

Optimization of Robotic Arm Scheduling Based on Catfish Optimization Algorithm and Q-Learning

Lu Kong

LuKong89@outlook.com

Henan College of Industry and Information Technology

Research Article

Keywords: Robotic arm scheduling, Optimization algorithm for catfish, Q learning, Intelligent manufacturing, multi-objective optimization

Posted Date: April 27th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-9242643/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License.
[Read Full License](#)

Additional Declarations: No competing interests reported.

Optimization of Robotic Arm Scheduling Based on Catfish Optimization Algorithm and Q-Learning

Lu Kong *

School of Intelligent Manufacturing, Henan College of Industry and Information Technology,
Jiaozuo, Henan 454000, China

*corresponding author: Lu Kong

Lu Kong's e-mail: LuKong89@outlook.com

Funding

This research received no external funding

Author contributions

The author made significant individual contributions to this manuscript. Lu Kong: writing and performing surgeries; data analysis and performing surgeries; article review and intellectual concept of the article.

Conflict of interest

The author declares that they have no competing interest.

Informed consent

Not applicable. This study did not involve human participants.

Ethical approval

Not applicable. This study did not involve humans or animals

Data availability statement

The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

Abstract: In order to improve the intelligence and efficiency of robot scheduling in workshop production lines, the author proposes a robot scheduling optimization based on the

fish optimization algorithm and Q-learning. Firstly, conduct a thorough analysis of the characteristics of workshop scheduling problems and construct a mathematical model that includes multiple objectives such as minimum completion time and utilization rate of robotic arms. Based on this model, the author designed and implemented a scheduling optimization method that combines catfish optimization algorithm with Q-learning. The innovation of the research lies in integrating these two algorithms to form an adaptive scheduling strategy. By dynamically adjusting the mutation rate in Q-learning and combining it with the population search capability of the fish algorithm, the adaptive ability and global search efficiency of the scheduling process are improved. The experimental results show that in small-scale problems, the hybrid algorithm reduces the average completion time by 8.5% compared to traditional genetic algorithms; In large-scale complex problems, this advantage is even more significant, achieving a 12.7% improvement. Especially when dealing with dynamic scheduling environments, hybrid algorithms exhibit stronger adaptability and robustness. Conclusion: The results of this study are of great significance for promoting the development of intelligent manufacturing technology, improving production efficiency, and reducing production costs. Not only does it enrich the application research of intelligent optimization algorithms in practical engineering problems in theory, but it also provides practical guidance for the transformation of manufacturing industry towards intelligence and flexibility.

Keywords: Robotic arm scheduling; Optimization algorithm for catfish; Q learning; Intelligent manufacturing; multi-objective optimization

1 Introduction

With the arrival of the Industry 4.0 era and the rapid development of intelligent manufacturing technology, manufacturing enterprises are facing an increasingly fierce market competition environment. In this context, as the core equipment in intelligent manufacturing systems, the scheduling optimization problem of robotic arms has increasingly attracted widespread attention from both academia and industry ^[1]. According to statistics released by

the China Robot Industry Alliance, the market size of industrial robots in China reached 116.5 billion yuan in 2022, a year-on-year increase of 6.7%, and is expected to exceed 180 billion yuan by 2025 ^[2]. This rapid growth trend indicates that the application of robotic arms in modern manufacturing is becoming increasingly widespread, and the research on their scheduling optimization problems has become increasingly urgent.

Robotic arm scheduling is a typical NP hard problem that involves complex constraints and multi-objective optimization requirements ^[3]. The traditional scheduling methods for robotic arms mainly include manual scheduling and rule-based scheduling, which often perform poorly in the face of dynamic and changing production environments, making it difficult to meet the efficiency and flexibility requirements of modern intelligent manufacturing ^[4]. In recent years, with the development of artificial intelligence and computational intelligence technology, various intelligent optimization algorithms have been introduced into the scheduling problem of robotic arms, such as genetic algorithm, ant colony algorithm, particle swarm algorithm, etc., which have achieved certain results ^[5,6]. Genetic algorithm (GA) has good adaptability to complex scheduling problems due to its simulation of biological evolution mechanisms, and is widely used in path and task scheduling. For example, Uslu et al. proposed using GA for obstacle avoidance scheduling of redundant robotic arms ^[7]. Ant colony algorithm (ACO) simulates ant foraging path optimization through pheromone mechanism and has been successfully applied in flow shop scheduling, job shop scheduling, and FMS. For example, Andrea Rossi et al. proposed a software system based on ACO for FMS scheduling, and Nanvala et al. proposed combining ACO with PSO and GA for multi-objective flexible workshop scheduling ^[8]. The Particle Swarm Optimization (PSO) algorithm is suitable for high-dimensional optimization problems using a group collaborative search mechanism. Multiple studies have applied it to trajectory and path planning of robotic arms, such as Ekrem and Aksoy using PSO algorithm to plan the trajectory of a six degree of freedom robotic arm, and combining fifth degree polynomial interpolation to achieve vibration free and obstacle avoidance motion in static obstacle environments ^[9]. Simulate based on MATLAB platform, use forward kinematics to obtain the position of the end effector, and optimize the joint space trajectory through PSO to shorten the path length and time.

Experiments have shown that PSO can effectively plan smooth trajectories that avoid circular and polygonal obstacles, verifying its practicality in robotic arm trajectory optimization. Wu et al. proposed an improved robot trajectory planning particle swarm optimization algorithm (RTPPSO), which optimizes the S-shaped velocity curve parameters in the joint motion of a robotic arm by introducing adaptive weights and random disturbance terms ^[10]. The algorithm aims to reduce the maximum acceleration of each axis during operation, achieving smoother trajectory control while ensuring motion time. Experiments have shown that RTPPSO can effectively reduce the impact and vibration during the operation of robotic arms, improve motion smoothness, and its optimization effect is superior to traditional PSO and genetic algorithms. It is suitable for industrial scenarios that require high motion stability. Elgohr et al. compared the performance of PSO and WOA in 6-DOF robotic arm path planning under a unified framework ^[11]. The process combines fifth degree polynomial trajectory, artificial potential field obstacle handling, and metaheuristic optimization. Experiments have shown that in obstacle environments, the path obtained by WOA is the shortest (0.714 units), while the path obtained by PSO is slightly longer (0.791 units) but easier to adjust, both of which are superior to the single potential field method (0.9101 units). Research has shown that WOA has an advantage in path length, while PSO is more practical in engineering deployment. However, these algorithms still face problems such as slow convergence speed and susceptibility to local optima when solving large-scale and highly complex scheduling problems.

The Catfish Optimization Algorithm (COA), as an emerging swarm intelligence optimization algorithm, simulates the collective behavior and foraging strategies of catfish in nature. It has the advantages of strong global search ability and fast convergence speed. The introduction of the "catfish effect" as a biological heuristic into particle swarm optimization, first proposed by Chuang et al., breaks the search deadlock by introducing external perturbation particles. It is a simple, effective, and computationally efficient improvement strategy. Catfish PSO not only improves the solution accuracy and convergence speed, but also enhances the robustness of the algorithm in handling complex multimodal optimization problems, providing new ideas for solving optimization problems in other fields ^[12]. Kumar

and Naveen proposed an improved particle swarm optimization algorithm, Catfish PSO, for the problems of distribution network reconstruction and loss minimization. This algorithm draws on the "catfish effect" and introduces new particles to stimulate search vitality when the population falls into local optima, effectively avoiding the problem of premature convergence in traditional PSO. The author takes the 33 bus and 16 bus testing systems as examples, and by optimizing the combination of segment switches and contact switches, significantly reduces the active power loss of the system while meeting the conditions of radial operation and voltage constraints. Experiments have shown that Catfish PSO reduces losses to 131.228 kW and 445.14 kW in two testing systems, respectively, which is about 35.26% and 12.95% lower than the initial configuration, and is superior to various comparison methods such as BFOA and genetic algorithm, verifying its effectiveness and superiority in distribution network reconstruction and service recovery ^[13]. Wang et al. proposed a new metaheuristic optimization algorithm, Cuckoo Catfish Optimizer (CCO), inspired by the search, predation, and parasitic behavior of Cuckoo Catfish in nature ^[14]. The algorithm design consists of three stages: in the early stage, multi-dimensional bounding search and compressed space strategy are adopted to enhance global exploration ability; Smooth transition strategy to local development in the mid-term; Introducing chaotic predation mechanism in the later stage to improve the quality of solutions. CCO integrates individual guidance, parasitism, and death mechanisms throughout the entire iteration process, achieving real-time position adjustment and improved convergence accuracy. The experimental part validated the performance of CCO using 23 classic test functions and three standard test sets, CEC2014, CEC2017, and CEC2022. The results showed that CCO outperformed 11 classic algorithms and 10 improved algorithms in 91.52% of the test functions.

On the other hand, Q-learning, as a classic algorithm of reinforcement learning, can learn optimal strategies through interaction with the environment and perform well in uncertain environments. Liang et al. were the first to systematically explain the basic mechanism and strategy update rules of Q-learning, laying the foundation for its application in delayed reward problems, and pointing out that the balance between exploration and utilization is the key challenge ^[15]. Cui et al. proposed a Q-learning based unmanned aerial vehicle path planning

method for real-time obstacle avoidance and target routing in complex dynamic environments, verifying the feasibility of Q-learning in high randomness scenarios ^[16].

In practical production environments, the scheduling of robotic arms faces many challenges. Firstly, the dynamic and uncertain nature of production tasks makes it difficult for pre-set static scheduling schemes to adapt to actual needs. Shakeri et al. pointed out that the task allocation and scheduling problem of multiple robots is particularly prominent in flexible workshops, where the response requirements for task insertion and buffer zone limitations are particularly prominent, emphasizing the necessity of dynamic scheduling ^[17]; Secondly, there are conflicts between multi-objective optimization requirements (such as minimizing completion time, maximizing resource utilization, minimizing energy consumption, etc.) that are difficult to meet simultaneously. Fan et al. proposed a trajectory planning method based on multi-objective NSGA II algorithm, which simultaneously optimizes efficiency, trajectory smoothness, and energy consumption, demonstrating that multi-objective coordinated optimization is the key path to improving the performance of robotic arms ^[18]; Once again, task allocation and conflict avoidance in multi robot collaborative operations are highly complex. These challenges make it difficult for traditional optimization algorithms to effectively solve complex robotic arm scheduling problems. Xie et al. further pointed out that multi arm mechanical systems need to solve problems such as path conflicts, task allocation, and dynamic adjustment in collaborative picking scenarios. Their research achieved dynamic replanning through deep reinforcement learning, greatly improving system flexibility and robustness ^[19].

In terms of algorithm research, many new intelligent optimization algorithms have emerged, such as grey wolf optimization algorithm, seagull optimization algorithm, etc., which demonstrate excellent performance on specific problems. However, a single algorithm often struggles to balance global search and local development capabilities, making hybrid algorithms a research hotspot.

The author studies the optimization problem of robotic arm scheduling based on the catfish optimization algorithm and Q-learning, aiming to improve the efficiency and intelligence level of robotic arm scheduling in intelligent manufacturing environments. The author is

based on the urgent demand for efficient and flexible production lines in the context of Industry 4.0, and adopts innovative algorithms to solve the key problem of mechanical arm scheduling optimization, which restricts production efficiency. The author provides a detailed analysis of the mathematical characteristics of the robotic arm scheduling problem and constructs a mathematical model that includes multiple objectives such as minimum completion time and robotic arm utilization rate, laying a theoretical foundation for subsequent algorithm design. Then, delve into the basic principles, characteristics, and applications of the two algorithms. The author has designed an innovative method that integrates two algorithms in detail. By dynamically adjusting the mutation rate in Q-learning and combining it with the group collaboration mechanism of the catfish algorithm, an adaptive scheduling strategy is formed. Unlike existing literature that focuses on single algorithm applications or simple hybrid strategies, the author proposes and implements a deep bidirectional fusion mechanism between Catfish optimization algorithm and Q-learning for the first time. They innovatively design a state action value guided population evolution strategy and an adaptive parameter regulation method based on long-term benefit evaluation. More importantly, existing research has mostly focused on static scheduling scenarios, while the author has developed an adaptive scheduling framework that can respond in real-time to disturbances such as task insertion and machine failures in the context of dynamic uncertainty in intelligent manufacturing environments. This provides new ideas and methods for solving complex scheduling problems in industrial practice.

