processes

MDPI

*Article*

# Multi-Objective Flexible Flow Shop Production Scheduling Problem Based on the Double Deep Q-Network Algorithm

**Hua Gong [1,2,*], Wanning Xu [3], Wenjuan Sun [1,2,*] and Ke Xu [1,2]**

[1] School of Science, Shenyang Ligong University, Shenyang 110159, China; xuke@sylu.edu.cn
[2] Liaoning Key Laboratory of Intelligent Optimization and Control for Ordnance Industry, Shenyang 110159, China
[3] School of Automation and Electrical Engineering, Shenyang Ligong University, Shenyang 110159, China; xuuwanning@163.com
* Correspondence: gonghua@sylu.edu.cn (H.G.); sunwenjuan@sylu.edu.cn (W.S.)

**Abstract:** In this paper, motivated by the production process of electronic control modules in the digital electronic detonators industry, we study a multi-objective flexible flow shop scheduling problem. The objective is to find a feasible schedule that minimizes both the makespan and the total tardiness. Considering the constraints imposed by the jobs and the machines throughout the manufacturing process, a mixed integer programming model is formulated. By transforming the scheduling problem into a Markov decision process, the agent state features and the actions are designed based on the processing status of the machines and the jobs, along with heuristic rules. Furthermore, a reward function based on the optimization objectives is designed. Based on the deep reinforcement learning algorithm, the Dueling Double Deep Q-Network (D3QN) algorithm is designed to solve the scheduling problem by incorporating the target network, the dueling network, and the experience replay buffer. The D3QN algorithm is compared with heuristic rules, the genetic algorithm (GA), and the optimal solutions generated by Gurobi. The ablation experiments are designed. The experimental results demonstrate the high performance of the D3QN algorithm with the target network and the dueling network proposed in this paper. The scheduling model and the algorithm proposed in this paper can provide theoretical support to make the production plan of electronic control modules reasonable and improve production efficiency.

**Keywords:** production scheduling; flexible flow shop; multi-objective optimization; deep Q-network; mixed integer programming



**Citation:** Gong, H.; Xu, W.; Sun, W.; Xu, K. Multi-Objective Flexible Flow Shop Production Scheduling Problem Based on the Double Deep Q-Network Algorithm. *Processes* **2023**, *11*, 3321. https://doi.org/10.3390/pr11123321

Academic Editor: Francisco Vazquez

Received: 19 October 2023
Revised: 22 November 2023
Accepted: 27 November 2023
Published: 29 November 2023



## 1. Introduction

Digital electronic detonators represent advanced industrial devices that integrate traditional detonators with electronic control technologies. Electronic control modules are utilized to perform various functions, such as time constraints and safety control. The integration of advanced features, including precision, safety measures, and remote-control capabilities significantly improve the effectiveness and safety of blasting operations. The electronic control module plays a vital role in the digital detonator, as it is responsible for precise programming and timing control to achieve explosion triggering with millisecond-level precision. The manufacturing process of the electronic control module comprises several sequential stages. It begins with the assembly and soldering of printed circuit boards (PCBs) in an automated production line of surface mount technology (SMT). Subsequently, these PCBs are processed using automated optical inspection (AOI) to transform them into semi-finished products. Then, these semi-finished products need to undergo a series of production and testing processes, including electrical performance testing of the semi-finished products, a visual inspection of the injection process, electrical performance testing of the finished products, a visual inspection of spot welding, electrical performance testing

of the finished products using all-in-one machines, and a visual inspection of the resistance and bridge wire. The production flow of the electronic control modules is shown in Figure 1.

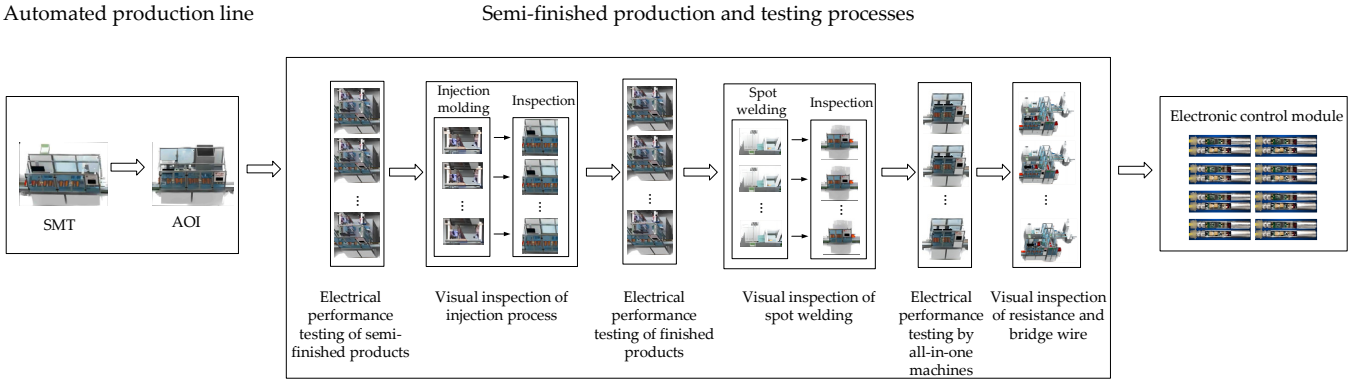



**Figure 1.** Production flow chart of electronic control modules.

During the practical production of electronic control modules, each manufacturing stage typically involves multiple processing and testing machines, where the production flow can be regarded as a typical flexible flow shop production environment. The allocation of machines and the sequencing of the job to be processed at each stage will directly affect the overall efficiency of the production. A good utilization of the multiple processing and testing machines is to minimize the makespan. The measurement of how well due dates are being met in practice is often represented by the objective of minimizing the total tardiness. Hence, it is necessary to find a scientific and reasonable scheduling scheme to ensure the efficiency of the overall production is achieved.

The flexible flow shop scheduling (FFSS) problem contains features of ordinary flow shop and parallel machine scheduling at each stage. The field of FFSS encompasses two types of scheduling: single-objective and multi-objective.

For the single-objective FFSS problem, Han et al. [1], Shi et al. [2], and Malekpour et al. [3] have first presented the development of intelligent optimization algorithms. These include the improved migratory bird optimization algorithm, the enhanced grey wolf algorithm, and the simulated annealing algorithm, designed to solve the FFSS problem with the objective of minimizing the makespan. Furthermore, Meng et al. [4] propose an enhanced genetic algorithm for minimizing energy consumption by incorporating energy-saving techniques and decoding methodologies. Azadeh et al. [5] adopt the minimization of the total delay time as the optimization objective and introduce an integrated algorithm that combines simulation, artificial neural networks, and genetic algorithms. Reinforcement learning is a significant machine learning technique that focuses on determining optimal behaviors within a given environment to maximize expected benefits. For the scheduling problems, reinforcement learning exhibits the capability to flexibly select actions and generate scheduling policies based on the state characteristics, aligning with actual production conditions. For the FFSS problem of minimizing the maximum completion time, Han et al. [6] and Zhu et al. [7] first adopt the Q-learning algorithm and the proximal policy optimization algorithm, respectively. Reyna and Jimenez [8] introduce an improved Q-learning methodology for FFSS to minimize the makespan. Zhao et al. [9] study a hybrid approach combining the water wave algorithm with the Q-learning methodology. The FFSS problem in distributed assembly contexts is effectively addressed by the incorporation of the Q-learning algorithm, which strikes a balance between the exploration and exploitation capabilities of the algorithm. Ren et al. [10] solve the FFSS problem by employing a neural network using reinforcement learning, in order to minimize the makespan.

