Research On Logistics Scheduling Model For Production Workshops Based On Deep Reinforcement Learning

Yan Wang, Xiufeng Cao*, Xiaosuo Luo, and Bin Zhou

School of Artificial Intelligence, Chongqing University of Education, Chongqing 400065, China

Corresponding author. E-mail: caoxf@cque.edu.cn

Received: Jun. 13, 2026; Accepted: Aug. 14, 2026

Aiming at solving difficult problems with traditional methods of logistics scheduling in production workshops, a logistics scheduling model based on deep reinforcement learning is studied. The proposed reinforcement learning method based on the Markov Decision Process fully explores the impact of various factors within the production workshop on logistics scheduling. Meanwhile, its correlation is analyzed, and the scheduling problem of the production workshop is solved through the double deep Q-network (DDQN) algorithm. Multi-source heterogeneous data such as people, machines, processes, and the environment in the manufacturing workshop will be processed and analyzed, which can extract the logical data for model training in the production workshop and guide the intelligence of the production workshop. Experimental results show that this method has better time allocation and higher efficiency, has better advantages than other algorithms, and is more adaptable to dynamic environments. Through the comparison of 100 experimental results, the probability of obtaining the optimal solution of the model by this method reaches 92working time obtained is shorter.

Keywords: Markov decision, deep reinforcement learning; production workshop; logistics scheduling; DDQN

http://dx.doi.org/10.6180/jase.202612_35.017

1. Introduction

1. introduction

The logistics scheduling problem of the production workshop is a key optimization technology to improve production efficiency, reduce logistics costs, and improve equipment utilization [1]. It is of great significance to reduce production costs and improve economic benefits by improving the satisfaction of each work program through effective scheduling optimization [2]. The scheduling problem of the production workshop is a kind of complex combinatorial problem of multi-objective optimization. Reasonable optimization is required for each process, each piece of equipment, and each workpiece, which is a typical NP problem. It needs to be optimized and solved by intelligent algorithms to achieve better results [3].

Reinforcement learning is a kind of machine learning, which obtains information through the interaction of the environment, and has a good prospect in the field of continuous decision-making. Applying deep reinforcement learning to the logistics scheduling of production workshops has important application value for improving the efficiency of production logistics [4]. Li et al. [5] realized the path planning and job scheduling of logistics vehicles through an optimized multi-objective algorithm in the scheduling of production workshops. Zhang et al. [6] also optimized the time and energy consumption of the production workshop through the optimization of the multi-objective backtracking algorithm. However, the design of this method lacks the consideration of actual factors, resulting in that the optimized result does not match the actual situation. Chen et al. [7] proposed an imitation learning algorithm to achieve algorithm optimization by learning scheduling rules. However, due to the extremely high time complexity of work-

shop scheduling and the lack of a large number of problem training, the applicability of this method is insufficient.

In recent years, deep reinforcement learning has made significant progress in the field of shop-floor scheduling. Li et al. [8] designed an intelligent scheduling method for multi-machine collaborative operations based on NSGA-III and an improved ant colony algorithm, aimed at optimizing production scheduling in smart factories. Workneh et al. [9] applied deep reinforcement learning to the adaptive flexible job-shop scheduling problem, effectively addressing the variability and uncertainty inherent in production environments. Although Tao et al. [10] did not directly address job-shop scheduling, the optimization methods they employed for green supply chain pricing strategies offer valuable methodological insights for the problem modeling in this study. Park et al. [11] combined graph neural networks with reinforcement learning; by employing graph embedding techniques to effectively capture the structural characteristics of the scheduling problem, they achieved performance superior to traditional DRL methods in job-shop scheduling. Lei et al. [12] proposed a large-scale dynamic flexible job-shop scheduling method based on hierarchical reinforcement learning, capable of effectively handling dynamic scenarios involving the stochastic arrival of new orders. Di et al. [13] employed a cooperative multi-agent reinforcement learning approach to solve the distributed hybrid flow-shop scheduling problem, demonstrating strong performance in multi-factory collaborative scenarios. Liu et al. [14] proposed an end-to-end real-time scheduling method for hybrid flow shops based on heterogeneous cooperative multi-agent deep reinforcement learning. However, existing methods still suffer from insufficient adaptability when dealing with dynamic disturbances, and most studies assume a static, deterministic scheduling environment, which deviates from actual production scenarios.

The primary limitations of existing methods can be summarized as follows: (1) traditional heuristic and mathematical programming approaches entail excessive computational complexity when addressing large-scale scheduling problems, making it difficult to obtain satisfactory solutions within a limited timeframe; (2) static scheduling methods fail to effectively handle dynamic disturbances during production, resulting in significant discrepancies between optimization results and actual execution outcomes; and (3) most existing deep reinforcement learning methods are designed for specific shop-floor layouts and lack a unified modeling framework capable of accommodating scenarios involving multiple operations, multiple pieces of equipment, and multiple constraints. To address these issues, the core objective of this research is to establish a unified logistics scheduling modeling framework that integrates Petri nets with Markov Decision Processes (MDPs) and to employ the DDQN algorithm for the autonomous optimization and learning of scheduling strategies, thereby enhancing the model's adaptability to complex, dynamic environments.

