APEM journal
Advances in Production Engineering & Management
Volume 18 | Number 2 | June 2023 | pp 137–151
https://doi.org/10.14743/apem2023.2.462
ISSN 1854-6250
Journal home: apem-journal.org
Original scientific paper

Real-time scheduling for dynamic workshops with random new job insertions by using deep reinforcement learning

Sun, Z.Y.[a,b], **Han, W.M.**[a,*], **Gao, L.L.**[a]

[a]School of Economics and Management, Jiangsu University of Science and Technology, Zhenjiang, Jiangsu, P.R. China
[b]School of Software, Pingdingshan University, Pingdingshan, Henan, P.R. China

ABSTRACT

Dynamic real-time workshop scheduling on job arrival is critical for effective production. This study proposed a dynamic shop scheduling method integrating deep reinforcement learning and convolutional neural network (CNN). In this method, the spatial pyramid pooling layer was added to the CNN to achieve effective dynamic scheduling. A five-channel, two-dimensional matrix that expressed the state characteristics of the production system was used to capture the state of the real-time production of the workshop. Adaptive scheduling was achieved by using a reward function that corresponds to the minimum total tardiness, and the common production dispatching rules were used as the action space. The experimental results revealed that the proposed algorithm achieved superior optimization capabilities with lower time cost than that of the genetic algorithm and could adaptively select appropriate dispatching rules based on the state features of the production system.

ARTICLE INFO

Keywords:
Real-time scheduling;
Machine learning;
Deep reinforcement learning (DRL);
Spatial pyramid pooling layer;
Artificial neural networks (ANN);
Convolutional neural networks (CNN)

**Corresponding author:*
wlmh63@163.com
(Han, W.M.)

Article history:
Received 7 December 2022
Revised 6 June 2023
Accepted 25 June 2023

1. Introduction

With the rapid development of information technology, China is gradually entering intelligent industry 4.0 [1]. With changing market demand, multiple-product small-batch order-type production has become prevalent in the manufacturing industry. Therefore, achieving sustainable and efficient workshop production under complex production conditions, responding quickly to market changes, and satisfying the diverse needs of customers are critical.

Although job shop scheduling problems (JSSP) [2, 3] primarily address static scheduling issues, many real-time disruption factors, such as equipment failure, dynamic order arrival, emergency order insertion, in the production process are ignored [4, 5]. The insertion of a new order can drastically change the number and mode of processing tasks. Error accumulation renders existing production scheduling schemes ineffective, which results in the failure of the production planning system [6]. Therefore, dynamic real-time production scheduling on the insertion of new

jobs is crucial for timely response to disruption events and ensuring production requirements are satisfied.

Current dynamic scheduling methods under order disturbance include heuristic algorithms [7-11] and dispatching rules [12]. Wang *et al*. [13] proposed an improved particle swarm optimization (PSO) algorithm to solve the dynamic job shop scheduling problem. Caldeira *et al*. [14] solved the flexible job shop scheduling problem on the arrival of a new job by using the improved backtracking search optimization algorithm that minimized makespan, energy consumption, and system stability. Ghaleb *et al*. [15] proposed three heuristic algorithms to address the real-time scheduling problem when new jobs are added and equipment fails. Tang *et al*. [16] considered minimum energy consumption and makespan as optimization objectives and proposed an improved PSO algorithm to solve the dynamic scheduling problem of flexible flow shops under new job arrival and equipment failure. In most heuristic algorithms, the dynamic scheduling problem is converted into a multi-stage static problem. Short-sightedness appears as the disturbance scale increases. The dispatching rules method can immediately respond to dynamic disturbance events and exhibits short computing time and high solution efficiency.

Hundreds of dispatching rules have been proposed for shop scheduling [17, 18]. Zhang *et al*. [19] proposed a job shop dispatching rule selection system based on semantics to achieve adaptive selection of dispatching rules by scheduling objectives. To reduce the time of job completion and complexity of process design in the conventional dispatching rule design process, Zhang *et al*. [20] proposed an improved genetic programming algorithm that evolves effective dispatching rules automatically. Ferreira *et al*. [21] combined machine learning with the problem domain reasoning to generate effective dispatching rules. Although dispatching rules can respond to dynamic disturbance events in real time and exhibit short computing time and high solving efficiency, these methods are prone to local optimum and cannot be adjusted adaptively to respond to various production states.

Reinforcement learning (RL) [22] has been widely used in production scheduling because of its excellent optimization ability and high computational speed. The continuous interaction between agent and environment maximizes cumulative rewards [23]. Dispatching rules can be dynamically and flexibly selected based on the real-time production status, which is suitable for the dynamic production scheduling problems. Wang *et al*. [24] used Q-learning to train a single machine agent and realized the dynamic selection of the three basic dispatching rules with the minimum average tardiness as the optimization objective. Chen *et al*. [25] proposed a rule-driven method for generating high-quality composite dispatching rules for the multi-objective dynamic job shop scheduling problem by using the Q-learning algorithm. Qu *et al*. [26] proposed a Q-learning algorithm that solves the dynamic job shop scheduling problem under random job arrival and equipment failure by combining the neighborhood search algorithm with the Q-factor. Although the conventional reinforcement learning algorithm has achieved excellent results in solving dynamic production scheduling problems, the algorithm is limited to situations in which the dimension and scale of the system state space are small and discrete. In the deep reinforcement learning (DRL) [27] algorithm, the perception ability of deep learning is effectively combined with the decision-making ability of reinforcement learning. Thus, DRL can effectively performs complex decision-making in the high-dimensional state space. Zhu *et al*. [28] proposed a proximal policy optimization (PPO) algorithm to solve the flexible flow shop scheduling problem by minimizing makespan. The PPO algorithm outperformed the conventional heuristic algorithm in terms of the quality of the solution. Luo *et al*. [29] proposed a deep Q-network (DQN) algorithm to solve the real-time workshop scheduling problem with dynamic job arrival to minimize total tardiness and achieved excellent results in the randomly generated data experiment. Yang *et al*. [30] used the A2C algorithm to train an intelligent model for the permutation flow job shop scheduling problem. This model outperformed the conventional heuristic algorithm in terms of the solving time and solution quality. Li *et al*. [31] proposed a hybrid DQN (HDQN) algorithm to solve the dynamic flexible job shop scheduling problem under transportation resource constraints. In most studies, the production system state is expressed through numerical features, which requires special manual design.

The CNN is used for the feature extraction of system state features expressed by multi-channel images to effectively reduce manual design difficulty and exhibits excellent generality. However, because of the limitation of CNN feature extraction on the input size of training images, static scheduling is widely used. Liu *et al*. [32] designed three channels of two-dimensional data matrices as system state features for job shop scheduling problems and solved these problems using the AC algorithm to achieve excellent results on benchmark examples. Han *et al*. [33] combined the CNN and DRL to achieve dynamic job shop scheduling. Wang *et al*. [34] used the PPO algorithm to solve the job shop scheduling problem outperform GA. Subsequently, random fine-tuning was performed on some examples to test the generalization ability of the model.

This method is simple and produces excellent results by describing the state features of the production system using multi-channel images. However, because of the structural characteristics of the CNN, applying the method to dynamic variable data of various image sizes is difficult. The static model can only process data of the same size, and it exhibits limited generalization. Multi-channel image system feature representation is yet to be used for dynamic scheduling. Therefore, in this study, an SPP layer [35] was added to the last layer of the CNN so that the neural network can handle any size of the input image information and achieve dynamic scheduling under the state image expression mode of the production system.

The contributions of this paper are as follows: (1) This study is the first to use SPP with neural networks to solve the dynamic scheduling problem to allow the system model to handle the input state data of any scale; (2) The state feature expression mode of the production system was improved. The limitations of the conventional three-channel image design was overcome, and the system state feature was represented by five-channel data, considering both the global and local features of system processing state; (3) To assess the effect of the dispatching rules selected at each decision point on the overall scheduling objective, a novel reward function targeting total tardiness was developed; (4) Comprehensive parameter sensitivity experiments and algorithm result comparisons were performed. The effectiveness of the calculation speed and optimization ability of the proposed scheme was demonstrated. The method achieved excellent generalization results.

