



# AHOA: Adaptively Hybrid Optimization Algorithm for Flexible Job-shop Scheduling Problem

Jiaxin Ye<sup>1,2</sup>, Dejun Xu<sup>1,2</sup>, Haokai Hong<sup>1,2</sup>, Yongxuan Lai<sup>1,2</sup>,  
and Min Jiang<sup>1,2</sup>(✉)

- <sup>1</sup> School of Informatics, Xiamen University, Xiamen 361005, Fujian, China  
minjiang@xmu.edu.cn
- <sup>2</sup> Key Laboratory of Digital Protection and Intelligent Processing of Intangible Cultural Heritage of Fujian and Taiwan, Ministry of Culture and Tourism, Fujian, China

**Abstract.** The Flexible Job-shop Scheduling Problem (FJSP) is a typical scheduling problem in industrial production that is proven to be NP-hard. The Genetic Algorithm (GA) is currently one of the most widely used algorithms to address the FJSP task. The major difficulty of using GA to solve FJSP lies in how to set the key hyperparameters and improve the convergence speed. In this paper, we propose a hybrid optimization method based on Reinforcement Learning (RL) called the Adaptively Hybrid Optimization Algorithm (AHOA) to overcome these difficulties. The proposed algorithm first merges GA and improved Variable Neighborhood Search (VNS), which aims to integrate the advantages of global and local search ability into the optimization process. Then the double Q-learning offers crossover and mutation rates according to the feedback from the hybrid algorithm environment. The innovation of this work lies in that our method can adaptively modify the key hyperparameters in the genetic algorithm. Furthermore, the proposed method can avoid the large overestimations of action values in RL. The experiment is evaluated on the most widely studied FJSP instances and compared with some hybrid and self-learning algorithms including dragonfly algorithm (DA), hybrid gray wolf weed algorithm (GIWO), and self-learning GA (SLGA), etc. The results show that the proposed method outperforms the latest related algorithms by more than 12% on average.

**Keywords:** Flexible Job-shop Scheduling Problem · Hybrid algorithm · Reinforcement learning

## 1 Introduction

Production scheduling [15] plays an essential role in the planning and scheduling of the manufacturing system. There are several well-known production scheduling problems including Job shop Scheduling Problem (JSP) and Flexible Job shop Scheduling Problem (FJSP) [15]. JSP is one of the most challenging parts

of these kinds of problems and had been proved to be an NP-hard (Non-deterministic Polynomial-time Hard) problem [13]. It can be described as a set of independent jobs to be processed on multiple available machines, and each job contains a series of operations with a specified order. However, each operation must be processed on a specified machine in JSP. It is incapable of meet the flexible scheduling requirements in the real world.

The FJSP is an essential extension of JSP [4] that can address flexible requirements. It allows each operation to be processed on different machines. This means FJSP can be decomposed into two sub-problems including operations sequencing and machines selection [13]. Where, JSP is only composed of the operations sequencing problem to find the best sequence of operations and machines selection problem is a extension of JSP task to assign suitable machines to each operation.

In recent years, a great deal of algorithms have been proposed to address FJSP like Evolutionary Algorithm (EA), Local Search (LS) Algorithm, and Hybrid Optimization Algorithm (HA). For instance, Ding *et al.* [10] proposed an improved Particle Swarm Optimization (PSO) method based on novel encoding and decoding schemes for the FJSP; Amiri *et al.* [1] presented a Variable Neighborhood Search(VNS) for the FJSP; Li *et al.* [26] proposed a HA based on the Genetic Algorithm (GA) and Tabu Search (TS).

These methods are limited by hyperparameters settings. There is evidence [8] showing that hyperparameters like crossover and mutation operator play a crucial role in the evolution of the population. This means that relying on experience to select parameters will affect the efficiency and performance of the algorithm. For instance, the adverse configuration may cause premature convergence to local extreme rather than the globally optimal. To overcome this defect, many researchers often fix or update in a predetermined way [11]. But it still lacks generalization. Moreover, some methods [5] based on RL can also adjust the hyperparameters adaptively. However, the problem of overestimation is not avoided [17].

Based on these motivations, this work proposes an Adaptively Hybrid Optimization Algorithm named AHOA to search the minimum makespan of FJSP. AHOA first utilizes GA to optimize globally and then introduces improved VNS based on the critical path to optimize locally. Finally, the double Q-learning offers crossover and mutation rates according to the feedback from the hybrid algorithm environment. The contributions are as follow:

- 1) Our algorithm can adaptively modify the hyperparameters in the HA based on double Q-learning. It can also avoid the overestimations problem of action values compared to other works based on RL.
- 2) We deploy a hybrid algorithm to solve this problem effectively. It shows an outstanding performance by combining the exploration ability of GA and the exploitation ability of VNS.

The remainder of this work is organized as follows. Section 2 introduces the related work. The problem formulation is presented in Sect. 3, while the proposed algorithm is given in Sect. 4. The experimental result is proposed in Sect. 5. Finally, Sect. 6 describes the conclusion and future work.

## 2 Related Work

Since Brucker and Schlie [3] first proposed the FJSP in 1990, numerous methods have been suggested to solve the FJSP task in recent years, and most existing algorithms can be classified into two categories: GA based methods and RL based methods. The GA based methods include simple GA and HA. The simple GA is a typically global optimization method [31]. It can scale robustly and easily to complex problems that are difficult to solve by traditional optimization algorithms. For instance, De *et al.* [7] presented an improved GA for the FJSP, in which a new operator based on LS is combined. Lu *et al.* [28] proposed a GA embedded with a concise chromosome representation to solve the distributed FJSP. However, the drawbacks of the simple GA are the poor local optimization ability, and its excessively low convergence speed. For solving these problems, HA based on the GA and LS is proposed. This hybrid method can greatly improve the quality of the solution and the efficiency of optimization. Specifically, the GA optimizer always yields an approximate optimal set or population. The best individual of the set is utilized as the starting point for the local search optimizer to run with. So that, the global searching ability of GA and the local searching ability of LS are combined to solve the FJSP effectively. Wang *et al.* [33] proposed a hybrid algorithm that combines GA and TS. This approach also calls for some improvements, it limited by GA that the hyperparameters cannot be adjusted adaptively during the optimization process.