To address the optimization problem of production workshop scheduling models, this paper establishes a deep reinforcement learning-based logistics scheduling model grounded in the Markov Decision Process and employs the DDQN algorithm to enhance production efficiency within the workshop. By analyzing the various influencing factors of logistics scheduling and using strategic adjustments, the model agent can achieve the highest return expectation, and provide solutions for related problems.

2. logistics scheduling of production workshop

2.1 Logistics scheduling

From the perspective of the production workshop, the entire logistics scheduling process is actually a product state transition, which gradually transitions products from raw materials and workpieces to finished products [8]. The process of product state conversion needs to go through multiple processes, and each process involves multiple equipment and various utensils.

The logistics scheduling of the production workshop aims to improve the connection efficiency of the stations to complete tasks, and realize the improvement of product production efficiency and economic benefits through the allocation of production resources and the sequencing of processes [15]. The scheduling of workshop logistics mainly includes two aspects: production job sequence scheduling and production resource allocation scheduling. The optimization of the production sequence is realized based on a certain optimization strategy. Through the estimation of the operation time, the scheduling of resources can be implemented, and the transfer of the product status can be completed according to the change of the schedule. Based on the deep reinforcement method, it can effectively deal with the adaptive scheduling problem in practical problems, and has important research value for solving problems in this direction [9].

A solar cell is a new energy device that generates electricity by absorbing sunlight. As the main carrier of photovoltaic power generation, its main processes include the preparation of silicon single crystals, the processing of silicon wafers, the production of solar cells, and the assembly of solar cells. In particular, the production process of solar cells includes several important process steps such as tex-

turing, diffusion, etching, and screen printing, as shown in Fig. 1.

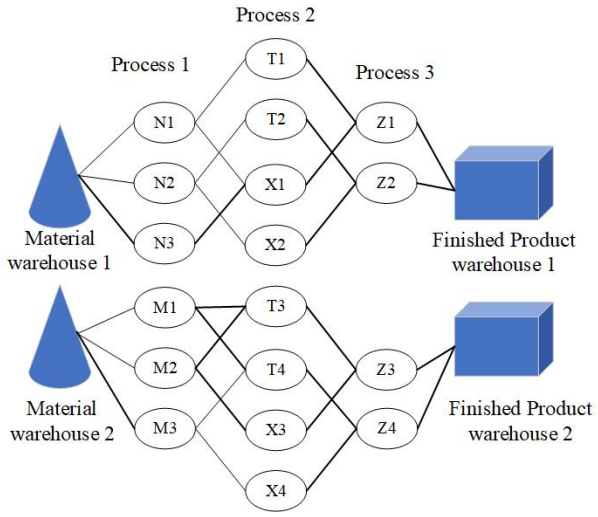

Fig. 1. Production workshop mode of solar cells

Taking a solar cell production workshop as an example, the logistics scheduling model is studied. During the production process of battery chips, the silicon wafers used as raw materials are sent from the warehouse to the production workshop, and are processed through the above-mentioned multiple processes to finally form products and finally shipped back to the warehouse. Assuming that in the above-mentioned production mode, the factory includes two workshops, and cells are produced through multiple main processes, and each workshop contains multiple parallel devices. The connection between processes is completed by transfer equipment, which can complete the transfer of processes in the workshop, and can also complete the process connection between workshops.

2.2 Constraint premise

The logistics scheduling of the solar cell production workshop requires a series of rules and constraints, as follows:

(1) Assuming that at time zero, all components to be processed are equipped and are waiting to be scheduled for processing at the start node of the workshop, and all machinery and equipment are in a normal and available state to be processed.

(2) The processing equipment of each process can only complete the processing of one group of cells at the same time. Before this process is completed, subsequent processing tasks cannot enter the product production process.

(3) The same group of cells can only be processed on one device, and the processing of the corresponding process does not involve other devices.

(4) There is a sequence in the processing procedures of the battery sheet, the procedures cannot be adjusted, and the product processing cannot be interrupted during the processing.

(5) During the processing of battery slices, the processing switching time between groups can be ignored.

Assumptions (1) and (5) mentioned above serve to simplify the model for the experimental validation presented in this paper. In actual solar cell production facilities, equipment may experience periods of unavailability due to preventive maintenance schedules, and batch transitions may involve auxiliary tasks such as cleaning and calibration. While this paper validates the effectiveness of the DDQN algorithm under idealized assumptions, future research will progressively incorporate real-world factors-such as equipment maintenance schedules and batch transition times-to further enhance the model's practical applicability.

2.3 Model construction

For the scheduling problem of the solar cell production workshop, the manufacturing petri net (MPN) mechanism can be used to complete the model construction, and the deep learning method is used to explore the optimal solution of the model [16].

The structure of the Petri net is defined as an array $PN = (P, T, I, O, M_0)$ containing five vectors, as follows:

(1) P is the quantity of the same equipment in the production workshop. Assuming that there are m devices, P can be defined as a set of devices of this type, which is $P = \{p_1, p_2, p_3, \ldots, p_m\}$.

(2) T is the trigger result of workpiece transfer between different processes. Assuming that there are n kinds of transfer results, it can be defined as $T = \{t_1, t_2, t_3, \ldots, t_n\}$, and $P \cup T \neq 0$, $P \cap T \neq \varnothing$.

(3) I represents the vector of the workpiece transferred from the warehouse to the processing workshop, and the number is N.