2. Overall scheduling framework

A dueling double DQN (DDDQN) algorithm framework was proposed to achieve dynamic job shop real-time scheduling with constant random new order insertion. The neural network of the DDDQN algorithm consists of a Q-network and a target network. Each network exhibits the same structure and is composed of the convolution layer and a full connection layer. To address the problem of the dynamic image size change caused by the dynamic order arrival, an SPP layer was added between the CNN convolution layer and full connection layer to ensure consistent output size of the CNN convolution layer. Thus, the scheduling problem was transformed into a multi-stage decision-making process by designing a state feature, action space, and reward function. The agent is trained through interaction with the environment, and the trained agent is applied to solve the online problem. The framework includes two parts, namely offline training and online application (Fig. 1).

The scheduling environment comprises equipment and order in the production system, which is used for the interaction with the agent and providing the current production system status information. The agent outputs the most appropriate dispatching rule and selects the highest priority operation for processing. The production system then enters the next state.

In the offline training phase, intermediate data generated in the learning process is stored in the replay memory, and a minibatch number of sample data are randomly sampled for training. The Q-network and the target network calculate the Q and target values of the system state, respectively. Q-network parameters are updated by the loss function calculated by the target value and Q value. The parameters of the target network are copied from the Q-network after a certain number of steps. The optimal action is selected according to the result of the Q value.

Although offline scheduling requires considerable time to train an agent, when an agent learns good policy, it can be widely used in online actual data scheduling to obtain optimal

scheduling results rapidly. The execution process only requires the Q-network to calculate the optimal Q value without updating various network parameters, calculating the reward value, or storing sample data and other operations.

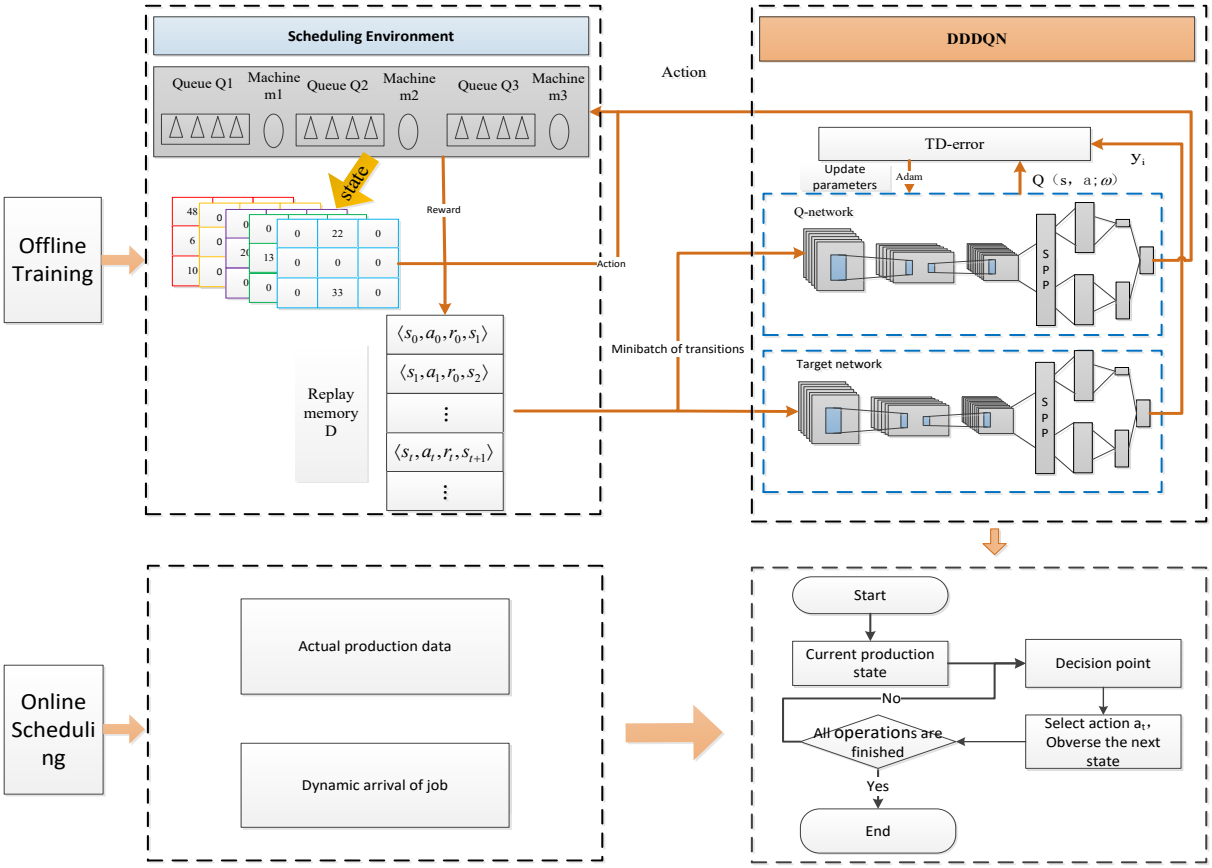


Fig. 1 Scheduling framework with the DDDQN

3. DRL for scheduling

3.1 Problem formulation

The JSPP with new order insertions can be described as follows: a processing system has N orders that are processed on M machines. Each order has n_j operations. The objective of production scheduling is to generate an optimal scheduling scheme based on the scheduling objectives and satisfy all constraints. However, when the new order arrives, the operations that were not started in the original scheduling scheme should be combined with operations in the new order for rescheduling. When creating the new scheduling scheme, factors such as the starting processing times and the number of the remaining operations of each order, should be considered. The scheduling problem should satisfy the following assumptions: (1) each order has a sequence constraint on the operations, that is, the next operation can only be processed after the previous operation is completed; (2) each operation of each order can only be processed by one machine; (3) each machine can only perform one operation at the same time; (4) when the new order arrives, the ongoing operation cannot be interrupted; (5) the processing time of each operation on the corresponding machine is known.

3.2 DQN principle

The DQN algorithm is used to solve the problem. The DQN is the most classical algorithms of DRL. Based on Q-learning, the deep neural network is used to represent value function f. The input of the neural network is the current state s, and the output is the state value func-

tion Q(s, a). In the DQN, empirical data are used to train the neural network, which is prone to instability and convergence difficulties. To solve these problems, replay memory and target network are used in the DQN. Experience replay stores the intermediate data in a fixed-size storage experience pool in the form of $\langle s_t, a_t, r_t, s_{t+1} \rangle$, which is generated in the learning process of Q-learning. The system randomly samples a certain number of small-batch samples from the replay memory for training. This random sampling not only breaks the correlation of training samples but also ensures an independent and homogeneous distribution of training samples. Target network reconstructs a network with the same structure as the original network. The original network is the Q-network, and the generated network is the target network. During training, only Q-network parameters are updated, and the parameters of the target network remain unchanged temporarily. After reaching a certain number of update steps C, the parameters of the Q-network are copied to the target network so that the value of the target network does not change in a certain update step to ensure system stability. The target value is calculated as follows:

$$Y^{DQN} \equiv r_{t+1} + \gamma \max_a \hat{Q}(s_{t+1}, a; \omega_t^-) \tag{1}$$

The neural network calculates TD errors through the target network and Q-network to update parameters.

In the standard DQN algorithm, the action with the largest Q value is selected, which results in an overestimation of the Q value. Therefore, the Double DQN (DDQN) algorithm is used to separate the action selected from the calculation of the Q value and two value functions are used. The target value of the DDQN is calculated as follows:

$$Y^{DoubleDQN} \equiv r_{t+1} + \gamma \hat{Q}(s_{t+1}, arg \max_a Q(s_{t+1}, a; \omega_t), \omega_t^-) \tag{2}$$

By using various value functions to decouple the target action and Q value, the DDQN algorithm mitigates the overestimation of Q values and achieves excellent stability.

In the DQN, the network outputs the Q value of action, but in practice, the Q value is associated with the action and state. Therefore, Dueling DQN improves the network structure of the DQN by adding state value function V(s, ω, a) related to the system state and the advantage function A(s, a, ω, β) related to the action before the network output layer and synthesizing these two functions to generate the action function in the final output layer. Thus, we have the following expression:

$$Q(s, a, \omega, \alpha, \beta) = V(s; \omega, \alpha) + (A(s, a, \omega, \beta) - \frac{1}{|\mathcal{A}|} \sum_{a_{t+1}} A(s, a_{t+1}, \omega, \beta)) \tag{3}$$

where ω is a parameter of the common part, α is a parameter of the value function, and β is a parameter of the advantage function.

In this study, dueling DQN and double DQN are used to solve the dynamic production scheduling problem.