Given the intricacies of the production environments and the scheduling challenges encountered in real-world scenarios, there is a growing interest in addressing multi-objective FFSS problems. Li et al. [11] propose a multi-objective optimization model and develop an

enhanced artificial bee colony algorithm to address the FFSS problem, to minimize both the makespan and the processing costs. Zhou et al. [12], Wang et al. [13], and Wang et al. [14] study the dual optimization objectives of minimizing the total energy consumption and the makespan. In order to address these challenges, the imperialist competitive algorithm, the decomposition-based hybrid multi-objective optimization algorithm, and the improved whale optimization algorithm are designed, respectively. The genetic algorithm (GA) is also commonly used to solve the multi-objective optimization problem (Rathnayake [15]). Kong et al. [16] design an improved GA to solve the FFSS problem with the objectives of minimizing the makespan, the total energy consumption, and the costs. To solve the FFSS problem, Lin et al. [17] derive a hybrid optimization algorithm, which integrates the harmony search algorithm and the GA, in order to minimize the makespan and the average flow time. To solve the FFSS problem and minimize the total completion time, the total energy consumption, and carbon emissions, Shi et al. [18] consider a variable-priority dynamic scheduling optimization algorithm based on the GA. Hasani et al. [19] present the non-dominated sorting genetic algorithm (NSGA-II) to solve the multi-stage FFSS problem, with the objective of minimizing the production costs and the total energy consumption. The NSGA-II algorithm is also employed by Wu et al. [20] and Feng et al. [21] for solving multi-objective FFSS problems. Gheisariha et al. [22] propose an enhanced algorithm based on the harmony search algorithm and the Gaussian mutation algorithm, to effectively optimize the makespan and average delay time simultaneously. Zhang et al. [23] employ a three-stage method based on decomposition to solve the FFSS problem, with the objective of minimizing the makespan and the total energy consumption. Mousavi et al. [24] present a heuristic algorithm based on the simulated annealing algorithm to solve the FFSS problem, with the objective of minimizing the makespan and the total tardiness. Schulz et al. [25] propose an iterated local search algorithm to solve the FFSS problem, with the objective of minimizing the makespan, the total energy costs, and the peak power. In addition, for reentrant FFSS, random FFSS, and blocking FFSS, some scholars have considered multiple optimization indicators, including the makespan, the total energy consumption, the total tardiness and earliness, and the advance quantity. The improved multi-objective ant lion algorithm [26], multi-objective artificial bee colony algorithm [27], and migrating birds optimization algorithm [28] have also been applied to find solutions to the multi-objective FFSS problem. In summary, with regard to the multi-objective FFSS problem, the current research primarily involves the development of intelligent optimization algorithms.

Although the application of intelligent optimization algorithms is widespread in the scheduling field, the solution results of the algorithms depend on the setting of the initial value to a great extent. If the initial value is not selected properly, it will greatly affect the convergence speed and the quality of the solution. Therefore, some scholars have tried new methods, such as deep reinforcement learning, to solve scheduling problems.

Deep reinforcement learning combines the robust applicability of reinforcement learning to cope with large-scale state spaces and dynamic changing environments. The function of deep learning is to acquire knowledge from historical data via multi-layer neural networks. The combination of deep learning and reinforcement learning enables the optimization of objective functions in more complex settings, which has gained significant attention to solve scheduling problems in recent years. Due to the complexity of multi-objective problems, few scholars have used deep reinforcement learning to solve FFSS problems. At present, most scholars have applied deep reinforcement learning to the flexible job shop scheduling (FJSS) problem. Du et al. [29] propose a Deep Q-Network (DQN) to address the FJSS problem involving the objectives of crane transportation and preparation time. Their experimental results show that the DQN algorithm can obtain better solutions than intelligent optimization algorithms such as the GA. Luo et al. [30] propose a two-hierarchy DQN algorithm to solve the dynamic FJSS problem, with the objective of optimizing both the total weighted tardiness and average machine utilization rate. Du et al. [31] propose a hybrid multi-objective optimization algorithm, combining the estimation of distribution algorithm and DQN algorithm, to address the FJSS problem, with the objective of opti-

mizing both the makespan and the total electricity price simultaneously. Wang et al. [32] investigated the occurrence of dynamic events in the scheduling problem and established a multi-objective FJSS model to simulate the production environment. On the basis of the DQN algorithm, they incorporated a target network and formulated a Double Deep Q-Network (DDQN) algorithm. Experiments demonstrate the superiority and stability of their approach in comparison to various combined rules, widely recognized scheduling rules, and conventional deep Q-learning algorithms. Wu et al. [33] propose the structure of a dual-layer DDQN algorithm to solve the dynamic FJSS problem with new job arrivals, in order to optimize both the total delay time and the makespan.

The contributions of this paper are as follows: In this paper, the multi-objective FFSS problem is motivated by the actual production process of electronic control modules in the electronic detonators industry. On the basis of the DQN algorithm, the overvaluation problem caused by the maximization process is solved by using a target network. Additionally, the action value is decomposed into the optimal state value and the optimal advantage value by using a dueling network. This approach is particularly effective in handling states that exhibit lower correlation with actions and enhance algorithmic stability. Furthermore, by integrating the idea of Experience Replay, a D3QN algorithm is designed to solve the multi-objective FFSS problem and obtain a feasible schedule.

The remaining sections of this paper are structured as follows: Section 2 describes the problem and presents a mixed integer programming (MIP) model for the multi-objective FFSS problem. In Section 3, we present the D3QN algorithm to solve this scheduling problem. Section 4 reports the experimental results, and lastly, Section 5 presents the conclusion.

## 2. Problem Description

The multi-objective FFSS problem studied in this paper can be described as follows: There is a set of $n$ jobs to be processed at $s$ stages, where each stage has several identical parallel machines. Each job is associated with the distinct processing time on each machine at each stage. The jobs are processed sequentially at all stages. Several assumptions are made as follows: (1) Each job can only be processed on one machine at any time. (2) Each machine can process only one job at any time. (3) Job processing on a machine cannot be interrupted. (4) The job can be processed on any machine at each stage. (5) Each job has its due date. Table 1 lists all the parameters used in the model.

**Table 1.** Parameters used in the proposed model.

| Parameter | Meaning |
| --- | --- |
| $N$ | set of jobs, $N = \{i \mid i = 1, 2, \cdots, n\}$ |
| $S$ | set of stages, $S = \{j \mid j = 1, 2, \cdots, s\}$ |
| $M$ | set of machines, $M = \{M_{1,1}, M_{1,2}, \cdots, M_{1,m_1}, M_{2,1}, M_{2,2}, \cdots, M_{2,m_2}, \cdots, M_{s,1}, M_{s,2}, \cdots, M_{s,m_s}\}$ |
| $m$ | number of machines, $m = \sum_{j=1}^{s} m_j$ |
| $p_{i,j}$ | processing time of job $i$ at stage $j$ |
| $d_i$ | due date of job $i$ |
| $S_{i,j}$ | starting time of job $i$ at stage $j$ |
| $C_{i,j}$ | completion time of job $i$ at stage $j$ |
| $C_{\max}$ | completion time of the last job, makespan, $C_{\max} = \max_{i \in N} C_{i,s}$ |
| $t_i$ | tardiness of job $i$, $t_i = \max\{C_{i,s} - d_i, 0\}$ |
| $T$ | total tardiness |
| $U$ | enormous positive number |
| $X_{i,j,k'}$ | If and only if the job $i$ is processed on the machine $k'$ at stage $j$, $X_{i,j,k'} = 1$, otherwise $X_{i,j,k'} = 0$ |
| $Y_{i,i',j}$ | If and only if job $i'$ is processed after job $i$ at stage $j$, $Y_{i,i',j} = 1$, otherwise $Y_{i,i',j} = 0$ |

The objective of the FFSS problem is to find a feasible schedule of production such that both the makespan and the total tardiness are minimized. The multi-objective FFSS

problem is described by a triplet $FF_s||C_{\max}, T$, where $FF_s$ means that the flexible flow shop involves $s$ stages and $C_{\max}$ and $T$ denotes the makespan and the total tardiness, respectively. Hence, the MIP model presented in this paper is formulated as follows.

Minimize:

$$\min C_{\max} \tag{1}$$

$$\min T = \sum_{i=1}^{n} t_i \tag{2}$$

Subject to:

$$\sum_{k'=1}^{m_j} X_{i,j,k'} = 1, i = 1, 2, \cdots, n; j = 1, 2, \cdots, s \tag{3}$$

$$Y_{i,i',j} + Y_{i',i,j} = 1, i = 1, 2, \cdots, n; i' = 1, 2, \cdots, n; j = 1, 2, \cdots, s \tag{4}$$

$$S_{i',j} - C_{i,j} + V \geq 0, i = 1, 2, \cdots, n; i' = 1, 2, \cdots, n; j = 1, 2, \cdots, s \tag{5}$$

where $V = U(3 - Y_{i,i',j} - X_{i,j,k'} - X_{i',j,k'}), i = 1, 2, \cdots, n; i' = 1, 2, \cdots, n; j = 1, 2, \cdots, s; k' = 1, 2, \cdots, m_j$

$$S_{i,j+1} \geq C_{i,j}, i = 1, 2, \cdots, n; j = 1, 2, \cdots, s \tag{6}$$

$$C_{i,j} = S_{i,j} + p_{i,j}, i = 1, 2, \cdots, n; j = 1, 2, \cdots, s \tag{7}$$

Formulas (1) and (2) represent the objectives to minimize the completion time of the last job at the final stage and the total tardiness for all jobs, respectively. Constraint (3) indicates that each job must be processed at all stages, and can be processed once on one machine at each stage. Constraint (4) signifies a sequence of the different jobs at the same stage. Constraint (5) shows the job sequence on the same machine. Constraint (6) implies that the starting time of a job at each stage is determined by its completion time at the previous stage. Constraint (7) implies that the completion time of a job at each stage is determined by its starting time and the processing time.

