International Journal of Industrial Engineering, 32(1), 34-51, 2025

A Reinforcement Learning and The Northern Goshawk Optimization Algorithm for Flexible Job Shop Scheduling Problem

Changshun Shao[1, 2], Zhenglin Yu[1, 2, *], Han Hou[2], Hongchang Ding[1, 2], Guohua Cao[1, 2], and Bin Zhou[1, 2]

[1]College of Mechanical and Electrical Engineering
Changchun University of Science and Technology
Changchun, China
[*]Corresponding author's e-mail: zhenglin_yu@sina.com

[2]Chongqing Research Institute
Changchun University of Science and Technology
Chongqing, China

This paper introduces northern goshawk optimization, a novel global search strategy for the flexible job shop scheduling problem. It uses two-stage encoding and random-key-based encoding to transform individual position vectors into flexible job shop scheduling problem solutions. To improve local search, reinforcement learning is integrated, converting the flexible job shop scheduling problem into a Markov decision process with 10 states and 6 rules. A reward function based on optimal completion time guides the search. The proposed hybrid northern goshawk optimization-Q-learning-state-action-reward-state-action framework combines global and local search strengths. Experiments on standard datasets show the algorithm's superior performance, validating its effectiveness and practicality in solving the flexible job shop scheduling problem and real-world production scheduling problems.

Keywords: Flexible Job Shop Scheduling Problem; Northern Goshawk Optimization; Reinforcement Learning.

(Received on September 16, 2024; Accepted on January 7, 2025)

1. INTRODUCTION

In the wave of global economic prosperity, manufacturing, as the backbone of the economic system, has increasingly emerged as a strategic sector. Ahmet Melik Öztürk *et al.* (2022) proposed a cluster-based priority list method to generate improved priority lists that reduce completion time, showing significant enhancements when compared to traditional methods using benchmark data. At the heart of manufacturing, the flexible job shop scheduling problem (FJSP) has become increasingly challenging due to the diversification and complex nature of production tasks, as well as the growing complexity of the execution environment. Traditional scheduling methods are struggling to maximize production efficiency in rapidly changing environments.

For a long time, traditional scheduling strategies based on static rules or heuristic algorithms have struggled to adapt to the complex production environment. While these methods perform well in certain specific scenarios, their inherent limitations, such as a lack of flexibility, adaptability, and global optimization capability, make it difficult to meet the modern manufacturing industry's demands for production efficiency, cost control, and rapid market response. As the complexity of production tasks increases and the number of uncertain factors in the production environment grows, traditional scheduling methods have increasingly shown problems in task allocation, resource utilization, and time management, leading to frequent problems such as low production efficiency and significant resource wastage.

However, over the past two decades, the development of computing technology and the rise of intelligent algorithms, especially in the field of the FJSP, have brought about revolutionary changes in production scheduling. The application of intelligent optimization techniques such as genetic algorithms, simulated annealing, and particle swarm optimization has provided new solutions. These intelligent algorithms, by mimicking natural evolutionary processes, physical phenomena, or the behavior of biological groups, can effectively search for optimal or near-optimal solutions in large-scale search spaces, demonstrating strong global search capabilities and good robustness.

Despite the significant achievements of intelligent algorithms in solving the FJSP, they still lack the ability to self-learn and adapt to complex environments. These methods often require manual parameter tuning and predefined rules, which can be time-consuming and may not always be suitable for complex environments. Fortunately, with the rapid advancement of artificial intelligence technology, more and more researchers are introducing reinforcement learning into the research of the

FJSP, infusing new vitality into this long-standing challenge. Reinforcement learning is a type of machine learning where an agent learns to make decisions by interacting with its environment.

The application of reinforcement learning to the FJSP enables a keen perception and flexible adaptation to changes in the production environment, significantly enhancing the overall efficiency and flexibility of the production line. In recent years, with the deep integration of deep learning and reinforcement learning, deep reinforcement learning has demonstrated tremendous application potential in the FJSP. While deep reinforcement learning demonstrates exceptional performance in state parsing and value estimation, it may also face the challenge of "aesthetic fatigue" over time, referring to the gradual decrease in novelty. Currently, the prevalent approach in academia is to integrate deep learning with reinforcement learning to accurately fit the value function, thereby achieving a comprehensive capture of state information. However, to maintain this advantage and seek further breakthroughs, we urgently need to explore new solutions based on existing achievements.

Based on the above background, this paper proposes an innovative approach by deeply studying previous achievements. This method aims to combine the global search capabilities of intelligent algorithms with the local search capabilities of reinforcement learning to achieve complementary advantages. Specifically, we introduce the northern goshawk optimization (NGO), which has attracted attention due to its excellent global search ability and efficient convergence speed. For the FJSP, the NGO not only demonstrates unprecedented novelty but also forms a perfect synergy with the local search capabilities of reinforcement learning.

The contributions of this paper are described as follows:

(1) This paper employs an innovative global search strategy, namely the NGO, to solve the FJSP. This algorithm combines a two-stage encoding technique, a random-key-based encoding method, and an efficient conversion mechanism that effectively transforms individual position vectors into solutions for the FJSP, thereby achieving an effective representation of the problem. By carefully selecting the population size, the NGO is deeply integrated with the FJSP, significantly enhancing the solution efficiency.

(2) This paper introduces reinforcement learning algorithms, specifically Q-learning and state-action-reward-state-action (SARSA), for local search. By carefully designing 10 states and 6 scheduling rules, the FJSP is transformed into a Markov decision process, and a reward function based on optimal completion time guides efficient local searches. Furthermore, a hybrid algorithm framework called Northern Goshawk Optimization-Q-learning-SARSA (NGO-QLSA) is proposed, integrating the global search capabilities of the NGO with the local search advantages of reinforcement learning. This combination enhances the algorithm's performance and broadens its application range.

(3) To validate the performance of the algorithm, we selected multiple standard datasets from Brandimarte, Dauzère, Barnes, and Fattahi for experimentation. By comparing with various advanced algorithms, we found that the proposed algorithm exhibits significant advantages and excellent performance in solving the FJSPs, fully demonstrating its effectiveness and practicality.

Section 1 describes the FJSP and introduces the contributions of this paper. Section 2 overviews recent related work. Section 3 details the FJSP, Section 4 introduces the proposed algorithm framework, and Section 5 presents the experimental results. The final section concludes the paper.

2. RELATED WORK

2.1 Intelligent algorithms for solving the FJSPs

The papers on using intelligent algorithms to solve the FJSPs are extensive; only a portion is listed here. For instance, Fan *et al*. (2022) proposed an improved genetic algorithm with problem-specific encoding and decoding strategies to address the FJSP with machine reconfiguration. Meng *et al*. (2023) developed an enhanced genetic algorithm with a unique population diversity check method to minimize the manufacturing span and investigate the FJSP with limited automated guided vehicles. Shen (2022) introduced an adaptive mutation probability function combined with an adaptive crossover operator and improved the fitness function to enhance the traditional genetic algorithm for solving the FJSP. Kong *et al*. (2024) proposed an enhanced discrete particle swarm optimization algorithm that incorporates Pareto optimization and nonlinear adaptive inertia weight strategies to minimize multiple objectives. Zhou *et al*. (2024) introduced an adaptive gray wolf fast optimization algorithm that enhances the traditional gray wolf algorithm with a non-linear convergence factor and spiral search mechanism to efficiently solve the FJSP. Fan *et al*. (2024) proposed a genetic chaos levy nonlinear tuna swarm optimization algorithm to address the FJSP with random machine breakdowns, aiming to minimize the maximum completion time and enhance stability. Mei *et al*. (2024) proposed an adaptive simulated annealing non-dominated sorting genetic algorithm II to optimize the multi-objective FJSP.