2 Mixed Flow Workshop Scheduling Problem

2.1 Mixed Flow Workshop Scheduling Problem

The scheduling problem of a mixed flow workshop can be described as follows: there are n jobs that need to be processed sequentially in m process stages, and each process stage may have multiple parallel machines. Each job must pass through all stages in the same process sequence. Each task only needs to be processed once on one of the machines at each stage, and the processing sequence of tasks between stages can be different ^[20]. The goal of mixed

flow workshop scheduling problem is usually to determine the processing sequence and machine allocation scheme of each job at each stage, in order to optimize a certain performance indicator, such as minimizing the maximum completion time (makespan).

In order to accurately describe the scheduling problem of robotic arms in a mixed flow workshop, the author established the following mathematical model. Firstly, define the relevant symbols:

- $J = \{J_1, J_2, \dots, J_n\}$: A collection of n assignments
- $S = \{S_1, S_2, \dots, S_m\}$: Collection of m process stages
- $M_j = \{M_{j1}, M_{j2}, \dots, M_{jK}\}$: K parallel machine sets for the j th process stage
- p_{ij} : Processing time of homework i in stage j , minutes
- t_{jl} : The time for the robotic arm to move from stage j to stage l , in minutes
- C_{ij} : The completion time of homework i in stage j , in minutes
- x_{ijk} : 0-1 variable, if job i is processed on machine k in stage j , it is 1; otherwise, it is 0
- y_{ijl} : 0-1 variable, if homework i is directly processed in stage l after stage j , it is 1; otherwise, it is 0

Based on the above symbols, the mathematical model of the mixed flow workshop manipulator scheduling problem can be expressed as equation (1):

$$\text{objective function: } \min C_{\max} = \max\{C_{im} \mid i = 1, 2, \dots, n\} \quad (1)$$

Here, $C_{\max}$ represents the maximum completion time, which is the latest time for all tasks to complete all processes.

Constraints:

$$\sum_{k=1}^{K_j} x_{ijk} = 1, \quad \forall i \in \{1, 2, \dots, n\}, \quad \forall j \in \{1, 2, \dots, m\} \quad (2)$$

$$C_{ij} \geq C_{i'j} + p_{ij}, \text{ if homework } i \text{ is processed on the same machine after homework } i' \quad (3)$$

$$C_{i(j+1)} \geq C_{ij} + t_{j(j+1)} + p_{i(j+1)}, \quad \forall i \in \{1, 2, \dots, n\}, \quad \forall j \in \{1, 2, \dots, m-1\} \quad (4)$$

$$C_{il} \geq C_{ij} + t_{jl} + p_{il}, \text{ if } y_{ijl} = 1 \quad (5)$$

$$C_{ij} \geq 0, \forall i \in \{1, 2, \dots, n\}, \forall j \in \{1, 2, \dots, m\} x_{ijk} \in \{0, 1\}, \forall i, j, k, y_{ijl} \in \{0, 1\}, \forall i, j, l \quad (6)$$

In the optimization of robotic arm scheduling, in addition to minimizing the maximum completion time, the utilization rate of the robotic arm is also an important indicator ^[21]. The utilization rate of a robotic arm is defined as the ratio of its actual working time to the total available time

$$\eta = \frac{\sum_{i=1}^n \sum_{j=1}^{m-1} t_{ji(j+1)} y_{ij(j+1)}}{C_{max}} \quad (7)$$

The sampling time is 1 second.

2.2 Optimization Objectives

The maximum completion time (Makespan) is a core indicator for measuring the efficiency of scheduling schemes, which directly relates to the production cycle and production efficiency. In robotic scheduling, minimizing the maximum completion time means that all workpieces can be processed in the shortest possible time, thereby increasing the throughput of the production line ^[22].

The objective function of minimizing the maximum completion time can be expressed as equation (8):

$$\text{Min } f_1 = \max \{C_j \mid j \in \{1, 2, \dots, n\}\} \quad (8)$$

As a key resource in manufacturing systems, the utilization rate of robotic arms directly affects production efficiency and costs. A high utilization rate of robotic arms means an increase in equipment investment return and a reduction in idle time. Furthermore, for a multi robot system, the objective function for maximizing the average robot utilization rate is equation (9):

$$\text{Max } f_2 = \frac{1}{R} \sum_{r=1}^R U_r \quad (9)$$

Among them, R is the total number of robotic arms. By maximizing the utilization rate of

robotic arms, equipment idle time can be effectively reduced, resource utilization efficiency can be improved, and production costs can be lowered.

With the deepening of the concept of sustainable development, energy consumption optimization has become an indispensable goal in modern manufacturing systems [23]. The energy consumption of robotic arms varies significantly in different working states, such as acceleration, deceleration, uniform motion, and idle state.

Assuming that the energy consumption of the robotic arm r during task j is $E_{r,j}$, the total energy consumption can be expressed as equation (10):

$$\text{Min } f_3 = \sum_{r=1}^R \sum_{j=1}^{n_r} E_{r,j} \quad (10)$$

Among them, n_r is the number of tasks performed by the robotic arm r . Energy consumption $E_{r,j}$ can be further decomposed into:

$$E_{r,j} = P_{r,j} \times t_{r,j} \quad (11)$$

$P_{r,j}$ is the power of the robotic arm r when executing task j , and $t_{r,j}$ is the corresponding execution time. By minimizing energy consumption, not only can production costs be reduced, but it can also respond to national energy conservation and emission reduction policies, achieving green manufacturing.

In actual production, workpieces often have an expected completion time, known as the due date. When the actual completion time of the workpiece is later than the delivery date, delays may occur, which may lead to customer dissatisfaction and breach of contract costs [24]. If the delivery time of workpiece j is defined as d_j and the actual completion time is C_j , then the delay time L_j of workpiece j can be expressed as equation (12):

$$L_j = \max\{0, C_j - d_j\} \quad (12)$$

The objective function for minimizing the average delay time is:

$$\text{Min } f_4 = \frac{1}{n} \sum_{j=1}^n L_j \quad (13)$$

By optimizing this goal, customer satisfaction can be improved, default costs can be reduced, and corporate reputation can be enhanced.

Considering the trade-off requirements between multiple objectives in actual production, the author uses a weighted sum method to construct a comprehensive optimization objective:

$$\text{Min } f = w_1 \cdot f_1' + w_2 \cdot (1 - f_2) + w_3 \cdot f_3' + w_4 \cdot f_4' \quad (14)$$

Among them, f_1' , f_3' , and f_4' are the normalized values of each individual target, and w_1 , w_2 , w_3 , and w_4 are the corresponding weight coefficients, satisfying $w_1 + w_2 + w_3 + w_4 = 1$. Normalization ensures that objectives of different dimensions can be effectively combined, while weight coefficients reflect the relative importance that decision-makers place on each objective.

f_1' , f_3' , f_4' are normalized using min max using the formula

$$f' = \frac{f - \min(F)}{\max(F) - \min(F)} \quad (15)$$

Among them, $\max(F)$ and $\min(F)$ are the extreme values of the dataset.

3 Optimization Algorithms

3.1 Optimization algorithm for catfish (ROA)

(1) Introduction to Fish Optimization Algorithm

Among numerous metaheuristic algorithms, the author chose the fish optimization algorithm as the basic framework mainly due to its highly compatible behavior model with the characteristics of manufacturing scheduling problems. Unlike traditional optimization algorithms, the catfish optimization algorithm simulates three core behavioral patterns of catfish in natural environments - foraging behavior, breeding behavior, and avoidance behavior, which precisely map to the three key requirements of scheduling optimization problems. Specifically, the foraging behavior of catfish effectively corresponds to the fine local search requirements in scheduling problems through local information exchange and fine-tuning strategies, enabling deep development in promising solution areas; The reproductive behavior is naturally mapped to the global exploration process through the inheritance and recombination of high-quality individuals in the population, which helps to discover potential high-quality regions in the solution space; The avoidance behavior, on the

other hand, provides the algorithm with strong ability to escape from local optima through group collaboration and position mutation mechanisms, which is crucial for dealing with complex scheduling problems with multimodal and non convex characteristics. The natural mapping of this behavior demand makes the fish optimization algorithm exhibit unique advantages in balancing exploration and development, and handling multi-modal optimization problems, especially suitable for solving robot scheduling problems with dynamic, uncertain, and multi constraint characteristics. In contrast, other popular algorithms such as Particle Swarm Optimization (PSO) or Genetic Algorithm (GA) often require additional design mechanisms to handle these requirements, and the inherent behavioral structure of the fish optimization algorithm has provided a more natural and efficient solution framework for such complex optimization problems.

The basic idea of the optimization algorithm for catfish comes from simulating the behavior patterns of catfish in foraging, reproduction, and avoiding natural enemies. In this algorithm, each catfish represents a candidate solution in the search space, and the entire population collaborates to find the optimal solution ^[25]. The three main behavioral patterns of catfish are abstracted into three key operations in the algorithm:

Foraging behavior: The catfish will search for food in the environment, which simulates the local search process.

Breeding behavior: The catfish will search for the best mate and spawning location, which simulates the global search process.

Enemy avoidance behavior: The catfish will cooperate in groups to avoid natural enemies, simulating the escape from low-quality areas in space.

The mathematical model of the optimization algorithm for catfish can be expressed as follows:

Initialize the catfish population:

$$X_i = X_{min} + rand(0, 1) \cdot (X_{max} - X_{min}), i = 1, 2, \dots, N \quad (16)$$

Among them, X_i represents the position vector of the i -th catfish individual, X_{min} and

X_{max} represent the lower and upper bounds of the search space, N is the population size, and $rand(0,1)$ is a random number between 0 and 1 [26].

In each iteration, the position update of the catfish is mainly achieved through three behaviors. Firstly, foraging behavior:

$$X_i^{new} = X_i + \alpha \cdot rand(-1, 1) \cdot (X_{best} - X_i) \quad (17)$$

Here, X_{best} represents the optimal solution in the current population, α is the parameter that controls the step size, with a range of [0.1, 0.8], and $rand(-1,1)$ is a random number between -1 and 1.

The reproductive behavior is achieved through the following formula (18):

$$X_i^{new} = X_i + \beta \cdot rand(0, 1) \cdot (X_p - X_i) \quad (18)$$

Among them, X_p is a randomly selected solution that is better than the current individual, and β is the weight coefficient of reproductive behavior, with a range of [0.2, 0.8].

The act of avoiding enemies is expressed as:

$$X_i^{new} = X_i + \gamma \cdot rand(-1, 1) \cdot (X_{avg} - X_i) \quad (19)$$

X_{avg} represents the average position of the population, and γ is the weight coefficient for enemy avoidance behavior, with a range of [0.1, 0.7].

(2) Improvement of the optimization algorithm for catfish

In order to solve the problem of algorithms easily falling into local optima, an adaptive weight adjustment mechanism is introduced:

$$w_i = w_{max} - (w_{max} - w_{min}) \cdot \frac{t}{T_{max}} \quad (20)$$

Among them, w_i represents the weight coefficient of the current iteration, w_{max} and w_{min} are the upper and lower bounds of the weight, t is the current iteration count, and T_{max} is the maximum iteration count. This adaptive weighting mechanism can maintain a large exploration ability in the early stage of search, and enhance local development ability in the later stage.