Therefore, the RL based methods are proposed to overcome this defect. The characteristic of the RL method is self-learning [30]. In other words, it can adaptively adjust the hyperparameters of GA based method. For example, Chen *et al.* [5] proposed a algorithm combined with the SARSA algorithm and Q-learning algorithm to adjust the hyperparameters. However, the problem of overestimation in RL and sparse Q-table are not avoided.

## 3 Problem Formulation

### 3.1 Problem Model

The FJSP can be classified into two types of problems including total FJSP and partial FJSP [18]. The total FJSP means each operation can be processed on every machines and the partial FJSP means each operation can be processed only on one or more machines. In this paper, our major research problem is partial FJSP which is described as follows. There are a set  $J = \{J_1, J_2, \dots, J_i, \dots, J_n\}$  of  $n$  jobs and a set  $M = \{M_1, M_2, \dots, M_k, \dots, M_m\}$  of  $m$  machines. The total number of operations is defined as  $o = \sum_{i=1}^n H_i$ . Each job  $J_i$  contains a series of

operations  $O_i = \{O_{i,1}, O_{i,2}, \dots, O_{i,H_i}\}$  with a specified order. The  $j^{th}$  operation  $O_{i,j}$  of the  $i^{th}$  job  $J_i$  can be processed on a machine  $M_k$  selected from a set of available machines and  $t_{i,j,k}$  is its processing time. The objectives in the FJSP are to find the best sequence of all operations and the most suitable machine for each operation to optimize the makespan, workload, etc. For an intuitive illustration, a specific partial FJSP instance is shown in Table 1. The numbers in the table refers to the processing time  $t_{i,j,k}$  and the symbol ‘-’ indicates that operation  $O_{i,j}$  can not be processed on a machine  $M_k$ . The typical assumptions [27] of FJSP are given as follows:

- (1) All jobs can not be processed and all machines are available at the initial time;
- (2) The order of precedence between operations of each job must be obeyed;
- (3) Each machine  $M_k$  can process only one operation in any time;
- (4) Each operation  $O_{i,j}$  owns at least one machine to be processed;
- (5) The processing cannot be interrupted until is finished;
- (6) The transportation time of operations and depreciation of machines are ignored.

**Table 1.** An instance of  $3 \times 3$  partial FJSP.

Jobs	Operation	$M_1$	$M_2$	$M_3$
$J_1$	$O_{1,1}$	3	2	-
	$O_{1,2}$	2	5	1
$J_2$	$O_{2,1}$	2	3	2
	$O_{2,2}$	4	-	2
	$O_{2,3}$	-	2	3
$J_3$	$O_{3,1}$	4	2	1
	$O_{3,2}$	2	-	6

### 3.2 Optimization Object

In this paper, the optimization object is to obtain a scheduling with the lowest makespan  $C_{max}$ . The object function can be presented by Eq. (1) and the constraints are listed in Eq. (2)–Eq. (4). Among these equations,  $s_{i,j,k}$  represents the work start time of operation  $O_{i,j}$  on  $M_k$  machine. Equation (2) shows that the processing time of all operations is positive. Equation (3) represents the order of precedence between operations of each job must be performed. Equation (4) represents that each machine  $M_k$  can process only one operation in any time. Equation (5) shows that each operation  $O_{i,j}$  owns at least one machine to be processed.

Object:

$$\min(C_{\max}) = \min(\max(C_i)) \tag{1}$$

Subject to:

$$t_{i,j,k} > 0, i = 1, 2, \dots, N; j = 1, 2, \dots, H_i; k = 1, 2, \dots, M \tag{2}$$

$$s_{i,j,k} + t_{i,j,k} \leq s_{i,j+1,k}, j = 1, 2, \dots, (H_i - 1) \tag{3}$$

$$\sum_{i=1}^N \sum_{j=1}^{H_i} X_{i,j,k} = 1, X_{i,j,k} = \begin{cases} 1 & \text{If } O_{i,j} \text{ is assigned to } k^{th} \text{ machine} \\ 0 & \text{Otherwise} \end{cases} \tag{4}$$

$$\sum_{k=1}^M X_{i,j,k} \geq 1 \tag{5}$$

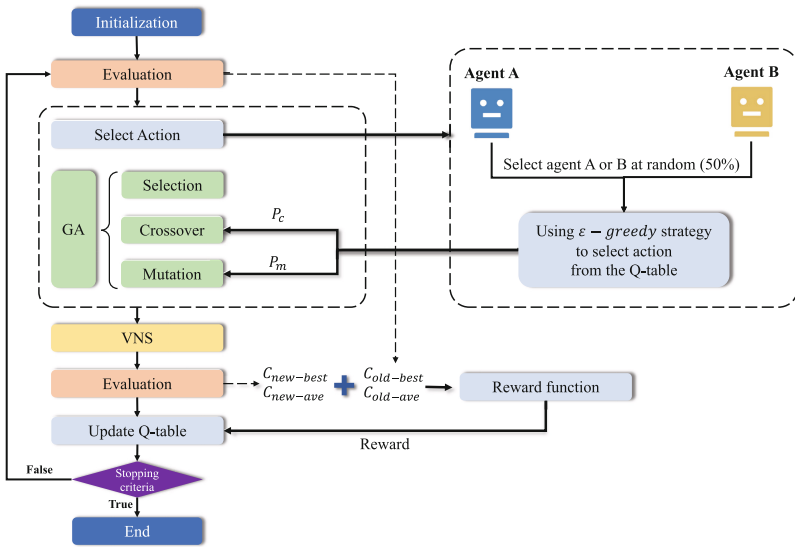


Fig. 1. The workflow of the proposed algorithm

## 4 Proposed Algorithm

### 4.1 Workflow of the Proposed AHOA

As Fig. 1 shows, AHOA is designed by merging the hybrid algorithm and double Q-learning algorithm. It accepts an instance of the FJSP as the inputs and uses HA that combines GA and VNS to optimize the makespan. To avoid the

