Computers & Industrial Engineering 180 (2023) 109255

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Dynamic scheduling for flexible job shop using a deep reinforcement learning approach

Yong Gui, Dunbing Tang *, Haihua Zhu, Yi Zhang, Zequn Zhang

College of Mechanical and Electrical Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210000, China

ARTICLE INFO

Keywords:
Dynamic flexible job-shop scheduling problem
Single dispatching rule
Markov decision process
Deep reinforcement learning
Deep deterministic policy gradient

ABSTRACT

Due to the influence of dynamic changes in the manufacturing environment, a single dispatching rule (SDR) cannot consistently attain better results than other rules for dynamic scheduling problems. Dynamic selection of the most appropriate rule from several SDRs based on the Deep Q-Network (DQN) offers better scheduling performance than using an individual SDR. However, the discreteness of action space caused by the DQN and the simplicity of the action as an SDR limit the selection range and restrict performance improvement. Thus, in this paper, we propose a scheduling method based on deep reinforcement learning for the dynamic flexible job-shop scheduling problem (DFJSP), aiming to minimize the mean tardiness. Firstly, a Markov decision process with composite scheduling action is provided to elaborate the flexible job-shop dynamic scheduling process and transform the DFJSP into an RL task. Subsequently, a composite scheduling action aggregated by SDRs and continuous weight variables is designed to provide a continuous rule space and SDR weight selection. Moreover, a reward function related to mean tardiness performance criteria is designed such that maximizing the cumulative reward is equivalent to minimizing the mean tardiness. Finally, a policy network with states as inputs and weights as outputs is constructed to generate the scheduling decision at each decision point. Also, the deep deterministic policy gradient (DDPG) algorithm is used to train the policy network to select the most appropriate weights at each decision point, thereby aggregating the SDRs into a better rule. Results from numerical experiments reveal that the proposed scheduling method achieves significantly better scheduling results than an SDR and the DQN-based method in dynamically changeable manufacturing environments.

1. Introduction

Production scheduling is an essential core process of production management in manufacturing enterprises that directly affects manufacturing costs and productivity. The adoption of effective scheduling optimization technology improves machine utilization (Pachpor et al., 2017), ensures timely delivery (J. C. Chen et al., 2008), and reduces inventory costs (Hanson et al., 2015). As one of the most widespread issues in this area, the flexible job-shop scheduling problem (FJSP) is an extension of the classical job-shop scheduling problem (JSP), which breaks through the constraints of resource uniqueness and has been studied extensively for decades. In traditional solutions for the FJSP, the optimal schedules of all operations are determined before the jobs enter the production system and remain unmodified throughout the entire working process. However, several dynamic events such as machine failures, the arrival of new jobs, due date changes, and so on inevitably exist in the actual production system. As a result, previously optimal schedules may no longer be appropriate for the changing production system and may even produce poor scheduling performance (Ouelhadj & Petrovic, 2009). Therefore, dynamic scheduling needs to be developed to deal with these sudden events so as to make the production system run continuously and optimally.

The dynamic FJSP (DFJSP) has been widely explored in recent years. Besides, numerous technologies, such as dispatching rules (Montazeri & Vanwassenhove, 1990), genetic algorithm (Pezzella et al., 2008), particle swarm optimization algorithm (Nouiri et al., 2018), and simulated annealing algorithm (Antonio Cruz-Chavez et al., 2017), have been proposed to solve the DFJSP. With the development of personalized product demand, dynamic scheduling algorithms for real-time response are pursued by manufacturing enterprises to improve market competitiveness (Y. Zhang et al., 2022). Compared with other algorithms, dispatching rules possess the advantages of rapid dispatch, high flexibility, and easy implementation, and are employed in various manufacturing enterprises. However, many studies (Baker, 1984; Blackstone et al.,

* Corresponding author at: College of Mechanical and Electrical Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China.
E-mail address: d.tang@nuaa.edu.cn (D. Tang).

https://doi.org/10.1016/j.cie.2023.109255
Received 8 June 2022; Received in revised form 29 March 2023; Accepted 16 April 2023
Available online 21 April 2023

Table 1
The works on dynamic scheduling based on DQN or Q-leaning.

Works	problem	Algorithms	State space	Action space	Action type	objective
(Aydin & Oztemel, 2000)	Job-shop scheduling	Q-learning	Discrete	Discrete	Single dispatching rule	Mean tardiness
(Wang & Usher, 2005)	Single-machine scheduling	Q-learning	Discrete	Discrete	Single dispatching rule	Maximum lateness; Number of tardy jobs; Mean lateness
(Shiue et al., 2018)	Flexible Job-shop scheduling	Q-learning	Continuous	Discrete	Single dispatching rule	Throughput; mean cycle time; Number of tardy jobs
(X. Chen et al., 2010)	Job-shop scheduling	Q-learning	Discrete	Discrete	Composite dispatching rule	Mean flow time; Mean tardiness
(Z. Zhang et al., 2007)	Parallel-machine scheduling	Q-learning	Discrete	Discrete	Single heuristic rule	Mean weighted tardiness
(Luo, 2020)	Flexible job-shop scheduling	DQN	Continuous	Discrete	Single heuristic rule	Total tardiness
(Lin et al., 2019)	Job-shop scheduling	DQN	Continuous	Discrete	Single dispatching rule	Makespan
(Han & Yang, 2020)	Job-shop scheduling	DQN	Continuous	Discrete	Single dispatching rule	Makespan
(Marchesano et al., 2021)	Flow job-shop scheduling	DQN	Continuous	Discrete	Single dispatching rule	Throughput
(Zhao et al., 2021)	Job-shop scheduling	DQN	Continuous	Discrete	Single dispatching rule	Delay time of all jobs

1982; Sabuncuoglu, 1998; Sabuncuoglu & Hommertzheim, 1992) have established that none of the single dispatching rules (SDRs) consistently attains better scheduling results than other rules under various shop configuration conditions. For instance, the mean tardiness (MT) performance of the shortest processing time (SPT) priority rule is effective when the due date is very tight but poor when the due date is loose (Baker, 1984). For changing shop configuration conditions in a dynamic environment, some researchers (Pierreval, 1992; Pierreval & Mebarki, 1997; Wu & Wysk, 1989) have highlighted that the dynamic selection of the most appropriate rule from several SDRs at each decision point may achieve superior performance than using the single dispatching rule (SDR), which is also the research motivation of this paper.