## 3. D3QN Algorithm for Solving Problem $FF_s||C_{\max}, T$

The problem $FF_s||C_{\max}, T$ has been proved to be NP-hard. As the size of the problem expands, its complexity increases exponentially, making it challenging for the traditional accurate algorithms to find optimal solutions. Reinforcement learning and neural networks are combined in the deep reinforcement learning approach; this approach autonomously facilitates the acquisition of representations of environments and tasks from raw data. This approach is more suitable to address intricate decision-making problems.

Initially, the selection of the machines at each stage and the job sequence to be processed on each machine are determined by a sequential decision-making process. Subsequently, the scheduling problem is converted into a Markov decision problem, where the state is defined by the parameters of the machines and the jobs, including the average utilization rate of all machines, the average processing completion rate of all jobs, and the average processing tardiness rate of all jobs. Concurrently, heuristic rules are utilized as actions, calculating immediate rewards while taking into account the present state of both the makespan and the total tardiness, as well as the change in state following the execution of the actions.

Considering the increasing computational complexity of the DQN algorithm, coupled with the challenges of parameter adjustment and the tendency of overfitting, the issue of Q-value overestimation in the algorithm is solved by the incorporation of a target network. Simultaneously, the idea of Q-value decomposition is introduced, based on the proposal of the D3QN algorithm, which combines the state value function, the advantage function by using a dueling network, and an experience replay buffer. This method can be applied to deal with the problem $FF_s||C_{\max}, T$, with the objective to optimize the scheduling scheme and ultimately achieve enterprise productivity.

### *3.1. Problem Transformation*

The problem $FF_s||C_{\max}, T$ is solved by the D3QN algorithm. The primary step is to transform the scheduling problem into a Markov decision process. In this process, the state is utilized to depict the variations and characteristics of the overall manufacturing system environment. Actions are used to represent the decision-making behavior of the agent, while the rewards are employed to reflect the outcomes of the interactions between the agent and the environment. The definitions of the state, the action, and the reward are outlined as follows.

#### 3.1.1. State Features

The state is defined by the variations in the characteristics of the machines and the jobs. Given the fluctuation of certain properties in them, along with the inconsistency of dimensions, six critical features are selected to describe the states of the scheduling problem. State features 1 and 2 describe the characteristics of the machines: State feature 1 represents the average utilization rate of all machines. State feature 2 represents the standard deviation of the average utilization rate of all machines. State features 3–6 describe the characteristics of the jobs: State feature 3 represents the average processing completion rate of all jobs. State feature 4 represents the standard deviation of the average processing completion rate of all jobs. State feature 5 represents the average processing tardiness rate of all jobs. State feature 6 represents the standard deviation of the average processing tardiness rate of all jobs.

Let $t$ denote a decision moment. A decision moment refers to the moment when the agent needs to choose the action that will be rewarded the most according to the state features. The agent makes decisions at the moment of the state transition. The decision moment $t$ is the moment when the state transitions for the $t$-th time. At the decision moment $t$, the definitions of the state features are shown as below.

State feature 1.

$$U_{ave}(t) = \frac{\sum_{k=1}^{m} U_k(t)}{m} \tag{8}$$

where $U_k(t) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{OP_i(t)} p_{i,j}\alpha}{CT_k(t)}$ represents the machine utilization. If job $i$ is processed on machine $k$, $\alpha = 1$, otherwise $\alpha = 0$. $CT_k(t)$ denotes the total overload time of machine $k$; $OP_i(t)$ denotes the number of the stages for the completed job $i$.

State feature 2.

$$U_{std}(t) = \sqrt{\frac{\sum_{k=1}^{m} (U_k(t) - U_{ave}(t))^2}{m}} \tag{9}$$

State feature 3.

$$CRJ_{ave}(t) = \frac{\sum_{i=1}^{n} CRJ_i(t)}{n} \tag{10}$$

where $CRJ_i(t) = \frac{OP_i(t)}{s}$ represents the processing completion rate of job $i$.

State feature 4.

$$CRJ_{std}(t) = \sqrt{\frac{\sum_{i=1}^{n} (CRJ_i(t) - CRJ_{ave}(t))^2}{n}} \tag{11}$$

State feature 5.

$$Tard_{ave}(t) = \frac{\sum_{i=1}^{n} Tard_i(t)}{n} \tag{12}$$

where $Tard_i(t) = \frac{C_{i,OP_i(t)} + \sum_{j=OP_i(t)}^{s} p_{i,j} - d_i}{C_{i,OP_i(t)} + \sum_{j=OP_i(t)}^{s} p_{i,j}}$ represents the tardiness rate of job $i$.

State feature 6.

$$Tard_{std}(t) = \sqrt{\frac{\sum_{i=1}^{n} (Tard_i(t) - Tard_{ave}(t))^2}{n}} \tag{13}$$

3.1.2. Action

The actions in the D3QN algorithm for solving the problem $FF_s||C_{max}, T$ are determined based on the decision of the jobs and the machines. To minimize the action space and describe the actual production process accurately, the rules for job selection and machine selection are designed with reference to the heuristic scheduling rule and objective functions $C_{max}$ and $T$. The rules for job selection are outlined as follows.

1. SPT Rule: Select a job using the shortest processing time rule (SPT rule). The jobs are indexed in the SPT rule;
2. LPT Rule: Select a job using the longest processing time rule (LPT rule). The jobs are indexed in the LPT rule;
3. EDD Rule: Select a job using the earliest due date rule (EDD rule). The jobs are indexed in the EDD rule;
4. ODD Rule: Select a job using the operation due date rule (ODD rule). The jobs are indexed in the ODD rule;
5. SRP Rule: Select a job using the shortest remaining processing time rule (SRP rule). The jobs are indexed in the SRP rule;
6. LNP Rule: Select a job using the longest processing time for the next process rule (LNP rule). The jobs are indexed in the LNP rule;
7. SNP Rule: Select a job using the shortest processing time for the next process rule (SNP rule). The jobs are indexed in the SNP rule.

The rules for the selection of a machine for jobs are as follows:

1. FCFS Rule: Select the machine using the first come first serve rule (FCFS rule). The machines are indexed in the FCFS rule;
2. WINQ rule: Select the machine using the shortest total processing time rule (WINQ rule). The machines are indexed in the WINQ rule.

By combining the above two rule sets together, a total of 14 combination rules are obtained, serving as the actions in the D3QN algorithm for solving the problem $FF_s||C_{max}, T$. The specific actions are shown in Table 2.

**Table 2.** Job–machine combination rules.

| Combination Rules | | Machine Rules | |
|---|---|---|---|
| | | **FCFS** | **WINQ** |
| Job rules | SPT | SPT-FCFS | SPT-WINQ |
| | LPT | LPT-FCFS | LPT-WINQ |
| | EDD | EDD-FCFS | EDD-WINQ |
| | ODD | ODD-FCFS | ODD-WINQ |
| | SRP | SRP-FCFS | SRP-WINQ |
| | LNP | LNP-FCFS | LNP-WINQ |
| | SNP | SNP-FCFS | SNP-WINQ |

In the initial state $s_0$, the agent proceeds to select a job and allocates it to a machine where all machines are idle. Subsequently, the system transitions into a new state when a job has finished processing on a machine. At a decision moment $t$, the agent selects an action $a_t$ based on $s_t$. The state then transfers into $s_{t+1}$ at the next decision moment, while the agent receives a time-delay reward $r_t$. As previously mentioned, state transitions occur at the time when any job completes a certain process on the machine. Then, the agent needs to select an action in this new state. Once all the jobs have completed the last process, the agent finishes the work.

3.1.3. Reward

In a deep reinforcement learning algorithm, the reward function can be formulated to guide the learning process of the agent to meet the requirements of multi-objective optimization. The reward function of the D3QN algorithm is formulated by evaluating

variations in the current state and the state after the action execution in two aspects: the makespan and the total tardiness. In the process of action selection, the $\varepsilon$-greedy strategy is adopted to facilitate a comprehensive exploration and exploitation, with the aim of better spotting the relationships, including the states, the actions, and the rewards. The specific steps are outlined as follows.

Step 1. Exploration. Using the $\varepsilon$-greedy strategy, a random number is compared with $\varepsilon$. If the random number is less than $\varepsilon$, the agent randomly selects an action from the 14 job–machine combination rules. This strategy ensures that the agent can explore the environment, rather than being limited solely to the existing optimal actions.

Step 2. Exploitation. According to the $\varepsilon$-greedy strategy, the exploitation phase is entered when the random number exceeds $\varepsilon$. By calculating the Q-value for each available action in the current state, the agent selects the action that maximizes the reward and executes it.