2.2 Deep reinforcement learning algorithms for solving the FJSPs

There are quite a few papers that use deep reinforcement learning to solve the FJSPs. Here are some examples: Zeng *et al*. (2022) developed a deep reinforcement learning solution combining graph neural networks (GNN) and multi-layer perceptron (MLP) with asynchronous advantage actor-critic optimization, effectively reducing training time. Saqlain *et al*. (2023) introduced a Monte Carlo tree search-based FJSP algorithm for real-time scheduling in complex environments. Lei *et al*. (2022) proposed an algorithm using multi-pointer graph networks and multi-policy optimization to learn high-quality allocation strategies. Jing *et al*. (2024) designed a multi-agent reinforcement learning structure based on graph convolutional networks for personalized manufacturing, addressing high flexibility and dynamic response. Yuan *et al*. (2023) transformed combinatorial optimization into a multi-stage decision-making problem using a multi-agent double deep Q-network. Wang *et al*. (2023) combined a self-attention model with deep reinforcement learning to handle complex relationships between operations and machines. Song *et al*. (2022) merged operation selection and machine allocation into a composite decision using an isomorphic graph neural network. Zhang *et al*. (2024) used graph attention networks and Transformer encoders to extract global state information and generate operation strategies. Du *et al*. (2022) presented a deep Q-network model for the multi-objective FJSP, optimizing completion time and energy consumption. Wan *et al*. (2024) proposed a deep actor-critic reinforcement learning approach for the FJSP. Wan *et al*. (2024) proposed a novel end-to-end deep reinforcement learning method with a meta-path-based heterogeneous graph neural network that efficiently solves the FJSP. Yuan *et al*. (2024) introduced a lightweight MLP-based framework to reduce computational complexity, enhancing state representation and action space design.

2.3 Intelligent algorithms and Reinforcement learning algorithms for solving the FJSPs

Papers that combine intelligent algorithms and reinforcement learning to solve the FJSPs are relatively rare, making this a promising research direction. Below are three examples:

Chen *et al*. (2020) proposed a self-learning genetic algorithm that utilizes a genetic algorithm as the basic optimization method and intelligently adjusts its key parameters based on reinforcement learning, addressing the problem of intelligent algorithms lacking self-learning and adaptability to environmental changes. Long *et al*. (2022) proposed an improved self-learning artificial bee colony algorithm based on reinforcement learning. In this algorithm, the update dimension of the artificial bee colony is intelligently selected based on reinforcement learning, improving convergence speed and accuracy. Chen *et al*. (2024) proposed a Q-Learning-based non-dominated sorting genetic algorithm II algorithm to address the dynamic FJSP with limited transportation resources, considering job cancellation, machine breakdown, and automated guided vehicle breakdown, demonstrating the algorithm's effectiveness in minimizing completion time and total energy consumption while maintaining rescheduling stability through simulation experiments.

3. PROBLEM DESCRIPTION

In the context of the FJSP, we consider a system comprised of m machines, denoted as $M\epsilon\{M_1, M_2, \cdots, M_m\}$. Each machine has specific functionalities and processing capabilities, potentially suitable for different types of operations. Additionally, there are n jobs that need to be processed, labeled as $J\epsilon\{J_1, J_2, \cdots, J_n\}$. Each job consists of a sequence of operations that must be executed in a predefined order. Notably, each operation is not limited to a single machine but can be performed on any machine within its allowable set, which adds flexibility to the scheduling process while also increasing its complexity.

For each job, its series of operations must be carried out in a strictly sequential manner, meaning that subsequent operations can only begin once all preceding operations have been completed. This dependency is defined by precedence constraints between operations. Furthermore, each operation can be executed on a designated set of machines, depending on the technical requirements of the operation and the capabilities of the machines. For example, some operations may require specific tools or equipment that are only available on certain machines. (Dauzère-Pérès, S *et al*. 2024, Demir, Y *et al*. 2014)

The Definition of Notation of the paper is shown as follows:

- O: The operation set, the index used include j, g;
- n_i: The number of operations of job i;
- M_k: k -th machine;
- O_{ij}: The j -th operation of i -th job;
- M_{ij}: The processing machine for O_{ij};
- C_{ij}: The completion time of operation O_{ij}.

Minimizing the maximum completion time (makespan) in the FJSP is a critical performance indicator that significantly enhances production efficiency and resource utilization. A shorter makespan leads to more streamlined production planning, reducing idle times and improving throughput. This not only maximizes the use of machines and labor but also reduces operational costs, including energy, maintenance, and labor expenses. Additionally, a minimized makespan can improve customer satisfaction by enabling faster delivery times, which is essential in today's competitive market. Overall, optimizing the makespan is key to achieving higher productivity, cost savings, and better responsiveness to market demands.

The formula for minimizing the makespan is shown in Equation 1.

$$f = \min\left(max\left(C_{i,n_i}\right)\right) \tag{1}$$

4. THE PROPOSED APPROACH FOR FJSP

The approach adopted in this paper is divided into two phases. In the first phase, the NGO is utilized, primarily leveraging its global search capabilities. In the second phase, a reinforcement learning algorithm is employed, primarily focusing on its local search abilities.

4.1 The first stage - Northern Goshawk optimization algorithm

4.1.1 Basic Algorithm Description

Mohammad *et al*. (2021) were inspired by the natural hunting behavior of the northern goshawk and proposed the NGO. The algorithm is based on two behaviors exhibited by the northern goshawk during hunting: prey identification and pursuit-evasion.

The ub and lb represent the upper and lower position boundaries of the northern goshawk, the initialization of the population positions in the NGO is represented by Equation 2.

$$x = lb + rand * (ub - lb) \tag{2}$$

Identification and Attack: This involves randomly selecting prey in the search space to enhance the algorithm's exploration capabilities, aiming at a global search of the space. r and I are random numbers, F_i is fitness, $x_{i,j}$ is the individual position of goshawk, $p_{i,j}$ is the individual position of prey. X_i is the population position of Goshawk and $F_i^{new,P1}$ is the objective function value of the first stage. As shown in Equation 3, and Equation 4 is the updated position according to whether the fitness is reduced.

$$x_{i,j}^{new,P1} = \begin{cases} x_{i,j} + r\left(p_{i,j} - Ix_{i,j}\right), F_{Pi} < F_i \\ x_{i,j} + r\left(x_{i,j} - p_{i,j}\right), F_{Pi} \geq F_i \end{cases} \tag{3}$$

$$X_i = \begin{cases} X_i^{new,P1}, F_i^{new,P1} < F_i \\ X_i, \qquad F_i^{new,P1} \geq F_i \end{cases} \tag{4}$$

Chase and Evasion: During the hunting process, once the northern goshawk attacks, the prey tries to escape. This ongoing chase and evasion require the goshawk to track its target continuously in various complex environments until it's finally caught. This natural hunting pattern is skillfully simulated in algorithm design, enhancing the algorithm's ability to conduct deep searches in local areas of the search space and significantly improving search efficiency and accuracy. The t is the iteration counter, T is the maximum number of iterations, and $F_i^{new,P2}$ is the objective function value of the second stage, the update formulas for this phase are represented by Equation 5 to Equation 7.

$$R = 0.02(1 - t/T) \tag{5}$$

$$x_{i,j}^{new,P1} = x_{i,j} + R(2r - 1)x_{i,j} \tag{6}$$

$$X_i = \begin{cases} X_i^{new,P2}, F_i^{new,P2} < F_i \\ X_i, \qquad F_i^{new,P2} \geq F_i \end{cases} \tag{7}$$

4.1.2 Combination of NGO and FJSP

Coding mechanism: This paper uses a two-segment encoding with random keys for these jobs. For a workshop with 4 jobs, each having 2 operations and element values ranging from -4 to 4, the individual position vector of NGO can be expressed as $X = \{x(1), x(2), \cdots, x(l), \cdots, x(2l)\}$, illustrated in Figure 1.

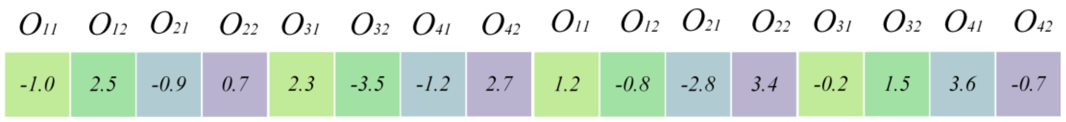
O_{11} O_{12} O_{21} O_{22} O_{31} O_{32} O_{41} O_{42} O_{11} O_{12} O_{21} O_{22} O_{31} O_{32} O_{41} O_{42}
-1.0 2.5 -0.9 0.7 2.3 -3.5 -1.2 2.7 1.2 -0.8 -2.8 3.4 -0.2 1.5 3.6 -0.7

Figure 1. Individual position vector

Conversion mechanism: This paper adopts the methods from references (Yuan Y *et al*., .2013) and (Wang *et al*.,2008) to achieve the conversion between continuous and discrete values.

(1) Conversion of continuous individual position values to discrete scheduling solutions

(a) Machine selection: Convert the continuous values of the individual position into indices of selectable machines for each operation, obtaining the machine numbers, $z(h)$ represents the number of optional machines for operation h; $u(h) \in [1, z(h)]$ represents the serial number of the selected machine in the process optional machine, as shown in Equation 8.

$$u(h) = round\left(\frac{x(h) + \epsilon}{2\epsilon}(z(h) + 1) + 1\right), 1 \leq h \leq l \tag{8}$$

(b) Operation sequencing: Each element is assigned a unique rank order value (ROV) using the ascending order sorting rules. Then, based on the ROV values, an operation sequencing scheme is constructed, as illustrated in Figure 2.

