.392	5.076	4.121	15.345
F6	10.033	1.828	2.006	4.006	4.953	3.115	13.733
F7	9.987	2.478	2.834	5.110	5.794	3.865	16.430
F8	9.278	1.935	1.994	3.122	4.686	3.003	14.002
F9	9.729	1.866	2.208	3.412	4.812	2.975	15.349
F10	9.752	2.039	2.311	3.431	4.945	3.125	15.627
F11	10.455	3.903	4.068	9.475	8.052	5.587	29.518
F12	10.441	3.993	4.385	9.268	8.362	5.425	31.904
F13	9.827	1.906	4.268	12.157	10.361	7.023	36.858
F14	8.317	1.803	1.913	2.936	4.805	3.239	13.105
F15	7.399	1.749	1.822	2.901	4.773	3.137	13.653
F16	6.383	1.906	1.748	2.680	4.868	3.093	13.247
F17	7.347	1.703	1.905	3.211	4.890	3.268	15.777
F18	8.991	2.081	2.052	3.558	4.909	3.259	16.998
F19	9.334	2.042	2.100	3.489	5.234	3.404	16.633
F20	9.502	2.280	2.231	4.294	5.536	3.678	17.137
Total	187.164	37.697	45.673	90.462	104.786	77.486	332.294

According to the results from Table 4 and Table 5, although RLVWOA exhibits longer running time and fails to converge to the theoretical optimal values on some test functions such as F5, F6 and F9, it demonstrates relatively better convergence accuracy and attains the best mean ranking. For the sake of brevity, this paper only presents the convergence figures of F1, F4, F9, F12, F17 and F20, which include two unimodal test functions, two multimodal test functions and two fixed-dimension test functions. To make these figures more intuitive, we use the same initial population and set $t_{max} = 50$.

As shown in the Figure 3, although RLVWOA requires more running time, it demonstrates better convergence performance, enabling faster convergence compared to other algorithms. Therefore, it fully demonstrates that the RLVWOA, which combines the reinforcement learning algorithm and the VNS algorithm, can solve the unstable optimization performance of the WOA well.



(a) The result of F1

(b) The result of F4

(c) The result of F9

(d) The result of F12

(e) The result of F17

(f) The result of F20

Figure 3. Convergence capability comparison figures.

5. Simulation

5.1. Model establishment

The problem of time-optimal trajectory planning for manipulators can be likened to solving a

constrained optimization problem to find the minimum value. It heavily relies on the algorithm's search capability to navigate through the vast solution space and identify the optimal trajectory that minimizes the completion time while satisfying the constraints imposed by the manipulator's dynamics and task requirements. The efficiency of the optimization algorithm plays a pivotal role in achieving time-optimal solutions, ensuring the manipulator's swift and precise execution of tasks in various industrial applications. To facilitate better understanding and avoid the need to learn about different manipulator structures, this paper chooses to use the common PUMA560 manipulator as the model for trajectory planning. Its modified D-H parameters and kinematic constraints are shown in Tables 6 and 7, respectively.

Table 6. The modified D-H parameters of PUMA560.

Joint	θ_i (°)	α_{i-1} (°)	α_{i-1} (mm)	d_i (mm)	Variable Range (°)
1	90	0	0	0	-160~160
2	0	-90	0	149.09	-225~45
3	-90	0	431.8	0	-45~225
4	0	-90	20.32	433.07	-110~170
5	0	90	0	0	-100~100
6	0	-90	0	0	-266~266

Table 7. The kinematic constraints parameters of PUMA560.

Joint	1	2	3	4	5	6
Angular velocity V(°/s)	100	95	100	150	130	110
Angular acceleration A(°/s^2)	45	40	75	70	90	80
Angular jerk J(°/s^3)	60	60	55	70	75	70

The goal of this paper is to find the time-optimal trajectory for the manipulator. Therefore, the fitness function of the algorithm is defined as depicted in Eq (31):

$$f = \sum_{i=1}^{10} \left(t_i - t_{i-1} \right) \tag{31}$$

In the Eq (31), f denotes the overall execution duration of the manipulator and t_i represents the time to reach the i-th path point.