performance being severely affected by the preset hyperparameters, the double Q-learning algorithm is introduced to intelligently adjust them. The overall workflow of the proposed algorithm is described as Algorithm 1, and the details are described in the following sub-sections.

---

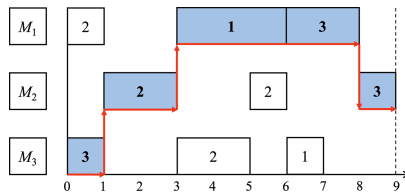
**Algorithm 1:** Adaptively Hybrid Optimization Algorithm

---

**Input:** Initial parameters and an FJSP instance.

**Output:** The best solution  $x$  and its makespan  $C_x$ .

- 1 Initialize two agents  $Q^A$  and  $Q^B$ , population  $Pop$  with  $P$  individuals and set  $Gen = 1$ ;
  - 2 **for**  $Gen = 1 \rightarrow loop_{max}$  **do**
  - 3     Evaluate  $Pop$  to obtain the average makespan  $C_{old-ave}$  and the minimal makespan  $C_{old-best}$ ;
  - 4     **if** *population has stagnated in evolution for 20 iterations* **then**
  - 5         | Reinitialize the population;
  - 6     **end**
  - 7     Select agent  $Q^A$  or  $Q^B$  randomly (50%);
  - 8     Select action (including crossover and mutation rates) from Q-table with  $\epsilon$ -greedy strategy;
  - 9     Apply the selection, crossover and mutation operators to generate the new population  $Pop_{new}$ ;
  - 10    Apply the improved VNS to promote the quality of each individual in  $Pop_{new}$ ;
  - 11    Evaluate  $Pop_{new}$  to obtain the  $C_{new-ave}$  and the  $C_{new-best}$ ;
  - 12    Update the Q-table according to the  $C_{old-ave}$ ,  $C_{old-best}$ ,  $C_{new-ave}$  and  $C_{new-best}$ ;
  - 13     $Gen \leftarrow Gen + 1$ ;
  - 14     $Pop \leftarrow Pop_{new}$ ;
  - 15 **end**
  - 16 Select the best solution  $x$  of  $Pop_{new}$ ;
  - 17 **return**  $x$  and  $C_x$ .
- 



**Fig. 2.** The example of the critical path.

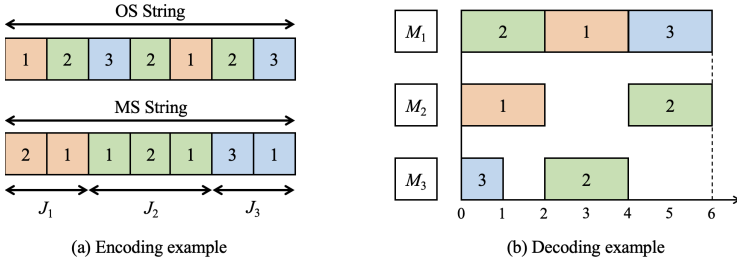


Fig. 3. The encoding and decoding example based on the Table 1

## 4.2 Hybrid Algorithm

### 4.2.1 Neighborhood Structure Based on Critical Path

The critical path is the longest path from the start to the end of all the operations in the Gantt chart. There is no interval between any adjacent operations, and the length of the path is equal to the makespan of the current solution. For instance, the combination of the blue blocks including  $O_{3,1} \rightarrow O_{2,1} \rightarrow O_{1,1} \rightarrow O_{3,2} \rightarrow O_{3,3}$  is the critical path in Fig. 2. The length of this critical path is 9 which is also the end of the Gantt chart. In the case that the critical path length remains the same, the makespan cannot be reduced [35]. Therefore, the operation based on the critical path is introduced to design a problem-specific neighborhood structure for the VNS and the mutation operators for the GA.

The neighbourhood structure can be decomposed into the following steps. First, all the critical paths in the current Gantt chart are found. Among the critical paths, one path is chosen randomly, and the available machine is selected for each operation on this critical path. For VNS, This neighborhood structure modifies the individuals from the GA to generate new neighborhood solutions. Moreover, this structure is also used as a mutation operator to generate new individuals for the new population in the GA.

### 4.2.2 Genetic Algorithm

In the proposed algorithm, the encoding method proposed by Gao *et al.* [12] is adopted. As shown in Fig. 3, the code composes of two strings. One is called OS (Operation Sequence), and the other is called MS (Machine Sequence). As mentioned in Sect. 1, two sub-problems are needed to solve FJSP. The first is to find the best sequence of operations and the second assigns suitable machines to each operation. The two strings correspond to these two sub-problems respectively. For OS string, the number  $i$  that appears  $j^{th}$  times represents it is the operation  $O_{i,j}$  of job  $J_i$ . For MS string, it presents the selected machines for the corresponding operations of each job. Moreover, the decoding method proposed by Gong *et al.* [14] is used in this work to minimize the makespan. It can minimize makespan while obeying the constraints during operations and machines.