To select the most appropriate SDR at each scheduling decision time point, many studies consider the dynamic scheduling process as a Markov decision process (MDP) and develop a dynamic selection strategy through the reinforcement learning (RL) algorithm. Aydin and Oztemel (Aydin & Oztemel, 2000) developed an improved Q-learning algorithm that trained an intelligent agent to select the most appropriate SDR from three SDRs according to the shop conditions in real-time. Wang and Usher (Wang & Usher, 2005) trained a single-machine agent using the Q-learning algorithm to learn the selection from three SDRs for a single-machine scheduling problem. Shiue et al. (Shiue et al., 2018) proposed an RL-based real-time scheduling mechanism using the Q-learning algorithm that trained the RL agent to select the most appropriate rule from several dispatching rules. Chen et al. (X. Chen et al., 2010) presented a rule-driven method for multi-objective dynamic scheduling that utilized Q-learning to learn the most appropriate weights from the given discrete value space and construct the composite rule. Zhang et al. (Z. Zhang et al., 2007) converted a scheduling problem into an RL problem by constructing a semi-MDP model and used Q-learning to learn the selection from five heuristic rules. In these works, the Q-learning algorithm was used to select the most appropriate dispatching rule at each state by constructing a Q table that displayed the estimated maximum value of the Q function for all discrete state-action pairs. However, actual production systems exist in a complex environment with high-dimensional continuous states that are impossible to list with a Q table.

Aiming to address this issue, deep reinforcement learning (DRL), which integrates deep learning and reinforcement learning, has attracted a great deal of attention. The Deep Q-Network (DQN), as one of the most popular DRLs, directly maps the continuous state to the maximum value of the Q function for each action. Through the Deep Q-Learning algorithm, the DQN can be trained to select the most appropriate action at each decision point and is now widely applied in real-time dynamic scheduling. Luo (Luo, 2020) used the DQN to train an intelligent agent that selected the most appropriate dispatching rule from six predefined composite dispatching rules according to seven continuous state features at each scheduling decision point for the DFJSP. Lin et al. (Lin et al., 2019) proposed an extended DQN method named MDQN that chose one SDR from seven common SDRs for each machine. Besides, Han et al. (Han & Yang, 2020) proposed a DRL scheduling framework based on disjunctive graph dispatching. They developed a dueling double DQN with a prioritized replay that trained an agent to obtain the best rules from eighteen SDRs. Marchesano et al. (Marchesano et al., 2021) adopted the DQN to develop a self-optimizing scheduling policy, selecting the most appropriate SDR from a set of known SDRs according to the system state. Zhao et al. (Zhao et al., 2021) developed an adaptive scheduling algorithm based on the DQN, then designed five state features of continuous value ranges and ten well-known SDRs as the input and action set of the DQN. Table 1 summarizes the above works on real-time dynamic scheduling based on DQN or Q-leaning. We can see that although DQN can handle scheduling problem tasks with continuous state space compared with Q-learning, it can only handle tasks with discrete action space. This directly limits the range of optional actions and may miss better scheduling decisions. Moreover, these works primarily concentrate on choosing an SDR at each scheduling decision point rather than exploring potentially more effective rules.

To improve this situation, in this paper, the weighted aggregation approach of SDRs and the deep deterministic policy gradient (DDPG) algorithm are applied. Compared with an individual SDR, the aggregation of multiple SDRs with appropriate weights offers a better adaptation of the rules to the manufacturing environment (Grabot & Geneste, 1994) and achieves better scheduling performance (Hershauer & Ebert, 1975; Kanet & Li, 2004; Kuczapski et al., 2010). Moreover, weights, as continuous parameters, can be aggregated with SDRs to construct a continuous rule space. DDPG, a policy-gradient-based DRL algorithm, can train an agent to learn the most appropriate action from a continuous action space compared with DQN, having been applied to various scheduling problems. Park et al. (I.-B. Park & Park, 2021) presented a scheduling method based on the DDPG algorithm for scheduling semiconductor operations and designed a continuous representation of an action to maintain the size of the action space even when the numbers of jobs were subject to change. Min et al. (Min & Kim, 2022) proposed a DRL-based apparent tardiness cost (ATC) dispatching rule for single-machine scheduling, in which the DDPG algorithm was used to train a scheduling agent that learned the relationship between raw job data and the continuous lookahead parameter of the ATC equation. Liu et al. (Liu et al., 2020) constructed a model with actor-critic architecture for the JSP and proposed a parallel training method. They combined asynchronous updates as well as DDPG to train the actor network to learn the continuous distributions of probabilities for different SDRs. Chen et al. (C. Chen et al., 2021) proposed a DRL algorithm framework that

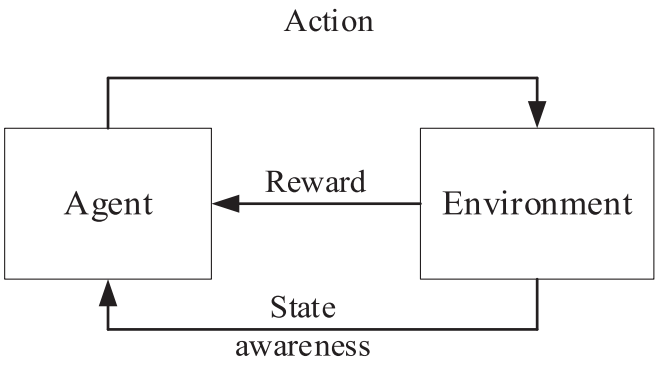


Fig. 1. The interactions between the agent and environment.

combined the DDPG and the convolutional neural network (CNN) for scheduling automated guided vehicles (AGVs) in an automated container terminal. Here, the DDPG was used to train the CNN to achieve optimal matching between the AGV and the container in the decision stage.

With the motivations noted above, in this paper, we train the scheduling agent using the DDPG algorithm to obtain the most appropriate weights at each decision point. This aggregates the SDRs into a better rule and minimizes the mean tardiness for the DFJSP. The main contributions of this paper are: (1) A composite scheduling action aggregated by SDRs and continuous weight variables is designed to provide a continuous rule space and SDR weight selection. By using the DDPG algorithm, a policy network that maps the production system state to the continuous weight variables is trained to select the most appropriate weights at each scheduling decision point for aggregating the SDRs into a better rule; (2) Numerical experiments confirm that our proposed method attains significantly better results than the SDR and the DQN-based scheduling method under most shop configuration conditions.

The remainder of this paper is organized as follows. Section 2 gives the background knowledge of reinforcement learning related to this paper, such as MDP, DQN, and DDPG. Section 3 provides the problem description and mathematical model of the DFJSP. In Section 4, a method based on the DDPG algorithm is proposed for the DFJSP. Numerical experiments that confirm the effectiveness and generality of the proposed method are developed in Section 5. Finally, Section 6 offers the conclusions of this study.

2. Background

2.1. RL and MDP

Reinforcement learning (RL) (Kaelbling et al., 1996) is a particular type of machine learning algorithm. Compared with supervised learning and unsupervised learning, the RL task involves learning behavior through interactions between an agent and a dynamic environment. As Fig. 1 illustrates, an intelligent agent interacts with the environment, performing an action on the environment according to the perceived state and then receiving a reward from the environment. Through continuous interaction, the agent is constantly trained by the RL algorithm to learn the best action policy, thereby maximizing the cumulative reward.