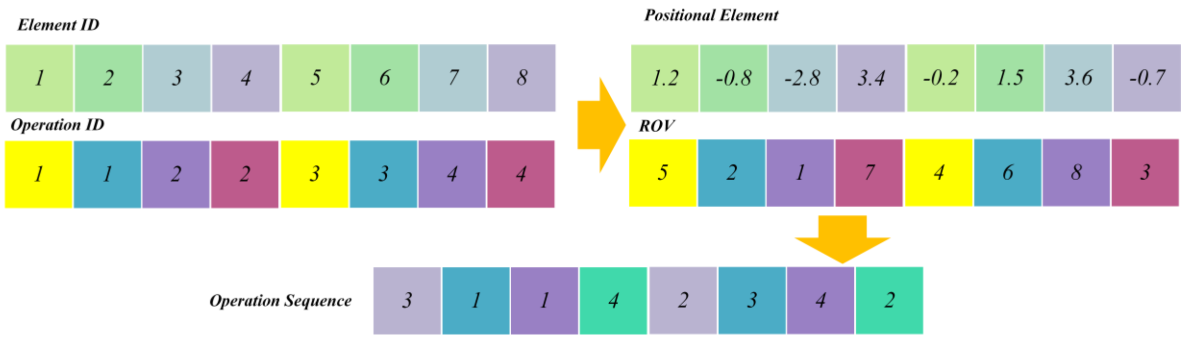

Figure 2. Position values to scheduling solutions

(2) Conversion of discrete scheduling values to continuous individual positions

(a) Machine selection: Select the machine according to Equation 9.

$$x(h) = \frac{2\epsilon}{z(h) - 1}(u(h) - 1) - \epsilon, z(h) \neq 1 \tag{9}$$

If $z(h) = 1$, indicating that an operation has only one selectable machine, then $x(h)$ can take any value within $[-\epsilon, \epsilon]$.

(b) Operation sequencing: Generate l random numbers within $[-\epsilon, \epsilon]$, assign each ROV in ascending order for operation sequencing, and determine the individual position vector element values based on the operation numbers. As is shown in Figure 3.

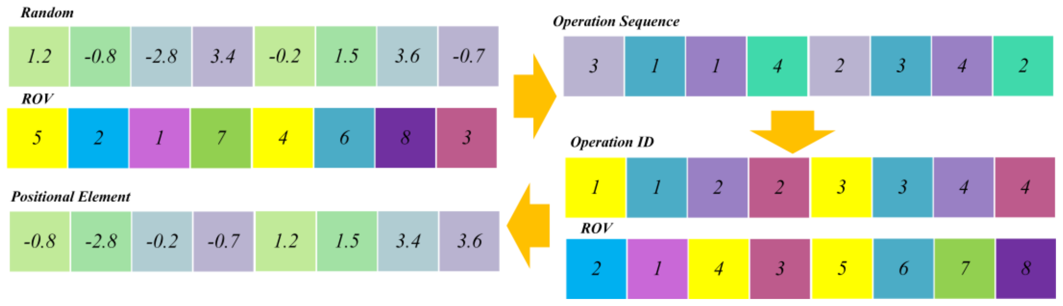

Figure 3. Scheduling values to position values

Population Initialization and Fitness Calculation: The method used in this paper for initializing the population adopts a random machine and random operation approach. For the operation part, the operations are obtained in the order of job numbers. For the machine selection part, an optional machine list is generated for each operation, and the machines are randomly selected for each operation. The fitness calculation involves determining the makespan for each individual. For a given job sequence, first, select a job and determine the machine and processing time based on machine availability. Then, find the earliest available slot for the machine. Repeat until all jobs are processed, and calculate the final completion time.

Algorithm Solution Steps:

Step 1: Initialize the maximum iteration count T and the number of individuals in the initial population N;
Step 2: Calculate the fitness function;
Step 3: Convert the initial population into the individual positions of the northern goshawk;
Step 4: Phase 1: Randomly select a prey and then update the individual positions according to Equation 3 and Equation 4;
Step 5: Phase 2: Update R based on Equation 5, and then update the individual positions using Equation 6 and Equation 7;
Step 6: Convert the individual positions of the northern goshawk into the operation sequencing process and select the machines;
Step 7: Determine if the algorithm meets the termination condition. If it does, proceed to Step 8; otherwise, go back to Step 2;
Step 8: The algorithm ends and outputs the results. The flowchart is shown in Figure 4.

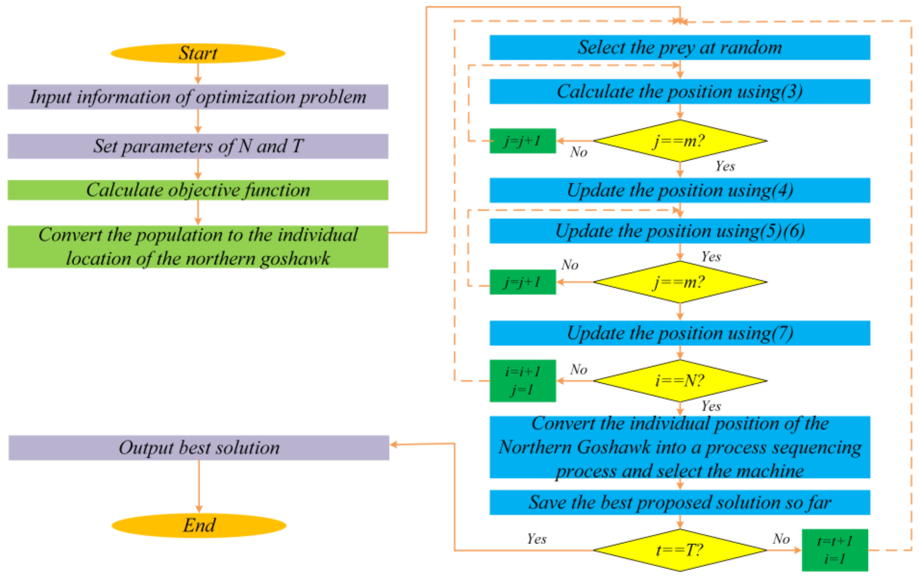
Start
Input information of optimization problem
Set parameters of N and T
Calculate objective function
Convert the population to the individual location of the northern goshawk
Output best solution
End
Select the prey at random
Calculate the position using(3)
j==m?
j=j+1
No
Yes
Update the position using(4)
Update the position using(5)(6)
j==m?
No
Yes
j=j+1
Update the position using(7)
i==N?
i=i+1
j=1
No
Yes
Convert the individual position of the Northern Goshawk into a process sequencing process and select the machine
Save the best proposed solution so far
t==T?
Yes
No
t=t+1
i=1

Figure 4. NGO algorithm flowchart

4.2 The Second Stage - Reinforcement Learning Algorithm

4.2.1 Basic Description

Reinforcement learning enables an agent to achieve a goal by learning an optimal policy through interactions with its environment. The agent explores different actions to receive reward signals from the environment, subsequently adjusting its policy to maximize the total return. Reinforcement learning comprises an agent, an environment, a state s_t, an action a_t, a reward $r(t)$, and a policy p_t. This process can be described as a Markov decision process (MDP). (Lei, *et al*., 2023, Li, *et al*., 2023). The policy can be represented by a state transition probability function p, as shown in Equation 10.

$$p_t(s_{t+1}|s_t, a) = p(S_{t+1} = s_{t+1}|S_t = s_t, A_t = a) \tag{10}$$

The key to adopting reinforcement learning is determining the policy, denoted by π. The evaluation of a policy's effectiveness is primarily determined by the cumulative reward. In reinforcement learning, the value of a policy is often derived by calculating the expected cumulative reward. This value comprises the action-value function and the state-value function, represented by q and v, respectively, U_t represents the cumulative reward, and E stands for taking the expectation, As is shown in Equation 11.

$$\begin{cases} \pi(a|s) = p(A = a|S = s) & (a) \\ q(s_t, a_t) = E(U_t|S_t = s_t, A_t = a) & (b) \\ v(s_t) = E\big(q(s_t, A_t)\big) & (c) \end{cases} \tag{11}$$

The goal of reinforcement learning is to enable the agent to learn an optimal policy through interactions with the environment, such that the agent can select the optimal action in any state to maximize the cumulative reward. In this paper, the value function approach of reinforcement learning is chosen to address the FJSP.