For selection operators [34], the elitist selection method is performed firstly and the tournament selection method is continuously performed until the size of the new population reaches the *Popsiz*e. Each string has its independent crossover and mutation operators. For the OS string, the Precedence Operation crossover (POX) [26] and the Job-Based crossover (JBX) [26] are performed randomly (50%). Moreover, two mutation operators for the OS string have been adopted. The first one is the insertion mutation that selects a random digit to insert before a random position. The second one is the swapping mutation that swaps two random positions. These two mutation operators for the OS string are also performed randomly (50%). For the MS string, the two-point crossover [26] is adopted as the crossover operator. The mutation operator based on the critical path is introduced in this paper. This operator only changes the machine of operation in the critical path which is explicitly described in Sect. 4.2.1.

### 4.2.3 Variable Neighborhood Search

The LS algorithm is introduced to improve GA in terms of the convergence speed, the quality of individuals, and the local search ability. The VNS is a meta-heuristic LS method proposed by Mladenovi *et al.* [29]. The core idea is neighborhood transformation which has been successfully applied to numerous combinatorial optimization problems.

The VNS contains two procedures called shaking and local search. These procedures only employ a neighborhood structure based on the critical path. The main structure of the improved VNS is composed of external and internal loops. The external loop performs the shaking procedure, and the internal loop performs a local search. The definition of related symbols is as follows. The symbol  $k_{max}$  is the max iterations of the external loop ( $k_{max} = 2$ ), and  $l$  is the max iterations of the internal loop ( $l_{max} = 2$ ). The symbol  $k_{popsiz$ e is delineated as the size of the shaking neighborhood ( $k_{popsiz$ e = 3), and  $l_{popsiz$ e is the size of the local search neighborhood ( $l_{popsiz$ e = 3). Besides,  $N(x)$  is presented as the neighborhood structure related to the critical path which is generated from a solution  $x$ .

As shown in Algorithm 2, in the first step, the initial solution  $x$  generated from the GA is denoted as the global optimal solution and the  $x'$  generated from the neighborhood structure  $N_k(x)$  is denoted as the local optimal solution. The shaking procedure firstly selects a random solution  $x'$  from the  $k^{th}$  neighborhood  $N_k(x)$  generated base on  $x$ . Then the local search procedure selects a random solution  $x''$  from the  $l^{th}$  neighborhood  $N_l(x')$  generated base on  $x'$ . If  $x''$  is better than  $x'$ , then set  $x'$  to  $x''$  and  $l$  to 1. The local search procedure continues to be performed until the end of the internal loop iteration. When the end of the local search procedure. If the local optimal solution  $x'$  is better than  $x$ , then set  $x$  to  $x'$  and  $k$  to 1. Finally, the procedure repeats until the end of the external loop iteration.



---

**Algorithm 2:** Variable Neighborhood Search Algorithm

---

**Input:** The old population  $P_{old}$ , and neighborhood parameters

$k_{max}, k_{popsize}, l_{max}, l_{popsize}$ .

**Output:** The new population  $P_{new}$ .

```

1 foreach individual  $x$  in  $P_{old}$  do
2   Evaluate the global makespan  $C_x$  of  $x$ ;
3   for  $k = 1 \rightarrow k_{size}$  do
4     Shaking procedure: pick a random solution  $x'$  from  $k^{th}$ 
       neighborhood  $N_k(x)$  of  $x$ ;
5     for  $l = 1 \rightarrow l_{size}$  do
6       Local search procedure: pick a random solution  $x''$  in
         neighborhood  $N_l(x')$  of  $x'$ ;
7       if  $C_{x''} < C_{x'}$  then
8         | Update the local optimal solution:  $x' \rightarrow x''$ , and  $l \rightarrow 1$ ;
9       else
10        |  $l \rightarrow l + 1$ ;
11      end
12    end
13    if  $C_{x'} < C_x$  then
14      | Update the global optimal solution:  $x \rightarrow x'$ , and  $k \rightarrow 1$ ;
15    else
16      |  $k \rightarrow k + 1$ ;
17    end
18  end
19   $P_{new} = P_{new} \cup \{x\}$ ;
20 end
21 return  $P_{new}$ .

```

---

### 4.3 Double Q-learning Procedure

The double Q-learning is a typical reinforcement learning method proposed by Hasselt *et al.* [17]. It aims to solve the overestimations problem of Q-value defined as  $Q(s, a)$ . The double Q-learning obtains two intelligent agents  $Q^A$  and  $Q^B$ . The agents interact continuously with the HA environment to find the most cumulative Q-value based on past experience and update the Q-value from the other agent for the next state. Moreover, each agent can obtain a reward or penalty while performing a selected action and learn to maximize the long-term reward after multi iterations.

The agent will take different actions to get crossover and mutation rates combined as the action in the Q-table. Specifically, the crossover rates group  $G_c$  is set to  $\{0.9, 0.8, 0.7, 0.6, 0.5\}$  and the mutation rates group  $G_m$  is establish to  $\{0.3, 0.2, 0.1\}$  in the action list  $A_{list}$ . The  $A_{list}$  is the Cartesian product of

$G_c$  and  $G_m$ . Besides, the symbol  $\alpha \in [0, 1]$  represents the learning rate. The  $\gamma \in [0, 1]$  represents the attenuation rate for the future reward. The  $R$  is defined as the reward shown in Eq. (8).