Generally, the RL task can be described as a Markov decision process (MDP), which consists of state space X, action space A, transition function P, reward function R, and discount factor γ. In the MDP, the agent perceives the current state $s_t \in X$ at decision time step t and chooses an action $a_t \in A$ to perform on the environment. Subsequently, the environment enters the next state s_{t+1} from s_t according to P and the agent receives immediate reward $r_t \in R$. All actions of an agent form a policy π and the goal is to find an optimal policy π^* that achieves the maximum cumulative reward.

2.2. Q-learning and DQN

Q-learning (Watkins & Dayan, 1992) is a common algorithm that is used to solve RL tasks in the early stage. It aims to estimate the maximum value $Q_\pi^*(s,a)$ of the state-action value function $Q_\pi(s,a)$ of action a in state s under the execution of policy π, as defined in Eq. (1).

$$Q_\pi^*(s,a) = \max_\pi Q_\pi(s,a) = \max_\pi \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots | s_t = s, a_t = a] \quad (1)$$

where, $Q_\pi(s,a)$, also known as the Q function, represents the cumulative reward expectation after executing action a from state s using policy π. γ is the discount factor that represents time-based penalization. A two-dimensional table named the Q table can be created to save the $Q_\pi^*(s,a)$ values for all discrete state-action pairs. At each decision point, the agent chooses the action with the maximum $Q_\pi^*(s,a)$ value in the current state through the Q table, thus obtaining the optimal policy π^*.

Since the Q table can only store a finite number of $Q_\pi^*(s,a)$ values, Q-learning cannot solve RL tasks with continuous state space. The Deep Q-Network (DQN) proposed by Mnih et al. (Mnih et al., 2013) makes up for this deficiency, fitting the $Q_\pi^*(s,a)$ by a deep neural network with the input as a continuous state and the output as $Q_\pi^*(s,a)$ of each action. Compared to Q-learning, DQN takes advantage of the strong representation ability of deep learning to improve the ability of the agent in RL tasks with continuous state space. At each decision point, the agent can acquire the $Q_\pi^*(s,a)$ of each action in the current state through the trained DQN to choose the action with the maximum $Q_\pi^*(s,a)$ value.

2.3. Ddpg

Although the application of DQN offsets the deficiency of Q-learning, it can only be used to solve RL tasks with discrete actions. The deep deterministic policy gradient (DDPG) algorithm proposed by Lillicrap et al. (Lillicrap et al., 2015) can be used to solve RL tasks with continuous actions. The DDPG algorithm adopts a framework called actor-critic, in which the actor is responsible for a policy network and the critic is responsible for a value network. The policy network is developed to fit the policy function through the deep neural network with a state as the input and a deterministic action as the output. The value network is developed to fit the value function through the deep neural network, where the input is a state and a performed action, while the output is $Q_\pi(s,a)$.

Through the interactive process between the actor and the environment, the samples generated by the interaction are constantly stored in the experience replay memory. In each time step, a small batch of samples is transmitted to the actor and the critic for calculation to continuously train the policy network and the value network. The specific procedure of the algorithm is shown in Algorithm 1.

Algorithm 1 The procedure of the DDPG algorithm

1. Initialize critic value network $Q(s,a;\theta^Q)$ and actor policy network $\mu(s;\theta^\mu)$ with weights θ^Q and θ^μ
2. Initialize critic target value network $Q'(s,a;\theta^{Q'})$ and actor target policy network $\mu'(s;\theta^{\mu'})$ with weights $\theta^{Q'} \leftarrow \theta^Q$ and $\theta^{\mu'} \leftarrow \theta^\mu$
3. Initialize replay memory M
4. **for** episode = 1 to L **do**
5. Initialize a random process
6. Receive initial state s_1
7. **for** $t = 1$ to T **do**
8. Calculate $a_t = \mu(s_t;\theta^\mu) + \mathcal{N}_t$ according to the current state s_t and exploration noise $\mathcal{N}_t$
9. Execute action a_t, then receive reward r_t and new state s_{t+1}
10. Store sample (s_t, a_t, r_t, s_{t+1}) in M
11. Randomly select a minibatch N samples (s_i, a_i, r_i, s_{i+1}) from M
12. Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1};\theta^{\mu'});\theta^{Q'})$

(continued on next page)

(*continued*)

Algorithm 1 The procedure of the DDPG algorithm

13. Update the critic value network by minimizing the loss:$L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i; \theta^Q))^2$
14. Update the actor policy network using the policy gradient method:
$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a; \theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s; \theta^\mu)|_{s_i}$
15. Update the target networks:
$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$
$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$
16. **end for**
17. **end for**

3. Problem description and mathematical model

The DFJSP is defined as follows. There are n jobs $\boldsymbol{J} = \{J_1, J_2, \cdots, J_n\}$ to be processed on m machines $\boldsymbol{M} = \{M_1, M_2, \cdots, M_m\}$. Each job J_i has a definite process route. The j_{th} operation of J_i is denoted by O_{ij}, which can be processed on any machine of the set of compatible machines $\boldsymbol{M_{ij}}(\boldsymbol{M_{ij}} \subset \boldsymbol{M})$. The processing time of operation O_{ij} on machine M_k is denoted by t_{ijk}. The actual start time and completion time of O_{ij} on machine M_k are denoted by S_{ijk} and C_{ijk}, respectively. The due date and actual completion time of the job J_i are denoted by D_i and C_i, respectively. The objective is to minimize mean tardiness by implementing machine selection and operation sequencing at each scheduling decision point. To simplify the problem, some constraints and assumptions are set as follows:

(1) All machines are ready at time 0.
(2) Each operation of a job should be processed in the sequence of the process route.
(3) Each machine can process at most one operation at a time.
(4) Each operation can be assigned to only one machine at a time.
(5) The transportation time and clamping time are not considered.

The notations required for the mathematical model are listed in Table 2.

Based on the above problem definition, constraints, assumptions and notations, the mathematical model can be established as follows.

$$\text{Minimize}\left\{\left(\sum_{i=1}^{n}\max\{C_i - d_i, 0\}\right)\Big/ n\right\} \tag{2}$$

Subject to:

Table 2
Notations of the proposed DFJSP.