The SARSA algorithm (Zhao *et al*., 2016) is an on-policy method, and its iterative expression is as follows:

$$q(s_t, a_t) \leftarrow q(s_t, a_t) + \alpha[r_{t+1} + \gamma\, q(s_{t+1}, a_{t+1}) - q\,(s_t, a_t)] \tag{12}$$

where, a_t and a_{t+1} are both selected using an ε-greedy policy based on the q-values.

Q-learning (Clifton *et al*., 2020) is an off-policy method, and its iterative update equation is expressed as follows:

$$q(s_t, a_t) \leftarrow q(s_t, a_t) + \alpha\left[r_{t+1} + \gamma \max_{a\in A(s_{t+1})} q(s_{t+1}, a) - q\,(s_t, a)\right] \tag{13}$$

where, a_t is sampled using an ε-greedy method based on the q-values, but a is chosen greedily. This is where Q-learning differs from SARSA: the behavior policy is ε-greedy, while the target policy is greedy.

4.2.2 Converting FJSP to MDP

State: The state design follows the reference (Shahrabi *et al*., 2017) with improvements, focusing on minimizing the maximum completion time, which also serves as the fitness for the NGO scenario. The final state value is determined by the minimum and average completion times of the population, using three equations, as shown in Equation 14.

$$\begin{cases} fit_{new}^t = \dfrac{\sum_{i=1}^N fit(x_i^t)}{N} & (a) \\ max_{new}^t = maxfit(x_i^t) & (b) \\ S_{new} = w_1 * \dfrac{fit_{new}^t}{fit_{new}^1} + w_2 * \dfrac{max_{new}^t}{max_{new}^1} & (c) \end{cases} \tag{14}$$

where $fit(x_i^t)$ represents the fitness of individual i in the t -th generation, fit_{new}^t is the average fitness of the t -th generation, and $maxfit(x_i^t)$ is the fitness of the optimal individual in the t -th generation. Weights w_1 and w_2 are both set to 0.5. S_{new} represents the state. To balance representation and exploration, 10 states are selected, denoted as $S_{new} = \{S_{new}(1), S_{new}(2), S_{new}(3), \cdots, S_{new}(10)\}$. The interval values of S_{new} are set to 0.1, i.e., if $S_{new} \in [0,0.1), S_{new} = S_{new}(1); S_{new} \in [0.1,0.2), S_{new} = S_{new}(2)$ and so on.

Action: This paper redesigns six rules as actions, and one of them is randomly selected when choosing an action.

Rule 1: Randomly select an operation sequence, then swap two randomly chosen elements corresponding to different jobs; Rule 2: Select two elements A and B (A before B) from the operation sequence, remove B, and insert it before A, maintaining the relative order of other elements; Rule 3: Randomly select start and end positions in the operation sequence, reverse the elements between these positions, and reinsert them; Rule 4: Randomly select two parent operation sequences, divide them into subsets, swap the genes of jobs in one subset between parents, and choose one offspring as the new solution based on a random number; Rule 5: Randomly select an operation, find available machines and their processing times, and select the machine with the minimum processing time; Rule 6: Initialize relevant parameters, iterate through operations to update machine and job statuses, and for the job with the maximum completion time, select the machine with the minimum processing time for its last operation.

Reward: The reward function aims to maximize the immediate impact of actions, with cumulative rewards aligning with the optimization direction of the objective function. In solving the FJSP, the most common reward function is based on single-step transition time. This paper's reward function follows this approach, as is shown in Equation 15.

$$r(t) = \frac{maxfit(x_i^t) - maxfit(x_i^{t-1})}{maxfit(x_i^{t-1})} \tag{15}$$

where， $maxfit(x_i^t)$ is the fitness of the optimal individual of the t-th generation population.

Policy: The Q-learning policy and SARSA policy is selected using Equation 16.

$$\pi(s_t, a_t) = \begin{cases} a(Random\ 6\ rules) & Random < \varepsilon \\ max_a Q(s_t, a) & Random \geq \varepsilon \end{cases} \tag{16}$$

The reinforcement learning algorithm adopted in this paper is a combination of Q-learning and SARSA algorithm, referred to as QLSA. Select by setting a range, as is shown in Equation 17.

$$QLSA = \begin{cases} SARSA & t < (T-t) * snum * anum * 0.01 \\ Q-learning & t \geq (T-t) * snum * anum * 0.01 \end{cases} \tag{17}$$

where t is the t-th generation population, that is, the t-th iteration; T is the total number of iterations; *snum* is the state dimension; *anum* is the action dimension. The algorithm flow is shown in Figure 5.

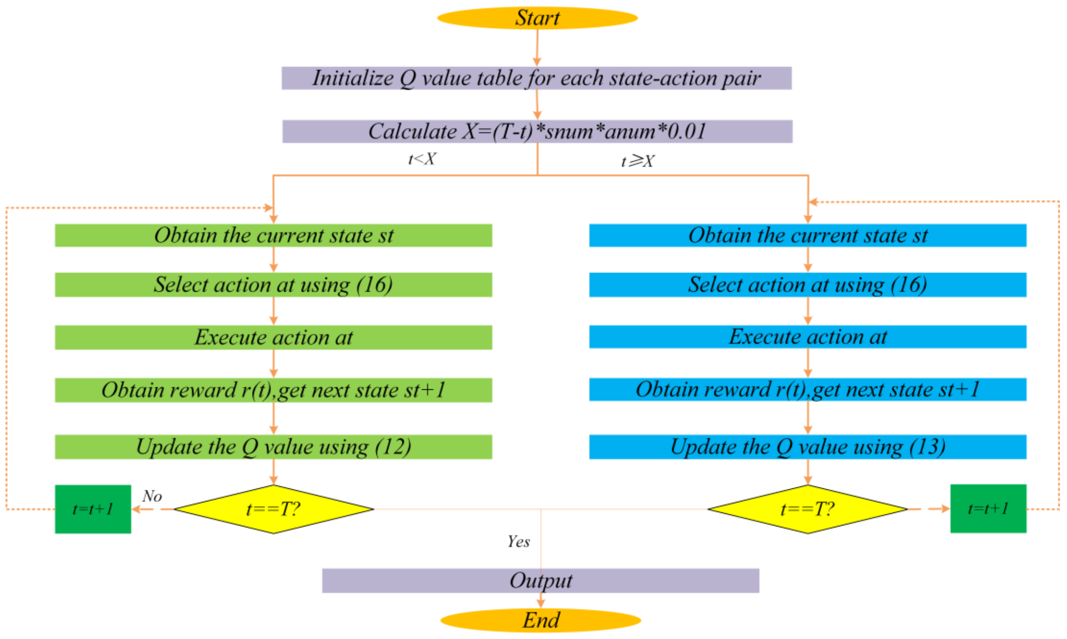
Start
Initialize Q value table for each state-action pair
Calculate X=(T-t)*snum*anum*0.01
t<X
t≥X
Obtain the current state st
Select action at using (16)
Execute action at
Obtain reward r(t),get next state st+1
Update the Q value using (12)
Obtain the current state st
Select action at using (16)
Execute action at
Obtain reward r(t),get next state st+1
Update the Q value using (13)
t=t+1
No
t==T?
t==T?
t=t+1
Yes
Output
End

Figure 5. QLSA flowchart

4.3 Overall Algorithm Flow

The algorithm ultimately adopted in this paper to solve the FJSP is a combination of the NGO and QLSA algorithms, referred to as NGO-QLSA.

```
NGO-QLSA Algorithm
1    Initialize population size N and number of iterations T; Initialize Q value table for each state-action pair; Initialize the transition conditions between NGO and QLSA: stagnate limit=round(T*0.3); Initialize the discount factor $\gamma$,  learning rate $\alpha$,  greed factor $\varepsilon$;
2    While 1: T:
3        If  t < stagnate limit:
4            Use NGO for global search;
5        else:
6            if  t<(T-t)*snum*anum*0.01:
7                Use SARSA for local search;
8            else:
9                Use Q-learning for local search;
10           End if;
11       End if;
12   End While.
```

5. EXPERIMENTAL RESULTS

5.1 Experimental preparation

To validate the proposed algorithm for solving the FJSP, this paper uses internationally recognized benchmark instances for experimental numerical analysis. Four groups of instances are selected (Behnke D *et al*., 2012):

1. The first 10 problem instances from Brandimarte's FJSP benchmark showcase moderate flexibility.
2. 18 job shop problem instances by Dauzère-Pérès and Paulli, adapted for the FJSP to test broader scenarios.
3. 21 FJSP-specific instances by Chambers and Barnes serving as key references.
4. 20 small-to-medium-sized FJSP instances by Fattahi *et al*., including 10 small-scale and 10 medium-scale instances.