Step 3. The reward is calculated by using the makespan and the total tardiness, as outlined below:

$$f_1(t) = -0.01 \cdot C_{\max}(t+1) \tag{14}$$

$$f_2(t) = -0.01 \cdot T(t+1) \tag{15}$$

where $f_1(t)$ represents the relationship between the makespan and the reward, and $f_2(t)$ represents the relationship between the total tardiness and the reward.

Step 4. The instant reward at the time of decision moment $t$, is as follows:

$$r_t = f_1(t) + f_2(t) \tag{16}$$

The final reward function, denoted as $R$, is defined as the summation of rewards across $K$ decision moments, as shown in Equation (17).

$$R = \sum_{t=1}^{K} r_t \tag{17}$$

where $K$ is the total number of moments the agent needs to make a decision.

By formulating the reward function mentioned above, the agent can be directed towards efficient learning and decision-making in addressing problem $FF_s \| C_{\max}, T$ with the objective of optimizing both the makespan and the total tardiness.

### 3.2. D3QN Algorithm

The traditional DQN algorithm uses a single neural network to fit the optimal action value function. To achieve a more efficient approximation of the optimal action value function, a dueling network comprising two subnetworks is designed for the D3QN algorithm. These two subnetworks are used to approximate the state value function and the dominance function, respectively. The D3QN algorithm decomposes the optimal action value into the optimal state value and the optimal advantage value by using the dueling network. In the D3QN algorithm, the inputs of the dueling network are the states of the machines and the jobs. The outputs of the two subnetworks are the value of the state and the advantage of each action, respectively. The value of the state and the advantage of the action can be used to determine the optimal action value. The specific calculation formula is outlined as follows:

$$Q(s,a;w) \triangleq V(s;w^V) + D(s,a;w^D) - \underset{a \in A}{mean} D(s,a;w^D) \tag{18}$$

where $Q$ represents a dueling network, and $Q(s,a;w)$ represents the action value of the optimal action $a$ in state $s$. $w = (w^V, w^D)$, where $w^V$ and $w^D$ are the parameters for the optimal state value and the optimal advantage network, respectively. $V(s;w^V)$ represents the state

value of state $s$. $D(s,a;w^D)$ represents the action advantage of $a$ in state $s$. $\underset{a\in A}{mean}D(s,a;w^D)$ represents the average of all action advantage values in state $s$.

The dueling network facilitates the agent with a more accurate and efficient capacity for learning the state value functions for the problem $FF_s\|C_{\max},T$. The network structure of the target network in the D3QN algorithm is identical to that of the dueling network. The issue of overestimating the action value is solved by calculating action values through the target network. The incorporation of the experience replay in the D3QN algorithm breaks the correlation of the data sequence, making the trained data mutually independent. Additionally, it permits the reutilization of the collected data to achieve the objective of the algorithm with less data. The D3QN algorithm is designed to solve the problem $FF_s\|C_{\max},T$ by combining the target network, the dueling network, and the experience replay. The implementation flow of the D3QN algorithm is depicted in Figure 2.

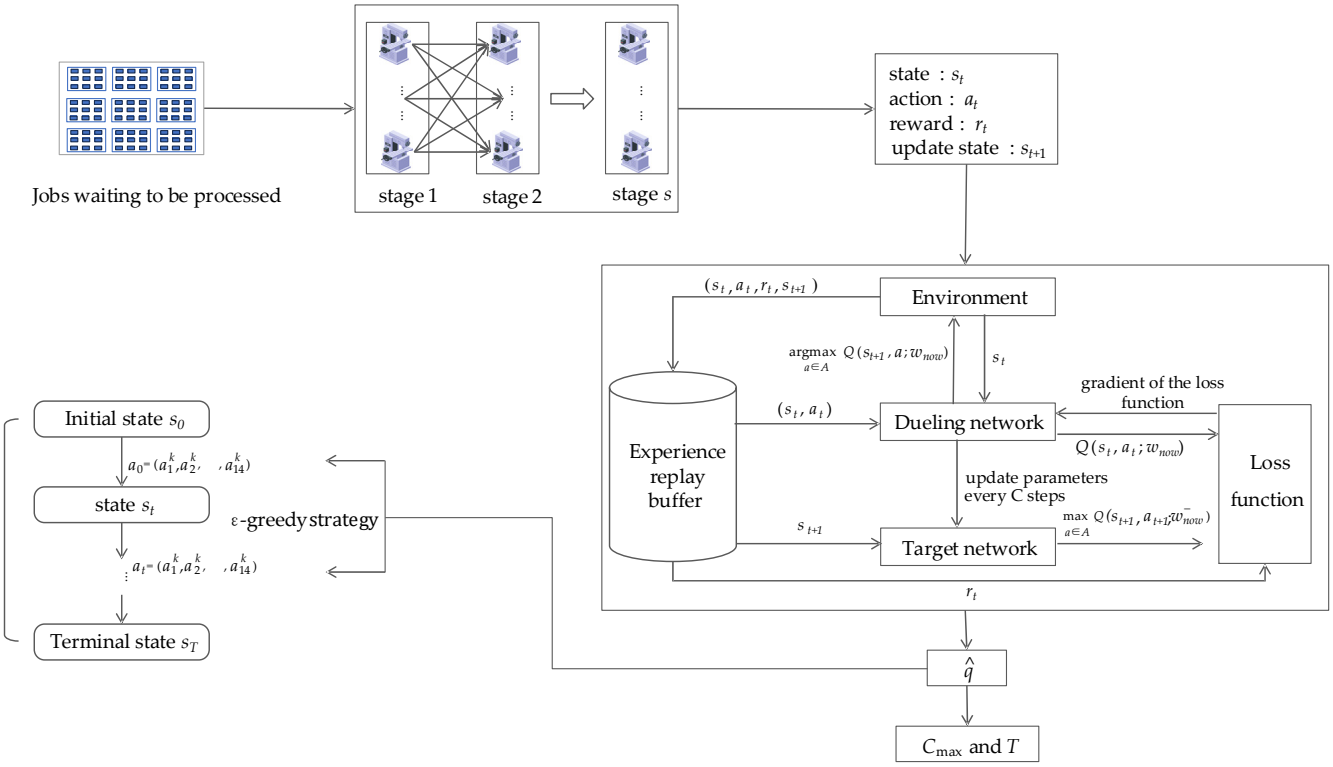



**Figure 2.** Implementation of the D3QN algorithm for solving the problem $FF_s\|C_{\max},T$.

The specific steps of the D3QN algorithm presented in this paper are given as below.

Step 1. Initialize the parameters of the problem $FF_s\|C_{\max},T$ and the D3QN algorithm. Schedule the problem parameters: the set of jobs $N$, the set of machines $M$, the processing time $p_{i,j}$ of job $i \in N$ at stage $j \in S$, and the due date $d_i$ of the job $i \in N$. Algorithm parameters: the learning rate $\alpha$, the exploration rate $\varepsilon$, the discount factor $\gamma$, the sample batch size *batch_size* for updating the network, the number of steps of the target network update $C$, and the maximum iteration number *Max_episode*. Initialize the dueling network $Q$ and the target network $Q^-$ by using the random dueling network parameters $w_{now}$ and the target network parameter $w^-_{now}$. Let *episode* = 0.

Step 2. Define the initial state, $s_0 = \{U_{ave}, U_{std}, CRJ_{ave}, CRJ_{std}, Tard_{ave}, Tard_{std}\}$.

Step 3. Choose an action $a_t$ randomly with the probability of $\varepsilon$ or use the dueling network to choose the optimal action for the current state $a_t = \underset{a\in A}{\arg\max} Q(s_t, a; w_{now})$. Execute action $a_t$, update the state to $s_{t+1}$, receive the reward $r_t$, and store the resulting quadruple $(s_t, a_t, r_t, s_{t+1})$ into the experience replay $D$.

Step 4. Check whether the quantity of data in $D$ is greater than *batch_size*. If it is greater, randomly take *batch_size* quadruples $(s_j, a_j, r_j, s_{j+1})$, $j \in \{0, 1, 2, \ldots, |D|\}$ for training; otherwise, return to Step 3. The training process is as follows:

(1) When the dueling network parameter is $w_{now}$, and the state is $s_j$, use the dueling network $Q$ for positive propagation. According to the Formula (18), obtain the $q$ value of the action $a_j$, $\hat{q}_j = Q(s_j, a_j; w_{now})$.

(2) Use the dueling network to select the action $a^* = \arg\max_{a \in A} Q(s_{j+1}, a; w_{now})$ with the maximum $q$ value at state $s_{t+1}$.

(3) Use the target network to obtain the $q$ value of output state $s_{t+1}$ under $a^*$, $\hat{q}_{j+1} = Q^-(s_{j+1}, a^*; w^-_{now})$.