Notation	Definition
n	Number of total Jobs
m	Number of total machines
i, i'	index of jobs
j, j'	index of operations
k, k'	index of machines
J_i	The i_{th} job
M_k	The k_{th} machine
$\boldsymbol{M_{ij}}$	The set of compatible machines of O_{ij}
O_{ij}	The i_{th} operation of J_i
t_{ijk}	The processing time of O_{ij} on M_k
S_{ijk}	The start time of O_{ij} on M_k
C_{ijk}	The completion time of O_{ij} on M_k
C_i	The completion time of J_i
D_i	The date time of J_i
δ_{ijk}	$\begin{cases} 1, & \text{if } O_{ij} \text{ is assigned on } M_k \\ 0, & \text{otherwise} \end{cases}$
$\varepsilon_{ij\_i'j'}$	$\begin{cases} 1, & \text{if } O_{ij} \text{ is ahead of } O_{i'j'} \\ 0, & \text{otherwise} \end{cases}$

$$C_{ijk} > S_{ijk} \tag{3}$$

$$S_{ijk} \geq C_{i(j-1)k'} \tag{4}$$

$$\sum_{k \in M_{ij}} \delta_{ijk} = 1 \tag{5}$$

$$\left(S_{ijk}, C_{ijk}\right) \cap \left(S_{i'j'k}, C_{i'j'k}\right) = \varnothing \tag{6}$$

$$\left(S_{ijk} - C_{i'j'k}\right) \bullet \delta_{ijk} \bullet \delta_{i'j'k} \bullet \varepsilon_{ij\_i'j'} \geq 0 \tag{7}$$

where Eq. (2) is the minimization of mean tardiness of all jobs. Eq. (3) indicates that the completion time of each job is after the start time. Eq. (4) represents the processing sequence constraint of a job. Each operation can be assigned to only one machine, and each machine can only process at most one operation at a time are shown in Eq. (5) and Eq. (6), respectively. Eq. (7) indicates that the machine should process the operation in the priority sequence.

4. Proposed method for the DFJSP

In this section, a scheduling method based on the DDPG is proposed for the DFJSP. First, a Markov decision process of DFJSP (MDP-DFJSP) with composite scheduling action is established. Then, definitions of the state features, composite scheduling action, and reward function are successively provided. Finally, the policy network structure is constructed, and the training framework of the DDPG algorithm is developed to train the policy network.

4.1. MDP-DFJSP with composite scheduling action

The scheduling process of DFJSP is a sequential decision-making process that starts with the arrival of the initial jobs and ends with the completion of all jobs. At each scheduling decision point, the scheduling agent makes a scheduling decision according to the current state of the shop production system. Moreover, the next state of the production system is only determined by the current state and the scheduling decision, which conforms to the Markov property. Thus, the scheduling process of DFJSP can be regarded as a Markov decision process for the interaction between the scheduling agent and the shop production system, which is also the motivation for using the reinforcement learning algorithm to solve the DFJSP. In the MDP-DFJSP, the action made by the scheduling agent is a composite scheduling action including machine selection and operation sequencing. Thus, at a certain scheduling decision point, the appropriate machine is selected for candidate jobs and the operation of the idle machines is sequenced. In this study, the composite scheduling action is represented by the machine selection rule and the operation sequencing rule.

As shown in Fig. 2, the MDP-DFJSP with composite scheduling action starts at the initial state s_1 when the initial jobs arrive at the shop. According to s_1, the scheduling agent makes a composite scheduling action a_t (i.e., machine selection rule a_{t1} and operation sequencing rule a_{t2}). Based on a_{t1}, each candidate job selects a machine from the compatible machine family and enters the corresponding buffer. Next, the idle machine acquires the highest priority operation from the buffer based on a_{t2}. After the composite scheduling action is implemented, the machines assigned to the operation begin processing, and the state of the production system is also updated.

When the operation of one job is completed, the next operation of the job requires the choice of an appropriate machine for processing, and the machine that has just finished processing also needs to be assigned a subsequent operation to process. Moreover, when a new job arrives, the first operation of the job also needs to be processed. Thus, the production system reaches a scheduling decision point every time a new job arrives or an operation is completed, and the scheduling agent again

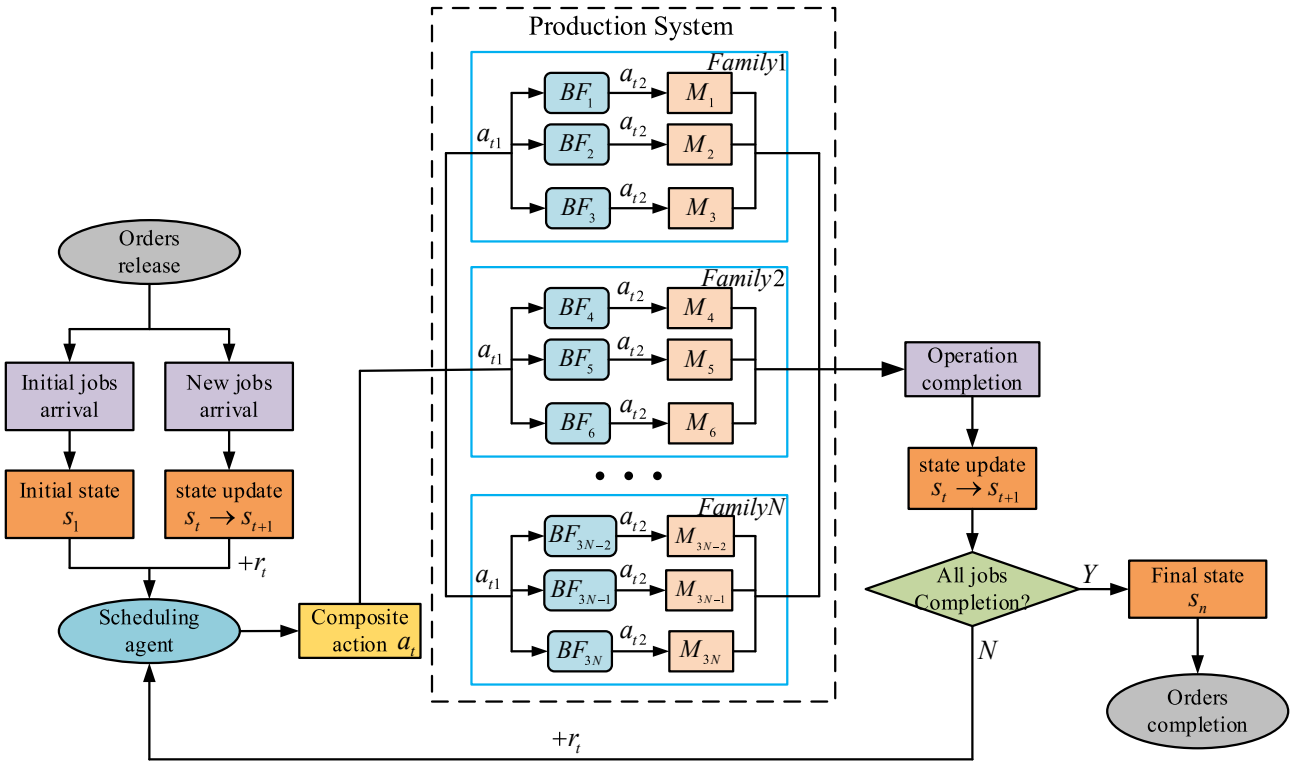


Fig. 2. The MDP-DFJSP with composite scheduling action.

makes a composite scheduling action that interacts with the production system. The interaction process is repeated until all the jobs are completed, the final state s_n is reached, and MDP-DFJSP ends.