These benchmarks provide a comprehensive dataset for evaluating the algorithm's performance across different problem sizes. The algorithm is implemented in MATLAB R2024a on a platform with an Intel Core i5-10600KF CPU and an NVIDIA GeForce GTX 1050 Ti GPU, running Windows 10 64-bit, ensuring a suitable environment for training and testing.

5.2 Reinforcement learning hyperparameter selection

In the NGO algorithm used in this paper, the population size is set to N=100. Due to the adaptive process of reinforcement learning, the hyperparameters of reinforcement learning need to be determined, including the discount factor γ, learning rate α, and greed factor ε. Three combinations are set to test the convergence rate of the algorithm, specifically: Combination 1: [γ=0.9, α=0.3, ε=0.9], Combination 2: [γ=0.8, α=0.2, ε=0.9], and Combination 3: [γ=0.7, α=0.15, ε=0.9].

Four benchmark instances are selected for experimentation: Mk01 from Brandimarte, 10a from Dauzère-Pérès, mt10c1 from Barnes, and Fattahi20 from Fattahi. Each instance is tested using the aforementioned three combinations. Each instance is trained 10 times under each combination, and Figure 6 shows one such result. (Combination 1 is blue, Combination 2 is red, and Combination 3 is yellow).

From the iteration plots, it is evident that combinations 2 and 3 exhibit more significant advantages in terms of iterative convergence. Considering the Markovian nature of reinforcement learning, which underscores the immediate state's direct influence on subsequent states, a smaller discount factor is preferred to place greater emphasis on immediate rewards. This approach facilitates quicker adaptation to changes in the environment, aligning well with the principles of reinforcement learning. Moreover, by reducing the learning rate, we ensure policy updates occur in a more stable and gradual manner, thus preventing substantial fluctuations that can arise from overly aggressive learning rates.

Taking these factors into account, this paper opts for combination 3. This choice is driven not only by its superior performance in iterative convergence but also by its ability to meet the stability requirements critical for practical applications. Combination 3 stands out as an optimal solution for enhancing the performance of the learning system. It achieves a balance between improving efficiency and ensuring reliability, providing a solid foundation for the practical deployment of reinforcement learning models. Therefore, combination 3 represents an ideal selection for optimizing the model's learning process and boosting its effectiveness in real-world scenarios.

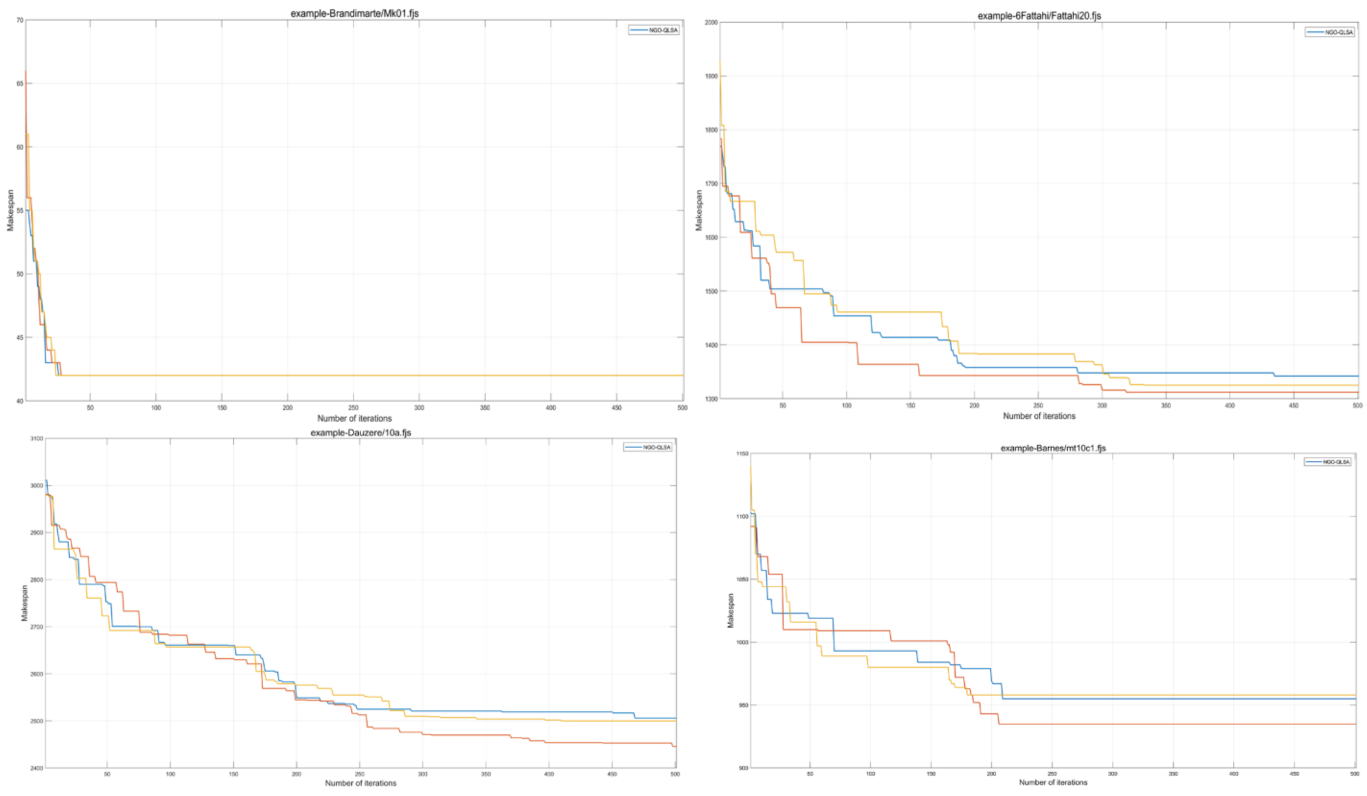

Figure 6. Hyperparameter selection iteration plots

5.3 Experimental Results

5.3.1 Percentage Deviation

Define the best percentage deviation (BPD) as the percentage deviation of C_{max} relative to the upper baseline (UB) (Behnke *et al.*,2012), with the formulas given in Equation 18.

$$BPD = \frac{C_{max} - UB}{UB} \times 100\% \quad (18)$$

where C_{max} is the makespan. To further validate the effectiveness of the proposed framework, this paper also conducts ablation studies, solving the FJSP using only the NGO and only the QLSA.

5.3.2 Brandimarte instance results and comparison

Select the first 10 problem instances from Brandimarte's generalized FJSP benchmarks, recording the C_{max}. Compare the algorithm proposed in this paper with the algorithm (GA-Q, GA-SARSA, SLGA) from reference (Chen *et al.*, 2020), the algorithm (MPGN-multi-PPO) from reference (Lei *et al.*, 2022), the algorithm (GNN-PPO) from reference (Yuan *et al.*, 2024). The first value is the makespan, and the second is the BPD (%) (same below), as shown in Table 1. (In this instance, UB refers to the theoretical optimal value).

Table 1. The results and comparison of Brandimarte instances

Instance	UB	GA-Q	GA-SARSA	SLGA	GNN-PPO	MPGN-multi-PPO	NGO	QLSA	NGO-QLSA
Mk01(10*6)	36	42	41	**40***	44	47	42	43	42
		16.7	13.9	**11.1**	22.2	30.6	16.7	19.4	16.7
Mk02(10*6)	24	31	30	27	31	30	29	31	**27***
		29.2	25.0	12.5	29.2	25.0	20.8	29.2	**12.5**
Mk03(15*8)	204	211	205	204	211	204	204	213	**204***
		3.4	0.5	0	3.4	0	0	4.4	**0**
Mk04(15*8)	48	75	67	**60***	78	76	67	81	66
		56.3	39.6	**25.0**	62.5	58.3	39.6	68.8	37.5
Mk05(15*4)	168	177	176	172	183	178	177	180	**172***
		5.4	4.8	2.4	8.9	6.0	5.4	7.1	**2.4**
Mk06(10*10)	33	73	72	69	74	79	67	87	**67***
		121	118	109	124	139	103	164	**103**
Mk07(20*5)	133	155	151	144	156	152	150	150	**143***
		16.5	13.5	8.3	17.3	14.3	12.8	12.8	**7.5**
Mk08(20*10)	523	526	533	523	524	541	523	524	**523***
		0.6	1.9	0	0.2	3.4	0	0.2	**0**
Mk09(20*10)	299	342	338	320	326	335	332	345	**311***
		14.4	13.0	7.0	9.0	12.0	11.0	15.4	**4.0**
Mk10(20*15)	165	281	278	254	241	236	250	294	**224***
		70.3	68.5	53.9	46.1	43.0	51.5	78.2	**35.8**

By comparing the performance of the algorithms on the 10 instances, we can observe the following points:

In multiple instances, the NGO-QLSA algorithm found superior completion times, indicating that this algorithm has better optimization capabilities in solving the FJSPs. When comparing NGO and QLSA in solving the FJSPs, NGO performed better on these 10 instances, indicating that global search played a significant role. Together with local search, this combination led to the superior performance of NGO-QLSA.