(4) O represents the vector of workpiece transfer from processing workshop to storage warehouse.

(5) M represents the state vector of each workpiece in the warehouse, and M_0 represents the initial state in the warehouse.

(6) $C(p_i)$ is the number of parallel equipment in the production workshop, and it can also represent the number of groups of solar cells that can be accommodated in the workshop.

(7) D_c represents the time matrix of the operation of solar cells on different process lines.

(8) W represents the set of all processing path vectors from the beginning to the end of the workpiece in the logistics scheduling production model.

(9) O_{rd} means scheduling orders in the production model.

Through the above definition, the production workshop of solar cells can be treated equivalently, and it can be regarded as a resource scheduling node. Through the processing of different batches of cells, as well as the scheduling trigger and multiple circulation of cells, the construction of the production workshop model is completed [17]. Drawing upon the Petri net approach to modeling discrete-event dynamic systems, this paper abstracts production workshop equipment states, process flows, and material transfers into Petri net places, transitions, and tokens to construct an MPN model oriented toward material flow scheduling. While the model focuses on state representation and transition logic within production logistics-sharing mathematical similarities at the level of modeling abstraction with traffic scheduling in computer networks-its application domain and optimization objectives are entirely different.

The final optimization goal of the model is to take the overall shortest overall processing time of all cells in the order as the optimization goal, which can be expressed as min T.

The logistics scheduling problem is formulated as the following optimization model.
The decision variables are:

$$\begin{cases} x_{ij}^k \in \{0,1\} \\ s_{ij} \geq 0 \\ c_{ij} \geq 0 \end{cases} \tag{1}$$

In formula Eq. (1), x_{ij}^k takes the value 1 if the i-th workpiece of operation j is processed on the k-th machine, and 0 otherwise; s_{ij} represents the start time of the i-th workpiece of operation j; and c_{ij} represents the completion time of the i-th workpiece of operation j.

Objective function: Minimize the maximum completion time, expressed as:

$$\min T = \min \left\{ \max_i c_{i,n} \right\} \tag{2}$$

Where n is the total number of process steps.
The constraints are:

(1) Process sequence constraints:

$$c_{i,j} \leq s_{i,j+1}, \quad \forall i, \forall j \tag{3}$$

(2) Equipment capacity constraints:

$$s_{ij} + p_{ij} \leq s_{i'j'} + M\left(2 - x_{ij}^k - x_{i'j'}^k\right), \quad \forall i, i', \forall j, j', \forall k \tag{4}$$

Where M is a sufficiently large positive number; when $x_{ij}^k = x_{i'j'}^k = 1$, the constraint reduces to $c_{ij} \leq s_{i'j'}$, whereas otherwise, the constraint is automatically satisfied.

(3) Processing time constraints:

$$c_{ij} = s_{ij} + p_{ij}, \quad \forall i, \forall j \tag{5}$$

Where p_{ij} is the processing time for the i-th workpiece at operation j.

(4) Uniqueness constraint on device allocation:

$$\sum_k x_{ij}^k = 1, \quad \forall i, \forall j \tag{6}$$

The aforementioned model describes the fundamental mathematical structure of solar cell production scheduling and serves as the basis for subsequent MDP modeling. Within the MDP framework, the state space S is composed of elements such as the occupancy status of each piece of equipment and the completion status of each process step. The action space A consists of scheduling decisions, such as assigning a specific workpiece at a particular process stage to a specific machine. State transition probabilities $P(s' \mid s, a)$ are determined by the firing rules of the Petri net. The reward function $R(s, a)$ is defined as the negative of the completion time, guiding the agent to optimize production objectives.

2.4 Reinforcement learning based on markov decision processes

Reinforcement learning is a type of machine learning. Through continuous interaction with the environment and accumulation of self-learning, better guidance strategies can be obtained. The experience learning of the agent can formulate an optimization strategy for it, and with the continuous exchange of the environment, the optimization effect of the strategy is getting better and better [10].

The method of reinforcement learning can carry out workshop logistics scheduling, and the model can be abstracted. Markov decision process can be used for interactive processing of data. Through the perception of the

current environment, the agent executes actions and updates the state to obtain the next environment state and form a loop.

For workshop logistics scheduling, the main environmental state of its operation is only related to the current state and has nothing to do with the historical state, so it can be considered to have Markov characteristics and meet the Markov conditions:

$$\pi\left(S_{t+1} \mid S_t, \cdots, S_0\right) = \pi\left(S_{t+1} \mid S_t\right) \tag{7}$$

In formula Eq. (7), S_t is corresponding to the t-th production state, and π is the strategy corresponding to this state.

In the training model of reinforcement learning, in order to enable the agent to obtain more positive rewards, it is necessary to adjust the strategy to maximize the expected reward. In the process of strategy evaluation, the role of the discount factor needs to be considered, and its expression is as follows.

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{8}$$

In formula Eq. (8), R_t is the reward feedback corresponding to the t-th state, r_{t+1} is the reward corresponding to the previous state, and γ is the discount factor, with a value between 0 and 1. For the mathematical expectation of the reward including the initial state, it can be expressed by the state value, and the expression is as follows:

$$V^{\pi}(s) = E_{\pi}\left[R_t \mid s_t = s\right] = E_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s\right] \tag{9}$$

In formula Eq. (9), $V^{\pi}(s)$ is the state value corresponding to the π-th strategy, and E_{π} is the mathematical expectation of the π-th strategy.