Based on the above-mentioned MDP-DFJSP, the DFJSP can be transformed into an RL task, whose key elements are listed in Table 3.

4.2. Definition of state features of the production system

In the DRL, state features represent the dynamically changing environmental information perceived by the agent. These features are the basis for predicting the most appropriate action at each decision point to maximize the cumulative reward. In this study, the action and the reward are related to the dispatching rules and the mean tardiness performance criteria, respectively. Therefore, system state features that can serve as a solid basis for predicting the most appropriate dispatching rule to optimize mean tardiness need to be designed.

Many state features of the production system were designed in early studies (C. C. Chen et al., 1999; Cho & Wysk, 1993; S. C. Park et al., 1997; Priore et al., 2018) that developed scheduling knowledge bases for predicting the most appropriate dispatching rule at each decision point according to the current state, thereby optimizing mean tardiness. Based on this, 20 candidate state features, which are listed in Table 4, are examined in this study.

Table 3
The key elements of the MDP-DFJSP.

Elements	Information
Agent	Schedule agent
Environment	Shop production system
State	Dynamic attributes of the production system
Action	Machine selection rule and operation sequencing rule
Reward	Scheduling Performance criteria

4.3. Definition of composite scheduling action

In most RL-based scheduling methods with dispatching rules as actions, the action space contains only a limited number of SDRs. However, the discreteness of the action space and the simplicity of the action as an SDR directly limit the selection range and restrict performance improvement. Moreover, the aggregation of SDRs with appropriate continuous weight variables values has a better adaptation of the rules to the manufacturing environment and works better than an SDR (Grabot & Geneste, 1994; Hershauer & Ebert, 1975; Kanet & Li, 2004; Kuczapski et al., 2010). Thus, a composite scheduling action aggregated by SDRs and continuous weight variables is designed in this paper to provide

Table 4
State features of the production system in this study.

ID	Description
1	Number of jobs in the system
2	The ratio of the number of late jobs to the total number of jobs in the system
3	The mean number of alternative machines for all candidate jobs
4	The ratio of the number of busy machines to the total number of machines
5	The mean utilization of machines
6	The difference between the maximum and minimum machine utilization
7	The ratio of the standard deviation of machine utilization to the mean machine utilization
8	The mean processing time of candidate jobs in the system
9	The minimum processing time of candidate jobs in the system
10	The minimum remaining time of candidate jobs in the system
11	The mean remaining time of candidate jobs in the system
12	The minimum slack time of candidate jobs in the system
13	The mean slack time of candidate jobs in the system
14	The maximum tardiness of candidate jobs in the system
15	The mean tardiness of candidate jobs in the system
16	The mean interval time of two recent arrival jobs
17	The mean interval time of five recent arrival jobs
18	The mean critical ratio of candidate jobs in the system
19	The ratio of the maximum workload to the mean workload
20	The ratio of the standard deviation of machine workload to the mean machine workload

Table 5
The single dispatching rules of machine selection.

ID	Rules	Description
1	SMPT	Selects the machine with the smallest processing time of the operation.
2	NINQ	Selects the machine with the smallest number of jobs in the buffer.
3	WINQ	Selects the machine with the smallest workload, i.e., the sum of operation processing time of jobs in the buffer.

Table 6
The single dispatching rules of operation sequencing.

ID	Rules	Description
1	SPT	Selects the job with the shortest processing time
2	SRPT	Selects the job with the shortest remaining processing time
3	EDD	Selects the job with the shortest due-date
4	MDD	Selects the job with the minimum modified due-date

continuous rule space and SDR weight selection.

Table 5 and Table 6 respectively list the SDRs for machine selection and operation sequencing from previous works (Doh et al., 2013; Fujimoto et al., 1995; Jeong & Kim, 1998; Priore et al., 2018). These SDRs have been verified to be effective in minimizing mean tardiness. Through the weighted aggregation approach, the priority values of the machine and the operation are shown respectively in Eqs. (8) and (9).

$$PM_k = \sum_{r=1}^{3} (wm_r \bullet zm_{kr}^*); zm_{kr}^* = \frac{zm_{kr}}{\sum_k^m zm_{kr}} \tag{8}$$

$$PO_i = \sum_{r=1}^{4} (wo_r \bullet zo_{ir}^*); zo_{ir}^* = \frac{zo_{ir}}{\sum_i^n zo_{ir}} \tag{9}$$

where PM_k and PO_i are the priority values of machine k and job i, respectively. wm_r and wo_r are the weights of the r_{th} SDR for machine selection and operation sequencing, respectively. zm_{kr} and zo_{ir} are respectively the values of machine k and job i that are calculated using the r_{th} SDR. zm_{kr}^* and zo_{ir}^* are respectively normalized by zm_{kr} and zo_{ir} to ensure that each single rule has a consistent and equal contribution toward the priority value.

As long as the weights $\boldsymbol{w} = (wm_1, wm_2, wm_3, wo_1, wo_2, wo_3, wo_4)$ are determined at each decision point, the composite scheduling action a_t can be performed in the shop production system. The RL task is to select the most appropriate weights at each decision point to obtain the best composite scheduling action.

4.4. Definition of the reward function

When moving from the current state s_t to the next state s_{t+1}, the RL agent receives an immediate reward r_t according to the reward function. Also, the goal of RL is to maximize the cumulative reward accumulated by all immediate rewards. To maximize the cumulative reward and simultaneously optimize the scheduling performance criteria, the design of the reward function should be closely related to the scheduling performance criteria.

Eq. (10) presents an immediate reward function r_t, which is designed based on the mean tardiness performance criteria.

$$r_t = MT_t - MT_{t+1} \tag{10}$$

where MT_t denotes the mean tardiness value of all jobs at decision time step $t(t = 1,2,\cdots,n-1)$. Based on Eq. (10), the cumulative rewards R can be calculated as shown in Eq. (11).

$$R = r_1 + r_2 + \cdots + r_{n-1}$$

$$= (MT_1 - MT_2) + (MT_2 - MT_3) \cdots + (MT_{n-1} - MT_n)$$

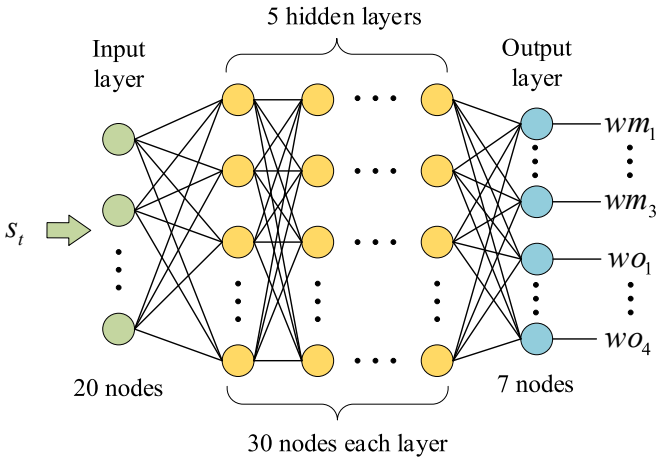


Fig. 3. Policy network structure.

$$= MT_1 - MT_n = 0 - MT = -MT \tag{11}$$

where, MT_1 signifies the mean tardiness in the first decision time step (i.e., when the initial workpiece arrives in the workshop), so $MT_1 = 0$. MT represents the mean tardiness when all jobs are completed. It can be seen that R and MT are opposites, so maximizing R is equivalent to minimizing MT.