In the AHOA, we assume that only one static state in the HA environment prevents the sparse Q-table. The specific algorithm is described as Algorithm 3 shows. Firstly, step in the double Q-learning is to initialize agent  $Q^A$  and agent  $Q^B$  by initializing each Q-table to a zero-value matrix. Then select agent  $Q^A$  or agent  $Q^B$  to use at random (50%). Assuming that agent  $Q^A$  is selected, the  $\varepsilon$ -greedy strategy as shown in Sect. 4.3.1 is next used to select the action  $a^*$  to be performed from the action list  $A_{list}$ . The HA obtains the crossover rate and mutation rate from  $a^*$  and executes to get an evaluation. Finally, the Q-table is updated according to Eq. (6). In Eq. (6), the above equation is used to update if the agent  $Q^A$  is selected. Otherwise, the following equation is used to update the table.

$$\begin{cases} Q^A(s, a) \leftarrow Q^A(s, a) + \alpha \times (R + \gamma \times Q^B(s, a^*) - Q^A(s, a)) & (1) \\ Q^B(s, a) \leftarrow Q^B(s, a) + \alpha \times (R + \gamma \times Q^A(s, a^*) - Q^B(s, a)) & (2) \end{cases} \quad (6)$$

---

### Algorithm 3: Double Q-Learning Algorithm

---

**Input:** The learning rate  $\alpha$ , attenuation rate  $\gamma$ ,  $\varepsilon$ -greedy rate  $\varepsilon$ , crossover rates group  $G_c$ , mutation rates group  $G_m$ , and makespans

$C_{old-ave}$ ,  $C_{old-best}$ ,  $C_{new-ave}$  and  $C_{new-best}$ .

- 1 Initialize  $Q^A$  and  $Q^B$ ;
  - 2 Select agent  $Q^A$  or  $Q^B$  randomly (50%);
  - 3 Select action  $a^*$  from Q-table with  $\varepsilon$ -greedy strategy;
  - 4 Apply the  $a^*$  (the crossover rate  $P_c$  and mutation rate  $P_m$ ) to HA;
  - 5 Calculate reward  $R$  according to the  $C_{old-ave}$ ,  $C_{old-best}$ ,  $C_{new-ave}$  and  $C_{new-best}$  from HA;
  - 6 **if** *Select*  $Q^A$  **then**
  - 7     /\* Update the table of  $Q^A$  \*/;
  - 8      $Q^A(s, a) \leftarrow Q^A(s, a) + \alpha \times (R + \gamma \times Q^B(s, a^*) - Q^A(s, a))$ ;
  - 9 **else if** *Select*  $Q^B$  **then**
  - 10    /\* Update the table of  $Q^B$  \*/;
  - 11     $Q^B(s, a) \leftarrow Q^B(s, a) + \alpha \times (R + \gamma \times Q^A(s, a^*) - Q^B(s, a))$ ;
  - 12 **end**
- 

#### 4.3.1 Selection Strategy

The selection strategy of double Q-learning is the  $\varepsilon$ -greedy strategy. In this strategy, an action  $a^*$  will be selected randomly from  $A_{list}$  if a random number  $rand_{[0-1]}$  is higher than the agreed value  $\varepsilon$ , otherwise the action with the most  $Q(s, a)$  will be selected. The definition is shown in Eq. (7).

$$a^* = \begin{cases} \arg \max_a Q(s, a) & rand_{[0-1]} < \varepsilon \\ A_{list}[random] & rand_{[0-1]} \geq \varepsilon \end{cases} \quad (7)$$

### 4.3.2 Reward Function

The reward function is designed according to the best and the average individual fitness as shown in Eq. (8). In the reward method,  $C_{old-ave}$  and  $C_{old-best}$  are delineated as the average and the best makespan in the previous population.  $C_{new-ave}$  and  $C_{new-best}$  are delineated as the average and the best makespan in the current population.

$$R = \begin{cases} 30 & \text{If } C_{new-ave} < C_{old-ave} \text{ and } C_{new-best} < C_{old-best} \\ 15 & \text{If } C_{new-ave} \geq C_{old-ave} \text{ and } C_{new-best} < C_{old-best} \\ -0.5 & \text{If } C_{new-ave} < C_{old-ave} \text{ and } C_{new-best} \geq C_{old-best} \\ -5 & \text{If } C_{new-ave} \geq C_{old-ave} \text{ and } C_{new-best} \geq C_{old-best} \end{cases} \quad (8)$$

## 4.4 Computation Complexity Analysis

For each generation of the AHOA, the computational complexity can be analyzed as follows. First it is with the computational complexity  $\mathcal{O}(P \log P)$  by using quick sorting method to evaluate the population. Then, it is with the computational complexity  $\mathcal{O}(P)$  to perform selection operator, with the computational complexity  $\mathcal{O}(P \times o)$  to perform crossover operator and with the computational complexity  $\mathcal{O}(P \times m \times o)$  to perform mutation operator. Finally, the AHOA applies the VNS based on the critical path with the computational complexity  $\mathcal{O}(P \times k_{size} \times k_{pop} \times l_{size} \times l_{pop} \times m^2 \times o)$ . Thus, the computational complexity for each generation of the AHOA is  $\mathcal{O}[P \times (\log P + o \times k_{size} \times k_{pop} \times l_{size} \times l_{pop} \times m^2)]$ .

## 5 Experiment and Discussions

### 5.1 Experimental Setup

To demonstrate the performance of the proposed algorithm, ten benchmark instances of the BRdata [2] are tested. The BRdata is a data set consists of 10 test instances (Mk01-Mk10), which are randomly generated using a uniform distribution between given limits. The parameters in the proposed algorithm for these instances are listed in Table 2. The proposed AHOA is implemented in C++ on the AMD Ryzen 5 5600X machine running at 3.7 GHz and the optimization objective of it is makespan. Besides, we select three categories represented algorithms with to compare, including EA based methods, HA based methods, and RL based methods. For instance, Wang *et al.* [32] proposed a hybrid algorithm named GIWO based on gray wolf and invasive weeds, and Chen *et al.* [5] and Han *et al.* [16] both proposed methods based on RL to solve the FJSP.