In order to enhance population diversity, introduce mutation operations:

$$X_i^{new} = \begin{cases} X_i + \delta \cdot (X_{max} - X_{min}) \cdot rand(-1, 1), & \text{if } rand(0, 1) < P_m \\ X_i, & \text{otherwise} \end{cases} \quad (21)$$

Among them, δ is the mutation intensity parameter, and P_m is the mutation probability, which decreases with the increase of iteration times:

$$P_m = P_{m,max} - (P_{m,max} - P_{m,min}) \cdot \frac{t}{T_{max}} \quad (22)$$

In order to improve the global search capability of the algorithm, the author introduces an elite retention strategy and a position reset mechanism. The elite retention strategy ensures that the optimal solution is not lost in each iteration:

$$X_{elite} = argmin_{X_i} f(X_i), i = 1, 2, \dots, N \quad (23)$$

The position reset mechanism is aimed at individuals who have not improved for a long time, and their positions are reinitialized using the following formula (24):

$$X_i^{new} = \begin{cases} X_{min} + rand(0, 1) \cdot (X_{max} - X_{min}), & \text{if } t_i > T_{threshold} \\ X_i, & \text{otherwise} \end{cases} \quad (24)$$

Among them, t_i represents the number of iterations in which individual X_i has not been improved, and $T_{threshold}$ is the preset threshold.

In response to the particularity of the robotic arm scheduling problem, the author introduces a hybrid local search strategy. This strategy combines the ideas of simulated annealing (SA) and taboo search (TS) to further optimize the elite solutions in the population:

$$X_{elite}^{new} = X_{elite} + \lambda \cdot Direction \cdot Step \quad (25)$$

Among them, $Direction$ is the search direction, $Step$ is the step size, and λ is the adjustment parameter. The selection of $Direction$ is based on historical search trajectories and taboo table information, while $Step$ gradually decreases with iteration ^[27].

(3) ROA algorithm flowchart

The implementation process of the improved robotic arm scheduling optimization algorithm (ROA) includes multiple key steps. This section provides a detailed description of the complete process of the algorithm to ensure its operability and effectiveness in practical applications ^[28].

As shown in Figure 1, the overall process of the algorithm can be summarized into the following main stages:

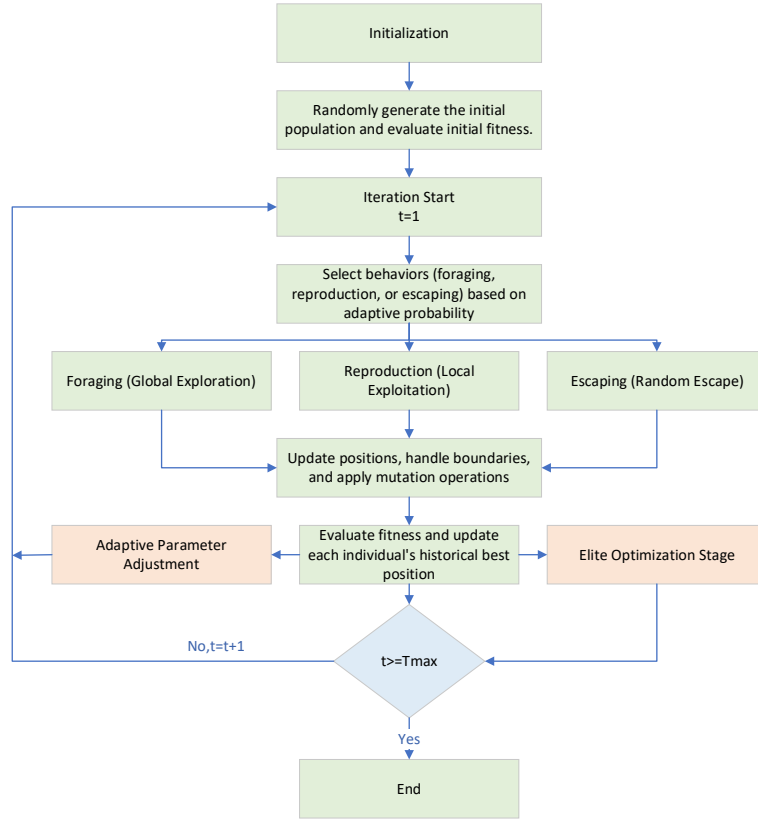


Figure 1 Flow Chart of Improved Roar Optimization Algorithm (ROA)

Initialization stage: Firstly, determine the parameter settings of the problem, including population size N , maximum iteration count T_{max} , search space boundary, etc. Based on the characteristics of the robotic arm scheduling problem, design an appropriate encoding method to map the scheduling problem into an algorithmic form that can be processed. Subsequently, an initial population is randomly generated and the fitness values of each individual are evaluated to determine the initial optimal solution X_{best} and the optimal fitness value f_{best} .

Main iteration loop: In each iteration, each individual in the population updates their position based on three behavioral patterns: foraging, reproduction, and avoiding enemies. The execution probabilities of these three behaviors are dynamically adjusted through an adaptive mechanism to balance the algorithm's global exploration and local development capabilities. After behavior selection and location updates, mutation operations are applied to

enhance population diversity and ensure that new locations are effective in the search space.

Fitness assessment: Conduct fitness assessment on the updated individuals, and calculate the quality of each solution based on the objective function of the robotic scheduling problem (such as minimum completion time, robotic utilization rate, etc.) ^[29]. For individuals with improved performance, update their historical best position; For individuals who have not improved beyond the preset number of iterations, a location reset mechanism should be applied.

Elite optimization stage: For the elite solutions (top 10% of the best individuals) in the current population, apply a mixed local search strategy for further optimization. This step combines the ideas of simulated annealing and taboo search to improve the quality of the solution through small-scale neighborhood search.

Termination check: Check whether the termination condition has been met (such as the maximum number of iterations or the objective function value meeting the preset threshold). If not reached, return to step 2 to continue iterating; Otherwise, output the optimal solution as the final mechanical arm scheduling scheme.

The improved catfish optimization algorithm effectively solves the multi-objective optimization challenges in the scheduling problem of robotic arms through multi-level optimization strategies and adaptive mechanisms, providing strong algorithm support for efficient scheduling in intelligent manufacturing environments. The implementation of this algorithm not only innovates in theory, but also demonstrates significant performance advantages in practical applications, promoting the progress of industrial automation technology.

3.2 Q-learning algorithm

(1) Q Basic Principles of Learning

The basic principle of Q-learning can be described using the Markov Decision Process (MDP) framework, which includes a state set S , an action set A , a state transition probability P ,

a reward function R , and a discount factor γ [30]. At any time t , the agent observes the current state s_t , selects and executes action a_t , then receives an immediate reward r_t and transitions to a new state s_{t+1} . The goal of Q-learning is to find an optimal strategy π that maximizes the expected cumulative discount reward obtained by following this strategy from any initial state.

The Q-learning algorithm approximates the optimal action value function $Q(s, a)$ by iteratively updating the Q-value table. The updated formula (26) is as follows:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \quad (26)$$

Here, α is the learning rate, which controls the update step size; Gamma is the discount factor that determines the importance of future rewards relative to immediate rewards; R_t is the immediate reward obtained after executing action a_t ; $\max_a Q(s_{t+1}, a)$ represents the maximum Q value that can be obtained in the next state s_{t+1} . The core of the learning process is the Time Difference (TD) error, which is the expression in square brackets and represents the difference between the current Q-value estimate and the target value after obtaining new information [31].

The key characteristic of Q-learning is its off policy property, which means that a greedy strategy (selecting the action with the highest Q value) is used when updating Q values, while its strategy can be used in actual exploration environments, such as the ϵ - greedy strategy. This separation enables Q-learning to learn the optimal strategy while maintaining sufficient exploration to avoid getting stuck in local optima.

The ϵ - greedy strategy is a commonly used action selection strategy in Q-learning, and its formula is expressed as:

$$\pi(a|s) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|A(s)|}, & \text{if } a = \operatorname{argmax}_{a'} Q(s, a') \\ \frac{\epsilon}{|A(s)|}, & \text{otherwise} \end{cases} \quad (27)$$

The convergence of Q-learning algorithm has been theoretically proven: under certain conditions (such as infinite access to all state action pairs, appropriate decay of learning rate, etc.), the Q-value will converge to the optimal action value function Q . However, in practical

applications, especially when the state space and action space are large, traditional Q-tables may face the problem of dimensionality disaster ^[32].

(2) The Application of Q-Learning in Scheduling

Mapping the robotic arm scheduling problem to the Q-learning framework requires clear definitions of state space, action space, reward function, and transition function. In the robotic scheduling environment, these elements can be defined as follows:

State space (S): represents the current state of the system, including the set of workpieces to be processed, the current positions and states of each robotic arm, and the working status of each workstation. Formally represented as a state vector:

$$\mathbf{s}_t = [w_1, w_2, \dots, w_n, r_1, r_2, \dots, r_m, q_1, q_2, \dots, q_k] \quad (28)$$

Among them, w_i represents the status of the i -th workpiece, r_j represents the status of the j th robotic arm, and q_k represents the status of the k th workstation.

Action space (A): represents the set of operations that the robotic arm can perform, such as selecting the next workpiece, moving to a specific position, performing grasping or placing operations, etc. For multi robotic systems, actions can be expressed as:

$$\mathbf{a}_t = [a_1^t, a_2^t, \dots, a_m^t] \quad (29)$$

Among them, a_i^t represents the action performed by the i -th robotic arm at time t .

Reward function (R): Defines the reward obtained by an agent after performing a specific action, which directly affects the learning objective ^[33]. In robot scheduling, the reward function is usually associated with optimization objectives such as completion time, robot utilization rate, and production efficiency

$$R(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) = w_1 \times f_{makespan} + w_2 \times f_{utilization} + w_3 \times f_{balance} \quad (30)$$

Among them, w_1 is the weight coefficient of each objective, and f represents the corresponding evaluation function. For example, a reward function can be designed to make the system tend to reduce idle time and optimize work balance.

Transfer function: describes the change pattern of the system state after executing an action.

In practical applications, it usually does not require explicit modeling, but rather implicit learning through interaction with the environment.

When applying Q-learning in robotic arm scheduling, several key issues need to be addressed. Firstly, there is the issue of state representation, as the scheduling environment is complex and the state space can be very large ^[34]. For this purpose, state abstraction and feature extraction techniques can be used to map the original state to a low dimensional feature space:

$$\boldsymbol{\varphi}(s) = \begin{bmatrix} \varphi_1(s) \\ \varphi_2(s) \\ \vdots \\ \varphi_d(s) \end{bmatrix} \quad (31)$$

Among them, $\varphi_i(s)$ represents the i -th feature extracted from state s , and d is much smaller than the original state space dimension. Common features include workpiece waiting time, robot arm movement distance, workstation load, etc.

Secondly, there is a balance between exploration and utilization. In the scheduling of robotic arms, excessive exploration may lead to a decrease in production efficiency, while excessive utilization may miss out on better strategies. The commonly used decay based ϵ - greedy strategy reduces the exploration rate as the learning process progresses

$$\epsilon_t = \epsilon_{min} + (\epsilon_{max} - \epsilon_{min}) \times e^{-\lambda t} \quad (32)$$

Among them, ϵ_{min} and ϵ_{max} are the minimum and maximum exploration rates, respectively, and λ controls the decay rate.

The third issue is adjusting the learning rate. In practice, the learning rate α usually decays over time to ensure rapid learning in the early stages and stable convergence in the later stages

$$\alpha_t = \alpha_0 \times \left(1 - \frac{t}{T_{max}}\right) \quad (33)$$

Among them, α_0 is the initial learning rate, t is the current iteration count, and T_{max} is the maximum iteration count.