In order to calculate the immediate reward at each decision point according to Eq. (10), the calculation procedure of MT_t is provided as shown in Algorithm 2, where $done_i$ denotes whether J_i is finished and num_t denotes the number of jobs that have arrived before decision time step t.

Algorithm 2 The calculation procedure of MT_t

1. Input: $done_i = \begin{cases} 1, if\ J_i\ is\ finished \\ 0, \quad otherwise \end{cases}$, t, D_i, C_i, num_t
2. Initialize the total tardiness$AT_t = 0$ of the jobs that have arrived before the decision time step t
3. **for** $i = 1$ to num_t **do**
4. **if** $done_i$==1 **then**
5. **if** $C_i <= D_i$ **then**
6. $AT_t = AT_t + 0$
7. **else**
8. $AT_t = AT_t + (C_i - D_i)$
9. **end if**
10. **else**
11. **if** $t <= D_i$ **then**
12. $AT_t = AT_t + 0$
13. **else**
14. $AT_t = AT_t + (t - D_i)$
15. **end if**
16. **end if**
17. **end for**
18. $MT_t = \frac{AT_t}{num_t}$
19. Return MT_t

4.5. Policy network structure

In this study, a policy network is constructed to embed in the scheduling agent for generating the weights $\boldsymbol{w} = (wm_1, wm_2, wm_3, wo_1, wo_2, wo_3, wo_4)$ according to the state of the production system at each decision point. As shown in Fig. 3, the policy network is a deep neural network consisting of seven fully connected layers with one input layer, one output layer, and five hidden layers. The numbers of nodes belonging to the input and output layers are the same as the dimension of state space and weights $\boldsymbol{w}$, respectively. The number of nodes in each hidden layer is 30. The "tanh" activation function is used for the hidden layers, and "sigmoid" for the output layer.

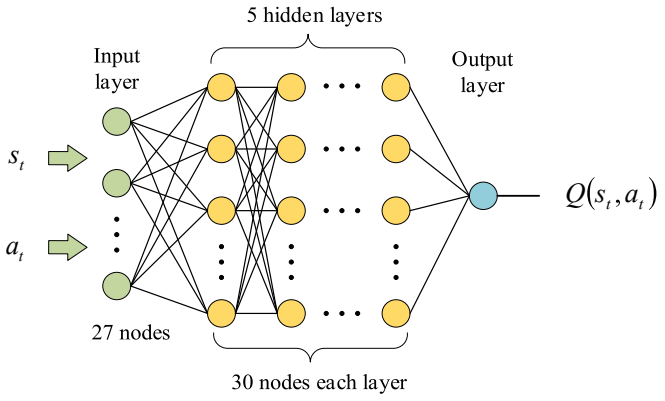


Fig. 4. Value network structure.

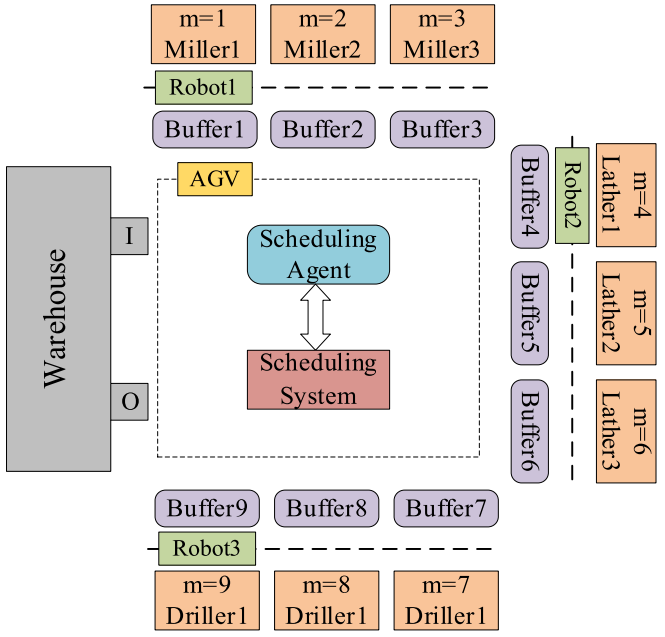


Fig. 5. The layout of a flexible job shop testbed.

4.6. Training framework of the DDPG algorithm

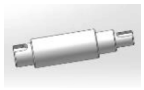
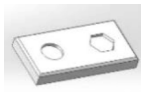
Here, the policy network is trained using the DDPG algorithm to select the most appropriate weights at each decision point. The training framework of the DDPG algorithm is provided in Algorithm 3, where the value network is designed as a deep neural network, as Fig. 4 illustrates. The "ReLU" activation function is used for the hidden layers, and "Linear'' for the output layer. It should be noted that in Algorithm 3, the outputs calculated by the policy network are the weights $\boldsymbol{w}_t$. Therefore, the sample (s_t, $\boldsymbol{w}_t$, r_t, s_{t+1}) should be stored in M to train the policy network.

Algorithm 3 Training framework of the DDPG algorithm

1. Initialize critic value network $Q(s,a;\theta^Q)$ and actor policy network $\mu(s;\theta^\mu)$ with weights θ^Q and θ^μ
2. Initialize critic target value network $Q'(s,a;\theta^{Q'})$ and actor target policy network $\mu'(s;\theta^{\mu'})$ with weights $\theta^{Q'} \leftarrow \theta^Q$ and $\theta^{\mu'} \leftarrow \theta^\mu$
3. Initialize replay memory M
4. **for** episode = 1 to L **do**
5. Initialize the shop production environment and an order to be processed
6. Receive initial state s_1 of the production system
7. **for** t = 1 to T **do**

(*continued on next column*)

Table 7
Jobs information.