**Table 2.** The AHOA parameters.

Parameters	Descriptions	
$\alpha$	Learning rate	0.6
$\gamma$	Attenuation rate	0.8
$\varepsilon$	$\varepsilon$ -greedy rate	0.85
$G_c$	Crossover rates group in the action list $A_{list}$	{0.9, 0.8, 0.7, 0.6, 0.5}
$G_m$	Mutation rates group in the action list $A_{list}$	{0.3, 0.2, 0.1}
$k_{max}$	Max iterations of external loop in VNS	2
$l_{max}$	Max iterations of internal loop in VNS	2
$k_{popsize}$	Size of the shaking neighborhood	3
$l_{popsize}$	Size of the local search neighborhood	3
$P$	Size of the population	80
$loop_{max}$	Max iterations of the proposed algorithm	200

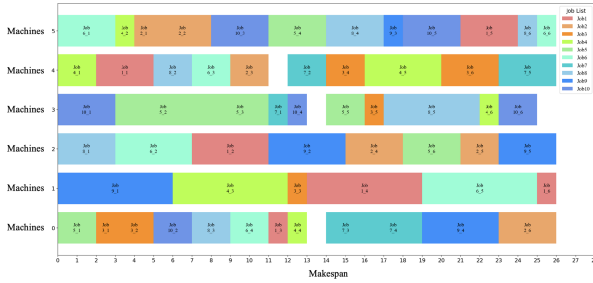
## 5.2 Experimental Results

The results of these comparisons are shown in the Table 3. For the EA based methods, the average improving performance of AHOA is 11%, of which the three testing instances Mk02, Mk04, and Mk06 improved by more than 22.8%; for the HA based methods, the average improving performance of AHOA is 6.7%, of which the two testing instances Mk02 and Mk06 improved by more than 16.7%. The most important difference between the AHOA and the above two methods is the ability to adjust the hyperparameters adaptively. The improved PSO [10] proposes some tuning schemes of parameters with exact mathematical methods. However, the accurate mathematical expression of the parameters tuning schemes may be unavailable as the schemes are affected by various factors, such as dynamic conditions. The result shows that the AHOA is more effective in large-scale complex environments. Besides, it can deal with situations in which the mathematical expression of the parameters tuning schemes is not available. These observations align with the findings in [6].

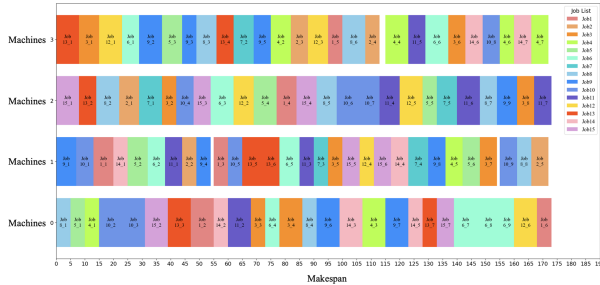
For the RL based methods, the average improving performance of AHOA is 22%, including the three testing instances Mk06, Mk07, and Mk10 increasing by more than 25%. However, the AHOA performs slightly worse than the self-learning GA (SLGA) [5] on Mk05. Because the setting range of the action list in the SLGA is more refined compared with the AHOA. It can find more suitable parameters for complex environments with enough iterations. The SLGA proposes a different way to combine two RL algorithms to enhance the performance of GA. Nevertheless, it still can not avoid the large overestimations of action values in the RL [17]. The result shows that AHOA performs better in terms of solution quality than other RL based methods. Besides, the Gantt chart of the solution obtained by AHOA for the Mk02 and Mk05 instance are presented in Fig. 4 and Fig. 5.

**Table 3.** The comparison of the makespan in BRdata

Methods	Year	Mk01	Mk02	Mk03	Mk04	Mk05	Mk06	Mk07	Mk08	Mk09	Mk10
		10 × 6	10 × 6	15 × 8	15 × 8	15 × 4	10 × 15	20 × 5	20 × 10	20 × 10	20 × 15
Deep RL [16]	2020	44	28	245	74	193	123	216	<b>523</b>	386	337
Dragonfly Algorithm [19]	2020	52	46	210	88	175	87	—	—	—	—
HLO-PSO [9]	2020	<b>40</b>	28	<b>204</b>	63	175	71	<b>144</b>	<b>523</b>	326	238
Improved PSO [10]	2020	<b>40</b>	29	<b>204</b>	66	175	77	145	<b>523</b>	320	239
SLGA [5]	2020	<b>40</b>	27	<b>204</b>	<b>60</b>	<b>172</b>	69	<b>144</b>	<b>523</b>	320	254
GIWO [32]	2021	<b>40</b>	32	<b>204</b>	65	177	84	156	<b>523</b>	331	242
GA	—	42	33	<b>204</b>	67	183	86	181	<b>523</b>	325	270
VNS	—	42	33	<b>204</b>	67	183	86	181	<b>523</b>	338	277
HA(GA+VNS)	—	<b>40</b>	31	<b>204</b>	64	179	78	150	<b>523</b>	315	258
<b>Our proposed method</b>	2021	<b>40</b>	<b>26</b>	<b>204</b>	<b>60</b>	173	<b>65</b>	<b>144</b>	<b>523</b>	<b>307</b>	<b>233</b>