3.3 CNN with the SPP layer

The insertion of new orders dynamically changes the size of multi-channel images expressed by state features. However, the full connection layer in the conventional CNN requires an input of fixed size. An SPP layer was added between the last convolutional layer and the first full connection layer in the CNN to divide the feature graph obtained after convolution into a fixed number of grids of various sizes. The grid is then pooled with mean values. Thus, the feature graph convolved with any size can be changed into the output of a fixed size so that the graph has the same dimension of feature vector with the following full connection layer. Thus, image convolution with any image size input can achieved as follows (Fig. 2).

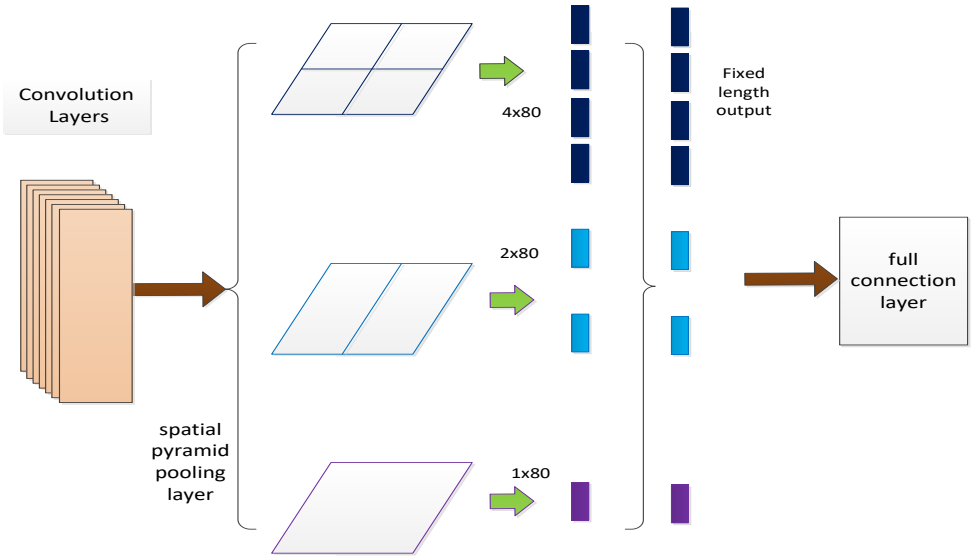


Fig. 2 SPP principle

3.4 Scheduling problem transformation

The problem transformation between scheduling and algorithm design is critical for applying DRL to JSPP and involves three aspects, namely state feature, action space, and reward function design.

State features expression

To improve the state changes of the production system, the following rules should be followed for describing the state features:

- State features should be able to reflect the features and changes of the production system, and both global and local state features should be considered.
- State features at each moment are represented by a universal feature set.
- State features should be represented numerically for easy calculation and standardization for uniform scaling of various features.

This study optimized and upgraded the production system state features based on literature [33]. The limitation of three channels in the conventional system state feature expression was overcome. Five channels were designed for characterization. The first channel is represented by the two-dimensional matrix of the order to be processed. The rows of the matrix represent the order, and the columns represent the operation. The initial value is the processing time of the corresponding operation in the order. The value of the corresponding position becomes zero on the completion of an operation. The second channel is represented by the two-dimensional matrix of the completed operations of the order. The initial value is zero. The value of the corresponding position is the processing time on completion of an operation. The third channel is the remaining processing time of the processing operation. The fourth channel is the processing time of each operation in the queue to be processed. The fifth channel is the waiting time of each operation in the waiting queue. The first and second channels express the global state feature, whereas the third, fourth, and fifth channels express the local state feature. All channel data are linearly normalized to the maximum value.

Definition of ACTIONS

The action selection involves selecting the most suitable operation waiting for processing, and the production scheduling rule can select an appropriate process at each scheduling decision point. In this article, 16 commonly used production dispatching rules are selected as the action space in DRL. The details are as follows: the select the job with the shortest processing time (SPT), select the job with the longest processing time (LPT), select the job with the longest remaining processing time (LWKR), select the job with the shortest remaining processing time (MWKR), select the job with the shortest processing time of subsequent operation (SSO), select

the job with the longest processing time of subsequent operation (LSO), select the job with the shortest remaining processing time in addition to the current operation (SRM), select the job with the longest remaining processing time in addition to the current operation (LRM), select the job that arrives first (FIFO), select the job with the earliest due date, select the job with the minimum sum of processing time of the current and subsequent operation (SPT+SSO), select the job with the maximum sum of processing time of the current and subsequent operation (LPT+LSO), select the job with the minimum ratio of current processing time to the total working time (SPT/TWK), select the job with the maximum ratio of current processing time to the total working time (LPT/TWK), select the job with the minimum product of the current processing time and total working time (SPT *TWK), select the job with the maximum product of current processing time and total working time (LPT * TWK). The diversity of dispatching rules is increased so that the agent can adaptively select dispatching rules.

Reward FUNCTION

The reward function is key to DRL and directly affects the direction of learning and is closely related to optimization. The reward function should be designed as follows: (1) the reward function should accurately express the immediate reward of the current action. (2) Cumulative reward should be closely related to the scheduling objective. (3) Reward function should be universal and can be used for problems of various scales. Because the overall scheduling objective is to minimize total tardiness, the following reward function should be designed:

$$r_k = Tard_{k-1} - Tard_{k,} \qquad Tard_k = \sum_{j=1}^{n} \delta_j(\tau) \tag{4}$$

where $\delta_j(\tau)$ represents the tardiness of job J at the current system state, $Tard_k$ is the total tardiness of all the jobs in the current system. For the job without tardiness, the tardiness is zero. Here, r_k represents the reward received at the decision-making time t_{k-1} after executing the action, then the system arrives at decision-making time t_k. The derivation process of cumulative reward function R is as follows:

$$\begin{aligned} \mathrm{R} &= \sum_{k=1}^{K} r_k = \sum_{k=1}^{K} Tard_{k-1} - Tard_k \\ &= Tard_0 - Tard_1 + Tard_1 - Tard_2 \\ &\quad + \cdots + Tard_{K-1} - Tard_K \\ &= Tard_0 - Tard_K = -Tard_K \end{aligned} \tag{5}$$

The derivation process of the cumulative reward formula reveals that it is inversely proportional to the total tardiness, that is, the smaller the total tardiness is, the larger the cumulative reward value is, which is consistent with the scheduling objective.

Training based on DRL

The scheduling process is a semi-Markov decision process. When any machine completes an operation or a new order arrives as a decision time point, the agent adaptively selects appropriate dispatching rules, and the highest priority operation is selected for processing. Subsequently, the machine enters the next state after receiving rewards. The cycle continues until all operations are finished, that is, a scheduling scheme is obtained. The process is as follows:

Algorithm 1 DDDQN-based training method

1: Initialize replay memory D, minibatch k, learning rate α, target network parameters update every C steps.
2: Initialize Q-network with random weights ω
3: Initialize target network $\widehat{Q}$ with weights $\omega^{-} = \omega$
4: For episode = 1 : M do
5: Reset the system scheduling status to s_0 and clear schedule results.
6: while True : (t is the decision time point at which an operation is completed or a new order arrives, the Boolean variable done terminates the loop when all operations are complete)

7: Select action a_t based on ε-greedy strategy
8: Execute action a_t, calculate the immediate reward r_t, observe the next state s_{t+1}
9: Store transition$\langle s_t, a_t, r_t, s_{t+1}, done\rangle$ in D
10: Random sampling minibatch of transitions $\langle s_i, a_i, i, s_{i+1}, done\rangle$ from D
11: $y_i = \begin{cases} r_i & \text{if done} = \text{true} \\ r_i + \gamma \hat{Q}(s_{i+1}, argmax_a(s_{i+1}, a; \omega), \omega^-) & \text{otherwise} \end{cases}$
12: Compute TD-error$(y_i - Q(s_i, a; \omega))^2$to update the parameters of Q-network
13: Every C steps update the parameters of the target network $\hat{Q}$: $\omega^- = \omega$
14: end while
15: end for

The training process is divided into inner and outer loops, the outer loop represents the times of training. After M loops of cyclic training, the agent gradually reaches the ability to adaptively select the optimal dispatching rules at various decision moments. The inner loop represents a complete scheduling scheme generation process, starting from the first operation until all operations are finished, which is an episode, lines 5-14 describe the execution of the inner loop that starts from the initial state of s_0, at the decision moment t the agent select and execute the action a_t based on the ε-greedy strategy, the suitable operation is scheduled. The reward r_k and the next state s_{t+1} are observed, the transition $\langle s_t, a_t, r_t, s_{t+1}, done\rangle$ is stored. Minibatch transitions are randomly sampled from the replay memory for system training. A loss was calculated according to lines 11 and 12, and gradient descent was used to update the parameters of Q-network. The parameters of the target network were updated by the parameters of Q-network according to the contents of 13 lines after every C step.