The action-state-value function can be expressed as follows.

$$\begin{aligned} Q^{\pi}(s,a) &= E_{\pi}\left[R_t \mid s_t = \mathrm{s}, a_t = a\right] \\ &= E_{\pi}\left[\gamma_{t+k+1} \mid s_t = \mathrm{s}, a_t = a\right] \end{aligned} \tag{10}$$

In formula Eq. (10), a is a specific execution action, and a_t is the execution action in the t-th state.

It can be seen from the Markov property that the above two formulas need to satisfy the functional relationship, and its formula is as follows.

$$Q^{\pi}(s,a) = E_{\pi}\left[r_{t+1} + \gamma^{V\pi}\left(s_{t+1}\right)\right] I_A \tag{11}$$

$$S^{\pi}(s) = E_{a\sim\pi(a|s)} \mid Q^{\pi}(s,a) \tag{12}$$

It can be seen that for the optimal choice of strategy $V^*(s)$, the following formula needs to be satisfied.

$$V^*(s) = \max V^{\pi}(s) \tag{13}$$

In the reinforcement learning process based on Markov Decision Processes, emphasis is placed on evaluating the operational status of each node, including factors such as job queue lengths, equipment processing times, and personnel skill proficiency. The scheduling decision of the generation process has nothing to do with the historical state, but only depends on the current production logic elements. Information such as station status can be described by the node dimension, which is mainly used to describe information in each production link, such as a series of information such as worker proficiency, equipment status, material processing status, and operating environment. The station status information can be described by the node dimension, which is mainly used to describe a series of information in each production link, such as worker proficiency, equipment status, material processing status, operating environment, etc.

2.5 DDQN algorithm

DQN (deep Q-network) is a deep neural network algorithm based on the Q-learning algorithm. Through the prediction of the Q value, the expected return judgment of the agent's action is carried out. Q value can be understood as the value of state action in logistics scheduling model. Due to the high error of the network in the DQN algorithm, a large number of overestimated actions are selected multiple times, which makes the accuracy of the algorithm insufficient [18].

The goal of the DQN algorithm is to make the trained neural network good enough. The input of the entire network is the state of the agent in the current year, and the output result is the Q value with n optional actions, and the output of the network is an n-dimensional vector. When the agent chooses an action, it will select the action with the largest Q value in the output result through the input of its own state, so that the agent enters a new state and updates it sequentially until all tasks are completed. The target selection of the DQN algorithm can be expressed by the formula as follows.

$$\begin{aligned} Q^*(s,a) = \max E\Big[&r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots \\ &\mid s_t = s, a_t = a, \pi\Big] \end{aligned} \tag{14}$$

DDQN is the abbreviation of Double DQN, based on DQN deep neural network. Its structural model is consistent with the DQN network, but there are two Q networks in the structure, which can effectively eliminate the overestimation of the DQN deep neural network algorithm, and its calculation accuracy is greatly improved.

The main difference between the DDQN algorithm and the DQN algorithm is that the objective functions of the two are not the same, which can be expressed by the following formulas.

$$Y_t^{DQN} = R_{t+1} + \gamma \max Q\left(S_{t+1}, a_t, \theta_t^-\right) \tag{15}$$

$$Y_t^{DDQN} = R_{t+1} + \gamma Q\left(S_{t+1}, \operatorname{argmax} Q\left(S_{t+1}, a_t, \theta_t\right), \theta_t^-\right) \tag{16}$$

The difference between the objective functions of the two algorithms is mainly that the optimal action selection of DQN comes from the parameter θ_t^- of the target Q network before updating. However, the optimal action selection of DDQN will be based on the parameter θ_t^- of the updated target Q network, so that the action selection is more reasonable, and the obtained Q value is closer to the real value.

3. experimental results and analysis

3.1 Experimental conditions

To verify the validity of the model in this paper, the production process of solar cells is used for verification. The experimental data set for verification is that: (1) The production workshop of solar cells includes 5 important processes, such as texturing, diffusion, etching, screen printing, and PECVD. Each process includes multi-step scheduling steps, and the number of machines and equipment contained in each step is 3 . (2) During the material handling process, the work of placing flower baskets and collecting flower baskets is carried out simultaneously. (3) In the process of material transportation, batch transportation is used as the unit. (4) The processing time of each process for a batch of materials is 192s, 325s, 203s, 183s and 205s respectively.

To evaluate the effectiveness of the DDQN algorithm in production workshop logistics scheduling, this study employs Python and neural network algorithms to conduct data analysis and experimental validation on a Markov Petri net model. The algorithm was constructed by taking into account the practical conditions of solar cell production. The input layer dimension was set to 10, the output layer dimension to 200, and the architecture included nine hidden layers, each containing varying numbers of neurons for data connectivity. Parameters such as basic settings, weights, and bias coefficients were defined for the experiment. The model was trained for 200,000 iterations, the discount factor for the target Q-value was set to 0.85, and the sample size was 50,000.

According to the principles of a reinforcement learning model grounded in the Markov decision process, TensorFlow is used to build the perception layer model of the system. The number of neurons in the input layer is 20, and the number of neurons in the output layer is 220. The data samples are randomly generated, and the generated data is input into the experience pool of the model, so it can facilitate the system's judgment of the state and the acquisition of rewards. When all the information is in the terminal state, it is considered that the system model has been trained and saved.