Based on the data in Tables 6 and 7, The selected path points that satisfy the kinematic constraints are shown in Table 8. Based on these path points, by substituting it into Eq (1) and Eq (31), the time-optimal trajectory planning for the manipulator is conducted.

Table 8. The table of path points.

Path points	Position of each joint (°)					
	1	2	3	4	5	6
1	10	–10	–30	–25	20	0
2	22	–30	–10	–45	0	15
3	45	–45	10	–60	–20	30
4	65	–60	30	–50	–40	40
5	40	–70	40	–40	–55	55
6	25	–45	60	–20	–40	70
7	15	–25	60	0	–25	90
8	0	0	75	5	–30	100
9	–10	10	85	15	–45	105
10	–20	25	90	20	–60	120

5.2. Trajectory planning

The time-optimal trajectory planning for the manipulator using the RLVWOA is conducted. In order to further validate the performance of the algorithm, the RSA, SO, WOA, IWOA, MSWOA and MWOA algorithms are also utilized for the trajectory planning of the manipulator. Each algorithm utilizes the same number of iterations $T = 300$ and population size $N = 30$, while other specific parameters are taken from the data presented in Table 3. The specific results are shown in Table 9, where the results obtained by the RLVWOA are highlighted in bold. The convergence comparison figure is shown in Figure 4.

Table 9. The table of path points.

Path points	Time (s)						
	RSA	SO	WOA	IWOA	MSWOA	MWOA	RLVWOA
1	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	1.844	1.762	2.097	1.771	1.868	1.982	1.694
3	2.840	2.563	4.599	2.656	2.685	3.249	2.423
4	4.266	3.650	5.451	4.419	3.733	4.765	3.722
5	5.437	4.947	8.860	5.310	5.346	6.687	4.885
6	6.923	6.142	10.253	8.119	6.682	8.175	6.195
7	8.013	7.405	12.134	9.429	7.995	9.526	7.195
8	9.145	8.497	13.608	10.941	9.055	11.065	8.350
9	9.994	9.108	14.375	12.691	10.133	11.899	9.008
10	11.899	10.956	17.763	14.513	12.167	13.798	10.767

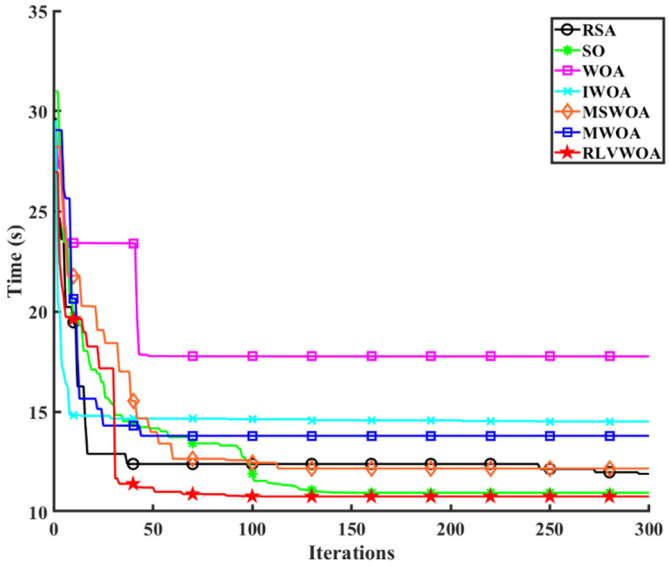
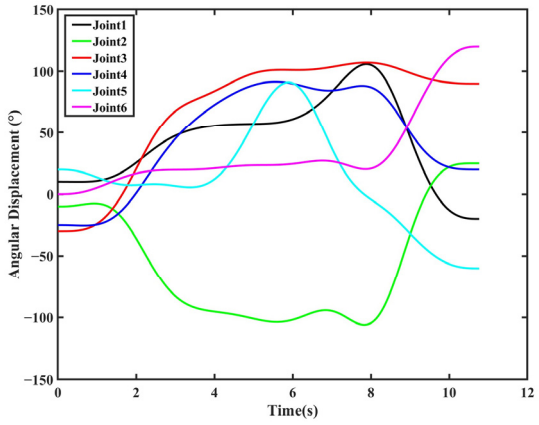
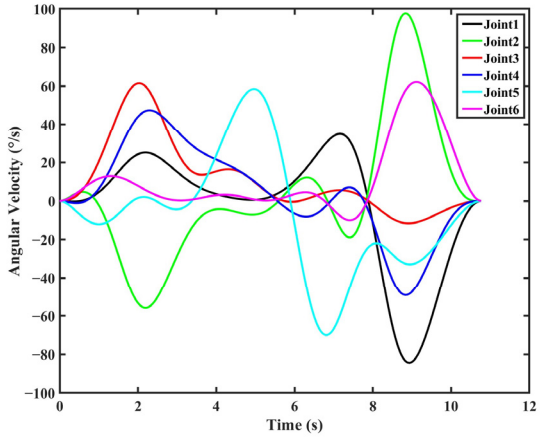