4. Numerical simulation

In numerical simulation, multiple groups of data were used to train the DDDQN. The optimal training model is then saved, and the model is tested in the new test data of multiple groups of various production scenarios. The data generation method proposed in a previous study [29] is randomly generated, and the parameters are presented in Table 1.

According to the arrival time and number of new orders, the number of machines, and the due date of orders, 81 groups of data of various production scenarios were randomly generated for the test. Assuming that 30 orders exist at the beginning of each production scenario, the arrival time of new jobs follow Poisson distribution. Therefore, the time interval between two consecutive new jobs is subjected to exponential distribution. The due date tightness (DDT) represents the urgency of orders. The due date of order i can be calculated as $D_i = A_i + (\sum_j^{n_i} t_{ij}) \cdot DDT$. Here, A_i is the arrival time of the order, n_i is the number of operations in the order, t_{ij} is the processing time of the j operation of the order. The smaller the DDT is, the more urgent the order due date is.

Table 1 Parameter settings of various production scenarios

Parameter	Value
Number of machines	5,10,15
Number of initial jobs	30
Number of new job insertions	10,30,50
Processing time of each operation	Unif[0,50]
Due date tightness	1.0,1.5,2.0
Average value of exponential distribution between two successive new job arrivals	25,50,100

4.1 Network structure and system parameter setting

The CNN structure of the DDDQN consists of four convolution layers and two full connection layers. To solve the problem of variable image size, a SPP layer was added between the convolution layer and the full connection layer. From the first layer to the fourth layer for the convolution layers, the size of the convolution kernel was 6 × 6, 4 × 4, 3 × 3, 2 × 2, the step size was 2, 2, 2, 1, and the number of output channels was 20, 40, 60, and 80. Because each element in the state feature image represents an operation, the pooling layer in the CNN results in incomplete

scheduling information. Therefore, the pooling layer was not used. The full connection layer consists of two branches of the full connection layer with 512 units. The two branches connect the state value and the advantage value. Finally, the state and advantage values were combined to obtain the final result output. The RELU activation function was used for every layer. The Adam optimizer was used to update the parameters.

Parameter setting considerably affected DDDQN performance. Five groups of data were randomly generated for the parameter sensitivity experiment under 10 new order insertions, 10 machines, 50 average time interval between new order arrival, and DDT tardiness coefficient of 1.5. The effects of the training batch, learning rate, replay memory buffer size, and target network parameter updating frequency on algorithm performance were verified. Fig. 3 displays the training effect under various parameter settings, the total number of training was set to 3000 episodes.

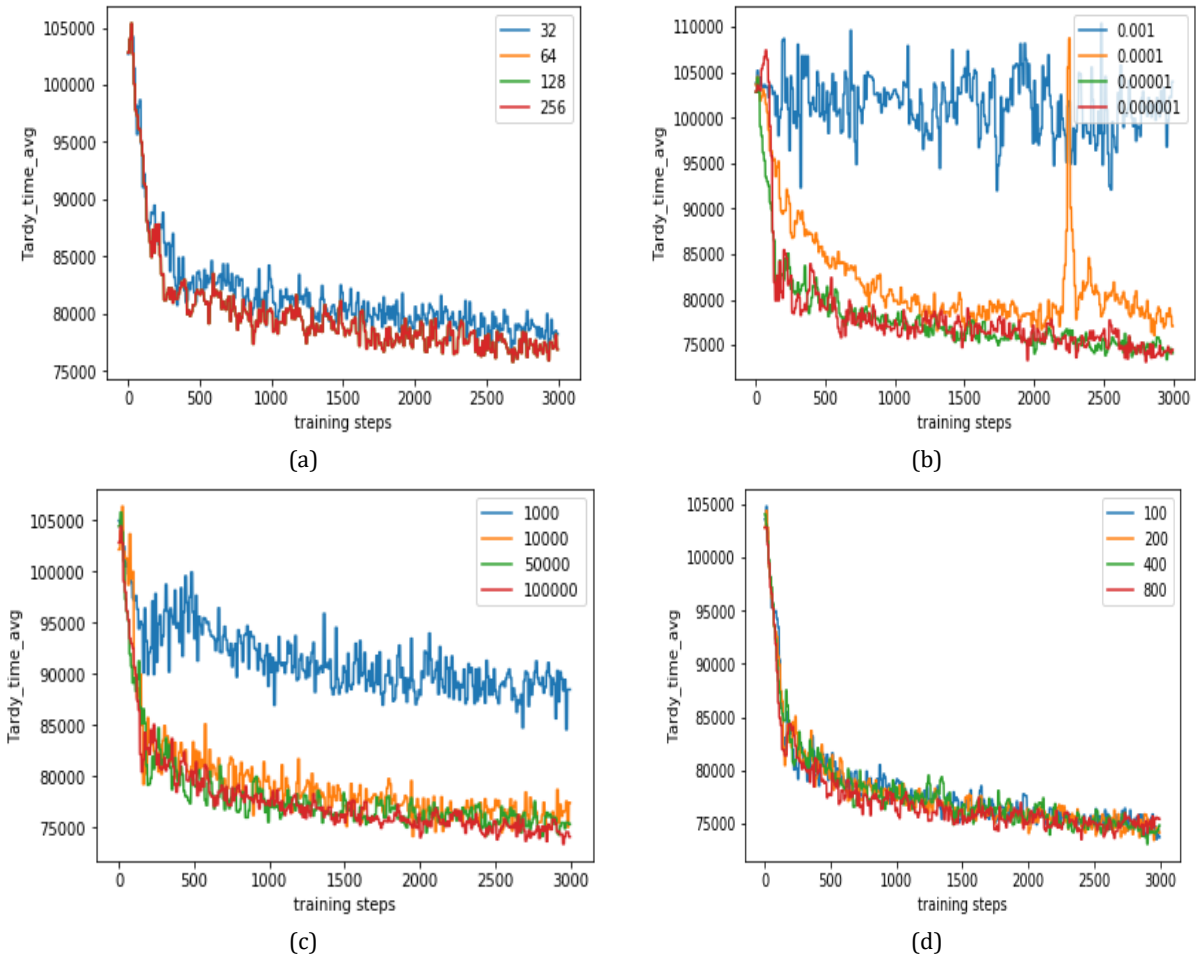


Fig. 3 Verification results of each hyperparameter: (a) Minibatch size; (b) Learning rate; (c) Replay memory buffer size (d) Target network updating frequency

Fig. 3(a) verifies the influence of the training batch on the algorithm. The figure reveals that all parameters exhibits excellent stability. The performance with a batch size of 32 decreased slightly. Fig. 3(b) displays the influence of various learning rates on the algorithm. The higher the learning rate is, the more the training effect is unstable. When the learning rate is 0.001, the algorithm does not even converge. Fig. 3(c) displays the influence of various replay memory buffer sizes on the algorithm. As displayed in the figure, the larger the replay memory is, the better the convergence of the algorithm is, and the replay memory with a capacity of 100000 exhibits superior stability. Fig. 3(d) displays the influence of the target network parameter updating frequency on algorithm performance, and the parameters exhibit an effect under various updating frequencies. According to the verification of various parameters, the final neural network parameter settings are presented in Table 2.

Table 2 Setting of neural network parameters

Parameters	Values
Number of episodes	3000
Explore times steps	3000 · total operation number · 0.3
Epsilon	$1-(1-\varepsilon_{min}\cdot \min(1, \text{current}_{\text{iter}}/\text{totalsteps}))$
Replay memory buffer size	100000
Learning rate	0.000001
Minibatch	256
Target network updating frequency	200
Discount factor	1.0

The ε-greedy action selection strategy was implemented according to the method mentioned in a previous study [33]. To ensure the maximum cumulative rewards learned by the agent, the discount factor is selected as 1.0, that is, the cumulative rewards are not discounted.