To ensure a fair comparison, the computational resource budgets for the algorithms are uniformly set as follows:

Genetic Algorithm (GA): population size of 100 , crossover probability of 0.85 , mutation probability of 0.05 , maximum of 500 iterations, and a total computation time limit of 45 minutes;

Ant Colony Optimization (ACO): 50 ants, pheromone evaporation coefficient of 0.1, $\alpha = 1, \beta = 2$, maximum of 500 iterations, and a total computation time limit of 45 minutes;

DDQN Algorithm: 200,000 total training steps, batch size of 64, experience replay buffer capacity of 50,000 , learning rate of 0.001 , target network update frequency of 1,000 steps, and a total training time of 42 minutes and 46 seconds.

All algorithms were executed on identical hardware (Intel Core i7-10700 CPU, NVIDIA GTX 1660 GPU, 32GB RAM) to guarantee a fair comparison. Although the DDQN requires a lengthy offline training period, its online inference time is extremely short (<0.002s per step) once training is complete, making it well-suited for real-time scheduling requirements in actual workshops-a key advantage of deep reinforcement learning methods.

3.2 Experimental results

3.2.1 The loss function of the model

Considering that the complexity of workshop logistics scheduling, a large amount of data training is required to improve the efficiency of the model. The number of model training is 200,000 times, and every 1000 times is regarded as a training cycle. The reward mechanism in this model is set as follows.

$$r = -\frac{T_{\text{completion}}}{T_{\max}} \tag{17}$$

In formula Eq. (17), $T_{\text{completion}}$ represents the completion time corresponding to the current scheduling scheme - specifically, the maximum time span required to complete the processing of all jobs-while $T_{\max}$ is the preset baseline for the maximum completion time (set to 500 time units in this study). Since the completion times of all valid scheduling schemes in this experiment do not exceed $T_{\max} = 500$, the ratio of these two values normalizes the reward to the range of [-1, 0]. The design philosophy behind this reward function is to use the negative normalized completion time as an immediate reward, thereby guiding the agent to make decisions at each step that tend to reduce the overall completion time.

The relationship between the model loss function and the number of training times is shown in Fig. 2.

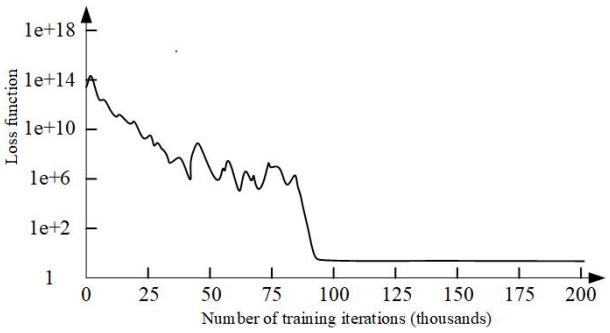

Fig. 2. The loss function curve of the model

As shown in Fig. 2, the model's loss function gradually decreases as training progresses (with the horizontal axis representing training iteration steps, where each step corresponds to an episode) and converges at approximately 95,000 steps, which can demonstrate the model's effectiveness. The completion time of the entire logistics scheduling also reaches stability after the model converges, the burden of the oscillation amplitude of the model is reduced, and the training effect is more obvious. The convergence curve of the model is gradually flat, which proves that the DDQN algorithm used in this paper has certain scientific and rationality in dealing with related problems.

To investigate the impact of key hyperparameters on model performance, this paper conducted comparative experiments by adjusting the learning rate ($l_r \in \{0.0005, 0.001, 0.005\}$) and the discount factor ($\gamma \in \{0.8, 0.85, 0.9\}$) while keeping other parameters constant. The results indicate that a learning rate of 0.001 yields a moderate convergence rate and the lowest final loss. A discount factor of 0.85 achieves the optimal balance between short-term rewards and long-term returns. An excessively high learning rate (0.005) exacerbates training oscillations, whereas an excessively low one (0.0005) leads to slow convergence. An excessively high discount factor (0.9) causes the agent to overemphasize long-term gains at the expense of immediate scheduling efficiency, while an excessively low one (0.8) can result in a limited planning horizon and a tendency to get trapped in local optima.

The specific data of the experimental results are shown in Table 1.

Table 1. Experimental result data

Index	Results
training time	42min46s
Training times	200,000 times
Completed time	125~236 time units
Response time per step	<0.002s

From the results in Table 1, it can be seen that when the number of training times of the model reaches 200,000, the scheduling time of the entire model tends to be stable. The total training time of the model is 42min 46s, and the average training time of each time is less than 0.01s, which proves that the model has extremely high learning efficiency. The test completion time of the model lies within 125~236-time units, and finally stabilizes to 138-time units. The average response step size of each step is 0.002s. It is proved that the model can give the corresponding scheduling guidance in a very short time, and it has a good effect of fast learning and scheduling guidance for the scheduling problem in the workshop.