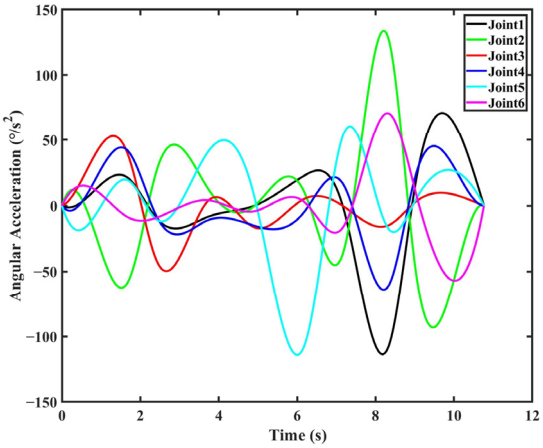
Figure 4. The convergence comparison figure.



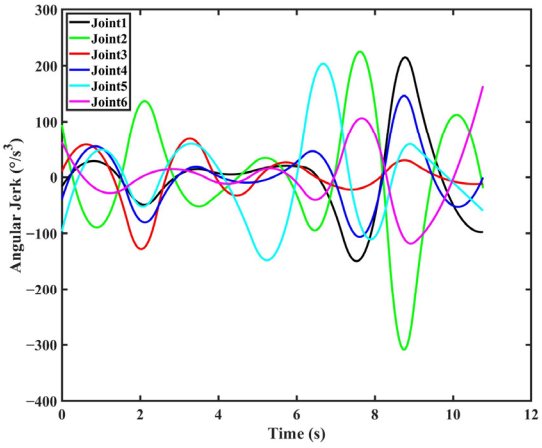
(a) Angular displacement curve



(b) Angular velocity curve



(e) Angular velocity curve



(d) Angular jerk curve

Figure 5. Trajectory curve graphs.

Based on the data in Table 9 and Figure 4, it can be observed that the RLVWOA achieves superior results in terms of obtaining the shortest running trajectory for the manipulator. The RLVWOA, compared to the standard WOA, achieves a reduction of 39.39% and compared to other improved WOAs achieve a minimum reduction of 11.51%. Additionally, the RLVWOA demonstrates faster convergence speed, further validating its superior search capability as proposed in this paper. The trajectory planning plot is depicted in the Figure 5.

According to Figure 5, all curves are uniform, continuous and devoid of any abrupt changes. Furthermore, they adhere to the kinematic constraints outlined in Table 7. Therefore, it can be concluded that the RLVWOA is capable of obtaining a superior time-optimal trajectory.

6. Conclusions

This paper proposes an improved RLVWOA that combines reinforcement learning to enhance global search capability and introduces VNS algorithm to improve local search capability. A comparison with other algorithms demonstrates the superior performance of RLVWOA. Subsequently, the RLVWOA is employed in conjunction with the quintic NURBS for trajectory planning of the manipulator. The result is a smooth, uniform and continuous trajectory, which outperforms the results obtained by other optimization algorithms in terms of reduced the manipulator operation time.