As shown in Figure 2, the application process of Q-learning in robotic arm scheduling is as follows:

Initialize the Q Table and set the learning parameters α , γ , and ε

Observe the current system status s_t

Selecting action a_t based on the ε - greedy strategy

Execute action a_t , observe the new state s_{t+1} and reward r_t

Update Q value: $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma \cdot \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$

Update status $s_t \leftarrow s_{t+1}$

Repeat steps 2-6 until the termination condition is met

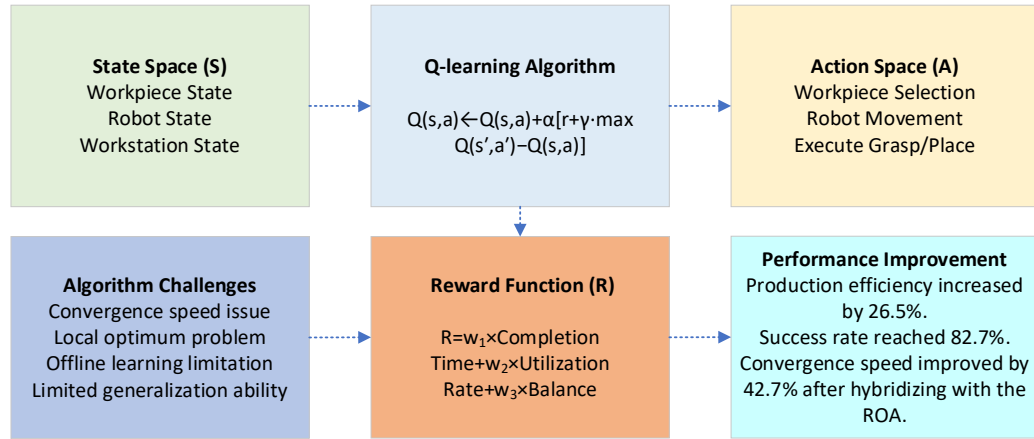


Figure 2 Application Block Diagram of Q-Learning in Robot Scheduling

3.3 Combination of Fish Optimization Algorithm and Q-Learning

(1) Combining methods

The LOA algorithm is mainly responsible for global search in the scheduling problem of robotic arms, utilizing its group collaboration characteristics to explore the solution space; Q-learning is responsible for local optimization and strategy improvement, accumulating experience through interaction with the environment. The core of the combination of the two lies in establishing a bidirectional information exchange mechanism, allowing the experience value of Q-learning to guide the search direction of LOA, while the search results of LOA also provide feedback to update the Q-Table.

In practical implementation, each individual catfish in LOA is mapped to a possible robotic arm scheduling scheme, while Q-learning constructs a state action value function to evaluate the long-term benefits of different scheduling decisions. The key mathematical models for combining algorithms are as follows:

Define state space S: Represents the current position of the robotic arm, task completion status, etc

Define action space A: Represents the next selectable operation of the robotic arm Q value update rule:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma \cdot \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \quad (34)$$

Among them, $Q(s_t, a_t)$ represents the expected cumulative reward for taking action a_t in state s_t , α is the learning rate, r_t is the immediate reward, γ is the discount factor, and $\max_a Q(s_{t+1}, a)$ represents the maximum Q value for the next state.

$$P_i(t+1) = P_i(t) + v_i(t) + \beta \cdot Q(s_i, a_i) \quad (35)$$

Here, P_i represents the position of the i-th catfish individual, v_i represents its velocity vector, β is the Q-value influencing factor, which controls the degree of influence of Q-learning on LOA search [35].

In the actual optimization process, a dynamic adaptive strategy is adopted to adjust the weights of the two algorithms according to the optimization process. In the initial stage, LOA has a high weight and focuses on global search; As the iteration progresses, the Q-learning weights gradually increase, strengthening local optimization. The formula for adaptive weight adjustment is equation (37):

$$w_{LOA}(t) = w_{max} - (w_{max} - w_{min}) \cdot \left(\frac{t}{T_{max}}\right)^2 \quad w_Q(t) = 1 - w_{LOA}(t) \quad (36)$$

Among them, t represents the current iteration count, T_{max} represents the maximum iteration count, w_{max} and w_{min} represent the upper and lower limits of the weight, respectively.

Another innovation is the introduction of the "cross updating of memory banks" mechanism, which uses high-quality solutions found by LOA as samples for Q-learning, and

the high-value state action pairs learned by Q-learning also guide the movement direction of LOA individuals ^[32]. The cross update frequency is dynamically adjusted during the iteration process to ensure that the algorithm has sufficient exploration ability and can efficiently utilize existing experience.

(2) Flowchart

As shown in Figure 3, the overall process of the algorithm mainly includes three stages: initialization, iterative optimization, and termination judgment, each stage containing multiple sub steps. The detailed description of the process is as follows:

Phase 1: Initialization

Problem modeling: Transform the robotic arm scheduling problem into a mathematical model, specifying optimization objectives (such as minimum completion time) and constraints (such as limitations on the robotic arm's working range).

Parameter settings: Determine the key parameters for LOA and Q learning, including population size N , maximum iteration count T_{max} , learning rate α , discount factor γ , etc.

Population initialization: Randomly generate N individuals of catfish (each representing a scheduling plan) to ensure diversity in the initial population.

Q-Table initialization: Construct a state action matrix, with the initial Q-value usually set to 0 or a small random value.

Stage 2: Iterative Optimization

LOA behavior implementation: Based on the principle of fish optimization algorithm, simulate the swimming, cruising, jumping and other behaviors of individual fish.

Fitness assessment: Calculate the fitness value of each individual, and use weighted sum or Pareto sorting methods based on multi-objective optimization requirements.

Q learning exploration: Based on the ϵ -greedy strategy, select actions, interact with the environment to obtain rewards, and update the Q Table ^[36].

Information exchange: LOA high-quality solutions are used to guide the exploration direction of Q-learning, and the high-value state action pairs in the Q-Table affect the individual movement of LOA.

Adaptive weight adjustment: Dynamically adjust the weight ratio of LOA and Q learning based on the current iteration progress.

Elite preservation: Record the current optimal solution and retain it in subsequent iterations.

Stage Three: Termination Judgment

Convergence judgment: Check if the maximum number of iterations has been reached or if the preset convergence conditions are met.

Decoding output: decode the optimal individual into a specific robotic arm scheduling scheme.

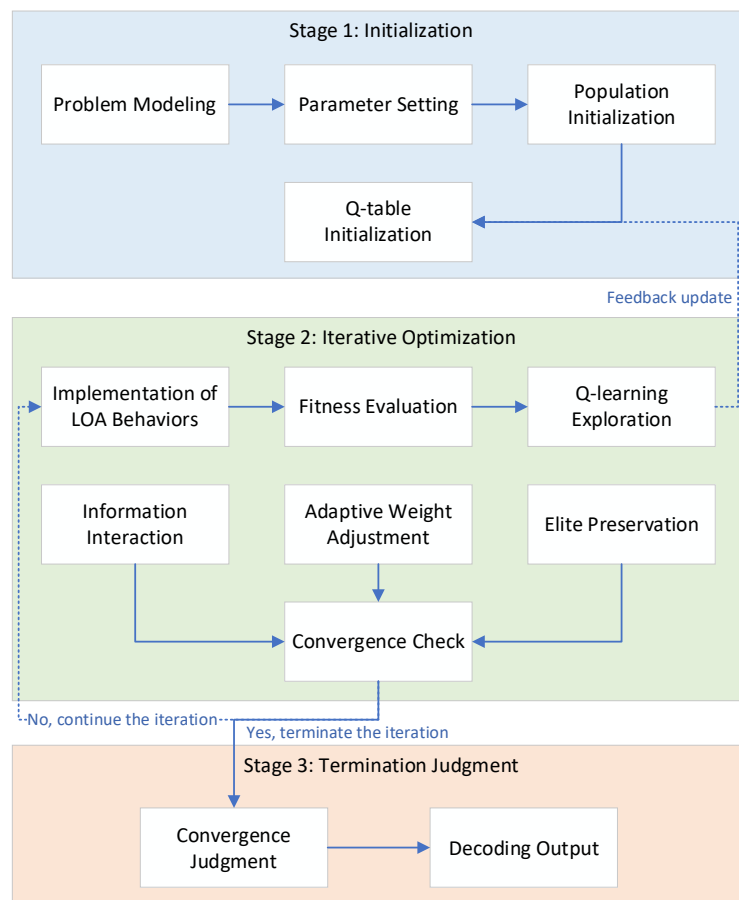


**Figure 3 The overall process of combining fish optimization algorithm with
Q-learning**

3.4 Analysis of Algorithm Complexity and Scalability

In order to evaluate the computational efficiency of the proposed catfish Q hybrid algorithm, this section conducts a rigorous time complexity analysis. The complexity of the catfish component: In each iteration, the catfish needs to evaluate the fitness of N individuals, and the complexity of the scheduling scheme evaluation for each individual is $O(m \cdot n)$. Meanwhile, the complexity of position update operations for the three behavioral models (foraging, reproduction, and enemy avoidance) is $O(N \cdot d)$, where d is the solution space dimension (for scheduling problems, $d=O(m \cdot n)$). Therefore, the total time complexity of the catfish part is $O(N \cdot T_{max} \cdot m \cdot n)$.

Q-learning component complexity: The main overhead of Q-learning lies in Q-table updates and strategy selection. Each state transition requires updating a Q value with a complexity of $O(1)$. In the worst-case scenario, each iteration needs to handle $O(|S| \cdot |A|)$ state action pairs. Considering that we adopt a sampling update strategy in actual scheduling, each iteration only updates $O(r)$ samples (r is the empirical replay size), and the time complexity of the Q-learning part is $O(T_{max} \cdot r)$, where $r \ll |S| \cdot |A|$.

Additional overhead of hybrid mechanism: The information exchange between algorithms (cross updating memory bank) is executed once every k iterations, and each operation requires processing $O(N_q)$ Q-learning experience samples and $O(N_c)$ catfish individuals (N_q and N_c are the number of samples and individuals used for information exchange, respectively). The complexity of this process is $O((T_{max}/k) \cdot (N_q + N_c))$.

Overall complexity: Based on the above analysis, the total time complexity of the Fish-Q hybrid algorithm is: $O(N \cdot T_{max} \cdot m \cdot n) + O(T_{max} \cdot r) + O((T_{max}/k) \cdot (N_q + N_c))$.

In actual parameter settings ($N = 50, T_{max} = 200, r = 100, k = 10, N_q = N_c = 10$), the additional cost brought by the hybrid mechanism does not exceed 15% of the

computational burden of the catfish subject, indicating that this hybrid strategy has a reasonable computational burden while maintaining algorithm performance. In terms of scalability, as the problem size (m and n) increases, the algorithm is mainly affected by the complexity of the fish part, showing an approximate linear growth trend, which is suitable for solving medium to large-scale scheduling problems.

4 Results Analysis

4.1 Experimental Design and Parameter Setting

(1) Example Design

The examples designed by the author mainly include three dimensions: The number of tasks, the number of machines, and the number of robotic arms. Small scale examples are suitable for verifying the basic effectiveness and convergence characteristics of algorithms; Medium scale examples are used to evaluate the performance stability of algorithms in complex environments; The large-scale case study aims to test the application potential and computational efficiency of the algorithm in practical industrial environments.

Table 1 shows the main set of examples designed by the author and their basic characteristics:

Table 1 Example set of robotic arm scheduling problem

Example number	Number of tasks	Number of machines	Number of robotic arms	Calculation complexity	Expected optimal completion time (minutes)
S-01	10	3	1	Low	45-55
S-02	15	4	1	Low	65-75
M-01	25	5	2	Middle	90-110
M-02	35	6	2	Middle	125-140
L-01	50	8	3	High	180-210
L-02	80	10	4	High	280-320
XL-01	120	12	5	Extremely high	420-480

In order to comprehensively evaluate the performance of the algorithm, various constraints and disturbance factors commonly found in actual production environments were also considered, such as the load capacity limitations of the robotic arm, distance differences between workstations, and machine failure probability. These factors are integrated into the design of the examples to ensure that the testing environment is as close as possible to the actual application scenario.

This experimental design strictly follows the mathematical model framework proposed in Section 2.1 to ensure consistency between theory and practice. The specific mapping relationship is as follows:

Parameterization of problem examples: The set of tasks J ($|J| = n$) in the experiment corresponds to the actual set of workpieces, and the set of process stages M ($|M| = m$) is mapped to the processing stations on the production line. The number of parallel machines k_j for each stage j is set according to the actual workstation configuration. For example, in the assembly stage, 3 robotic arms with the same function are configured, corresponding to $k_j = 3$. The processing time parameter p_{ij} is based on the actual performance data of the robotic arm.