4.2 Comparison of various status features expression

To verify the validity of the state feature expression of the proposed five-channel images, the influence of three state feature expression modes of three-channel images, four-channel images, and five-channel images on the algorithm were compared. In three-channel images, the production system state feature expression method in literature is adopted [33]. Five-channel images were the proposed production system state feature expression method, and four-channel images were separated from five-channel images to remove the waiting time channel of each operation in the waiting queue.

Five groups of data were randomly generated for testing under the following production configurations: the number of machines was 10, the average time interval between two consecutive new order arrival was 50, DDT was 1.5, and the number of new order insertion were 10, 30, and 50. The results are displayed in Fig. 4. The figure reveals that with the increase in the new order insertion scale, the expression methods of three-channel and four-channel both fluctuated considerably, whereas the proposed expression method exhibited excellent stability.

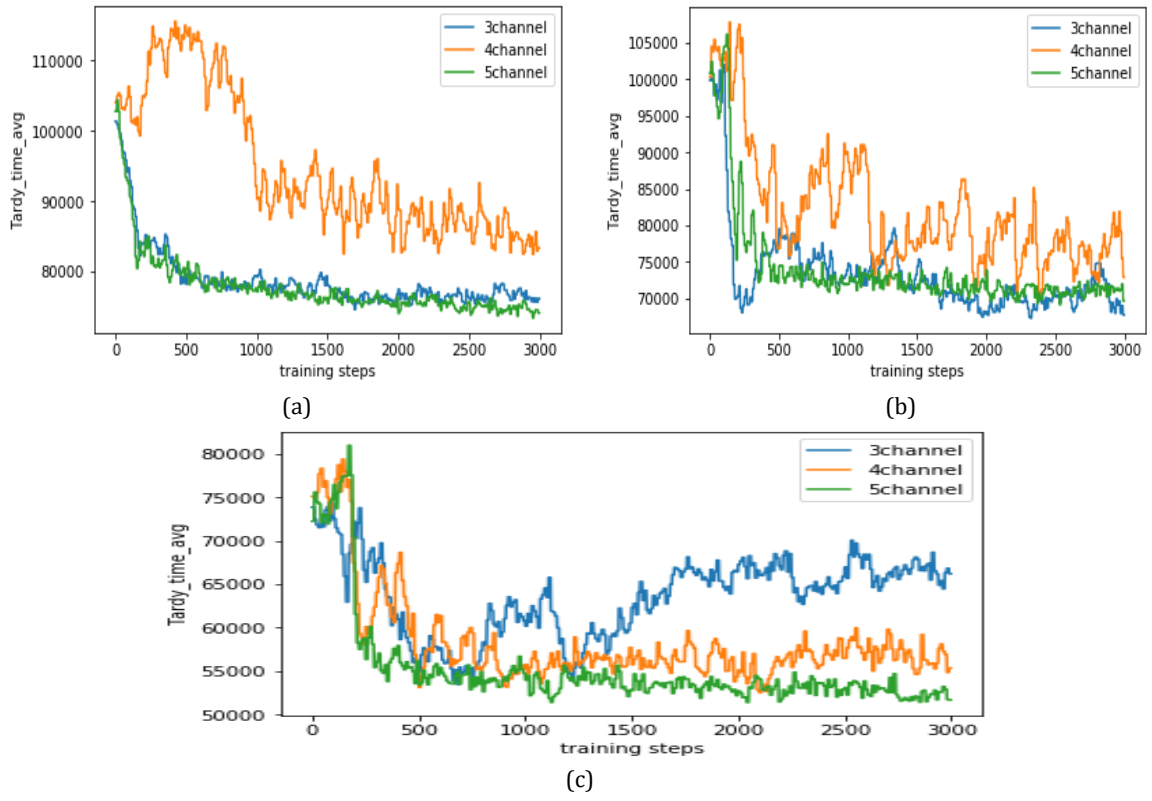
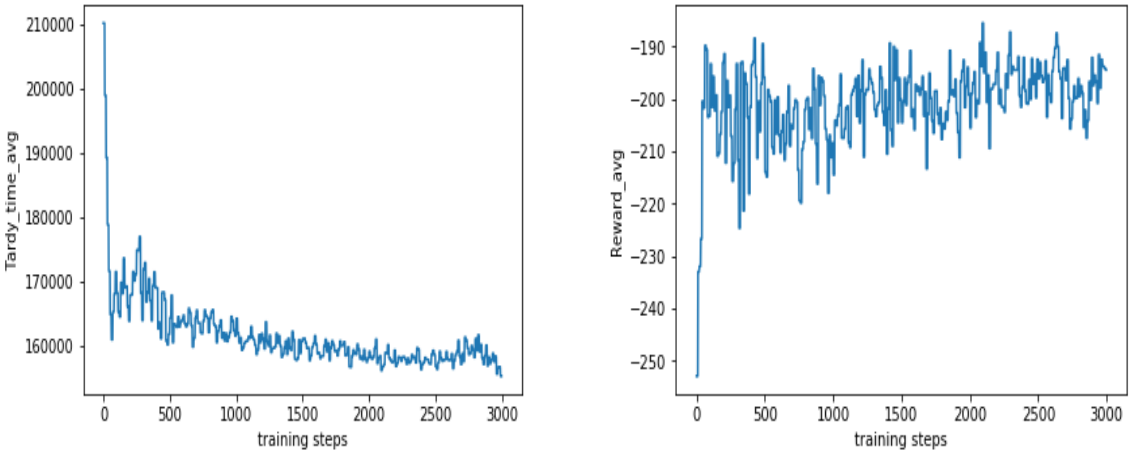


Fig. 4 Verification results of various state features: (a) 10 new job insertions; (b) 30 new job insertions; (c) 50 new order insertions

4.3 Training process of the DDDQN

The DDDQN was used to train a model with certain generalization ability. The model was tested with various test data. The model was trained according to the number of machine and the time interval between the dynamic arrival of orders. Each model was trained for 3000 episodes. In the training process, 12 groups of randomly generated data were used according to the number of new orders of 10, 30, and 50 and the DDT of 1.0 and 2.0. Fig. 5 displays the model training process with five machines and 100 times intervals.



(a) (b)

Fig. 5 Model training process in five machines and 100 times intervals: (a) Average total tardiness during training; (b) Average reward during training

Figs. 5(a) and 5(b) display the change process of the total tardiness and reward during the training process. The reward value gradually increases and becomes stable with the increase in training episodes, whereas the total tardiness decreases. After 1000 episodes of training, the model becomes stable, which indicates that the DDDQN has learned to adaptively select the appropriate dispatching rules at the decision time. The curve trend of the average reward value is similar to that of the average total tardiness, which indicates that the designed reward function exhibited a high correlation with the optimization objective with the minimum total tardiness. A small fluctuation was observed after the model convergence. The fluctuation was related to the exploration mechanism of DRL.

4.4 Comparison with conventional dispatching rules

To verify whether the training model can select appropriate dispatching rules at various decision moments, 16 dispatching rules were compared on the test data set. Test data were configured for 81 production scenarios according to all parameter configurations in Table 1, and 30 groups of test data were randomly generated for each production scenario. Tables 3-5 displays the comparison results under various number of machines. The data in the table are the average values of test data. The results of the optimal values are displayed in bold for easy identification. The test results indicate that the algorithm model of the DDDQN is superior to the single scheduling rule in most cases, which reveals satisfactory solution solving ability and generalization ability of various problems. Finding a scheduling rule that can perform well in all production scenarios is difficult.