Job name	3D sketch	Process route	Operations	Processing time
Shaft		Lathe → Mill	Lathe Mill	Unif[50,100] Unif[10,50]
Plate		Mill	Mill	Unif[50,100]
Flange		Lathe → Mill → Drill	Lathe Mill Drill	Unif [100,150] Unif[50,100] Unif[50,100]

(*continued*)

Algorithm 3 Training framework of the DDPG algorithm

8. Calculate weights $\boldsymbol{w}_t = \mu(s_t;\theta^\mu) + \mathcal{N}_t$ according to the state s_t and exploration noise $\mathcal{N}_t$
9. Aggregate the weights $\boldsymbol{w}_t$ and SDRs into a new dispatching rule a_t
10. Execute a_t in the production system and receive reward r_t and new state s_{t+1}
11. Store sample (s_t, $\boldsymbol{w}_t$, r_t, s_{t+1}) in M
12. Randomly select a minibatch N samples (s_i, $\boldsymbol{w}_i$, r_i, s_{i+1}) from M
13. Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1};\theta^{\mu'});\theta^{Q'})$
14. Update the critic value network by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i,a_i;\theta^Q))$
15. Update the actor policy network using the sampled policy gradient:
$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s,a;\theta^Q)|_{s=s_i,a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s;\theta^\mu)|_{s_i}$
16. Update the target networks:
$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$
$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$
17. **end for**
18. **end for**

5. Numerical experiments

In this section, numerical experiments are developed to confirm the effectiveness of our method for DFJSP. First, the experimental design is explained and then the training details of the DDPG algorithm are provided. Finally, the results of our method are compared with those of the SDRs and the DQN-based method.

5.1. Experimental design

As shown in Fig. 5, a flexible job-shop testbed is established to test the performance of different scheduling methods. There are three machine families, including three mills ($m = 1,2,3$), three lathes ($m = 4,5,6$), and three drills ($m = 7,8,9$), in the manufacturing shop. Each machine has a buffer with sufficient capacity to store the jobs to be processed. The robot and the AGV are used to clamp and transport jobs, respectively. The scheduling agent is embedded in the scheduling system, providing real-time scheduling decisions for the shop throughout the entire working process.

The test instances of orders used in this experiment are randomly generated, and each job of the instance is randomly generated from three types of job: shaft, plate, and flange. Due to the personalized demand of various job sizes, the processing time of the job operation is changing and is set to a uniform distribution in this experiment. Table 7 lists the operations and the processing times of each type of job.

For each instance, it is assumed that some jobs arrive at the initial time, then new jobs arrive successively at an interval. To test the generality of the algorithms under different shop configuration conditions, the number N_i of initial jobs is set to a constant value. Besides, three variables, including the total number N_a of new arrival jobs, arrival interval I_a of a new job, and due date tightness DDT, are controlled to

Table 8
The settings of shop configuration parameters.

Parameter	Value
The number of initial jobs (N_i)	20
The total number of new arrival jobs (N_a)	{20,50,100}
The average value of I_a (θ_a)	{50, 100, 200}
Due date tightness (*DDT*)	{1, 2, 3, 4}

Table 9
Parameters setting of the DDPG algorithm.

Algorithms Parameter	Value
Number of episodes	5000
The learning rate of actor	0.001
The learning rate of critic	0.001
Reward gamma	0.99
Soft replacement ratio tau	0.01
Normal distribution of exploration noise	$N(0, a^2)$, where a decays from 1 by multiplying by 0.99998 at each step
Memory capacity	10,000
Batch size	256

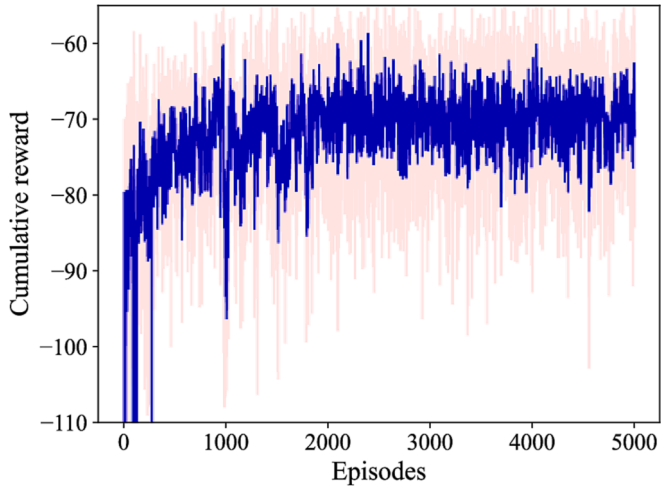


Fig. 6. Cumulative reward in the training process.

change the size of the orders, the workload of the machines and the job due dates, respectively. As a result, various shop configuration conditions are constructed. I_a is subject to exponential distribution and can be sampled by the probability density function $f(I_a) = \begin{cases} \frac{1}{\theta_a} e^{\frac{-I_a}{\theta_a}}, I_a > 0 \\ 0, I_a \leq 0 \end{cases}$, where θ_a is equal to the average value of I_a. Moreover, *DDT* can be used to calculate the job due date D_i by $D_i = A_i + \left(\sum_j^n \frac{\sum_{M_k \in M_{ij}} t_{ijk}}{d_{M_{ij}}} \right) \bullet DDT$, where A_i denotes the arrival time of a job J_i and $d_{M_{ij}}$ signifies the dimension of M_{ij}. The settings of the above parameters are listed in Table 8, where N_a, θ_a, and *DDT* are set to different values to create various shop configuration conditions.

5.2. DDPG training details

The training process of the DDPG algorithm is implemented by Python 3.6 and Tensorflow2.0 on a PC with Intel Core i7-10700 K @ 3.8 GHz CPU, 16 GB RAM, and Windows 10 64-bit operating system. The parameter settings of the DDPG algorithm are listed in Table 9. In this section, the shop configuration parameters N_a, θ_a, and *DDT* are set as 50, 100, and 1, respectively. At the beginning of each training episode, an order instance is randomly generated under the given configuration parameters. Furthermore, the cumulative reward and mean tardiness are recorded at the end of each training episode. After 5000 training episodes, the results of the cumulative reward and mean tardiness of all episodes are respectively shown in Figs. 6 and 7, where the light red curve is the initial result and the blue curve is the result after the moving average.

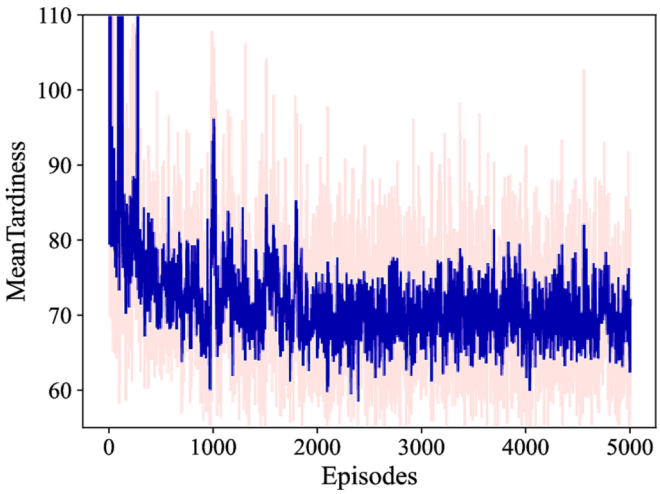


Fig. 7. Mean tardiness in the training process.