3.2.2 Comparison of model algorithms

To compare the optimization effect of the algorithm and the stability of the calculation results, the reinforcement learning algorithm of the model in this paper is compared with the genetic algorithm and ant colony algorithm in the literature [19] for the scheduling problem of the solar cell production workshop. Under the same experimental conditions, the algorithm is used to conduct 100 experiments to compare the optimal scheduling results obtained by the algorithm. The experimental results are shown in Table 2.

It can be seen from the results in Table 2 that under the same experimental conditions, 100 experiments are performed respectively. In the end, the number of times that the ant colony algorithm reaches the optimal solution is 78 times, and the number of times that the genetic algorithm reaches the optimal solution is 36 times, which are far lower than the 92 times of the DDQN algorithm in this paper. In comparison, the optimization success rate of the proposed algorithm exceeds that of the genetic algorithm and the ant colony algorithm by 56 and 14 percentage points, respectively. Furthermore, the standard deviation of the proposed

Table 2. Comparison of 100 test results of different algorithms

Algorithms	Mean value	Optimal value	Get the optimal number of times
Genetic algorithm	925.1 ± 12.7	925	36
Ant Colony Algorithm	978.6 ± 8.9	974	78
DDQN(ours)	926.5 ± 3.2	926	92

Note: Data in the table are presented as "mean ± standard deviation." The Wilcoxon signed-rank test was used to analyze the significance of differences between the algorithms; at a significance level of $\alpha = 0.05$, the performance differences between the proposed algorithm and both the genetic algorithm ($p = 0.009$) and the ant colony algorithm ($p = 0.013$) were statistically significant.

algorithm (3.2) is significantly lower than those of the genetic algorithm (12.7) and the ant colony algorithm (8.9), which can indicate reduced performance fluctuation and greater robustness across multiple experiments. These results demonstrate that the proposed algorithm achieves optimal solutions with a higher probability and exhibits more stable and reliable performance when solving the scheduling model.

3.2.3 Comparison with other deep reinforcement learning algorithms

To further validate the effectiveness of the DDQN algorithm, this paper presents additional comparative experiments against three mainstream deep reinforcement learning algorithms: DQN, PPO (Proximal Policy Optimization), and SAC (Soft Actor-Critic). All algorithms were trained within the same MDP framework for 200,000 iterations, with each algorithm executed 100 times independently; the experimental results are shown in Table 3.

The results in Table 3 indicate that DDQN outperforms DQN (156.3 ± 8.7), PPO (148.7 ± 7.2), and SAC (142.5 ± 6.5) in terms of both average completion time (128.8 ± 3.2) and optimal completion time (126). DDQN's advantage over DQN stems from the double Q-network structure's effectiveness in mitigating overestimation bias; compared to PPO and SAC, DDQN's value function decomposition aligns better with the discrete scheduling action space, which can resulte in superior scheduling performance. Although SAC performs excellently in continuous action spaces, DDQN demonstrates better suitability for the discrete scheduling decision scenario in this study. Regarding training time, DDQN (42 minutes 46 seconds) outperforms PPO and SAC; while it takes slightly longer than DQN, the improvement in solution quality achieved by DDQN more than compensates for this minor difference in training cost. Notably, Workneh et al. [9] proposed the TDQN (Triple DQN) method, achieving performance superior to that of DDQN; this offers a valuable direction for future research. In this paper, DDQN is adopted as the baseline method, while future work will explore the application and adaptation of superior algorithms such as TDQN.

3.2.4 Comparison of other heuristic rules

The reinforcement learning method for the production scheduling problem of workshop logistics can also use the heuristic algorithm to design scheduling rules. Meanwhile, according to different algorithm results, the workpiece processing procedures are adjusted and optimized to realize the scheduling of the physical workshop. However, the effects of different scheduling methods are quite different. The reinforcement learning method in this paper is compared with other scheduling algorithms. Comparing the policy indicators such as the response time of the algorithm, the rules of different heuristic algorithms are different, and the rule descriptions are shown in Table 4.

Based on the case of this paper, the calculation results of the above algorithm and the DDQN algorithm in this paper are compared horizontally, and the results are shown in Fig. 3.

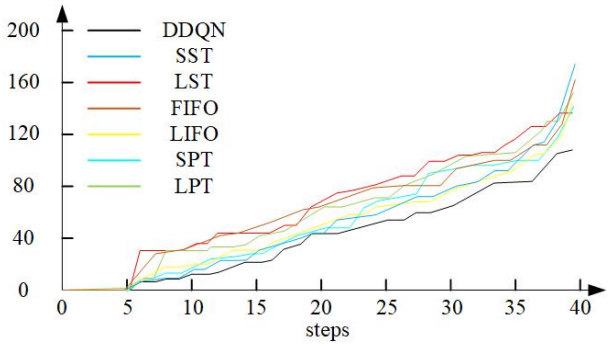

Fig. 3. Comparison of the results of different reinforcement algorithms

From the results in Fig. 3, it can be seen that the trend in the scheduling process will be quite different with different strengthening algorithms. Each algorithm adopts different strategies in process selection, which also leads to obvious differences in processing time in the scheduling process. In terms of the trend of various algorithms, as the number of processing steps increases, the processing time will gradually increase. However, from the point of view of the overall steps, the processing time of DDQN algorithm is less than other algorithms. It proves that the scheduling effect of the algorithm proposed in this paper is

Table 3. Performance comparison of different deep reinforcement learning algorithms.