The main contribution of this paper is the proposal of an improved RLVWOA that exhibits superior search capability compared to other algorithms. However, there are still some issues that need to be addressed in future work. This paper only combines reinforcement learning algorithms. It would be worthwhile to explore the use of deep learning algorithms such as deep q-learning network (DQN) algorithm, deep deterministic policy gradient (DDPG) algorithm and twin delayed deep deterministic policy gradient (TD-3) algorithm as potential alternatives. Additionally, while the introduction of the VNS algorithm has improved the search capability, it has also increased the algorithm's runtime significantly. Future work could involve redesigning more suitable neighborhoods or adding termination thresholds to control the runtime of the algorithm.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This study was supported by the National Natural Science Foundation of China (grant number 62272336) and the key R&D projects of Shanxi province (grant number 202102150401009).

Conflict of interest

The authors declare there is no conflict of interest.

References

1. L. Wang, Q. Wu, F. Lin, S. Li, D. Chen, A new trajectory-planning beetle swarm optimization algorithm for trajectory planning of robot manipulators, *IEEE Access*, **7** (2019), 154331–154345. https://doi.org/10.1109/ACCESS.2019.2949271
2. H. Zhao, B. Zhang, L. Yang, J. Sun, Z. Gao, Obstacle avoidance and near time-optimal trajectory planning of a robotic manipulator based on an improved whale optimization algorithm, *Arab. J. Sci. Eng.*, **47** (2022), 16421–16438. https://doi.org/10.1007/s13369-022-06926-y
3. S. Jia, J. Shan, Finite-time trajectory tracking control of space manipulator under actuator saturation, *IEEE Trans. Ind. Electron.*, **67** (2020), 2086–2096. https://doi.org/10.1109/TIE.2019.2902789
4. T. Zhang, M. Zhang, Y. Zou, Time-optimal and smooth trajectory planning for robot manipulators, *Int. J. Control Autom. Syst.*, **19** (2021), 521–531. https://doi.org/10.1007/s12555-019-0703-3
5. A. Abe, Minimum energy trajectory planning method for robot manipulator mounted on flexible base, in *2013 9th Asian Control Conference (ASCC)*, (2013), 1–7. https://doi.org/10.1109/ASCC.2013.6606088
6. D. Chen, Y. Zhang, Minimum jerk norm scheme applied to obstacle avoidance of redundant robot arm with jerk bounded and feedback control, *IET Control Theory Appl.*, **10** (2016), 1896–1903. https://doi.org/10.1049/iet-cta.2016.0220
7. X. Zhang, G. Shi, Multi-objective optimal trajectory planning for manipulators in the pre-sence of obstacles, *Robotica*, **40** (2021), 1–19. https://doi.org/10.1017/S0263574721000886
8. J. Liu, H. Wang, X. Li, K. Chen, C. Li, Robotic arm trajectory optimization based on multiverse algorithm, *Math. Biosci. Eng.*, **20** (2023), 2776–2792. https://doi.org/10.3934/mbe.2023130
9. L. Zhang, Y. Wang, X. Zhao, P. Zhao, L. He, Time-optimal trajectory planning of serial manipulator based on adaptive cuckoo search algorithm, *J. Mech. Sci. Technol.*, **35** (2021), 3171–3181. https://doi.org/10.1007/s12206-021-0638-5
10. X. Gao, Y. Mu, Y. Gao, Optimal trajectory planning for robotic manipulators using impr-oved teaching-learning-based optimization algorithm, *Ind. Robot*, **43** (2016), 308–316. https://doi.org/10.1108/IR-08-2015-0167
11. Y. Du, Y. Chen, Time optimal trajectory planning algorithm for robotic manipulator based on locally chaotic particle swarm optimization, *Chin. J. Electron.*, **31** (2022), 906–914. https://doi.org/10.1049/cje.2021.00.373
12. X. Zhang, F. Xiao, X. Tong, J. Yun, Y. Liu, Y. Sun, et al., Time optimal trajectory planning based on improved sparrow search algorithm, *Front. Bioeng. Biotechnol.*, **10** (2022).https://doi.org/10.3389/fbioe.2022.852408
13. L. Sun, J. Huang, J. Xu, Y. Ma, Feature selection based on adaptive whale optimization algorithm and fault-tolerance neighborhood rough sets, *Pattern Recognit. Artif. Intell.*, **35** (2022), 150–165. https://doi.org/10.16451/j.cnki.issn1003-6059.202202006
14. W. Yang, K. Xia, S. Fan, L. Wang, T. Li, J. Zhang, et al., A multi-strategy whale optimization algorithm and its application, *Eng. Appl. Artif. Intell.*, **108** (2022), 104558. https://doi.org/10.1016/j.engappai.2021.104558
15. J. Anitha, S. I. A. Pandian, S. A. Agnes, An efficient multilevel color image thresholding based on modified whale optimization algorithm, *Expert Syst. Appl.*, **178** (2021), 115003. https://doi.org/10.1016/j.eswa.2021.115003