(4) Calculate the TD target $\widetilde{y}_j = r_j + \gamma \hat{q}_{j+1}$ and TD error $\delta_j = \hat{q}_j - \widetilde{y}_j$.

(5) Perform the reverse spread to the dueling network and obtain the gradient $\nabla_w Q(s_j, a_j; w_{now})$.

(6) Update the dueling network parameters by the average stochastic gradient descent algorithm, $w_{new} \leftarrow w_{now} - \alpha \delta_j \nabla_w Q(s_j, a_j; w_{now})$.

(7) Assign the parameters of the dueling network to the target network after $C$ steps.

Step 5. Determine whether all jobs are processed or not and go to Step 6 if completed. Otherwise, return to Step 3.

Step 6. Determine whether *episode* reaches *Max_episode*, and if it is less than *Max_episode*, *episode* = *episode* + 1, return to Step 2. Otherwise, output the optimal scheduling solutions, ending the process.

The flow chart of the D3QN algorithm is shown in Figure 3.

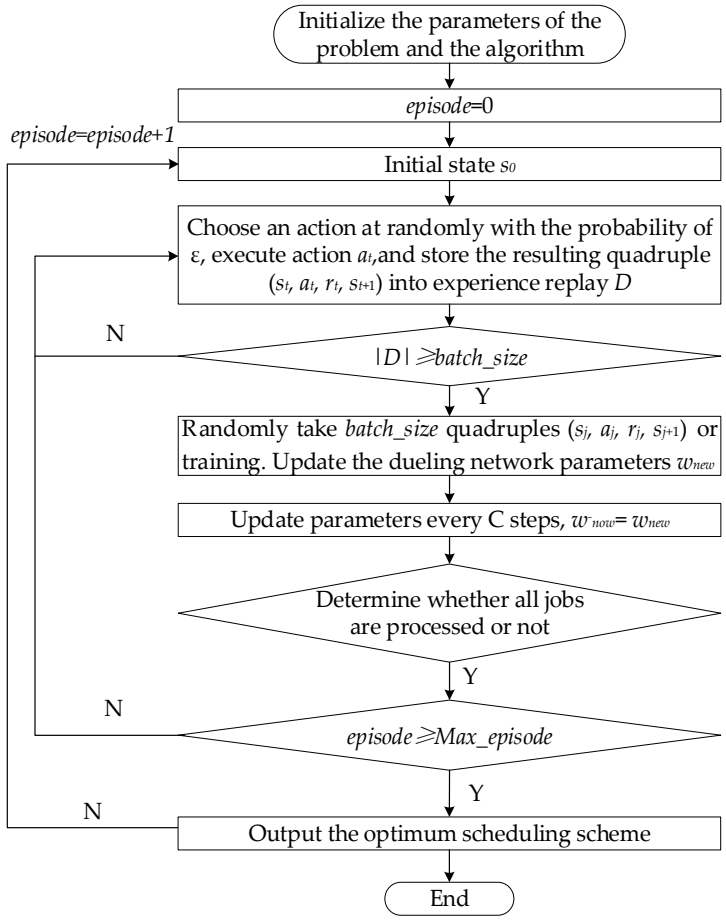



**Figure 3.** The flow chart of the D3QN algorithm.

## 4. Computational Experiments

In this section, we report the computational experiments to evaluate the performance of the D3QN algorithm for the problem $FF_s||C_{\max}, T$. To validate the adaptability of the D3QN algorithm for various problem sizes, the test instances are randomly generated and compared with Gurobi and the heuristic rules.

### *4.1. Experimental Environment and Parameter Settings*

The D3QN algorithm is coded in Python, and the program was run on the PyCharm Community Edition 2021.3.3. The experiments are conducted on a personal computer with Intel (R) Core i5-6300HQ CPU @2.30 GHz, and 8.00 GB RAM.

Based on the D3QN algorithm, the parameters are set as follows: $\alpha$ is the learning rate, which controls the magnitude of the weight parameters and updates in the training process of the neural network, and it is set at $\alpha = 0.001$; $\gamma$ is the discount factor used to calculate the cumulative rewards, and it is set at $\gamma = 0.95$; $\varepsilon$ represents the greedy factor, and it is set at $\varepsilon = 0.6$. The upper limit of iterations *Max_episode* is 500. The parameters of the problem $FF_s||C_{\max}, T$ are set as follows: the processing time of the jobs at each stage are generated by the uniform distributions, the processing times of the electrical performance testing $p_{i1}, p_{i3}, p_{i5} \sim U[1, 10]$, and the processing times of the visual testing $p_{i2}, p_{i4}, p_{i6} \sim U[11, 20]$.

### *4.2. The Experimental Results of the Model and the D3QN Algorithm*

In order to validate the effectiveness of the model and the D3QN algorithm, Gurobi optimization software (Gurobi Optimizer version 10.0.2 build v10.0.2rc0 (win64)) is employed to solve five groups of fifteen experimental instances spanning different sizes. Each instance is subject to a maximum allowable runtime constraint of one hour. The due dates of the jobs at each stage follows a uniform distribution, denoted as $d_i \sim U[12, 40]$. The comparation of the solutions generated using Gurobi and the D3QN algorithm is shown in Table 3. Columns 4–8 represent the incumbents, the bestbounds, and the runtimes obtained using Gurobi for the instances of the multi-objective mixed integer programming model. The symbol "-" denotes that Gurobi is unable to obtain the global optimal solutions within one hour.

**Table 3.** $C_{\max}$, $T$, and the runtime obtained by Gurobi and the D3QN algorithm.

| $n$ | $s$ | $m_j$ | Gurobi | | | | | D3QN | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | $C_{\max}$ | | $T$ | | Runtime | $C_{\max}$ | $T$ | Runtime |
| | | | Incumbent | BestBound | Incumbent | BestBound | | | | |
| 5 | 2 | 2, 1 | 69 | 69 | 113 | 113 | 2.112 s | 69 | 125 | 30.111 s |
| | 2 | 2, 2 | 40 | 40 | 40 | 40 | 6.383 s | 40 | 41 | 27.391 s |
| | 3 | 2, 1, 2 | 75 | 75 | 113 | 113 | 13.334 s | 75 | 113 | 37.733 s |
| 6 | 2 | 2, 1 | 85 | 85 | 178 | 178 | 15.383 s | 87 | 178 | 36.455 s |
| | 2 | 2, 2 | 45 | 45 | 67 | 67 | 26.768 s | 47 | 68 | 34.721 s |
| | 3 | 2, 1, 2 | 89 | 89 | 178 | 178 | 49.575 s | 89 | 185 | 41.421 s |
| 7 | 2 | 2, 1 | 99 | 99 | 251 | 251 | 66.824 s | 101 | 251 | 35.125 s |
| | 2 | 2, 2 | 54 | 54 | 99 | 99 | 1308.473 s | 55 | 101 | 40.372 s |
| | 3 | 2, 1, 2 | 103 | 103 | 251 | 251 | 2251.252 s | 103 | 259 | 48.642 s |
| 8 | 2 | 2, 1 | 111 | 111 | 333 | 333 | 435.219 s | 113 | 333 | 38.348 s |
| | 2 | 2, 2 | 58 | 58 | 137 | 137 | 1945.409 s | 60 | 138 | 39.874 s |
| | 3 | 2, 1, 2 | 115 | 115 | 333 | 333 | 2187.866 s | 117 | 333 | 54.423 s |
| 9 | 2 | 2, 1 | 126 | 126 | 436 | 436 | 2612.291 s | 128 | 436 | 43.576 s |
| | 2 | 2, 2 | - | 66 | - | 184 | 3600 s | 70 | 184 | 42.457 s |
| | 3 | 2, 1, 2 | - | 130 | - | 436 | 3600 s | 130 | 447 | 65.327 s |

Table 3 lists the scheduling solutions of the problem instances solved by Gurobi and the D3QN algorithm. The $C_{\max}$ generated by the D3QN algorithm is longer than that of

Gurobi by 6.061% and shorter than that of Gurobi by 0. The total tardiness $T$ generated by the D3QN algorithm is longer than that of 10.618%, and shorter than that of Gurobi by 0. As the size of the problem increases, the D3QN algorithm runs faster than Gurobi, and Gurobi cannot solve the instance of the problem with a size of $n = 9$ within one hour.

### *4.3. The Results for Large-Size Instances*

In order to verify the effectiveness of the D3QN algorithm, the experiments are conducted with problems of various sizes. The parameters of the problem $FF_s||C_{\max}, T$ are set as follows: the number of the stages $s = 6$, the number of the machines $m_j$ at stage $j$ are 8, 2, 8, 1, 4, and 1, respectively. The due dates of the jobs at each stage follow a uniform distribution, $d_i \sim U[36, 90]$. The number of jobs $n$ is 15, 30, 50, 100, and 200, respectively. The makespan $C_{\max}$ and the total tardiness $T$ obtained by the different heuristic algorithms, the GA, and the D3QN algorithm, are shown in Table 4. The algorithms Rule1–Rule14 in Table 4 are constructed according to the heuristic rules corresponding to the actions of the job and the machine.