Table 2 shows the optimal solutions compared to a collaborative agent reinforcement learning (CARL), genetic algorithm (GA), particle swarm optimization (PSO) in reference (Zhang *et al*., 2024), as well as two scales mentioned in reference (Song *et al*., 2022) and construction scheduling rules (Heuristic) (Ziaee, M *et al*., 2014), a multi-agent model algorithm based on chemical reaction optimization and greedy meta-heuristics (MACROG) (Marzouki *et al*., 2017)

Table 2. The results and comparison of Brandimarte instances

Instance	UB	GA	PSO	Song 20×5	Song 20×10	Heuristic	MAC-ROG	CARL	NGO-QLSA
Mk01(10*6)	36	42	46	55	48	42	**40***	44	42
		16.7	27.8	52.8	33.3	16.7	**11.1**	22.2	16.7
Mk02(10*6)	24	41	35	42	43	28	32	31	**27***
		70.8	45.8	75	79.2	16.7	33.3	29.2	**12.5**
Mk03(15*8)	204	238	212	232	213	204	204	207	**204***
		16.7	3.9	13.7	4.4	0	0	1.5	**0**
Mk04(15*8)	48	74	71	82	75	75	**64***	69	66
		54.2	47.9	70.8	56.3	56.3	**33.3**	43.8	37.5
Mk05(15*4)	168	188	185	205	185	179	179	177	**172***
		11.9	10.1	22.0	10.1	6.6	6.6	5.4	**2.4**
Mk06(10*10)	33	115	98	110	103	69	85	77	**67***
		248	197	233	212	109	158	133	**103**
Mk07(20*5)	133	183	176	215	214	154	172	151	**143***
		37.6	32.3	61.7	60.9	15.8	29.3	13.5	**7.5**
Mk08(20*10)	523	523	557	525	523	555	552	531	**523***

Instance	UB	GA	PSO	Song 20×5	Song 20×10	Heuristic	MAC-ROG	CARL	NGO-QLSA
		0	6.5	0.4	0	6.1	5.5	1.5	**0**
Mk09(20*10)	299	361	345	424	333	342	421	334	**311***
		20.7	15.4	41.8	11.4	14.4	40.8	11.7	**4.0**
Mk10(20*15)	165	319	247	266	266	242	358	**186***	224
		93.3	49.7	61.2	61.2	46.7	117	**12.7**	35.8

After conducting the aforementioned detailed comparative analysis, we can clearly see that the algorithm proposed in this paper exhibits significant advantages in performance compared to other reinforcement learning algorithms and intelligent algorithms. This remarkable superiority not only demonstrates the exceptional efficacy of our algorithm but also fully validates its high effectiveness and reliability.

5.3.3 Dauzère instance results and comparison

We have selected 18 problem instances from the generalized FJSP benchmarks proposed by Dauzère and the C_{max} was recorded. Our algorithm is compared with the UB in reference (BehnkeD *et al*., 2012), the algorithm (CARL) from reference (Zhang *et al*., 2024), the algorithm (MPGN-multi-PPO) from reference (Lei *et al*., 2022), the algorithm (GNN-PPO) from reference (Yuan *et al*., 2024), NGO and QLSA. Table 3 displays the results.

Table 3. The results and comparison of Dauzère instances

Instance	UB	CARL	MPGN-multi-PPO	GNN-PPO	NGO	QLSA	NGO-QLSA
01a	2518	2980	3109	2947	2697	2855	**2604***
		18.4	23.5	17.0	7.1	13.4	**3.4**
02a	2231	2483	2603	2495	2385	2509	**2335***
		11.3	16.7	11.8	6.9	12.5	**4.7**
03a	2229	2366	2366	2320	2423	2472	**2280***
		6.2	6.2	4.1	8.7	10.9	**2.3**
04a	2503	3055	3093	3037	2640	2758	**2569***
		22.1	23.6	21.3	5.5	10.2	**2.6**
05a	2216	2422	2573	2496	2428	2421	**2333***
		9.3	16.1	12.6	9.6	9.3	**5.3**
06a	2196	2330	2508	2286	2347	2375	**2283***
		6.1	14.2	4.1	6.9	8.2	**4.0**
07a	2283	2854	2857	2820	2566	2629	**2478***
		25.0	25.1	23.5	12.4	15.2	**8.5**
08a	2069	2327	2386	**2259***	2588	2417	2365
		12.5	15.3	**9.2**	25.1	16.8	14.3
09a	2066	2171	2209	**2155***	2299	2508	2361
		5.1	6.9	**4.3**	11.3	21.4	14.3
10a	2291	2819	2805	2721	2594	2658	**2497***
		23.1	22.4	18.8	13.2	16.0	**9.0**
11a	2063	2352	2356	**2206***	2596	2414	2312
		14.0	14.2	**6.9**	25.8	17.0	12.1
12a	2030	**2147***	2186	2157	2348	2344	2272
		5.8	7.7	6.3	15.7	15.5	11.9
13a	2257	2836	2811	2755	2643	2768	**2514***
		25.6	24.6	22.1	17.1	22.6	**11.4**
14a	2167	2351	2353	**2277***	2553	2657	2471
		8.5	8.6	**5.1**	17.8	22.6	14.0
15a	2165	2301	2385	**2202***	2528	2661	2521
		6.3	10.2	**1.7**	16.8	22.9	16.4

Instance	UB	CARL	MPGN-multi-PPO	GNN-PPO	NGO	QLSA	NGO-QLSA
16a	2255	2815	2861	2680	2661	2754	**2510***
		24.8	26.9	18.9	18.0	22.1	**11.3**
17a	2140	2392	2365	**2309***	2648	2614	2408
		11.8	10.5	**7.9**	23.7	22.2	12.5
18a	2127	2242	2363	**2216***	2693	2564	2508
		5.4	11.1	**4.2**	26.6	20.6	17.9

After a thorough comparative analysis, it is evident that our proposed algorithm demonstrates significant advantages over the algorithms in 10 out of the 18 instances. This result strongly validates the superiority of our algorithm and its efficient optimization capabilities in solving the FJSPs. When comparing NGO and QLSA in solving the FJSPs, NGO performed better on these 18 instances, similar to the previous performance, indicating that global search played a significant role. However, without local search, the solution quality was not as advantageous. Therefore, the combination of the NGO's global search with QLSA's local search contributed to the superior performance of NGO-QLSA.

5.3.4 Barnes instance results and comparison

The 21 instances specifically constructed by Chambers and Barnes for the FJSP were used in this paper, and the C_{max} was recorded. Our results are compared against the UB from reference (Behnke D *et al*., 2012), the algorithm (MPGN-multi-PPO) from reference (Lei *et al*., 2022), the algorithm (GNN-PPO) from reference (Yuan *et al*., 2024), NGO and QLSA. The outcomes are displayed in Table 4.