Table 3 Test results compared with dispatching rules under five machines

Eavg	DDT	n_{add}	DDDQN	SPT	LPT	LWKR	MWKR	SSO	LSO	SRM	LRM	FIFO	EDD	SPT+ SSO	LPT+ LSO	SPT/ TWK	LPT/ TWK	SPT* TWK	LPT* TWK
25	1.0	10	**17357**	19686	28070	17510	29855	20858	27870	18753	29631	27863	18483	18352	27245	22350	27779	17884	27068
		30	**32305**	38132	57816	32469	64009	40996	60024	35495	64420	52818	36364	34790	56965	44306	57974	33911	56239
		50	**45832**	53785	87842	45887	101770	57570	96690	50743	103294	74163	52510	48775	87674	64105	87011	48203	85888
	1.5	10	**14908**	17218	25612	15059	27386	18393	25401	16299	27162	25395	15896	15887	24780	19883	25310	15417	24614
		30	**28547**	34363	54069	28717	60239	37230	56253	31743	60648	49046	32326	31024	53201	40542	54208	30146	52510
		50	**40911**	48853	82932	40980	96842	52642	91754	45841	98358	69226	47351	43851	82756	59182	82083	43278	81013
	2.0	10	**12571**	14771	23215	12742	24920	15972	22935	13974	24694	22926	13377	13487	22342	17472	22855	13006	22266
		30	**24932**	30656	50447	25145	56486	33541	52504	28151	56878	45275	28323	27369	49518	36926	50492	26516	48963
		50	**36174**	44031	78168	36321	91953	47829	86824	41178	93428	64290	42214	39115	77991	54527	77227	38562	76427
50	1.0	10	**16838**	19093	27552	16932	29539	20199	27498	18072	29347	26825	17965	17791	26742	21664	27282	17274	26818
		30	**22246**	24766	39989	22278	43158	27871	39063	24663	42527	34269	24550	23414	38877	28719	39383	22724	39588
		50	26119	27639	49293	26601	51064	32367	45594	30007	49455	38772	28549	26413	45654	32019	48605	**25648**	47967
	1.5	10	**14346**	16584	25049	14444	27028	17692	24988	15572	26834	24313	15400	15284	24240	19155	24773	14765	24329
		30	**18721**	21317	36542	18822	39756	24381	35655	21185	39110	30782	21034	19946	35451	25304	35889	19270	36192
		50	21876	23515	44928	22292	47016	28009	41471	25624	45358	34521	24043	22165	41481	27941	44191	**21458**	43722
	2.0	10	**12003**	14118	22644	12113	24533	15235	22492	13222	24329	21800	12904	12865	21800	16748	22298	12355	21981
		30	**15568**	18155	33328	15700	36565	21106	32420	17997	35857	27380	17561	16755	32305	22259	32545	16166	33112
		50	18074	19912	41027	18597	43405	24099	37736	21828	41625	30466	19865	18475	37867	24497	40117	**17910**	40067
100	1.0	10	**15173**	17021	25108	15352	26651	18225	24509	16480	26259	23709	16125	16014	24234	19305	24820	15536	24473
		30	**15748**	17485	27199	15902	27681	19793	25101	17467	27181	24300	16815	16495	25888	19741	27063	15994	26604
		50	**14768**	16025	25094	14954	25826	17951	23477	16130	25365	22904	15595	15418	23825	18441	24543	14841	24458
	1.5	10	**12705**	14556	22641	12893	24196	15747	22049	14013	23799	21234	13584	13548	21785	16846	22330	13070	22025
		30	**12998**	14804	24421	13163	25045	17017	22444	14700	24530	21577	14014	13789	23212	17087	24245	13304	23876
		50	**12189**	13498	22442	12376	23327	15326	20982	13507	22852	20335	13043	12847	21304	15935	21861	12311	21896
	2.0	10	**10417**	12202	20294	10641	21823	13362	19656	11744	21393	18780	11157	11218	19466	14562	19914	10746	19771
		30	**10560**	12375	21850	10756	22603	14459	19976	12243	22066	18967	11374	11343	20777	14748	21616	10924	21456
		50	**9874**	11166	20028	10097	20997	12896	18637	11209	20480	17860	10602	10512	19004	13700	19346	10034	19620

Table 4 Test results compared with dispatching rules under ten machines

Eavg	DDT	nadd	DDDQN	SPT	LPT	LWKR	MWKR	SSO	LSO	SRM	LRM	FIFO	EDD	SPT+ SSO	LPT+ LSO	SPT/ TWK	LPT/ TWK	SPT* TWK	LPT* TWK
25	1.0	10	20981	22591	31577	21132	32255	25542	28239	21785	31198	30193	21617	22329	30241	24314	31177	**20934**	30617
		30	**33733**	37593	55837	33878	59908	43261	52166	35861	57985	50718	36140	37020	54543	40785	55596	34486	54841
		50	**54353**	60774	95044	54712	105769	71725	88968	57266	102888	81281	58799	60474	90305	66574	93949	55778	92471
	1.5	10	15896	17492	26486	16055	27155	20447	23143	16704	26098	25092	16393	17230	25154	19218	26081	**15834**	25540
		30	**26682**	30549	48797	26854	52863	36216	45126	28843	50938	43670	28891	29974	47514	33743	48551	27442	47835
		50	**44665**	51057	85352	45018	96054	62014	79251	47592	93170	71562	48740	50757	80611	56868	84235	46064	82813
	2.0	10	11218	12496	21544	11469	22067	15486	18173	12142	21006	19992	11478	12280	20249	14388	21066	**10899**	20735
		30	**20163**	23680	41964	20383	45856	29296	38215	22321	43910	36623	21926	23159	40714	27071	41575	20637	41182
		50	**35443**	41592	75924	35946	86383	52458	69664	38515	83493	61844	38873	41317	71231	47648	74616	36716	73633
50	1.0	10	**19524**	20891	29663	19760	30786	23937	26574	20428	29779	27904	20322	20955	28548	22412	29925	19604	29158
		30	**26583**	28375	45016	27452	47407	33446	39682	28857	44996	37856	28725	28281	43301	30408	45299	26766	44594
		50	33767	35203	61152	35939	60039	44526	49236	37475	56292	47583	37539	36658	56320	37714	61198	**33730**	60224
	1.5	10	**14441**	15795	24575	14687	25691	18845	21486	15359	24682	22804	15189	15856	23472	17331	24826	14507	24112
		30	**19467**	21245	37726	20195	40338	26211	32554	21585	37958	30614	21423	21100	36089	23334	37989	19633	37378
		50	24179	25669	51299	26152	50594	34705	39658	27685	46865	37768	27502	27022	46608	28274	51316	**24152**	50496
	2.0	10	9738	10907	19714	10149	20688	13936	16602	10799	19671	17706	10285	11003	18614	12662	19854	**9662**	19438
		30	**13470**	14995	31111	13963	34026	19701	26216	15344	31656	23763	14672	14816	29650	17493	31154	13625	30927
		50	**544470**	18053	42552	18184	42803	26247	31776	19565	39129	29031	18947	19225	38292	21088	42314	16826	42168
100	1.0	10	**17211**	18062	26805	17587	27139	21091	23701	18277	26008	24274	17847	18353	25360	19446	26445	17240	26114
		30	17389	18268	27961	18330	27786	21831	23995	18980	26194	24260	18605	18548	26906	19710	27501	**17384**	27545
		50	19839	20904	31334	21012	31206	24680	27157	21590	29455	27729	21248	21469	29566	22318	31428	**19836**	30832
	1.5	10	**12234**	13112	21796	12588	22226	16078	18769	13282	21107	19283	12781	13378	20406	14518	21417	12275	21164
		30	**11913**	12824	22216	12632	22401	16191	18572	13283	20822	18697	12879	13048	21299	14331	21721	11922	21848
		50	**13601**	14726	24832	14437	25137	18308	20932	15070	23434	21397	14678	15240	23231	16205	24887	13601	24408
	2.0	10	**7743**	8658	17171	8316	17646	11477	14229	8998	16517	14436	8156	8897	15876	10249	16670	7859	16747
		30	**7413**	8379	17390	8232	17813	11454	14046	8774	16273	13799	8194	8571	16635	10075	16779	7527	17197
		50	**8800**	9903	19595	9633	20152	13151	15982	10256	18472	16105	9647	10369	18204	11612	19556	8910	19386