Algorithms	Average completion time (s)	Optimal completion time (s)	Training time
DQN	156.3 ± 8.7	142	38min12s
PPO	148.7 ± 7.2	135	51min05s
SAC	142.5 ± 6.5	132	58min34s
DDQN(Ours)	128.8 ± 3.2	126	42min46s

Table 4. Rules of different heuristic algorithms

Algorithms	Rules
SPT	Optimizing the process with the shortest processing time
LPT	Optimizing the process with the longest processing time
SST	Optimizing the process with the shortest set-up time
LST	Optimizing the process with the longest preparation time
FIFO	applying the optimization principle of first come first processing
LIFO	applying the optimization principle of last come first processing

higher, and the result is more in line with the requirements of production operation.

Considering that the stability of the algorithm, the above algorithm is used to do a repeatability test for the experiment, and the model response performance after 100 test experiments is compared. The results are shown in Table 5.

From the results in Table 5, it can be seen that after the multi-step scheduling policy response, the scheduling time of various algorithms will be quite different. The average completion time of the six heuristic algorithms is about 200-time units, the average response time of each step is about 0.03˜0.05s, and the average total response time is about 1.4˜1.8s. However, the average completion time of the DDQN algorithm in this paper is less than 130-time units, the average response time of each step is about 0.015s, and the total response time is 0.65s, which is far lower than the other six heuristic algorithms.

Therefore, it can be further explained that the DDQN algorithm has better responsiveness and stronger adaptability to the production workshop. It can effectively meet the scheduling requirements of logistics workshops and has obvious performance advantages.

4. conclusions

To address the logistics scheduling problem in production workshops, this paper employs a deep reinforcement learning method based on the Markov Decision Process. By abstracting key workshop factors, the production logic and operational mechanisms are clarified, thereby enhancing the realism and accuracy of the model. A decision-making model is constructed through the analysis of workshop data, with the training dataset formed by correlating multi-dimensional state and behavioral data regarding the workshop's various production elements. Finally, the DDQN algorithm is utilized to schedule intelligent production within a solar cell manufacturing workshop.

Experimental results demonstrate that the logistics scheduling model constructed using a deep reinforcement learning approach based on the Markov Decision Process effectively enhances production efficiency in the workshop. A comparison of 100 experimental trials reveals that the proposed method achieves the optimal solution with a probability of 92%, significantly outperforming the Genetic Algorithm (36%) and the Ant Colony Algorithm (78%). Furthermore, when compared with other reinforcement learning methods, the proposed approach yields the shortest average processing time; the average response time per step is approximately 0.015 seconds, and the total response time is 0.65 seconds-figures substantially lower than those of the other six heuristic algorithms.

Although the method proposed in this paper has achieved promising experimental results, it still has certain limitations that warrant further exploration in future research. The model is designed for a specific solar cell manufacturing process; consequently, if significant changes occur in the workshop layout, equipment configuration, or process flow, the existing definitions of the MDP state and action spaces will no longer be applicable, necessitating the reconstruction and retraining of the model. Future research could investigate more generalized state representation methods to enable the model to adapt to workshop layouts of varying scales and topological structures.

Acknowledgements

The work is supported by Natural Science Foundation of Chongqing of China (Grant Nos. CSTB2025NSCQ-GPX0114, CSTB2025NSCQ-GPX0113), Science and Technology Research Project of the Chongqing Municipal Edu-

Table 5. Comparison of test results of 100 different algorithms

Algorithms	Average completion time (s)	Average response time per step (s)	Total response time (s)
SPT	226.4 ± 15.3	0.042 ± 0.008	1.78 ± 0.12
LPT	198.1 ± 13.8	0.044 ± 0.007	1.80 ± 0.11
SST	210.8 ± 14.5	0.038 ± 0.006	1.45 ± 0.10
LST	196.0 ± 12.9	0.037 ± 0.005	1.41 ± 0.09
FIFO	208.5 ± 14.1	0.036 ± 0.006	1.45 ± 0.10
LIFO	190.3 ± 13.2	0.042 ± 0.007	1.62 ± 0.11
DDQN(ours)	128.8 ± 3.2	0.015 ± 0.003	0.65 ± 0.05

cation Commission of China (Grant No. KJQN202301620), Scientific Research Foundation of Chongqing University of Education (Grant No. 2023BSRC018), Project Supported by Scientific and Technological Research Program of Chongqing Municipal (Grant No. XJ2026002802), and Key Project on Higher Education Teaching Reform Research and Practice in Henan Province in 2026 (Grant No. 2026SJGLX184).

References

[1] J. Fu, B. Yang, Z. Chang, Y. Zhang, J. Wang, X. Wang, and L. Wang, (2025) *"Integrated Production–Logistics Scheduling in Flexible Assembly Shops Using an Improved Genetic Algorithm"* **Machines *13***(12): 1090. DOI: 10.3390/machines13121090.

[2] Y. Fu, M. Zhou, X. Guo, L. Qi, K. Gao, and A. Albeshri, (2024) *"Multiobjective scheduling of energy-efficient stochastic hybrid open shop with brain storm optimization and simulation evaluation"* **IEEE Transactions on Systems, Man, and Cybernetics: Systems *54***(7): 4260–4272. DOI: 10.1109/TSMC.2024.3376292.