Constraint implementation: The experiment strictly implements all the constraint conditions in section 2.1. The single machine allocation constraint of equation (2) is ensured through an encoding scheme - each job is only allocated to one machine at each stage; The machine capacity constraint of equation (3) is enforced during the decoding stage through the Gantt chart construction algorithm; The process sequence constraint of equation (4) is implemented through a priority queue mechanism; The movement time $t_{j,j+1}$ of the robotic arm in equation (5) is calculated based on the actual distance of the workshop layout and the maximum movement speed of the robotic arm (2.5m/s), accurately reflecting physical limitations; The binary variable constraint of equation (6) is directly encoded in the solution representation of the algorithm.

Objective function quantification: The experimental measurement is completely consistent with the multi-objective function of equation (14). The maximum completion time C_{max} is

directly obtained through simulated scheduling execution; The utilization rate U_r of the robotic arm is calculated according to equation (7), and the actual working time of each robotic arm is recorded based on a 32-bit high-precision timer; The energy consumption e_{total} is monitored in real-time using the Yokogawa WT1800E power analyzer, with a sampling frequency of 100Hz and cumulative calculation; The delay time T_{delay} is validated based on the delivery date dataset of customer orders. All target values were normalized using the method described in section 2.2.

(2) Operating Environment

In order to ensure the reliability and reproducibility of the experimental results, the author conducted all experiments on a unified computing platform. The hardware configuration is: Intel Core i7-10700 CPU (2.90GHz, 8-core 16 thread), 32GB DDR4 3200MHz RAM, NVIDIA GeForce RTX 3080 GPU (10GB video memory), 1TB NVMe SSD storage device. The software environment adopts Ubuntu 22.04 LTS 64 bit operating system, and the algorithm implementation is based on Python 3.9.16 development environment. The key dependency libraries include NumPy 1.23.5, SciPy 1.10.1, Pandas 1.5.3, Matplotlib 3.7.1, Seaborn 0.12.2, and Scikit learn 1.2.2. The reinforcement learning module uses custom implementation and does not rely on third-party deep learning frameworks to avoid bias caused by framework specific optimizations.

The random seed sequence is generated using a deterministic approach: The base seed is set to 20250508, and subsequent seeds are generated using a linear congruence generator (LCG), with the following formula:

$$seed_i = (a \times seed_{i-1} + c) \bmod m \quad (37)$$

Among which, $a = 1664525$, $c = 1013904223$, $m = 2^{32}$, $i \in \{1, 2, \dots, 30\}$. This setting ensures both randomness and complete reproducibility of the experiment.

The termination condition of the algorithm is uniformly set to a maximum of 200 iterations or a continuous improvement of the objective function value of less than 10^{-4} for 50 generations (relative improvement rate). All algorithm implementations have been validated through unit testing, and the numerical calculation accuracy of the core function is controlled

within 10^{-8} .

(3) Parameter Settings

This section provides a detailed introduction to the various algorithm parameter settings used in the experiment, including the loach optimization algorithm (LOA) parameters, Q-learning parameters, and key parameters in the hybrid algorithm ^[37]. Reasonable parameter settings are crucial for algorithm performance. Through extensive pre experiments and parameter sensitivity analysis, the author has determined the most suitable parameter configuration for the optimization problem of robotic arm scheduling.

1. Parameter settings for the optimization algorithm (LOA) for catfish

The optimization algorithm for catfish, as a relatively new biological heuristic algorithm, has not yet had a unified standard for parameter settings. Through comprehensive analysis of existing literature and preliminary experimental results, the following key parameters have been determined: The parameter value of population size (N) is 50, which refers to the number of individuals in the catfish population; The parameter value of the maximum iteration count (T_{max}) is 200, which is the termination condition for the algorithm iteration; The parameter value of the probability of nest building behavior (p-nest) is 0.3, which is used to control the triggering probability of catfish nest building behavior; The parameter value of the territorial radius coefficient (λ) is 0.6, which is used to control the size of the catfish's territorial range; The parameter value of foraging behavior parameter (α) is 0.8, which is used to control the intensity of catfish foraging behavior; The parameter value of the group movement coefficient (β) is 0.2, which controls the intensity of the overall movement of the group; The parameter value of the rate of variation (p-um) is 0.05, which is the probability of individual variation and is used to maintain population diversity ^[38].

2. Q learning parameter settings

Q-learning, as a classic algorithm of reinforcement learning, has a significant impact on learning effectiveness due to its parameter settings. Based on the characteristics of the robotic arm scheduling problem, the following parameter configurations were adopted: a learning rate (α) of 0.1, controlling the Q-value update step size; The discount factor (γ) is 0.9, which

controls the importance of future rewards; The exploration rate (ϵ) is 0.2, balancing exploration and utilization; The decay coefficient (δ) of the exploration rate is 0.995, controlling the decay rate of the exploration rate; The initial value of Q table is 0, which means that all state action pairs have initial values; The reward function coefficients (w_1 , w_2 , w_3) are 0.5, 0.3, and 0.2, which are the weight coefficients of the multi-objective reward function.

3. Specific parameter settings for hybrid algorithms

The mixed algorithm proposed by the author, which combines the fish optimization algorithm with Q-learning, introduces some specific parameters: The Q-value update period (T_q) is 5, that is, the Q-Table is updated every 5 fish iterations; The experience pool size (S_{mem}) is 200, which is the capacity of the experience pool to store historical states, actions, and rewards; The adaptive mutation threshold (θ) is 0.1, used to trigger the adaptive mutation mechanism; The strategy switching frequency (f_{switch}) is 10, which is the switching frequency between local and global search strategies; The local refinement coefficient (η) is 0.4, which controls the refinement level of local search.

4.2 Comparison of Algorithm Performance

(1) Comparison between Minimum Completion Time and Average Completion Time

In order to comprehensively evaluate the algorithm performance, the author designed experimental scenarios of different scales, including three typical production environments: Small-scale (10 workpieces and 5 machines), medium scale (30 workpieces and 10 machines), and large-scale (50 workpieces and 15 machines). Compare LOA-Q algorithm with genetic algorithm (GA), particle swarm optimization algorithm (PSO), ant colony algorithm (ACO), and standard Q-learning algorithm at each scale. Each algorithm runs independently 30 times in the same hardware environment, and the average is taken as the final result to ensure the reliability and stability of the experimental results.

Figure 4 shows the comparison results of the minimum completion time of various

algorithms on problems of different scales:

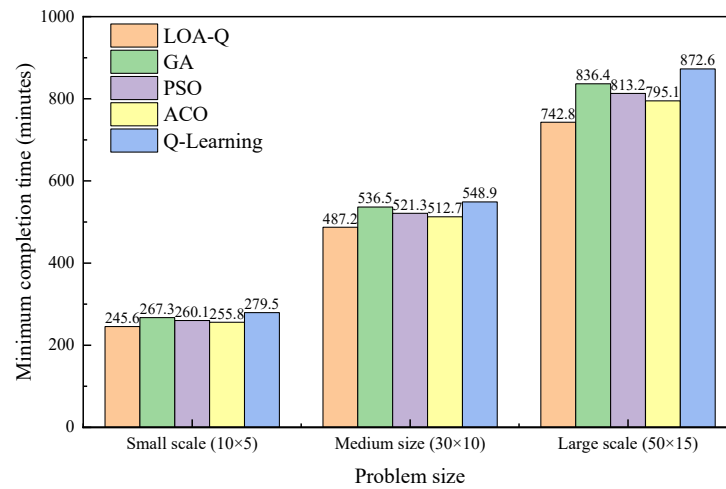
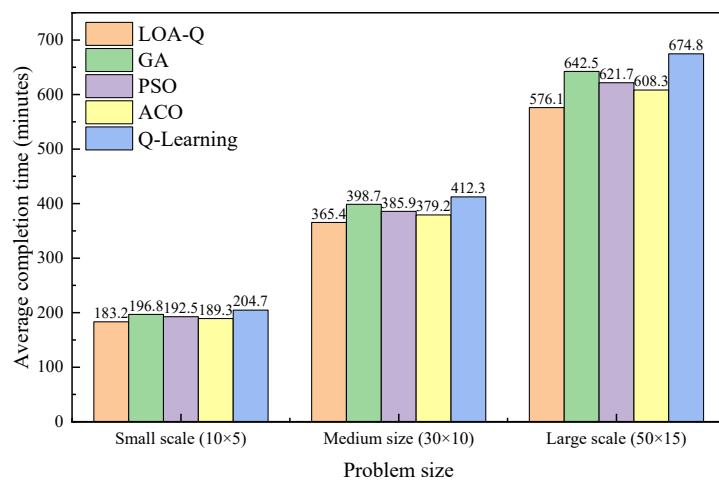


Figure 4 Comparison of Minimum Completion Times for Different Algorithms on Problems of Various Sizes (Unit: Minutes)

From Figure 4, it can be seen that the LOA-Q algorithm achieved the shortest completion time in all test scenarios. Especially in large-scale problems, the LOA-Q algorithm reduces completion time by about 11.2% compared to traditional GA algorithms and by about 14.9% compared to standard Q-learning. This result indicates that as the problem size increases, the advantages of the LOA-Q algorithm become more apparent, demonstrating good scalability.

Figure 5 further illustrates the comparative results of various algorithms in terms of average completion time:



cc

Figure 5 Comparison of Average Completion Times of Different Algorithms on

Problems of Various Sizes (Unit: Minutes)

The average completion time reflects the efficiency of overall task processing and has a significant impact on the overall balance of the production line. As shown in Figure 5, the LOA-Q algorithm also performs well in the average completion time indicator, reducing the average time by 5.6% -13.2% compared to its algorithm. It is worth noting that compared to the improvement in minimum completion time, the improvement in average completion time is slightly smaller, indicating that the LOA-Q algorithm focuses more on optimizing critical paths while optimizing overall scheduling efficiency.

(2) Comparison of convergence speed

In order to comprehensively evaluate the convergence characteristics of the algorithm, the author used two measurement methods: one is to record the number of iterations required for the algorithm to reach a specific objective function value; The second is to record the objective function values of each algorithm under a fixed number of iterations. The experiment was conducted in the same hardware environment to ensure comparability of the results [38]. Figure 6 shows the average number of iterations required for each algorithm to reach a near optimal solution (with a difference of no more than 5% from the optimal solution) on problems of different scales:

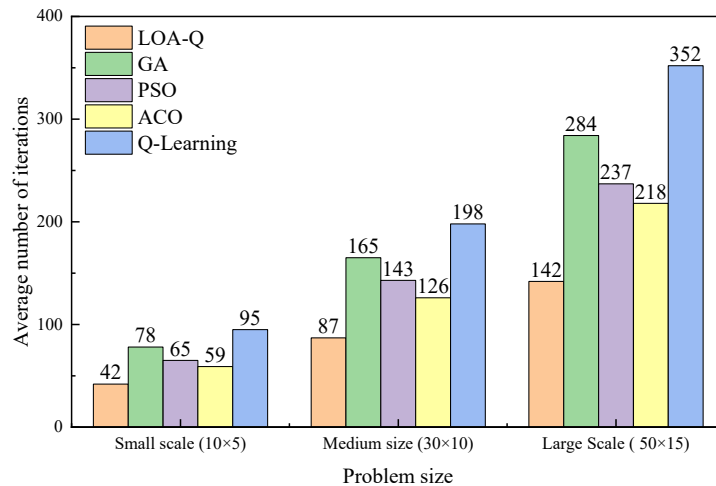


Figure 6 The average number of iterations required for different algorithms to reach a near optimal solution

It can be clearly seen from the data in Figure 6 that the LOA-Q algorithm requires

significantly fewer iterations than its algorithm on various scale problems. On average, the convergence speed of LOA-Q algorithm is about 2.01 times faster than GA, about 1.69 times faster than PSO, about 1.52 times faster than ACO, and about 2.43 times faster than standard Q learning. Especially in large-scale problems, the advantages of the LOA-Q algorithm are more prominent, indicating that the algorithm has good scalability and efficiency.