Table 5 Test results compared with dispatching rules under fifteen machines

Eavg	DDT	nadd	DDDQN	SPT	LPT	LWKR	MWKR	SSO	LSO	SRM	LRM	FIFO	EDD	SPT+ SSO	LPT+ LSO	SPT/ TWK	LPT/ TWK	SPT* TWK	LPT* TWK
25	1.0	10	22058	23233	31924	22220	31724	26677	27884	22971	30298	29570	22884	23507	30281	24643	31552	**22022**	31409
		30	**43428**	47477	66167	43666	69721	54727	58929	44931	67134	60692	45554	48028	62792	50423	65351	44820	65288
		50	**62010**	69072	101619	62266	112907	80190	90506	64825	109394	88322	65971	68955	96448	74621	100463	64080	100294
	1.5	10	14403	15539	24249	14606	24024	18982	20203	15343	22598	21870	15239	15812	22619	16960	23866	**14331**	23779
		30	**31929**	35961	54661	32194	58207	43217	47418	33470	55619	49176	34300	36519	51295	38915	53842	33308	53851
		50	**46976**	54008	86584	47275	97842	65126	75445	49835	94329	73255	50662	53895	81410	59562	85398	49019	85297
	2.0	10	7334	8288	17063	8336	16362	11655	12974	8995	14954	14210	8249	8596	15475	10082	16485	**7202**	16848
		30	**21257**	24694	43555	21926	46752	31911	36133	23125	44153	37669	23234	25351	40249	27998	42562	22244	43059
		50	**32904**	39352	71957	33690	82912	50335	60800	36150	79406	58199	36263	39264	66871	45302	70549	34677	70861
50	1.0	10	**21029**	21865	29949	21193	30426	25171	26476	21726	28820	27873	21855	22078	28617	23108	29644	21066	29784
		30	**32289**	33714	51706	32983	52959	40450	43539	34230	49804	43101	34524	34815	48151	35389	51248	32480	50473
		50	38373	38744	63575	40050	62705	47647	51403	41284	59086	49754	41548	40156	58561	40858	61844	**38085**	62948
	1.5	10	**13358**	14180	22297	13574	22754	17491	18809	14122	21152	20183	14013	14396	20978	15452	21965	13396	22172
		30	**21044**	22477	40327	21716	41791	29076	32281	22884	38709	31722	23277	23554	36876	24233	39815	21236	39162
		50	23960	24485	48881	25429	48478	33044	37038	26681	44917	35094	26630	25716	43984	26674	47048	**23727**	48349
	2.0	10	**6461**	7219	15262	7335	15351	10340	11753	7849	13776	12608	7349	7381	13961	8828	14691	6494	15355
		30	**11824**	13065	30074	12674	31974	18937	22485	13555	29025	21046	12782	14019	26907	15283	29286	12116	29269
		50	**13149**	13821	36133	14338	37025	20928	25650	15344	33588	22410	14083	14595	31933	16432	33995	13226	36082
100	1.0	10	18289	19347	27279	19016	26979	22250	23649	19559	25844	24482	19284	19640	25772	20125	26824	**18281**	26956
		30	**20760**	21456	32037	21560	30585	25180	26591	22381	28830	27015	22220	22138	29728	22390	31795	20778	31263
		50	**22665**	23284	35045	23668	32977	27470	29123	24374	31529	29369	24431	23971	33033	24490	34917	22666	34470
	1.5	10	10811	11865	19689	11512	19569	14712	16171	12039	18434	16913	11776	12159	18248	12704	19215	**10803**	19424
		30	**11766**	12475	22497	12301	21643	15950	17502	13102	19956	17813	12990	13113	20475	13539	22283	11809	21898
		50	**12626**	13301	24121	13092	22945	17011	18764	13839	21531	18970	13640	13940	22381	14583	23942	12638	23689
	2.0	10	**4589**	5622	12967	5758	12852	8112	9721	6253	11736	9797	5394	5876	11688	6840	12424	4707	13028
		30	**5302**	5996	15253	6139	14699	8970	10770	6872	13072	10376	6079	6604	13564	7505	14796	5426	14896
		50	**5889**	6585	16380	6687	15730	9626	11608	7302	14393	11133	6549	7092	15003	8084	16061	6069	16318

4.5 Comparison with the GA algorithm

To prove the computational speed and optimization ability of the model, the DDDQN was compared with the GA. In the GA, an active decoding approach and an elite retention strategy are used. For the medium- and large-scale problems, the GA first generates an initial scheduling scheme based on the initial order data. When a new order arrives, the GA reschedules to generate a new scheduling scheme. The start processing time of all orders differs considerably, and the number of the remaining operations of each order also differs. The parameters of the GA are set as follows: population size is 50, the crossover rate is 0.9, the mutation rate is 0.1, and the iteration number is 300. Some representative data were selected for verification. Each group of test data consists of 30 randomly generated data. The results are presented in Table 6. The data in the table are the average values of test data. The scheduling results and calculation time of the model are superior to the GA in all cases. The average calculation time of the DDDQN model to generate the scheduling scheme for test data at each decision moment was 0.05 s, which was almost instantaneous. Thus, the model can be used for real-time scheduling.

Table 6 Comparison results of DDDQN and GA

				Total tardiness		CPU times (s)	
m	Eavg	DDT	nadd	DDDQN	GA	DDDQN	GA
5	25	1.0	10	17357	26263	0.04	46.29
			50	45832	64650	0.02	57.38
		2.0	10	12571	21354	0.04	46.26
			50	36174	53352	0.02	56.34
	100	1.0	10	15173	22436	0.04	35.42
			50	14768	23195	0.01	8.11
		2.0	10	10417	18107	0.04	35.82
			50	9874	17173	0.01	8.10
10	25	1.0	10	20981	30552	0.10	92.18
			50	54353	107956	0.05	163.86
		2.0	10	11218	20531	0.10	91.82
			50	35443	88138	0.05	164.19
	100	1.0	10	17211	28167	0.10	82.69
			50	19839	36521	0.04	28.24
		2.0	10	7743	18470	0.10	82.35
			50	8800	22720	0.04	28.30
15	25	1.0	10	22058	32677	0.18	139.78
			50	62010	130030	0.08	266.73
		2.0	10	7334	17217	0.15	138.98
			50	32904	98901	0.08	265.63
	100	1.0	10	18289	31864	0.18	131.80
			50	22665	48197	0.08	57.58
		2.0	10	4589	17439	0.16	131.53
			50	5889	26732	0.07	58.57

5. Conclusion

A DRL algorithm, namely the DDDQN, was proposed to solve real-time dynamic job shop scheduling with new order insertions. SPP technology was applied to the neural network structure. A five-channel production system state feature expression method that considered both global and local feature information was considered. As the action space, 16 commonly used dispatching rules were used, and the corresponding reward function was designed to minimize total tardiness. Finally, considerable data from various production scenarios were generated at random to train and test the system model.

Compared with conventional dispatching rules and heuristic algorithms, the results revealed that the algorithm outperformed the single scheduling rule method in most cases, which indicated that the algorithm can select dispatching rules adaptively in various production states. Com-

pared with the GA, the computational speed and optimization ability of the trained models were validated, and real-time optimization and online decision were performed in dynamic event disturbance.

In the future, numerous uncertain factors, such as emergency orders, order cancellations, uncertain processing times, equipment failures, and other multiple disturbance factors, will be studied. Compared with the pure full connection layer neural network, the CNN exhibits a complex structure, which renders model training speed slow. The DQN in this study is a value-based method that cannot directly optimize the policy. Therefore, policy-based DRL methods, such as A3C and PPO, should be studied to improve the quality of solutions and the training speed.

References

[1] Zhou, J., Li, P.G., Zhou, Y.H., Wang, B.C., Zang, J.Y., Meng, L. (2018). Toward new-generation intelligent manufacturing, *Engineering*, Vol. 4, No. 1, 11-20, doi: 10.1016/j.eng.2018.01.002.

[2] Wang, X.H., Duan, H.B. (2014). A hybrid biogeography-based optimization algorithm for job shop scheduling problem, *Computers & Industrial Engineering*, Vol. 73, 96-114, doi: 10.1016/j.cie.2014.04.006.

[3] Çaliş, B., Bulkan, S. (2015). A research survey: Review of AI solution strategies of job shop scheduling problem, *Journal of Intelligent Manufacturing*, Vol. 26, No. 5, 961-973, doi: 10.1007/s10845-013-0837-8.

[4] Baykasoğlu, A., Karaslan, F.S. (2017). Solving comprehensive dynamic job shop scheduling problem by using a GRASP-based approach, *International Journal of Production Research*, Vol. 55, No. 11, 3308-3325, doi: 10.1080/00207543.2017.1306134.

[5] Bokrantz, J., Skoogh, A., Ylipää, T., Stahre, J. (2016). Handling of production disturbances in the manufacturing industry, *Journal of Manufacturing Technology Management*, Vol. 27, No. 8, 1054-1075, doi: 10.1108/JMTM-02-2016-0023.

[6] Zhang, F.F., Mei, Y., Nguyen, S., Zhang, M.J. (2020). Evolving scheduling heuristics via genetic programming with feature selection in dynamic flexible job-shop scheduling, *IEEE Transactions on Cybernetics*, Vol. 51, No. 4, 1797-1811, doi: 10.1109/TCYB.2020.3024849.