[3] L. Sun, W. Shi, J. Wang, H. Mao, J. Tu, and L. Wang, (2023) *"Research on production scheduling technology in knitting workshop based on improved genetic algorithm"* **Applied Sciences *13***(9): 5701. DOI: 10.3390/app13095701.

[4] L. Cai, W. Li, Y. Luo, and L. He, (2023) *"Real-time scheduling simulation optimisation of job shop in a production-logistics collaborative environment"* **International Journal of Production Research *61***(5): 1373–1393. DOI: 10.1080/00207543.2021.2023777.

[5] R. Li, J. Mao, X. Wu, W. Zhou, C. Qian, and H. Du, (2026) *"A new framework for job shop integrated scheduling and vehicle path planning problem"* **Sensors *26***(2): 543. DOI: 10.3390/s26020543.

[6] Z. Zhang, X. He, Y. Man, and Z. He, (2023) *"Multi-objective scheduling in dynamic of household paper workshop considering energy consumption in production process"* **Journal of Smart Environments and Green Computing *3***(3): 87–105. DOI: 10.20517/jsegc.2023.05.

[7] J.-F. Chen, L. Wang, H. Ren, J. Pan, S. Wang, J. Zheng, and X. Wang, (2022) *"An imitation learning-enhanced iterated matching algorithm for on-demand food delivery"* **IEEE Transactions on Intelligent Transportation Systems *23***(10): 18603–18619. DOI: 10.1109/TITS.2022.3163263.

[8] S. Li, M. Zhang, N. Wang, R. Cao, Z. Zhang, Y. Ji, H. Li, and H. Wang, (2023) *"Intelligent scheduling method for multi-machine cooperative operation based on NSGA-III and improved ant colony algorithm"* **Computers and Electronics in Agriculture *204***: 107532. DOI: 10.1016/j.compag.2022.107532.

[9] A. D. Workneh, M. El Mouhtadi, and A. El Hilali Alaoui, (2024) *"Deep reinforcement learning for adaptive flexible job shop scheduling: coping with variability and uncertainty"* **Smart Science *12***(2): 387–405. DOI: 10.1080/23080477.2024.2345951.

[10] Z.-j. Tao, X.-p. Chen, and P.-H. Koo, (2025) *"Optimizing pricing strategies in cross-border green supply chains with revenue sharing and flexible exchange rate mechanisms: Z. Tao et al."* **Operational Research *25***(4): 101. DOI: 10.1007/s12351-025-00975-5.

[11] J. Park, J. Chun, S. H. Kim, Y. Kim, and J. Park, (2021) *"Learning to schedule job-shop problems: representation and policy learning using graph neural network and reinforcement learning"* **International journal of production research *59***(11): 3360–3377. DOI: 10.1080/00207543.2020.1870013.

[12] K. Lei, P. Guo, Y. Wang, J. Zhang, X. Meng, and L. Qian, (2023) *"Large-scale dynamic scheduling for flexible job-shop with random arrivals of new jobs by hierarchical reinforcement learning"* **IEEE Transactions on Industrial Informatics *20***(1): 1007–1018. DOI: 10.1109/TII.2023.3272661.

[13] Y. Di, L. Deng, and L. Zhang, (2024) *"A collaborative-learning multi-agent reinforcement learning method for distributed hybrid flow shop scheduling problem"* **Swarm and Evolutionary Computation *91***: 101764. DOI: 10.1016/j.swevo.2024.101764.

[14] S. Liu, H. Zhu, L. Shen, Y. Li, and J. Zhang, (2026) *"The end-to-end real-time scheduling method for hybrid flow shop based on heterogeneous cooperative multi-agent deep reinforcement learning"* **Journal of Manufacturing Systems *85***: 487–512. DOI: 10.1016/j.jmsy.2026.02.011.

[15] Z. Wu and M. Tu. "Workshop Logistics Planning for Small and Medium Manufacturing Enterprises Based on SLP Method—with a Furniture Manufacturing Factory as an Example". In: *The International Conference on Artificial Intelligence and Logistics Engineering*. Springer. 2024, 290–300. DOI: 10.1007/978-3-031-72017-8_27.

[16] Z. He, N. Li, N. Ran, and L. Li, (2026) *"Scheduling of flexible manufacturing systems based on place-timed petri nets and basis reachability graphs"* **IEEE Transactions on Control Systems Technology**: DOI: 10.1109/TCST.2026.3664466.

[17] D. Yang, Z. Cheng, W. Zhang, H. Zhang, and X. Shen, (2023) *"Burst-aware time-triggered flow scheduling with enhanced multi-CQF in time-sensitive networks"* **IEEE/ACM transactions on networking *31***(6): 2809–2824. DOI: 10.1109/TNET.2023.3264583.

[18] A. Ly, R. Dazeley, P. Vamplew, F. Cruz, and S. Aryal, (2024) *"Elastic step DQN: A novel multi-step algorithm to alleviate overestimation in Deep Q-Networks"* **Neurocomputing *576***: 127170. DOI: 10.1016/j.neucom.2023.127170.

[19] M. F. Uslu, S. Uslu, and F. Bulut, (2022) *"An adaptive hybrid approach: Combining genetic algorithm and ant colony optimization for integrated process planning and scheduling"* **Applied Computing and Informatics *18***(1-2): 101–112. DOI: 10.1016/j.aci.2018.12.002.