Table 4. The results and comparison of Barnes's instances

Instance	UB	MPGN-multi-PPO	GNN-PPO	NGO	QLSA	NGO-QLSA
mt10x	918	1176	1084	982	960	**937***
		28.1	18.1	7.0	4.6	**2.1**
mt10xx	918	1176	1084	982	**940***	948
		28.1	18.1	7.0	**2.4**	3.3
mt10xxx	918	1176	1084	1001	935	**934***
		28.1	18.1	9.0	1.9	**1.7**
mt10xy	905	1148	1055	973	967	**936***
		26.9	16.6	7.5	6.9	**3.4**
mt10xyz	847	1180	1020	923	921	**873***
		39.3	20.4	9.0	8.7	**3.1**
mt10c1	927	1240	1096	988	968	**950***
		33.8	18.2	6.6	4.4	**2.5**
mt10cc	910	1226	1068	961	945	**919***
		34.7	17.4	5.6	3.9	**1.0**
setb4x	925	1210	1022	1027	974	**967***
		30.8	10.5	11.0	5.3	**4.5**
setb4xx	925	1210	1015	991	973	**941***
		30.8	9.7	7.1	5.2	**1.7**
setb4xxx	925	1210	1040	1013	990	**961***
		30.8	12.4	9.5	7.0	**3.9**
setb4xy	916	1111	1019	993	963	**946***
		21.3	11.2	8.4	5.1	**3.3**
setb4xyz	905	1095	1000	964	981	**935***
		21.0	10.5	6.5	8.4	**3.3**
setb4c9	914	1225	1037	1000	990	**978***
		34.0	13.5	9.4	8.3	**7.0**
setb4cc	909	1180	993	991	951	**944***

Instance	UB	MPGN-multi-PPO	GNN-PPO	NGO	QLSA	NGO-QLSA
		29.8	9.2	9.0	4.6	**3.9**
seti5x	1201	1592	1363	1310	1291	**1273***
		32.6	13.5	9.1	7.5	**6.0**
seti5xx	1199	1592	1344	1323	1303	**1264***
		32.8	12.1	10.3	8.7	**5.4**
seti5xxx	1197	1592	1340	1317	1310	**1273***
		33.0	12.0	10.0	9.4	**6.4**
seti5xy	1136	1438	1271	1273	1284	**1204***
		26.6	11.9	12.1	13.0	**6.0**
seti5xyz	1125	1298	1254	1244	1248	**1182***
		15.4	11.5	10.6	10.9	**5.1**
seti5c12	1174	1467	1346	1270	1273	**1234***
		25.0	14.7	8.2	8.4	**5.1**
seti5cc	1136	1438	1271	1257	1225	**1198***
		26.6	11.9	10.7	7.8	**5.5**

After a thorough comparative analysis, our proposed NGO-QLSA algorithm, when solving these 21 instances, demonstrates a consistent gap between the obtained results and the known UB ranging from 1.0% to 7.0%, and it is evident that our proposed algorithm demonstrates significant advantages over the algorithms in 20 out of the 21 instances. This significant performance fully proves the remarkable optimization capability of the NGO-QLSA in addressing the FJSPs. When comparing NGO and QLSA in solving the FJSPs, QLSA performed better on these 21 instances, which is different from the previous observations, indicating that local search played a significant role. However, without a global search, the solution's effectiveness was not optimal. Therefore, the combination of the NGO's global search with QLSA's local search contributed to the superior performance of NGO-QLSA.

5.3.5 Fattahi instance results and comparison

The 20 randomly generated small-to-medium-sized FJSP instances introduced by Fattahi *et al*. have also been included in our testing. Our results are compared against the UB from the reference (Behnke *et al*., 2012) and the algorithm from the reference (Marzouki *et al*., 2017). Table 5 displays the outcomes.

Table 5. The results and comparison of Fattahi instances

Instance	UB	MACRO	MACROG	NGO	QLSA	NGO-QLSA
Fattahi1	66	66	66	66	66	**66***
		0	0	0	0	**0**
Fattahi2	107	107	107	107	107	**107***
		0	0	0	0	**0**
Fattahi3	221	221	221	221	221	**221***
		0	0	0	0	**0**
Fattahi4	355	355	355	355	355	**355***
		0	0	0	0	**0**
Fattahi5	119	119	119	119	119	**119***
		0	0	0	0	**0**
Fattahi6	320	320	320	320	320	**320***
		0	0	0	0	**0**
Fattahi7	397	397	397	397	397	**397***
		0	0	0	0	**0**
Fattahi8	253	253	253	253	263	**253***
		0	0	0	4.0	**0**
Fattahi9	210	210	210	210	220	**210***

Instance	**UB**	**MACRO**	**MACROG**	**NGO**	**QLSA**	**NGO-QLSA**
		0	0	0	4.8	**0**
Fattahi10	516	516	516	516	516	**516***
		0	0	0	0	**0**
Fattahi11	468	477	470	469	530	**468***
		1.9	0.4	0.2	13.3	**0**
Fattahi12	446	464	464	448	469	**448***
		4.0	4.0	0.5	5.2	**0.4**
Fattahi13	466	578	564	**480***	498	505
		24.0	21.0	**3.0**	6.9	8.4
Fattahi14	554	745	745	580	648	**565***
		34.5	34.5	4.7	17.0	**2.0**
Fattahi15	514	708	696	551	610	**530***
		37.7	35.4	7.2	18.7	**3.1**
Fattahi16	635	836	836	662	704	**637***
		31.7	31.7	4.3	10.9	**0.3**
Fattahi17	879	1465	1454	910	1104	**890***
		66.7	65.4	3.5	25.6	**1.3**
Fattahi18	884	1934	1934	962	1032	**884***
		119	119	8.8	16.7	**0**
Fattahi19	1088	2965	2953	1178	1257	**1125***
		173	171	8.3	15.5	**3.4**
Fattahi20	1267	5223	5223	1380	1486	**1294***
		312	312	8.9	17.3	**2.1**

After a comprehensive comparative analysis, it is clearly evident that the NGO-QLSA proposed in this paper exhibits significant advantages in solving the FJSPs. This superiority not only lies in the algorithm's performance but also in its efficiency and practicality, providing new insights and effective tools for the research and application of the FJSP. When comparing NGO and QLSA in solving the FJSPs, NGO performed better on these 20 instances, indicating that global search played a significant role. However, without local search, the solution quality was not as advantageous. Therefore, the combination of the NGO's global search with QLSA's local search contributed to the superior performance of NGO-QLSA.

To demonstrate the effectiveness of our approach, we chose 07a (2478) from the Dauzère benchmarks mt10cc (919) from the Barnes benchmarks to showcase in a Gantt chart. These selected instances exhibit notable optimization results, as shown in Figure 7.

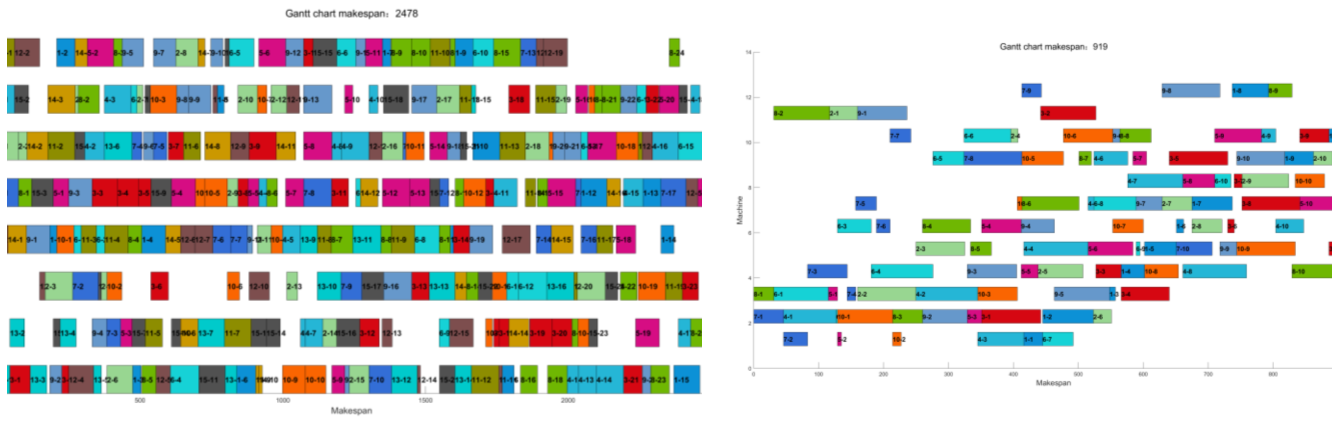

Figure 7. Gantt chart of 07a and mt10cc

6. CONCLUSION

Through in-depth research and experimental validation, this paper successfully proposes an NGO-QLSA hybrid algorithm framework for solving the FJSPs. This algorithm framework combines the global search capability of NGOs with the local search advantages of Q-learning and SARSA algorithms in reinforcement learning. Through carefully designed encoding techniques, transition mechanisms, and reward functions, it achieves efficient solutions to the FJSPs.

Experimental results on multiple standard datasets show that compared to existing advanced algorithms. The NGO-QLSA algorithm exhibits significant advantages and excellent performance in solving the FJSPs. To validate the importance of NGO's global search and QLSA's local search within this framework, the paper also conducted an ablation study. Although NGO and QLSA performed well when used individually, they were still not as effective as when integrated into the composite framework. This achievement not only proves the effectiveness and practicability of the algorithm but also provides strong support for the research and practical application of the FJSP.