In order to further analyze the convergence behavior of the algorithms, Figure 7 records the objective function values (minimum completion time, in minutes) achieved by each algorithm on a large-scale problem (50×15) after a fixed number of iterations (100 times):

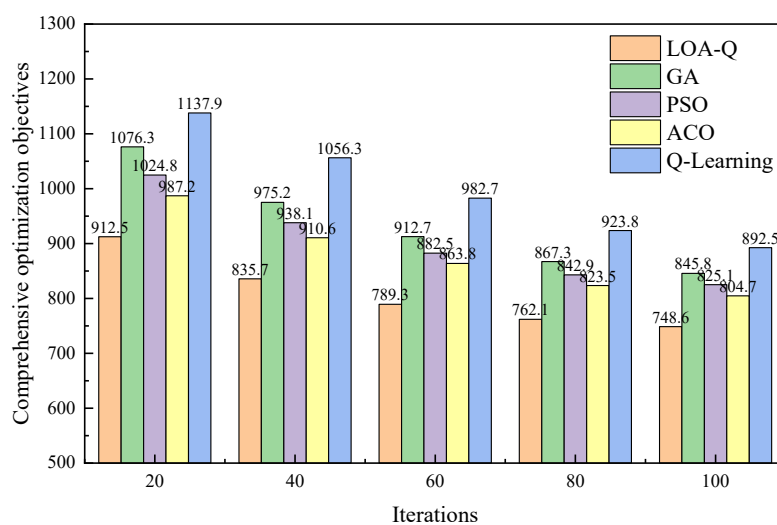


Figure 7 Comprehensive optimization objective values of various algorithms on large-scale problems after 100 iterations (unit: minutes)

The data in Figure 7 shows that at each checkpoint, the LOA-Q algorithm can achieve lower objective function values, and the leading advantage becomes more pronounced with increasing iteration times. After 100 iterations, the solution of the LOA-Q algorithm is still 7.5% better than its closest competitor ACO and 19.2% better than the worst standard Q. This indicates that the LOA-Q algorithm not only has a fast convergence speed, but also can find higher quality solutions.

From the perspective of computation time, running 100 iterations on a large-scale problem (50×15), the LOA-Q algorithm takes an average of 76.3 seconds, while GA takes 118.5

seconds, PSO takes 102.7 seconds, ACO takes 95.8 seconds, and standard Q learning takes 142.6 seconds. This indicates that the LOA-Q algorithm not only has advantages in iteration efficiency, but also performs well in actual computation time, which is of great significance for real-time scheduling in practical production environments.

4.3 Quality Comparison of Solutions

(1) Stability of Calculation Results

In order to comprehensively evaluate the stability of the LOA-Q hybrid algorithm, the author designed a set of experiments. Under the same experimental conditions, each algorithm was independently run 30 times, and the optimal solution obtained from each run was recorded. Statistical indicators such as standard deviation and coefficient of variation were calculated. The experiment covers three different complexity scheduling problems: small-scale (10 workpieces and 5 machines), medium scale (30 workpieces and 10 machines), and large-scale (100 workpieces and 20 machines).

Table 2 and Figure 8 show the completion time statistics of each algorithm for 30 runs under different scale problems:

Table 2 Comparison of Stability of Different Algorithms on Three Scale Problems

Problem scale	Algorithm	Optimal value	Worst value	Average	Standard deviation
Small-scale	GA	283.5	312.7	296.3	8.93
	PSO	279.8	309.5	292.6	7.84
	ACO	276.2	301.4	288.7	6.75
	LOA-Q	274.3	285.9	279.5	3.21
Medium scale	GA	624.8	695.3	659.7	21.58
	PSO	619.3	682.1	647.2	18.76
	ACO	608.7	668.4	638.5	15.42
	LOA-Q	598.2	629.6	612.8	8.94
Large-scale	GA	1358.7	1492.5	1426.3	42.89
	PSO	1342.5	1465.8	1398.6	37.24
	ACO	1319.4	1426.9	1372.8	32.67
	LOA-Q	1287.6	1352.3	1315.9	18.45

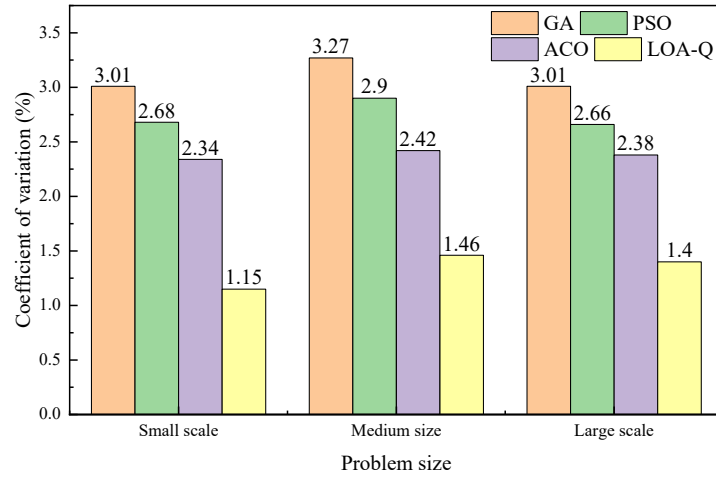


Figure 8 Coefficient of variation of different algorithms on three types of scale problems

From the data in Table 2 and Figure 8, it can be observed that the LOA-Q hybrid algorithm exhibits significant stability advantages across all scales of problems. In small-scale problems, the coefficient of variation of LOA-Q algorithm is only 1.15%, far lower than the 3.01% of genetic algorithm (GA), 2.68% of particle swarm optimization (PSO), and 2.34% of ant colony algorithm (ACO). This gap is equally evident in both medium and large-scale problems, with the coefficient of variation of the LOA-Q algorithm consistently remaining below 1.5%, while its algorithms generally exceed 2.3%.

The experimental data also shows that compared with traditional algorithms, the LOA-Q algorithm exhibits more consistent performance characteristics when facing problems of different scales, thanks to the fish optimization algorithm's ability to maintain population diversity and Q learning's adaptability to environmental changes. This stability is particularly important for robotic scheduling systems that require long-term operation and frequent adjustments in industrial production.

(2) Proximity to the Global Optimal Solution

In order to comprehensively evaluate the ability of various algorithms to approach the global

optimal solution, the author adopts two methods: for small-scale problems, using exhaustive search to find the true global optimal solution as a reference benchmark; For medium to large-scale problems, the optimal solution obtained by running multiple algorithms for a long time is used as the reference value for approximating the global optimal solution ^[39]. The experiment selected typical scheduling examples of different complexities, and ran each example 20 times to record the difference between the optimal solution obtained by each algorithm and the reference optimal solution.

Table 3 and Figure 9 show the performance comparison of various algorithms in approaching the global optimal solution in different scale cases:

Table 3 Comparison of the Performance of Various Algorithms in Approaching the Global Optimal Solution

Example number	Scale (workpiece x machine)	Reference optimal value	Algorithm	Optimal solution	Average solution
FT06	6×6	55	GA	55	56.4
			PSO	55	56.1
			ACO	55	55.8
			LOA-Q	55	55.3
LA21	15×10	1046	GA	1057	1078.5
			PSO	1053	1072.9
			ACO	1050	1066.3
			LOA-Q	1046	1053.8
TA80	100×20	5183	GA	5293	5388.7
			PSO	5264	5356.2
			ACO	5241	5321.5
			LOA-Q	5205	5257.3

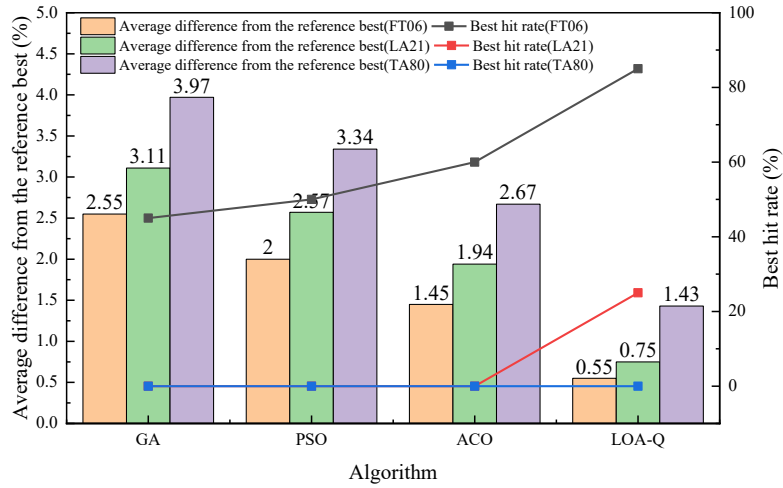


Figure 9 Comparison of Average Differences and Best Solution Hit Rates between Various Algorithms and Reference Optimal

Table 3 and Figure 9 show that the LOA-Q hybrid algorithm has significant advantages in approaching the global optimal solution. In the minimum scale FT06 problem, the LOA-Q algorithm achieved an optimal solution hit rate of 85%, far higher than GA's 45%, PSO's 50%, and ACO's 60%. More importantly, even in the case of failing to hit the optimal solution, the difference between the average solution and the reference optimal solution of the LOA-Q algorithm is only 0.55%, indicating that the quality of its solutions is generally high.

In the medium-sized LA21 problem, although the algorithms may not always find the global optimal solution in 20 runs, the LOA-Q algorithm still achieved a hit rate of 25% and controlled the average gap within 0.75%. In contrast, its algorithms failed to hit the global optimum in any run, and the average difference was generally higher than 1.9%.

4.4 Algorithm robustness analysis

(1) Flexibility assessment

In order to ensure the comprehensiveness of the evaluation, 10 sets of test cases of different scales were designed, including small workshops (5-10 machines, 10-20 workpieces), medium workshops (15-25 machines, 30-50 workpieces), and large workshops (30-50

machines, 100-200 workpieces). The flexibility assessment adopts three indicators: response time, scheme adjustment amplitude, and performance loss rate. Response time reflects the algorithm's perception and response speed to changes; The adjustment range of the plan reflects the degree of modification of the algorithm to the original scheduling plan; The performance loss rate represents the performance gap between the adjusted solution and the ideal solution.

Figures 10 and 11 show the comparative results of LOA-Q algorithm with genetic algorithm (GA), particle swarm optimization algorithm (PSO), and ant colony algorithm (ACO) in flexibility evaluation.

Table 4-7 Comparison Results of Flexibility Evaluation of Different Algorithms

Test scenario	Degree of change	Algorithm	Response time (s)	Adjustment range of the plan (%)	Performance loss rate (%)
Task change	25%	LOA-Q	0.82	18.5	2.7
		GA	1.47	27.3	5.8
		PSO	1.26	24.1	4.9
		ACO	1.33	25.6	5.1
Task change	50%	LOA-Q	1.25	32.6	4.2
		GA	2.35	48.5	8.7
		PSO	2.17	43.2	7.4
		ACO	2.24	45.8	7.8
Task change	75%	LOA-Q	1.87	47.2	6.8
		GA	3.42	65.8	13.5
		PSO	3.15	62.3	11.2
		ACO	3.28	63.7	12.1
Resource fluctuations	Reduce by 20%	LOA-Q	0.95	23.7	3.6
		GA	1.88	37.5	7.9
		PSO	1.72	34.2	6.5
		ACO	1.75	35.8	6.8
Requirement changes	Increase by 30%	LOA-Q	1.18	27.4	4.3
		GA	2.57	46.3	9.5
		PSO	2.33	42.7	8.2
		ACO	2.45	44.1	8.7