**Table 4.** $C_{\max}$ and $T$ obtained by the different heuristic algorithms, the GA, and the D3QN algorithm.

| Symbol | Algorithm | $C_{\max}$ | | | | | $T$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $n = 15$ | $n = 30$ | $n = 50$ | $n = 100$ | $n = 200$ | $n = 15$ | $n = 30$ | $n = 50$ | $n = 100$ | $n = 200$ |
| Rule1 | SPT-FCFS | 339 | 640 | 1022 | 2009 | 3916 | 2224 | 9757 | 27,149 | 111,886 | 444,280 |
| Rule2 | LPT-FCFS | 352 | 670 | 1124 | 2222 | 4363 | 2618 | 11,007 | 32,743 | 133,063 | 528,625 |
| Rule3 | EDD-FCFS | 280 | 524 | 844 | 1654 | 3270 | 1499 | 6552 | 18,803 | 77,392 | 314,297 |
| Rule4 | ODD-FCFS | 280 | 524 | 844 | 1654 | 3261 | 1499 | 6599 | 18,785 | 77,439 | 315,513 |
| Rule5 | SRP-FCFS | 280 | 524 | 844 | 1654 | 3270 | 1499 | 6552 | 18,803 | 77,392 | 314,297 |
| Rule6 | LNP-FCFS | 352 | 670 | 1124 | 2222 | 4363 | 2618 | 11,007 | 32,743 | 133,063 | 528,625 |
| Rule7 | SNP-FCFS | 339 | 640 | 1022 | 2009 | 3916 | 2224 | 9757 | 27,149 | 111,886 | 444,280 |
| Rule8 | SPT-WINQ | 324 | 629 | 1020 | 2008 | 3916 | 2014 | 9460 | 27,055 | 111,789 | 444,280 |
| Rule9 | LPT-WINQ | 352 | 686 | 1123 | 2214 | 4372 | 2618 | 11,471 | 32,683 | 132,245 | 530,195 |
| Rule10 | EDD-WINQ | 280 | 524 | 844 | 1654 | 3270 | 1499 | 6552 | 18,803 | 77,392 | 314,297 |
| Rule11 | ODD-WINQ | 283 | 524 | 844 | 1654 | 3266 | 1511 | 6550 | 18,799 | 77,387 | 314,111 |
| Rule12 | SRP-WINQ | 280 | 524 | 844 | 1654 | 3270 | 1499 | 6552 | 18,803 | 77,392 | 314,297 |
| Rule13 | LNP-WINQ | 352 | 686 | 1123 | 2214 | 4372 | 2618 | 11,471 | 32,683 | 132,245 | 530,195 |
| Rule14 | SNP-WINQ | 324 | 629 | 1020 | 2008 | 3916 | 2014 | 9460 | 27,055 | 111,789 | 444,280 |
| GA | Genetic Algorithm | 281 | 517 | 840 | 1654 | 3230 | 1566 | 6567 | 19,183 | 78,323 | 315,776 |
| our | D3QN | 272 | 516 | 838 | 1643 | 3228 | 1478 | 6538 | 18,704 | 77,016 | 310,512 |

As shown in Table 4, the D3QN algorithm is compared with several heuristic algorithms for the different instances of the scheduling problem. $C_{\max}$ is reduced by at least 0.19% and at most 26.17%. $T$ is reduced by at least 0.18% and at most 42.92%. The experimental results illustrate that the D3QN algorithm provided for the problem $FF_s||C_{\max}, T$ can effectively obtain better solutions than 14 heuristic rules and the GA.

### *4.4. Ablation Experiment Results of the D3QN Algorithm*

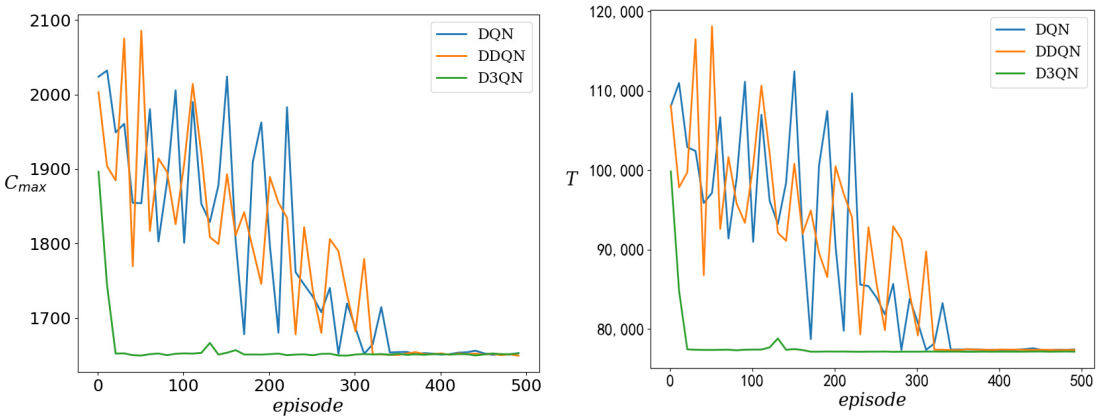
The purpose of an ablation experiment is to assess the influence of individual elements on performance by systematically eliminating or modifying them. Based on the DQN algorithm, the DDQN algorithm is constructed by incorporating a target network, while the D3QN algorithm is formulated by integrating both a target network and a dueling network. Ablation experiments are designed for these three algorithms to assess the influence of incorporating a target network and a dueling network in resolving the problem $FF_s||C_{\max}, T$ of different sizes, as shown in Table 5.

Based on the outcomes presented in Table 5, the improvement offered by the DDQN algorithm is not evident, which solely incorporates the target network compared with the original DQN algorithm. However, with the incorporation of both the target network and the dueling network in the D3QN algorithm, significant improvements in performance are achieved when solving scheduling problems of five different sizes. The D3QN algorithm not only yields superior results in terms of the makespan and the total tardiness, but also exhibits enhanced convergence stability. Taking 100 jobs as an example, the variation trends

of the makespan with respect to the number of iterations are shown in Figure 4a, and the variation trends of total tardiness with respect to the number of iterations are shown in Figure 4b. The experimental results indicate that the target network and the dueling network can mutually boost each other and achieve a better performance.

**Table 5.** Comparison of objective function values for the different algorithms.

| Algorithm | $C_{max}$ | | | | | $T$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $n$ = 15 | $n$ = 30 | $n$ = 50 | $n$ = 100 | $n$ = 200 | $n$ = 15 | $n$ = 30 | $n$ = 50 | $n$ = 100 | $n$ = 200 |
| DQN(baseline) | 277 | 522 | 844 | 1647 | 3260 | 1488 | 6587 | 18,754 | 77,369 | 313,847 |
| +Target Network | 277 | 524 | 842 | 1647 | 3253 | 1486 | 6553 | 18,842 | 77,318 | 310,862 |
| +Target Network and Dueling Network | 272 | 516 | 838 | 1643 | 3228 | 1478 | 6538 | 18,704 | 77,016 | 310,512 |




(**a**) (**b**)

**Figure 4.** Convergence curve of different objectives: (**a**) convergence curve of the makespan; (**b**) convergence curve of the total tardiness.

### *4.5. Results Analysis of Scheduling Problem Based on D3QN Algorithm*

Taking the number of the jobs with $n$ = 15 as an instance, the D3QN algorithm can obtain the optimal scheduling solution with a makespan of 272 and a total tardiness of 1478 for the problem $FF_s||C_{max}, T$. The Gantt chart of the optimal schedule for the instance with $n$ = 15 is shown in Figure 5.