The successful application of the NGO-QLSA demonstrates the potential and advantages of hybrid algorithm frameworks in solving complex optimization problems and also provides new directions for future research on the FJSP. With the increasing demand for efficient scheduling algorithms in the manufacturing industry, the NGO-QLSA is expected to be widely applied in actual production, further enhancing production efficiency and management levels. At the same time, this algorithm also provides new ideas and methods for solving other similar problems, possessing significant theoretical and practical value.

REFERENCES

Behnke, D. and Geiger, M.J. (2012). Test instances for the flexible job shop scheduling problem with work centers. Arbeitspapier/Research Paper/Helmut-Schmidt-Universität, Lehrstuhl für Betriebswirtschaftslehre, insbes. Logistik-Management.

Chen, R., Yang, B., Li, S. and Wang, S. (2020). A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem. Computers & industrial engineering, 149, pp. 106778.

Chen, R., Wu, B., Wang, H., Tong, H. and Yan, F. (2024). A Q-Learning based NSGA-II for dynamic flexible job shop scheduling with limited transportation resources. Swarm and Evolutionary Computation, 90, pp.101658.

Clifton, J. and Laber, E. (2020). Q-learning: Theory and applications. Annual Review of Statistics and Its Application, 7, pp.279-301.

Dauzère-Pérès, S., Ding, J., Shen, L. and Tamssaouet, K. (2024). The flexible job shop scheduling problem: A review. European Journal of Operational Research, 314(2), pp.409-432.

Demir, Y. and İşleyen, S.K. (2014). An effective genetic algorithm for flexible job-shop scheduling with overlap** in operations. International Journal of Production Research, 52(13), pp.3905-3921.

Dehghani, M., Hubálovský, Š. and Trojovský, P. (2021). Northern goshawk optimization: a new swarm-based algorithm for solving optimization problems. Ieee Access, 9, pp.162059-162080.

Du, Y., Li, J., Li, C. and Duan, P. (2022). A reinforcement learning approach for flexible job shop scheduling problem with crane transportation and setup times. IEEE Transactions on Neural Networks and Learning Systems.

Fan, C., Wang, W., & Tian, J. (2024). Flexible job shop scheduling with stochastic machine breakdowns by an improved tuna swarm optimization algorithm. Journal of Manufacturing Systems, 74, 180-197.

Fan, J., Zhang, C., Liu, Q., Shen, W. and Gao, L. (2022). An improved genetic algorithm for flexible job shop scheduling problem considering reconfigurable machine tools with limited auxiliary modules. Journal of Manufacturing Systems, 62, pp.650-667.

Jing, X., Yao, X., Liu, M. and Zhou, J. (2024). Multi-agent reinforcement learning based on graph convolutional network for flexible job shop scheduling. Journal of Intelligent Manufacturing, 35(1), pp.75-93.

Kong, J., & Wang, Z. (2024). Research on Flexible Job Shop Scheduling Problem with Handling and Setup Time Based on Improved Discrete Particle Swarm Algorithm. Applied Sciences, 14(6), 2586.

Lei, K., Guo, P., Zhao, W., Wang, Y., Qian, L., Meng, X. and Tang, L. (2022). A multi-action deep reinforcement learning framework for flexible Job-shop scheduling problem. Expert Systems with Applications, 205, p.117796.

Lei, K., Guo, P., Wang, Y., Zhang, J., Meng, X. and Qian, L. (2023). Large-scale dynamic scheduling for flexible job-shop with random arrivals of new jobs by hierarchical reinforcement learning. IEEE Transactions on Industrial Informatics.

Li, C., Zheng, P., Yin, Y., Wang, B. and Wang, L. (2023). Deep reinforcement learning in smart manufacturing: A review and prospects. CIRP Journal of Manufacturing Science and Technology, 40, pp.75-101.

Long, X., Zhang, J., Qi, X., Xu, W., **, T. and Zhou, K. (2022). A self-learning artificial bee colony algorithm based on reinforcement learning for a flexible job-shop scheduling problem. Concurrency and Computation: Practice and Experience, 34(4), p.e6658.

Marzouki, B., Driss, O.B. and Ghédira, K. (2017). Multi agent model based on chemical reaction optimization with greedy algorithm for flexible job shop scheduling problem. Procedia Computer Science, 112, pp.81-90.

Mei, Z., Lu, Y., & Lv, L. (2024). Research on Multi-Objective Low-Carbon Flexible Job Shop Scheduling Based on Improved NSGA-II. Machines, 12(9), 590.

Meng, L., Cheng, W., Zhang, B., Zou, W., Fang, W. and Duan, P. (2023). An improved genetic algorithm for solving the multi-AGV flexible job shop scheduling problem. Sensors, 23(8), p.3815.

Öztürk, A. M., & Lee, C. (2022). CLUSTER-BASED PRIORITY LIST GENERATION FOR RESOURCE-CONSTRAINED PROJECT SCHEDUL ING PROBLEMS. International Journal of Industrial Engineering, 29(2).

Saqlain, M., Ali, S. and Lee, J.Y. (2023). A Monte-Carlo tree search algorithm for the flexible job-shop scheduling in manufacturing systems. Flexible Services and Manufacturing Journal, 35(2), pp.548-571.

Shahrabi, J., Adibi, M.A. and Mahootchi, M. (2017). A reinforcement learning approach to parameter estimation in dynamic job shop scheduling. Computers & Industrial Engineering, 110, pp.75-82.

Shen, Y. (2022). Application of Improved Genetic Algorithm in FJSP. Academic Journal of Computing & Information Science, 5(11), pp.1-9.

Song, W., Chen, X., Li, Q. and Cao, Z. (2022). Flexible job-shop scheduling via graph neural network and deep reinforcement learning. IEEE Transactions on Industrial Informatics, 19(2), pp.1600-1610.

Wan, L., Cui, X., Zhao, H., Li, C. and Wang, Z. (2024). An effective deep actor-critic reinforcement learning method for solving the flexible job shop scheduling problem. Neural Computing and Applications, pp.1-23.

Wan, L., Fu, L., Li, C. and Li, K. (2024). Flexible job shop scheduling via deep reinforcement learning with meta-path-based heterogeneous graph neural network. Knowledge-Based Systems, p.111940.

Wang, R., Wang, G., Sun, J., Deng, F. and Chen, J. (2023). Flexible Job Shop Scheduling via Dual Attention Network-Based Reinforcement Learning. IEEE Transactions on Neural Networks and Learning Systems.

Wang L. (2008). Particle Swarm optimization and scheduling algorithms[M]. Beijing: Tsinghua University Press,117-118.

Yuan, M., Huang, H., Li, Z., Zhang, C., Pei, F. and Gu, W. (2023). A multi-agent double Deep-Q-network based on state machine and event stream for flexible job shop scheduling problem. Advanced Engineering Informatics, 58, p.102230.

Yuan, E., Wang, L., Cheng, S., Song, S., Fan, W. and Li, Y. (2024). Solving flexible job shop scheduling problems via deep reinforcement learning. Expert Systems with Applications, 245, p.123019.

Yuan, Y., Xu, H. and Yang, J. (2013). A hybrid harmony search algorithm for the flexible job shop scheduling problem. Applied soft computing, 13(7), pp.3259-3272.
Zeng, Z., Li, X. and Bai, C. (2022). A deep reinforcement learning approach to flexible job shop scheduling. In 2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC) (pp. 884-890). IEEE.

Zhao, D., Wang, H., Shao, K. and Zhu, Y. (2016). Deep reinforcement learning with experience replay based on SARSA. In 2016 IEEE symposium series on computational intelligence (SSCI) (pp. 1-6). IEEE.

Zhou, K., Tan, C., Zhao, Y., Yu, J., Zhang, Z., & Wu, Y. (2024). Research on solving flexible job shop scheduling problem based on improved GWO algorithm SS-GWO. Neural Processing Letters, 56(1), 26.

Zhang, W., Zhao, F., Li, Y., Du, C., Feng, X. and Mei, X. (2024). A novel collaborative agent reinforcement learning framework based on an attention mechanism and disjunctive graph embedding for flexible job shop scheduling problem. Journal of Manufacturing Systems, 74, pp.329-345.

Ziaee, M. (2014). A heuristic algorithm for solving flexible job shop scheduling problem. The International Journal of Advanced Manufacturing Technology, 71, pp.519-528.