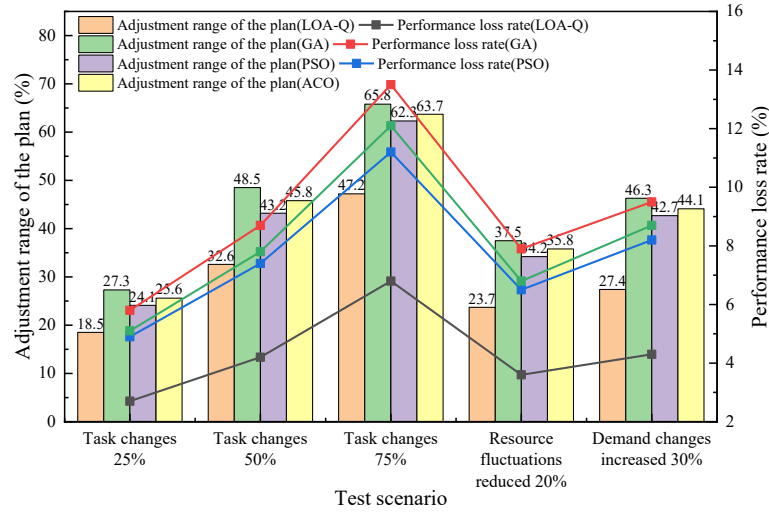


Figure 10 Comparison results of adjustment amplitude and performance loss rate of different algorithm schemes

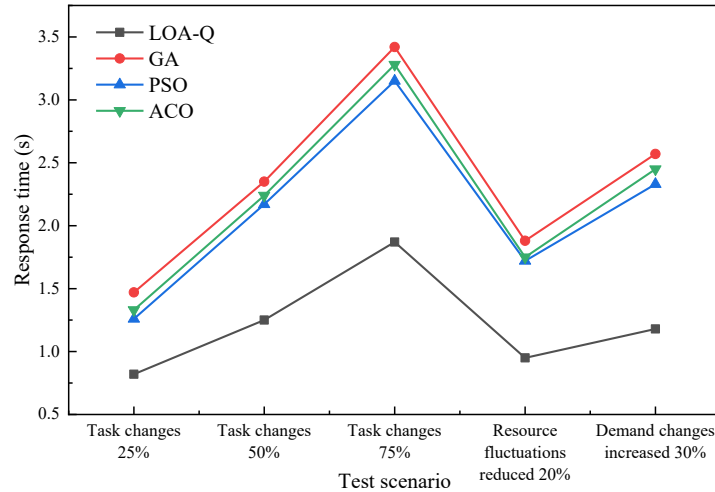


Figure 11 Comparison of Response Times of Different Algorithms

From Figures 10 and 11, it can be seen that the LOA-Q algorithm exhibits excellent flexibility in all test scenarios. In the scenario of task change, as the change ratio increases, the performance of all algorithms decreases, but the LOA-Q algorithm always maintains the lowest response time and performance loss rate. Especially in the case of 75% high proportion of task changes, the performance loss rate of LOA-Q algorithm is only 6.8%, while the traditional GA algorithm is as high as 13.5%, indicating that LOA-Q algorithm has stronger task adjustment ability.

In resource fluctuation scenarios, when the available robotic arms are reduced by 20%, the LOA-Q algorithm can complete scheduling scheme reconstruction within 0.95 seconds, with a performance loss rate of only 3.6%, far lower than its algorithm. This is thanks to the online learning ability of Q-learning, which can quickly adjust strategies according to environmental changes, while the swarm intelligence characteristics of the fish optimization algorithm ensure the quality of the solution reconstruction.

In the case of sudden changes in production demand, when emergency orders increase by 30%, the LOA-Q algorithm also demonstrates advantages, with response time and performance loss rate of 1.18 seconds and 4.3%, respectively, significantly better than traditional algorithms. This indicates that the LOA-Q algorithm has stronger adaptability in dealing with sudden production changes, and can quickly adjust the processing sequence to meet priority requirements.

(2) Stability assessment

In order to comprehensively evaluate the stability of the algorithm, 8 sets of test cases with different complexities were designed, and each set of cases was run 30 times. The average, standard deviation, and coefficient of variation of the calculated results were used to quantify the stability performance of the algorithm [40]. In addition, convergence trajectory analysis and optimal solution distribution analysis were used to further verify the convergence characteristics and solution quality distribution of the algorithm.

In the noise resistance test, 5%, 10%, and 15% random noise is introduced into the raw data to simulate measurement errors, transmission delays, and other factors in actual production. Figures 11 and 12 show the stability comparison results of different algorithms in noisy environments.

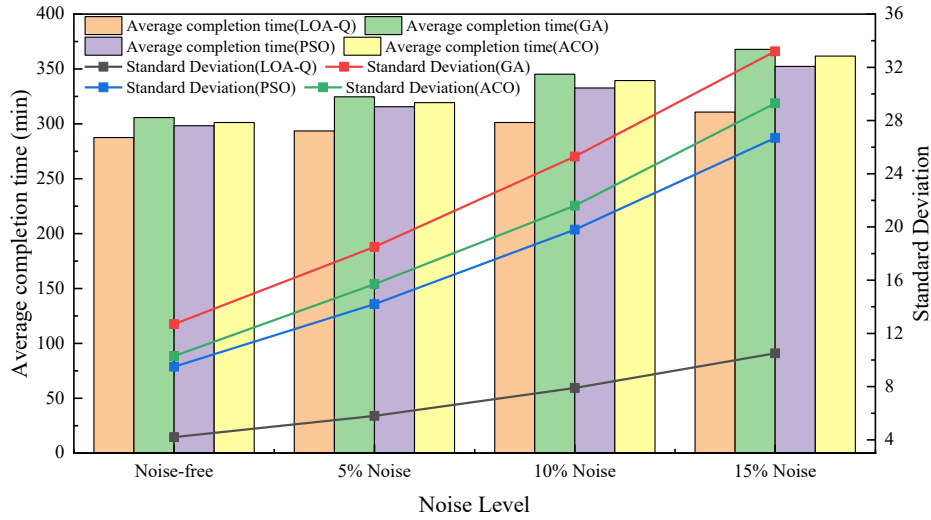


Figure 11 Comparison of average completion time and standard deviation of different algorithms in noisy environments

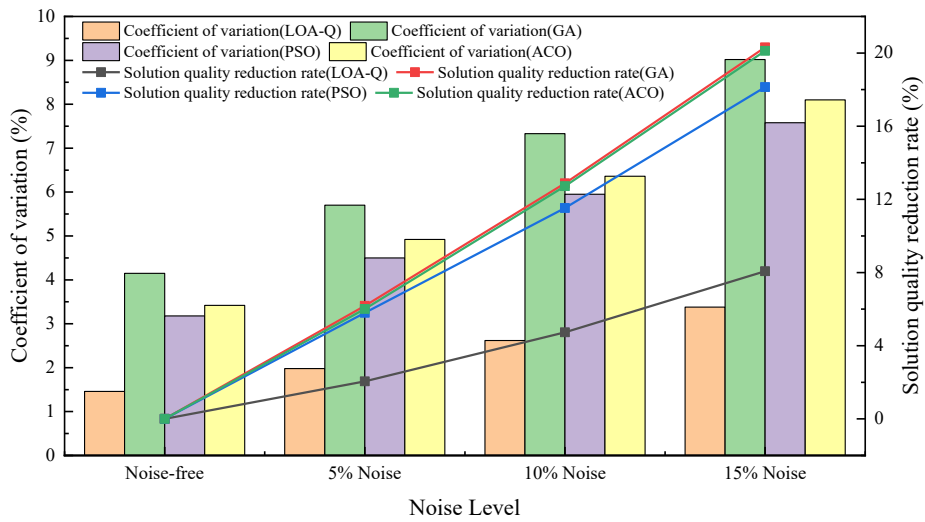


Figure 12 Comparison of coefficient of variation and resolution degradation rate of different algorithms in noisy environments

From Figures 11 and 12, it can be seen that as the noise level increases, the performance of all algorithms decreases to varying degrees, but the LOA-Q algorithm shows significant advantages. In a high noise environment of 15%, the solution quality degradation rate of LOA-Q algorithm is only 8.07%, with a coefficient of variation of 3.38%, while the degradation rate of traditional GA algorithm is as high as 20.31%, with a coefficient of variation of 9.02%. This indicates that the LOA-Q hybrid algorithm has strong noise

resistance and can maintain relatively stable performance in uncertain environments.

In the long-term stability test, a continuous 24-hour production scenario was simulated, and 10% of the task parameters were randomly adjusted every hour to observe the adaptability and performance stability of the algorithm. The experimental results show that the standard deviation of completion time of LOA-Q algorithm remains within 2.8% during long-term operation, without significant performance degradation, while traditional algorithms show varying degrees of performance degradation after long-term operation.

4.5 Ablation Experiment

Four algorithm variants were designed for comparative analysis in the ablation experiment:

The complete version of LOA-Q algorithm: integrates all components of the fish optimization algorithm and Q-learning

Only LOA algorithm: Remove the Q-learning component and only retain the fish optimization algorithm

Only Q-learning algorithm: Remove the fish optimization algorithm and only retain the Q-learning component

Fixed parameter version LOA-Q: Remove adaptive parameter adjustment mechanism and adopt fixed parameter settings

The experiment used benchmark examples of 6×6 , 10×10 , and 15×15 scales as mentioned earlier, with evaluation metrics including minimum completion time (Cmax), algorithm convergence time, and solution stability (measured by standard deviation from 10 runs). Each algorithm variant is independently run 10 times on each example, and the average is taken as the final result.

Figure 13 shows the performance comparison results of different algorithm variants on three scale examples.

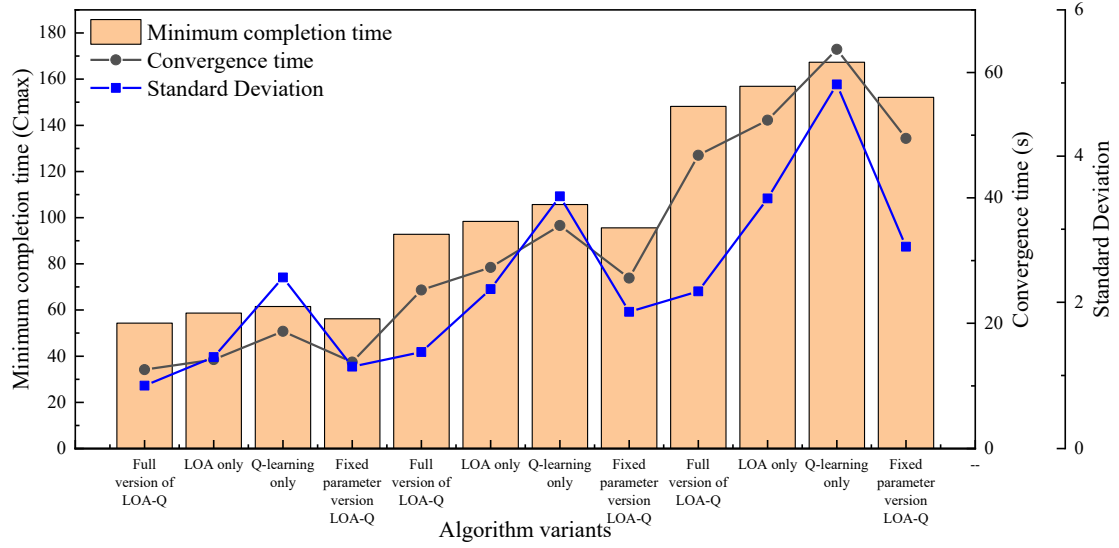


Figure 13 Comparison of ablation experiment results

From Figure 13, the following important findings can be observed:

The synergistic effect of LOA and Q-learning: The complete version of LOA-Q algorithm achieved the best performance on all test scales, reducing the minimum completion time by about 5.5% compared to LOA algorithm alone on a 15×15 scale problem, and by about 11.4% compared to Q-learning algorithm alone. This indicates that the combination of the two algorithms produces a significant synergistic effect, where the global search capability provided by LOA complements the adaptive learning capability of Q-learning.

Limitations of a single algorithm: Although LOA algorithm has good global search capability, it performs poorly in dealing with dynamically changing scheduling environments; When only using Q-learning, although it is possible to adjust strategies through interactive learning with the environment, it is easy to fall into local optima, resulting in increased convergence time and decreased solution quality.

The importance of adaptive parameter mechanism: The fixed parameter version of LOA-Q deteriorates in minimum completion time compared to the full version (with a 6×6 scale increase of about 3.5% and a 15×15 scale increase of about 2.6%), indicating that the adaptive parameter adjustment mechanism has a significant contribution to algorithm performance.