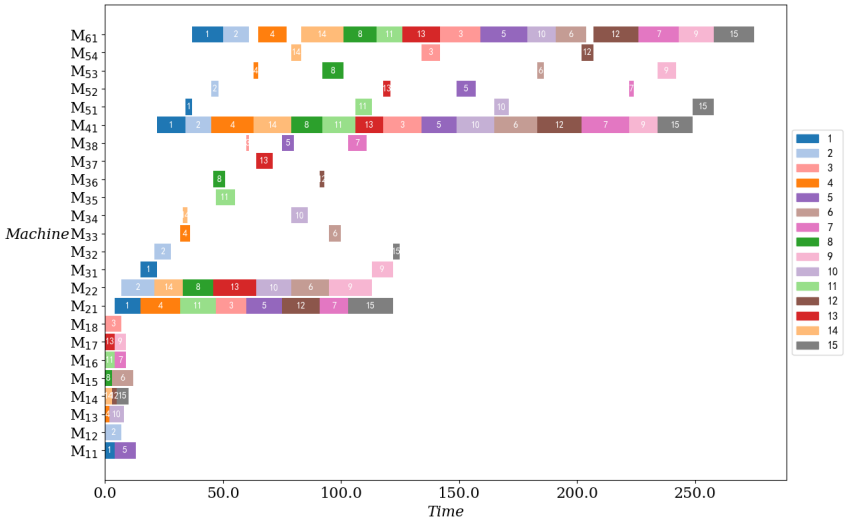



**Figure 5.** The Gantt chart of the optimal schedule.

#### 4.5.1. Action Selections Based on the D3QN Algorithm

In order to analyze the usage frequency of various actions involved in the optimal strategy based on the experimental results, a usage frequency distribution diagram of the 14 job–machine combination rules is generated. As shown in Figure 6, the actions that have been used more than 5000 times include ODD-FCFS, SRP-FCFS, EDD-WINQ, and SRP-WINQ. These actions have made significant contributions to achieve optimal solutions. The usage frequency of other actions exhibits a relatively even distribution, and the performance is not particularly noteworthy.

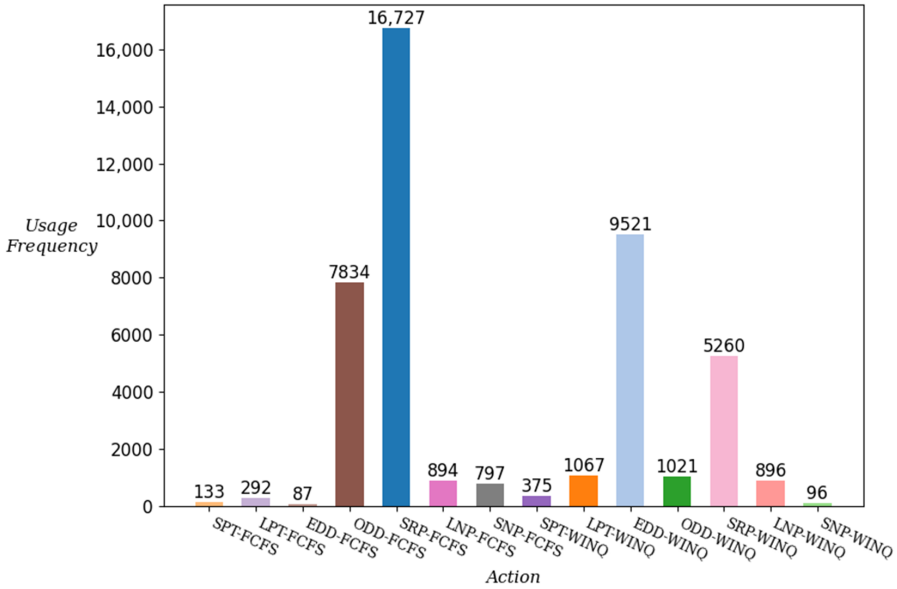



**Figure 6.** The distribution of the actions in the D3QN algorithm.

#### 4.5.2. The Variation Trend of Objective Functions Based on the D3QN Algorithm

The variation trend of the reward value with respect to the iteration times in the D3QN algorithm is shown in Figure 7. The cumulative reward of the algorithm tends to rise as the iteration time increases. The algorithm tends to converge approximately after 50 iterations. The variation trends of the makespan with respect to the number of the iterations are shown in Figure 8a, and variation trends of the total tardiness with respect to the number of the iterations are shown in Figure 8b. From this figure, as the iteration time increases, both the makespan and the total tardiness show a declining trend. This indicates that the D3QN algorithm can obtain better near-optimal solutions and outperform the heuristic rules in both effectiveness and efficiency measurement for the problem $FF_s||C_{\max}, T$.

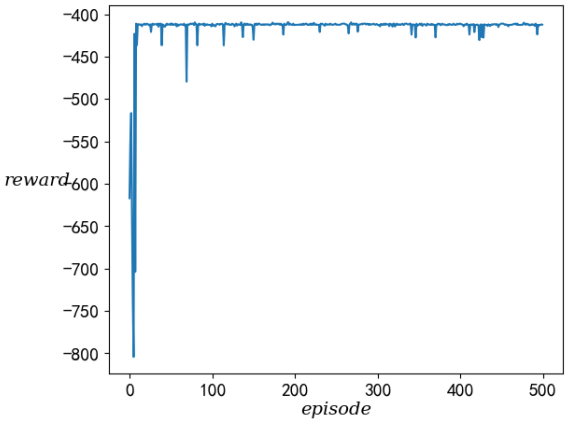
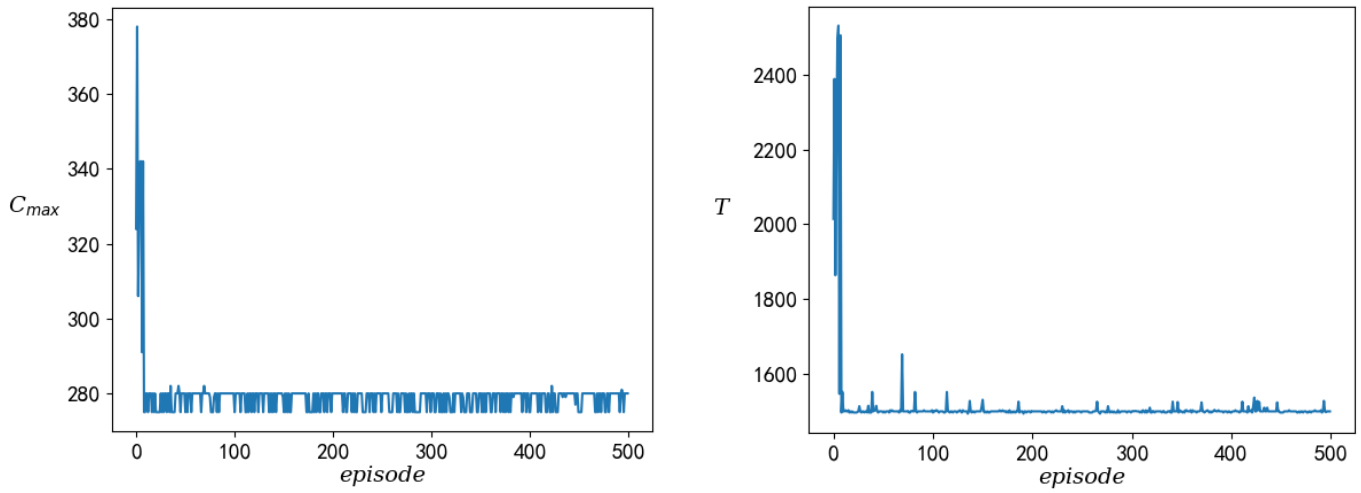



**Figure 7.** Variation trend of the reward.




(**a**) (**b**)

**Figure 8.** Variation trends of different objectives: (**a**) variation curve of the makespan; (**b**) variation curve of the total tardiness.

## 5. Conclusions

In this paper, the problem $FF_s||C_{\max}, T$, arising from the production process of electronic control modules in the digital electronic detonators industry, is considered. The objective is to minimize both the makespan and the total tardiness. The scheduling problem is described as a multi-objective MIP model. The D3QN algorithm is designed based on the DQN algorithm, which integrates a target network, a dueling network, and an experience replay buffer, to solve the proposed scheduling problem. The experiments that compared the D3QN algorithm with the heuristic rules and the GA illustrate that the incorporation of the target network, the dueling network, and the experience replay buffer accelerates the speed at which the problem is solved, improves the quality of near-optimal scheduling solutions, and enhances the effectiveness of the algorithm. Ablation experiments validate the significant advantages of the D3QN algorithm in terms of both the quality and the convergence rate when it is compared with the DQN algorithm and the DDQN algorithm to solve the problem $FF_s||C_{\max}, T$. An interesting future issue will be to consider the problem with uncertain constraints, such as dynamic arrival jobs and random processing times. It would also be interesting to consider the problem by taking other objective functions into account.

**Author Contributions:** Conceptualization, H.G.; methodology, W.X. and W.S.; software, W.X. and K.X.; validation, H.G. and W.S.; formal analysis, H.G. and K.X.; resources, K.X.; data curation, W.X.; writing—original draft preparation, W.X.; writing—review and editing, H.G. and W.S.; supervision, H.G.; project administration, H.G.; funding acquisition, H.G. and W.S. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by the Liaoning Province Xingliao Talents Plan Project (Grant No. XLYC2006017) and the Scientific Research Funds Project of the Educational Department of Liaoning Province (Grant Nos. JYTMS20230201 and LJKZ0260).