16. L. Piegl, W. Tiller, *The NURBS Book*, Springer Science & Business Media, (1996).
17. X. Li, H. Zhao, X. He, H. Ding, A novel cartesian trajectory planning method by using triple NURBS curves for industrial robots, *Robot Comput. Integr. Manuf.*, **83** (2023), 102576. https://doi.org/10.1016/j.rcim.2023.102576
18. W. Ma, T. Hu, C. Zhang, T. Zhang, A robot motion position and posture control method for freeform surface laser treatment based on NURBS interpolation, *Robot Comput. Integr. Manuf.*, **83** (2023), 102547. https://doi.org/10.1016/j.rcim.2023.102547
19. S. Li, X. Zhang, Research on planning and optimization of trajectory for underwater vision welding robot, *Array*, **16** (2022), 100253. https://doi.org/10.1016/j.array.2022.100253
20. W. Wang, Q. Wang, R. Zhong, L. Chen, X. Shi, Stacking sequence optimization of arbitrary quadrilateral laminated plates for maximum fundamental frequency by hybrid whale optimization algorithm, *Compos. Struct.*, **310** (2023), 116764. https://doi.org/10.1016/j.compstruct.2023.116764
21. L. P. Kaelbling, M. L. Littman, A. W. Moore, Reinforcement learning: A Survey, *J. Artif. Intell. Res.*, **4** (1996), 237–285. https://doi.org/10.1613/jair.301
22. M. Fayyazi, M. Abdoos, D. Phan, M. Golafrouz, M. Jalili, R. N. Jazar, et al., Real-time self-adaptive Q-learning controller for energy management of conventional autonomous vehicles, *Expert Syst. Appl.*, **222** (2023), 119770. https://doi.org/10.1016/j.eswa.2023.119770
23. R. Chen, B. Yang, S. Li, S. Wang, A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem, *Comput. Ind. Eng.*, **149** (2020), 106778. https://doi.org/10.1016/j.cie.2020.106778
24. V. Helder, T. Filomena, L. Ferreira, G. Kirch, Application of the VNS heuristic for feat-ure selection in credit scoring problems, *Mach. Learn. Appl.*, **9** (2022), 100349. https://doi.org/10.1016/j.mlwa.2022.100349
25. R. S. Sutton, A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, (2018).
26. S. Chakraborty, S. Sharma, A. K. Saha, A. Saha, A novel improved whale optimization algorithm to solve numerical optimization and real-world applications, *Artif. Intell. Rev.*, **55** (2022), 4605–4716. https://doi.org/10.1007/s10462-021-10114-z
27. L. Abualigah, M. A. Elaziz, P. Sumari, Z. W. Geem, A. H. Gandomi, Reptile search algorithm (RSA): A nature-inspired meta-heuristic optimizer, *Expert Syst. Appl.*, **191** (2022), 116158. https://doi.org/10.1016/j.eswa.2021.116158
28. F. A. Hashim, A. G. Hussien, Snake optimizer: A novel meta-heuristic optimization algorithm, *Knowl. Based Syst.*, **242** (2022), 108320. https://doi.org/10.1016/j.knosys.2022.108320

AIMS Press