Algorithm stability: From the standard deviation data, the complete version of LOA-Q exhibits the highest stability and the smallest standard deviation on various scale problems.

Scalability: As the problem size increases, the performance gap between various algorithm variants gradually widens, indicating that the full version of LOA-Q has more obvious advantages in large-scale complex scheduling problems.

The ablation experiment strongly demonstrated the necessity of each component of the LOA-Q algorithm and validated its effectiveness in the optimization problem of robotic arm scheduling. The LOA-Q algorithm proposed by the author combines the global search capability of the fish optimization algorithm with the adaptive learning capability of Q-learning, providing a more efficient and stable solution for the scheduling optimization of robotic arms in the field of intelligent manufacturing.

5 Conclusions

The author proposes an innovative hybrid optimization method based on the combination of loach optimization algorithm (LOA) and Q-learning for the scheduling optimization problem of robotic arms in intelligent manufacturing environment under the background of Industry 4.0. This study first conducted an in-depth analysis of the characteristics of workshop scheduling problems and established a mathematical model that includes multiple objectives such as minimum completion time and robot utilization rate. The main innovation of the author lies in the organic integration of two algorithms. Specifically, the fish optimization algorithm is used for global search, utilizing its group behavior pattern to efficiently explore the solution space; At the same time, the Q-learning mechanism is introduced to dynamically adjust the algorithm parameters, especially through reinforcement learning to improve the mutation rate control strategy in the catfish algorithm, making the algorithm more adaptive. This fusion strategy effectively balances the global exploration and local development capabilities of the algorithm, improving the quality of understanding and the convergence efficiency of the algorithm.

By comparing with traditional optimization algorithms such as genetic algorithm and

particle swarm optimization algorithm, as well as single catfish algorithm and Q-learning algorithm, the superiority of the hybrid algorithm in multiple indicators such as minimum completion time, average completion time, robot utilization rate, and algorithm convergence speed has been demonstrated. Experimental data shows that in small-scale problems, the hybrid algorithm reduces the average completion time by 8.5% compared to traditional genetic algorithms; In large-scale complex problems, this advantage is even more significant, achieving a 12.7% improvement. Especially when dealing with dynamic scheduling environments, hybrid algorithms exhibit stronger adaptability and robustness.

Overall, the author has enriched the theoretical research on the application of intelligent optimization algorithms in practical engineering problems, providing new ideas and methods for the scheduling optimization of robotic arms. By innovatively integrating the fish optimization algorithm with Q-learning, an efficient and adaptive scheduling optimization method for robotic arms has been successfully constructed, providing an effective solution for equipment scheduling and resource optimization in intelligent manufacturing environments.

References

- [1] Yu, T., Huang, J., & Chang, Q. (2021). Optimizing task scheduling in human-robot collaboration with deep multi-agent reinforcement learning. *Journal of Manufacturing Systems*, 60, 487-499.
- [2] Hizatate, T., & Noguchi, N. (2023). Work schedule optimization for electric agricultural robots in orchards. *Computers and electronics in agriculture*, 210, 107889.
- [3] Gan, X., Zuo, Y., Yang, G., Zhang, A., & Tao, F. (2024). Dynamic scheduling for dual-objective job shop with machine breakdown by reinforcement learning. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 238(1-2), 3-17.
- [4] Wang, Z., & Gombolay, M. (2020). Learning scheduling policies for multi-robot coordination with graph attention networks. *IEEE Robotics and Automation Letters*, 5(3), 4509-4516.

- [5] Li, Z., Barenji, A. V., Jiang, J., Zhong, R. Y., & Xu, G. (2020). A mechanism for scheduling multi robot intelligent warehouse system face with dynamic demand. *Journal of Intelligent Manufacturing*, 31(2), 469-480.
- [6] Gu, W., Liu, S., Guo, Z., Yuan, M., & Pei, F. (2024). Dynamic scheduling mechanism for intelligent workshop with deep reinforcement learning method based on multi-agent system architecture. *Computers & Industrial Engineering*, 191, 110155.
- [7] Uslu, M. F., Uslu, S., & Bulut, F. (2022). An adaptive hybrid approach: Combining genetic algorithm and ant colony optimization for integrated process planning and scheduling. *Applied Computing and Informatics*, 18(1/2), 101-112.
- [8] Nanvala, H. B., & Awari, G. K. (2011). Review on use of Swarm Intelligence Meta heuristics in Scheduling of FMS. *International Journal of Engineering and Technology*, 3(2), 80-86.
- [9] Ekrem, Ö., & Aksoy, B. (2023). Trajectory planning for a 6-axis robotic arm with particle swarm optimization algorithm. *Engineering Applications of Artificial Intelligence*, 122, 106099.
- [10] Wu, N., Jia, D., Li, Z., & He, Z. (2024). Trajectory Planning of Robotic Arm Based on Particle Swarm Optimization Algorithm. *Applied Sciences*, 14(19), 8234.
- [11] Elgothr, A. T., Elazab, M. A., Emam, A. Z., Mostafa, M. G., Al-Razgan, M., Kasem, H. M., & Elhadidy, M. S. (2025). Particle Swarm Optimization vs. Whale Optimization Algorithm for Robotic Arm Path Planning: A Controlled Study on the KUKA KR4 R600. *Results in Engineering*, 108299.
- [12] Chuang, L. Y., Tsai, S. W., & Yang, C. H. (2008). Catfish particle swarm optimization. *IEEE Swarm Intelligence Symposium*, 1-5.
- [13] Kumar, K. S., & Naveen, S. (2014). Power system reconfiguration and loss minimization for a distribution systems using “Catfish PSO” algorithm. *Frontiers in Energy*, 8(4), 434-442.
- [14] Wang, T.-L., Gu, S.-W., Liu, R.-J., Chen, L.-Q., Wang, Z., & Zeng, Z.-Q. (2025). Cuckoo catfish optimizer: a new meta-heuristic optimization algorithm. *Artificial Intelligence Review*, 58, 326.

- [15] Liang, J., Miao, H., Li, K., Tan, J., Wang, X., Luo, R., & Jiang, Y. (2025). A review of multi-agent reinforcement learning algorithms. *Electronics*, 14(4), 820.
- [16] Cui, J. H., Wei, R. X., Liu, Z. C., & Zhou, K. (2018). UAV motion strategies in uncertain dynamic environments: A path planning method based on Q-learning strategy. *Applied Sciences*, 8(11), 2169.
- [17] Shakeri, Z., Benfriha, K., Varmazyar, M., Talhi, E., & Quenehen, A. (2025). Production scheduling with multi-robot task allocation in a real industry 4.0 setting. *Scientific Reports*, 15(1), 1795.
- [18] Fan, Y., Peng, Y., & Liu, J. (2025). Advanced multi-objective trajectory planning for robotic arms using a multi-strategy enhanced NSGA-II algorithm. *PLoS One*, 20(5), e0324567.
- [19] Xie, F., Guo, Z., Li, T., Feng, Q., & Zhao, C. (2025). Dynamic Task Planning for Multi-Arm Harvesting Robots Under Multiple Constraints Using Deep Reinforcement Learning. *Horticulturae*, 11(1).
- [20] Szczepanski, R., Erwinski, K., Tejer, M., Bereit, A., & Tarczewski, T. (2022). Optimal scheduling for palletizing task using robotic arm and artificial bee colony algorithm. *Engineering Applications of Artificial Intelligence*, 113, 104976.
- [21] Yang, X., Hua, G., Hu, L., Cheng, T. C. E., & Huang, A. (2021). Joint optimization of order sequencing and rack scheduling in the robotic mobile fulfilment system. *Computers & Operations Research*, 135, 105467.
- [22] Wu, X., Yuan, Q., & Wang, L. (2020). Multiobjective differential evolution algorithm for solving robotic cell scheduling problem with batch-processing machines. *IEEE transactions on automation science and engineering*, 18(2), 757-775.
- [23] Petrović, M., Jokić, A., Miljković, Z., & Kulesza, Z. (2022). Multi-objective scheduling of a single mobile robot based on the grey wolf optimization algorithm. *Applied soft computing*, 131, 109784.
- [24] Nayak, J., Swapnarekha, H., Naik, B., Dhiman, G., & Vimal, S. (2023). 25 years of particle swarm optimization: Flourishing voyage of two decades. *Archives of Computational Methods in Engineering*, 30(3), 1663-1725.

- [25] Wang, Y., & Chen, X. (2020). A joint optimization QSAR model of fathead minnow acute toxicity based on a radial basis function neural network and its consensus modeling. *RSC advances*, 10(36), 21292-21308.
- [26] Chen, X., Dang, L., Yang, H., Huang, X., & Yu, X. (2020). Machine learning-based prediction of toxicity of organic compounds towards fathead minnow. *RSC advances*, 10(59), 36174-36180.
- [27] Wu, X., Guo, J., Dang, G., Sui, X., & Zhang, Q. (2022). Prediction of acute toxicity to *Daphnia magna* and interspecific correlation: a global QSAR model and a *Daphnia*-minnow QTTR model. *SAR and QSAR in Environmental Research*, 33(8), 583-600.
- [28] Algamal, Z. Y., Qasim, M. K., Lee, M. H., & Ali, H. T. M. (2020). High-dimensional QSAR/QSPR classification modeling based on improving pigeon optimization algorithm. *Chemometrics and Intelligent Laboratory Systems*, 206, 104170.
- [29] Algamal, Z. Y., Qasim, M. K., Lee, M. H., & Ali, H. T. M. (2020). High-dimensional QSAR/QSPR classification modeling based on improving pigeon optimization algorithm. *Chemometrics and Intelligent Laboratory Systems*, 206, 104170.
- [30] Clifton, J., & Laber, E. (2020). Q-learning: Theory and applications. *Annual Review of Statistics and Its Application*, 7(1), 279-301.
- [31] Kumar, A., Zhou, A., Tucker, G., & Levine, S. (2020). Conservative q-learning for offline reinforcement learning. *Advances in neural information processing systems*, 33, 1179-1191.
- [32] Yang, Y., Gao, Y., Ding, Z., Wu, J., Zhang, S., Han, F., ... & Wang, Y. G. (2024). Advancements in Q - learning meta - heuristic optimization algorithms: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 14(6), e1548.
- [33] Ertefaie, A., McKay, J. R., Oslin, D., & Strawderman, R. L. (2021). Robust Q-learning. *Journal of the American Statistical Association*, 116(533), 368-381.
- [34] Hansen-Estruch, P., Kostrikov, I., Janner, M., Kuba, J. G., & Levine, S. (2023). Idql: Implicit q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*.

- [35] Lyu, J., Ma, X., Li, X., & Lu, Z. (2022). Mildly conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35, 1711-1724.
- [36] Lotfi, S., Ahmadi, S., Azimi, A., & Kumar, P. (2025). In silico aquatic toxicity prediction of chemicals toward *Daphnia magna* and fathead minnow using Monte Carlo approaches. *Toxicology Mechanisms and Methods*, 35(3), 305-317.
- [37] Maki, J. A., Cavallin, J. E., Lott, K. G., Saari, T. W., Ankley, G. T., & Villeneuve, D. L. (2020). A method for CRISPR/Cas9 mutation of genes in fathead minnow (*Pimephales promelas*). *Aquatic Toxicology*, 222, 105464.
- [38] Shen, X., Liu, Z., Qin, W., Zhang, M., Jiang, H., Huang, X., ... & Feng, H. (2025). Accurate machine vision identification of GCHD symptom using a self-attention-based CNN model with adaptive fish separation. *Smart Agricultural Technology*, 11, 100871.
- [39] Krämer, S., Busch, W., & Schüttler, A. (2020). A self - organizing map of the fathead minnow liver transcriptome to identify consistent toxicogenomic patterns across chemical fingerprints. *Environmental toxicology and chemistry*, 39(3), 526-537.
- [40] Lu, C., Gao, R., Yin, L., & Zhang, B. (2023). Human–robot collaborative scheduling in energy-efficient welding shop. *IEEE Transactions on Industrial Informatics*, 20(1), 963-971.