**Data Availability Statement:** The processing time of the job at each stage and the due date was randomly created as explained in the paper.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References


1. Han, D.Y.; Tang, Q.H.; Zhang, Z.K.; Li, Z.X. An improved migrating birds optimization algorithm for a hybrid flow shop scheduling within steel plants. *Mathematics* **2020**, *8*, 1661. [CrossRef]
2. Shi, W.G.; Song, C.L. Improved grey wolf optimization to solve the hybrid flow shop scheduling problem. *Comput. Integr. Manuf. syst.* **2021**, *27*, 3196–3208.
3. Malekpour, H.; Hafezalkotob, A.; Khalili-Damghani, K. Product processing prioritization in hybrid flow shop systems supported on Nash bargaining model and simulation-optimization. *Expert Syst. Appl.* **2021**, *180*, 115066. [CrossRef]
4. Meng, L.L.; Zhang, C.Y.; Shao, X.Y.; Ren, Y.P.; Ren, C.L. Mathematical modelling and optimisation of energy-conscious hybrid flow shop scheduling problem with unrelated parallel machines. *Int. J. Prod. Res.* **2019**, *57*, 1119–1145. [CrossRef]
5. Azadeh, A.; Goodarzi, A.H.; Kolaee, M.H.; Jebreili, S. An efficient simulation-neural network-genetic algorithm for flexible flow shops with sequence-dependent setup times, job deterioration and learning effects. *Neural Comput. Appl.* **2019**, *31*, 5327–5341. [CrossRef]
6. Han, W.; Guo, F.; Su, X.C. A reinforcement learning method for a hybrid flow-shop scheduling problem. *Algorithms* **2019**, *12*, 222. [CrossRef]
7. Zhu, J.L.; Wang, H.G.; Zhang, T. A deep reinforcement learning approach to the flexible flow-shop scheduling problem with makespan minimization. In Proceedings of the 2020 IEEE 9th Data Driven Control and Learning Systems Conference, Liuzhou, China, 20–22 November 2020.
8. Fonseca-Reyna, Y.C.; Martínez-Jiménez, Y. Adapting a reinforcement learning approach for the flow shop environment with sequence-dependent setup time. *Rev. Cuba. Cienc. Informáticas* **2017**, *11*, 41–57.
9. Zhao, F.Q.; Zhang, L.X.; Cao, J.; Tang, J.X. A cooperative water wave optimization algorithm with reinforcement learning for the distributed assembly no-idle flowshop scheduling problem. *Comput. Ind. Eng.* **2021**, *153*, 107082. [CrossRef]
10. Ren, J.F.; Ye, C.M.; Yang, F. Solving flow-shop scheduling problem with a reinforcement learning algorithm that generalizes the value function with neural network. *Alex. Eng. J.* **2021**, *60*, 2787–2800. [CrossRef]
11. Li, X.X.; Tang, H.T.; Yang, Z.P.; Wu, R.; Luo, Y.B. Integrated optimization approach of hybrid flow-shop scheduling based on process set. *IEEE Access* **2020**, *8*, 223782–223796. [CrossRef]
12. Zhou, R.; Lei, D.M.; Zhou, X.M. Multi-objective energy-efficient interval scheduling in hybrid flow shop using imperialist competitive algorithm. *IEEE Access* **2019**, *7*, 85029–85041. [CrossRef]
13. Wang, S.J.; Wang, X.D.; Chu, F.; Yu, J.B. An energy-efficient two-stage hybrid flow shop scheduling problem in a glass production. *Int. J. Prod. Res.* **2020**, *58*, 2283–2314. [CrossRef]
14. Wang, Y.K.; Wang, S.L.; Li, D.; Shen, C.F.; Yang, B. An improved multi-objective whale optimization algorithm for the hybrid flow shop scheduling problem considering device dynamic reconfiguration processes. *Expert Syst. Appl.* **2021**, *174*, 114793.
15. Rathnayake, U. Migrating storms and optimal control of urban sewer networks. *Hydrology* **2015**, *2*, 230–241. [CrossRef]
16. Kong, L.; Wang, L.M.; Li, F.Y.; Wang, G.; Fu, Y.; Liu, J. A new sustainable scheduling method for hybrid flow-shop subject to the characteristics of parallel machines. *IEEE Access* **2020**, *8*, 79998–80009. [CrossRef]
17. Lin, C.C.; Liu, W.Y.; Chen, Y.H. Considering stockers in reentrant hybrid flow shop scheduling with limited buffer capacity. *Comput. Ind. Eng.* **2020**, *139*, 106154. [CrossRef]
18. Shi, L.; Guo, G.; Song, X.H. Multi-agent based dynamic scheduling optimization of the sustainable hybrid flow shop in a ubiquitous environment. *Int. J. Prod. Res.* **2021**, *59*, 576–597. [CrossRef]
19. Hasani, A.; Hosseini, S.M.H. A bi-objective flexible flow shop scheduling problem with machine-dependent processing stages: Trade-off between production costs and energy consumption. *Appl. Math. Comput.* **2020**, *386*, 125533. [CrossRef]
20. Wu, X.L.; Shen, X.L.; Cui, Q. Multi-Objective Flexible Flow Shop Scheduling Problem Considering Variable Processing Time due to Renewable Energy. *Sustainability* **2018**, *10*, 841. [CrossRef]
21. Feng, Y.; Kong, J. Multi-objective hybrid flow-shop scheduling in parallel sequential mode while considering handling time and setup time. *Appl. Sci.* **2023**, *13*, 3563. [CrossRef]
22. Gheisariha, E.; Tavana, M.; Jolai, F.; Rabiee, M. A simulation optimization model for solving flexible flow shop scheduling problems with rework and transportation. *Math. Comput. Simul.* **2021**, *180*, 152–178. [CrossRef]
23. Zhang, B.; Pan, Q.K.; Gao, L.; Meng, L.L.; Li, X.Y.; Peng, K.K. A three-stage multi-objective approach based on decomposition for an energy-efficient hybrid flow shop scheduling problem. *IEEE Trans. Syst. Man Cybern. Syst.* **2020**, *50*, 4984–4999. [CrossRef]
24. Mousavi, S.M.; Zandieh, M.; Yazdani, M. A simulated annealing/local search to minimize the makespan and total tardiness on a hybrid flowshop. *Int. J. Adv. Manuf. Technol.* **2013**, *64*, 369–388. [CrossRef]
25. Schulz, S.; Neufeld, J.S.; Buscher, U. A multi-objective iterated local search algorithm for comprehensive energy-aware hybrid flow shop scheduling. *J. Clean. Prod.* **2019**, *224*, 421–434. [CrossRef]
26. Geng, K.F.; Ye, C.M.; Dai, Z.H.; Liu, L. Bi-objective re-entrant hybrid flow shop scheduling considering energy consumption cost under time-of-use electricity tariffs. *Complexity* **2020**, *2020*, 1–17. [CrossRef]
27. Fu, Y.P.; Wang, H.F.; Wang, J.W.; Pu, X.J. Multi-objective modeling and optimization for scheduling a stochastic hybrid flow shop with maximizing processing quality and minimizing total tardiness. *IEEE Syst. J.* **2021**, *15*, 4696–4707. [CrossRef]
28. Aqil, S.; Allali, K. Two efficient nature inspired meta-heuristics solving blocking hybrid flow shop manufacturing problem. *Eng. Appl. Artif. Intell.* **2021**, *100*, 104196. [CrossRef]

29. Du, Y.; Li, J.Q.; Li, C.D.; Duan, P.Y. A reinforcement learning approach for flexible job shop scheduling problem with crane transportation and setup times. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *10*, 1–15. [CrossRef]
30. Luo, S.; Zhang, L.X.; Fan, Y.S. Dynamic multi-objective scheduling for flexible job shop by deep reinforcement learning. *Comput. Ind. Eng.* **2021**, *159*, 107489. [CrossRef]
31. Du, Y.; Li, J.Q.; Chen, X.L.; Duan, P.Y.; Pan, Q.K. Knowledge-based reinforcement learning and estimation of distribution algorithm for flexible job shop scheduling problem. *IEEE Trans. Emerg. Top. Comput. Intell.* **2023**, *7*, 1036–1050. [CrossRef]
32. Wang, H.; Cheng, J.F.; Liu, C.; Zhang, Y.Y.; Hu, S.F.; Chen, L.Y. Multi-objective reinforcement learning framework for dynamic flexible job shop scheduling problem with uncertain events. *Appl. Soft Comput.* **2022**, *131*, 109717. [CrossRef]
33. Wu, Z.F.; Fan, H.B.; Sun, Y.M.; Peng, M.Y. Efficient multi-objective optimization on dynamic flexible job shop scheduling using deep reinforcement learning approach. *Processes* **2023**, *11*, 2018. [CrossRef]