[7] Kundakcı, N., Kulak, O. (2016). Hybrid genetic algorithms for minimizing makespan in dynamic job shop scheduling problem, *Computers & Industrial Engineering*, Vol. 96, 31-51, doi: 10.1016/j.cie.2016.03.011.

[8] Cao, H.J., Zhou, J., Jiang, P., Hon, K.K.B., Yi, H., Dong, C.Y. (2020). An integrated processing energy modeling and optimization of automated robotic polishing system, *Robotics and Computer-Integrated Manufacturing,* Vol. 65, Article No. 101973, doi: 10.1016/j.rcim.2020.101973.

[9] Ning, T., Huang, M., Liang, X., Jin, H. (2016). A novel dynamic scheduling strategy for solving flexible job-shop problems, *Journal of Ambient Intelligence and Humanized Computing*, Vol. 7, No. 5, 721-729, doi: 10.1007/s12652-016-0370-7.

[10] Fan, W., Zheng, L.Y., Ji, W., Xu, X., Lu, Y.Q., Wang, L.H. (2021). A machining accuracy informed adaptive positioning method for finish machining of assembly interfaces of large-scale aircraft components, *Robotics and Computer-Integrated Manufacturing*, Vol. 67, Article No. 102021, doi: 10.1016/j.rcim.2020.102021.

[11] Zhang, S.C., Wong, T.N. (2017). Flexible job-shop scheduling/rescheduling in dynamic environment: A hybrid MAS/ACO approach, *International Journal of Production Research*, Vol. 55, No. 11, 3173-3196, doi: 10.1080/00207543.2016.1267414.

[12] Park, S.C., Raman, N., Shaw, M.J. (1997). Adaptive scheduling in dynamic flexible manufacturing systems: A dynamic rule selection approach, *IEEE Transactions on Robotics and Automation*, Vol. 13, No. 4, 486-502, doi: 10.1109/70.611301.

[13] Wang, Z., Zhang, J.H., Yang, S.X. (2019). An improved particle swarm optimization algorithm for dynamic job shop scheduling problems with random job arrivals, *Swarm and Evolutionary Computation,* Vol. 51, Article No. 100594, doi: 10.1016/j.swevo.2019.100594.

[14] Caldeira, R.H., Gnanavelbabu, A., Vaidyanathan, T. (2020). An effective backtracking search algorithm for multi-objective flexible job shop scheduling considering new job arrivals and energy consumption, *Computers & Industrial Engineering,* Vol. 149, Article No. 106863, doi: 10.1016/j.cie.2020.106863.

[15] Ghaleb, M., Zolfagharinia, H., Taghipour, S. (2020). Real-time production scheduling in the Industry-4.0 context: Addressing uncertainties in job arrivals and machine breakdowns, *Computers & Operations Research,* Vol. 123, Article No. 105031, doi: 10.1016/j.cor.2020.105031.

[16] Tang, D.B., Dai, M., Salido, M.A., Giret, A. (2016). Energy-efficient dynamic scheduling for a flexible flow shop using an improved particle swarm optimization, *Computers in Industry*, Vol. 81, 82-95, doi: 10.1016/j.compind.2015.10.001.

[17] Panwalkar, S.S., Iskander, W. (1977). A survey of scheduling rules, *Operations Research*, Vol. 25, No. 1, 45-61, doi: 10.1287/opre.25.1.45.

[18] Lu, M.-S., Romanowski, R. (2013). Multicontextual dispatching rules for job shops with dynamic job arrival, *International Journal of Advanced Manufacturing Technology*, Vol. 67, 19-33, doi: 10.1007/s00170-013-4765-8.

[19] Zhang, H., Roy, U. (2019). A semantics-based dispatching rule selection approach for job shop scheduling, *Journal of Intelligent Manufacturing*, Vol. 30, No. 7, 2759-2779, doi: 10.1007/s10845-018-1421-z.

[20] Zhang, F.F., Mei, Y., Zhang, M.J. (2019). A new representation in genetic programming for evolving dispatching rules for dynamic flexible job shop scheduling, In: Liefooghe, A., Paquete, L. (eds.), *Evolutionary Computation in Combinatorial Optimization. EvoCOP 2019. Lecture Notes in Computer Science,* Vol 11452. Springer, Cham, Switzerland, 33-49, doi: 10.1007/978-3-030-16711-0_3.
[21] Ferreira, C., Figueira, G., Amorim, P. (2022). Effective and interpretable dispatching rules for dynamic job shops via guided empirical learning, *Omega*, Vol. 111, Article No. 102643, doi: 10.1016/j.omega.2022.102643.
[22] Kaelbling, L.P., Littman, M.L., Moore, A.W. (1996). Reinforcement learning: A survey, *Journal of Artificial Intelligence Research*, Vol. 4, 237-285, doi: 10.1613/jair.301.
[23] Sutton, R.S., Barto, A.G. (2018). *Reinforcement learning: An introduction*, Second edition, MIT press, Cambridge, Massachusetts, USA.
[24] Wang, Y.-C., Usher, J.M. (2004). Learning policies for single machine job dispatching, *Robotics and Computer-Integrated Manufacturing,* Vol. 20, No. 6, 553-562, doi: 10.1016/j.rcim.2004.07.003.
[25] Chen, X.L., Hao, X.C., Lin, H.W., Murata, T. (2010). Rule driven multi objective dynamic scheduling by data envelopment analysis and reinforcement learning, In*: Proceedings of 2010 IEEE International Conference on Automation and Logistics,* Hong Kong, China, 396-401, doi: 10.1109/ICAL.2010.5585316.
[26] Qu, S.H., Wang, J., Shivani, G. (2016). Learning adaptive dispatching rules for a manufacturing process system by using reinforcement learning approach, In: *Proceedings of 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, Berlin, Germany, 1-8, doi: 10.1109/ETFA.2016.7733712.
[27] Arulkumaran, K., Deisenroth, M.P., Brundage, M., Bharath, A.A. (2017). Deep reinforcement learning: A brief survey, *IEEE Signal Processing Magazine*, Vol. 34, No. 6, 26-38, doi: 10.1109/MSP.2017.2743240.
[28] Zhu, J., Wang, H., Zhang, T. (2020). A deep reinforcement learning approach to the flexible flowshop scheduling problem with makespan minimization, In: *Proceedings of 2020 IEEE 9th Data Driven Control and Learning Systems Conference (DDCLS)*, Liuzhou, China, 1220-1225, doi: 10.1109/DDCLS49620.2020.9275080.
[29] Luo, S. (2020). Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning, *Applied Soft Computing*, Vol. 91, Article No. 106208, doi: 10.1016/j.asoc.2020.106208.
[30] Yang, S., Xu, Z., Wang, J. (2021). Intelligent decision-making of scheduling for dynamic permutation flowshop via deep reinforcement learning, *Sensors*, Vol. 21, No. 3, Article No. 1019, doi: 10.3390/s21031019.
[31] Li, Y.X., Gu, W.B., Yuan, M.H., Tang, Y.M. (2022). Real-time data-driven dynamic scheduling for flexible job shop with insufficient transportation resources using hybrid deep Q network, *Robotics and Computer-Integrated Manufacturing*, Vol. 74, Article No. 102283, doi: 10.1016/j.rcim.2021.102283.
[32] Liu, C.-L., Chang, C.-C., Tseng, C.-J. (2020). Actor-critic deep reinforcement learning for solving job shop scheduling problems, *IEEE Access*, Vol. 8, 71752-71762, doi: 10.1109/ACCESS.2020.2987820.
[33] Han, B.-A., Yang, J.-J. (2020). Research on adaptive job shop scheduling problems based on dueling double DQN, *IEEE Access*, Vol. 8, 186474-186495, doi: 10.1109/ACCESS.2020.3029868.
[34] Wang, L.B., Hu, X., Wang, Y., Xu, S.J., Ma, S.J., Yang, K.X., Liu, Z.J., Wang, W.D. (2021). Dynamic job-shop scheduling in smart manufacturing using deep reinforcement learning, *Computer Networks*, Vol. 190, Article No. 107969, doi: 10.1016/j.comnet.2021.107969.
[35] He, K.M., Zhang, X.Y., Ren, S.Q., Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 37, No. 9, 1904-1916, doi: 10.1109/TPAMI.2015.2389824.