**Fig. 4.** Gantt chart of the solution of the Mk02 instance



**Fig. 5.** Gantt chart of the solution of the Mk05 instance

### 5.3 Ablation Experiment

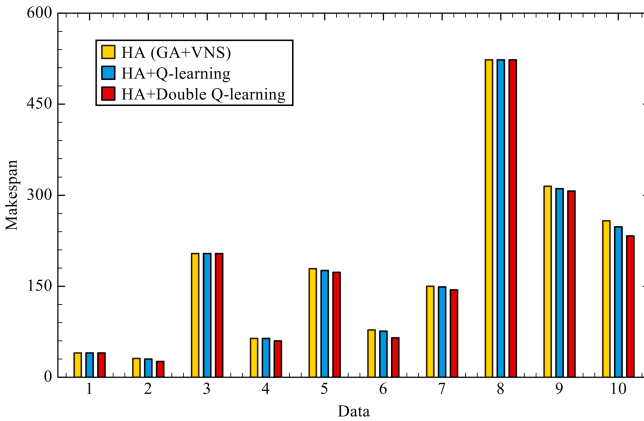
As mentioned in Sect. 4, we introduce the critical path and reinforcement learning to solve the FJSP task. In order to illustrate the improvement in the proposed algorithm, we make ablation experiments on these two parts.

### 5.3.1 The Discussion of the Critical Path

As shown in Table 4, an algorithm with critical path can achieve better performance than an algorithm without critical path. The experiment results show that the makespan of the AHOA is 10.7% more than the method without critical path on the ten instances on average. It indicates that the critical path can effectively improve search quality. As the complexity analysis of AHOA in Sect. 4.4, searching the critical path will increase the time cost of the algorithm, which means that the AHOA will spend more time in a single iteration. However, it can reach convergence with fewer iterations.

**Table 4.** Experiments observing the influence of the critical path on 10 instances.

Method	Mk01	Mk02	Mk03	Mk04	Mk05	Mk06	Mk07	Mk08	Mk09	Mk10
AHOA without critical path	42	32	<b>204</b>	67	181	84	159	<b>523</b>	326	272
AHOA with critical path	<b>40</b>	<b>26</b>	<b>204</b>	<b>60</b>	<b>173</b>	<b>65</b>	<b>144</b>	<b>523</b>	<b>307</b>	<b>233</b>
Improvement	5.0%	23.1%	0.0%	11.7%	4.6%	29.2%	10.4%	0.0%	6.2%	16.7%



**Fig. 6.** Histograms reflecting the influence of the different reinforcement learning on 10 instances.

### 5.3.2 The Discussion of the Reinforcement Learning

We also conduct an ablation experiment on reinforcement learning. It aims to explore the influence of different reinforcement learning methods on HA. As shown in Fig. 6, the makespan of the histogram performs best on the double Q-learning algorithm, which outperforms the HA and the Q-learning algorithm by more than 5.9% and 4.6% respectively. Compared with the HA, the algorithm introduced with reinforcement learning has the characteristics of adaptation which can better adapt to different environments. However, the Q-learning overestimate attempted actions, which will affect the convergence results to a certain extent. The double Q-learning can avoid the positive bias in estimating

the action values and achieves better performance. This result shows that the strategy with the double Q-learning has achieved the best results on the ten instances. It indicates that the double Q-learning can improve performance by enhancing generalization and preventing overestimation problems.

## 6 Conclusion

This paper discusses the FJSP task by proposing a novel algorithm AHOA, combining the hybrid and double Q-learning algorithms. Compared with the widely deployed methods for FJSP, our algorithm can modify the key hyperparameters adaptively in the HA and avoid the overestimations problem of action values compared to other methods based on RL. Moreover, the proposed algorithm is more efficient due to the neighborhood structure based on the critical path. Finally, the experiment results verify that AHOA is suitable for FJSP instances with the high flexibility.

For the future work, we notice that the existing algorithms have achieved remarkable results in static scheduling, but there are few studies on the dynamic scheduling. We would like to investigate the dynamic multi-objective FJSP based on the transfer learning [21, 23–25] and domain adaptation learning [20, 22]. Besides, the performance of AHOA in practical applications still needs further testing, our future work will also focus on enhancing the generalization capabilities of the algorithm on the larger FJSP data sets.

**Acknowledgement.** This work was supported in part by the National Natural Science Foundation of China under Grant 61673328, and in part by the Collaborative Project Foundation of Fuzhou-Xiamen-Quanzhou Innovation Demonstration Zone under Grant 3502ZCQXT202001.

## References

1. Amiri, M., Zandieh, M., Yazdani, M., Bagheri, A.: A variable neighbourhood search algorithm for the flexible job-shop scheduling problem. *Int. J. Prod. Res.* **48**(19), 5671–5689 (2010)
2. Brandimarte, P.: Routing and scheduling in a flexible job shop by tabu search. *Ann. Oper. Res.* **41**(3), 157–183 (1993)
3. Brucker, P., Schlie, R.: Job-shop scheduling with multi-purpose machines. *Computing* **45**(4), 369–375 (1990)
4. Chaudhry, I.A., Khan, A.A.: A research survey: review of flexible job shop scheduling techniques. *Int. Trans. Oper. Res.* **23**(3), 551–591 (2016)
5. Chen, R., Yang, B., Li, S., Wang, S.: A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem. *Comput. Ind. Eng.* **149**, 106778 (2020)
6. Dai, P., Yu, W., Wen, G., Baldi, S.: Distributed reinforcement learning algorithm for dynamic economic dispatch with unknown generation cost functions. *IEEE Trans. Ind. Inf.* **16**(4), 2258–2267 (2019)

7. De Giovanni, L., Pezzella, F.: An improved genetic algorithm for the distributed and flexible job-shop scheduling problem. *Eur. J. Oper. Res.* **200**(2), 395–408 (2010)
8. Deb, K., Agrawal, S.: Understanding interactions among genetic algorithm parameters. *Found. Genet. Algorithms* **5**(5), 265–286 (1999)
9. Ding, H., Gu, X.: Hybrid of human learning optimization algorithm and particle swarm optimization algorithm with scheduling strategies for the flexible job-shop scheduling problem. *Neurocomputing* **414**, 313–332 (2020)
10. Ding, H., Gu, X.: Improved particle swarm optimization algorithm based novel encoding and decoding schemes for flexible job shop scheduling problem. *Comput. Oper. Res.* **121**, 104951 (2020)
11. Du, Y., Fang, J., Miao, C.: Frequency-domain system identification of an unmanned helicopter based on an adaptive genetic algorithm. *IEEE Trans. Ind. Electron.* **61**(2), 870–881 (2013)
12. Gao, L., Peng, C., Zhou, C., Li, P.: Solving flexible job shop scheduling problem using general particle swarm optimization. In: *Proceedings of the 36th CIE Conference on Computers & Industrial Engineering*, pp. 3018–3027. Citeseer (2006)
13. Garey, M.R., Johnson, D.S., Sethi, R.: The complexity of flowshop and jobshop scheduling. *Math. Oper. Res.* **1**(2), 117–129 (1976)
14. Gong, X., Deng, Q., Gong, G., Liu, W., Ren, Q.: A memetic algorithm for multi-objective flexible job-shop problem with worker flexibility. *Int. J. Prod. Res.* **56**(7), 2506–2522 (2018)
15. Graves, S.C.: A review of production scheduling. *Oper. Res.* **29**(4), 646–675 (1981)
16. Han, B., Yang, J.: A deep reinforcement learning based solution for flexible job shop scheduling problem. *Int. J. Simul. Model. (IJSIMM)* **20**(2), 375–386 (2021)
17. Hasselt, H.: Double q-learning. *Adv. Neural Inf. Process. Syst.* **23**, 2613–2621 (2010)
18. Ho, N.B., Tay, J.C.: Genace: an efficient cultural algorithm for solving the flexible job-shop problem. In: *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753)*, vol. 2, pp. 1759–1766. IEEE (2004)
19. Hu, S., Zhou, L.: Research on flexible job-shop scheduling problem based on the dragonfly algorithm. In: *2020 International Conference on Artificial Intelligence and Electromechanical Automation (AIEA)*, pp. 241–245. IEEE (2020)
20. Jiang, M., Huang, W., Huang, Z., Yen, G.G.: Integration of global and local metrics for domain adaptation learning via dimensionality reduction. *IEEE Trans. Cybern.* **47**(1), 38–51 (2015)
21. Jiang, M., Huang, Z., Qiu, L., Huang, W., Yen, G.G.: Transfer learning-based dynamic multiobjective optimization algorithms. *IEEE Trans. Evol. Comput.* **22**(4), 501–514 (2017)
22. Jiang, M., Qiu, L., Huang, Z., Yen, G.G.: Dynamic multi-objective estimation of distribution algorithm based on domain adaptation and nonparametric estimation. *Inf. Sci.* **435**, 203–223 (2018)
23. Jiang, M., Wang, Z., Guo, S., Gao, X., Tan, K.C.: Individual-based transfer learning for dynamic multiobjective optimization. *IEEE Trans. Cybern.* **51**(10), 4968–4981 (2021)
24. Jiang, M., Wang, Z., Hong, H., Yen, G.G.: Knee point-based imbalanced transfer learning for dynamic multiobjective optimization. *IEEE Trans. Evol. Comput.* **25**(1), 117–129 (2020)
25. Jiang, M., Wang, Z., Qiu, L., Guo, S., Gao, X., Tan, K.C.: A fast dynamic evolutionary multiobjective algorithm via manifold transfer learning. *IEEE Trans. Cybern.* **51**(7), 3417–3428 (2021)



26. Li, X., Gao, L.: An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem. *Int. J. Prod. Econ.* **174**, 93–110 (2016)
27. Liu, B., Fan, Y., Liu, Y.: A fast estimation of distribution algorithm for dynamic fuzzy flexible job-shop scheduling problem. *Comput. Ind. Eng.* **87**, 193–201 (2015)
28. Lu, P.-H., Wu, M.-C., Tan, H., Peng, Y.-H., Chen, C.-F.: A genetic algorithm embedded with a concise chromosome representation for distributed and flexible job-shop scheduling problems. *J. Intell. Manuf.* **29**(1), 19–34 (2018). <https://doi.org/10.1007/s10845-015-1083-z>
29. Mladenović, N., Hansen, P.: Variable neighborhood search. *Comput. Oper. Res.* **24**(11), 1097–1100 (1997)
30. Qi, X., Luo, Y., Wu, G., Boriboonsomsin, K., Barth, M.: Deep reinforcement learning enabled self-learning control for energy efficient driving. *Transp. Res. Part C Emerg. Technol.* **99**, 67–81 (2019)
31. Rangel-Merino, A., López-Bonilla, J., y Miranda, R.L.: Optimization method based on genetic algorithms. *Apeiron* **12**(4), 393–408 (2005)
32. Wang, Y., Song, Y.C., Zou, Y.J., Lei, Q., Wang, X.K.: A hybrid gray wolf weed algorithm for flexible job-shop scheduling problem. In: *Journal of Physics: Conference Series*, vol. 1828, p. 012162. IOP Publishing (2021)
33. Wang, Y., Zhu, Q.: A hybrid genetic algorithm for flexible job shop scheduling problem with sequence-dependent setup times and job lag times. *IEEE Access* **9**, 104864–104873 (2021)
34. Whitley, D.: A genetic algorithm tutorial. *Stat. Comput.* **4**(2), 65–85 (1994)
35. Zhang, G., Shao, X., Li, P., Gao, L.: An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. *Comput. Ind. Eng.* **56**(4), 1309–1318 (2009)