As can be seen from Fig. 6, the cumulative reward gradually rises with an increasing number of training episodes and finally converges in a stable range. This shows that the policy network has effectively learned the policy of maximizing the cumulative reward based on the training of the DDPG algorithm. Moreover, Fig. 7 reveals that the mean tardiness gradually drops as the number of training episodes increases. Eventually, the mean tardiness value becomes stable, which suggests that the policy network has learned how to select the most appropriate weights to obtain a weighted combination rule with good performance. By comparing the two figures, we can see that the reward curve and the mean tardiness curve are symmetrical about the *x*-axis, indicating that the mean tardiness is minimized while maximizing the cumulative reward. Therefore, the reward function proposed in Section 4.4 is effective for training the DDPG algorithm.

5.3. Comparisons with the SDRs and DQN-based method

To verify the generality and superiority of our method compared with SDRs and the DQN-based method for DFJSP, we compare the results of our method with those of the 12 SDRs and the DQN-based method under various shop configuration conditions in this section. The 12 SDRs consist of the three machine selection rules and four operation sequencing rules that are mentioned in Section 4.3. The DQN-based method uses the DQN algorithm to select the most appropriate SDR from the 12 SDRs at each decision point, as per the work of Luo (Luo, 2020). The three parameters N_a, θ_a, and *DDT* are set to different values according to Table 8, forming 36 different shop configuration conditions. Under each shop configuration condition, 20 orders generated randomly according to Table 7 are used to test the mean tardiness of each method. The mean values of mean tardiness obtained by each method are listed in Table 10 with the best result highlighted in bold and our method identified with the term “ours”. Comparisons of our method with the SDRs and DQN-based method are provided in the following sections.

5.3.1. Comparisons with the SDRs

The results in Table 10 reveal that the mean tardiness of the 12 SDRs and our method is affected by different shop configuration conditions

Table 10
The mean values of mean tardiness obtained by our method, the DQN-based method, and 12 SDRs under 36 shop configuration conditions.

N_a	θ_a	*DDT*	Ours	DQN	SMPT				NINQ				WINQ			
					SPT	SRPT	EDD	MDD	SPT	SRPT	EDD	MDD	SPT	SRPT	EDD	MDD
20	50	1	**180.4**	194.7	195.7	199.8	219.3	199.6	254.5	260.7	291.2	260.7	260.7	264.5	292.7	264.4
		2	**63.7**	78.6	101.3	100.8	91.3	88.7	131.3	133.2	131.9	127.3	136.9	140.2	135.3	129.8
		3	**12.4**	35.7	47.7	46.0	25.0	26.2	59.9	59.1	36.7	37.4	66.6	62.2	41.0	42.5
		4	**0.1**	6.5	18.7	19.3	5.7	5.6	20.1	16.1	2.8	3.3	28.6	20.2	6.2	6.8
	100	1	126.0	**125.5**	125.9	129.1	137.1	129.0	170.0	171.3	174.3	171.3	167.8	171.5	178.6	171.4
		2	**39.3**	49.8	57.1	58.4	56.2	54.7	71.4	72.1	71.2	70.5	73.6	74.1	75.9	72.4
		3	**7.2**	12.6	20.1	19.0	16.0	16.3	19.2	17.4	12.6	12.6	25.0	24.5	20.0	19.0
		4	**0.4**	1.9	5.3	4.8	3.0	3.0	3.3	1.9	1.2	1.2	8.4	7.4	5.2	5.3
	200	1	**101.1**	107.9	107.9	110.6	114.0	110.5	134.8	136.7	138.3	136.7	131.8	137.5	142.1	137.4
		2	**33.9**	43.6	43.8	44.8	44.1	44.1	57.4	57.7	56.6	54.6	55.7	59.7	59.0	56.8
		3	**3.4**	9.9	13.8	13.5	11.1	11.3	13.3	11.6	10.0	10.0	17.1	17.1	15.1	13.1
		4	**0.1**	1.2	3.8	3.7	1.9	2.0	2.8	1.7	1.3	1.3	7.4	6.0	5.5	4.9
50	50	1	**108.3**	135.4	135.4	138.1	156.7	138.0	179.6	181.0	195.8	181.2	169.8	170.7	191.6	170.7
		2	**42.9**	45.8	73.1	77.5	73.6	68.0	83.5	84.5	84.4	77.2	75.5	76.1	78.9	73.9
		3	**11.2**	22.1	41.0	43.3	29.0	28.8	40.4	41.6	25.2	25.1	35.7	35.1	22.8	22.9
		4	**0.8**	3.1	21.7	21.3	9.7	9.9	15.8	17.2	3.7	4.0	13.0	12.6	4.5	5.0
	100	1	**73.3**	75.1	78.8	80.3	86.2	80.3	104.1	105.9	110.5	106.0	103.3	105.7	110.2	105.7
		2	**23.4**	36.0	39.1	40.2	41.2	37.8	41.6	42.4	42.8	41.1	41.3	42.4	40.8	39.3
		3	**4.7**	9.9	17.7	17.2	15.7	14.9	11.4	11.7	9.7	9.7	12.8	12.3	8.9	9.3
		4	1.6	1.5	5.9	5.4	3.8	3.8	2.1	1.5	1.0	**0.7**	2.8	2.8	1.6	1.8
	200	1	**62.6**	66.7	66.7	68.4	71.3	68.4	81.2	81.3	84.3	81.5	77.8	78.7	81.4	78.7
		2	**20.9**	30.8	30.9	31.3	31.7	31.0	34.1	34.1	33.8	32.9	31.1	31.4	32.3	31.4
		3	**2.4**	8.8	12.0	11.8	10.3	10.6	8.3	7.2	6.2	6.2	7.6	6.9	6.6	6.5
		4	**0.4**	1.8	3.5	2.7	1.9	1.8	1.8	0.8	0.7	0.7	2.5	2.0	1.4	1.5
100	50	1	**76.9**	93.7	93.7	95.7	110.3	95.6	129.4	129.0	139.8	129.1	132.8	131.1	141.6	131.1
		2	**27.4**	42.4	49.9	50.1	51.2	44.8	51.7	50.3	51.6	48.2	52.9	51.4	52.7	49.4
		3	**4.1**	19.7	29.2	28.1	20.2	20.5	23.6	21.6	13.0	13.6	23.1	21.3	13.8	14.4
		4	**0.4**	1.6	16.0	13.8	4.3	5.0	8.0	6.2	2.0	1.9	8.0	6.9	1.5	1.5
	100	1	**44.2**	48.1	48.2	49.2	52.5	49.2	72.5	73.3	76.6	73.4	73.2	74.8	76.7	74.8
		2	**15.3**	20.9	23.4	23.7	23.5	22.5	25.2	25.1	25.7	24.8	25.2	25.6	26.0	24.5
		3	**2.2**	4.2	9.6	8.8	7.3	7.5	7.8	6.9	5.3	5.2	6.8	7.4	5.8	5.9
		4	**0.6**	1.2	3.1	1.7	1.0	1.0	1.6	1.2	**0.6**	**0.6**	0.9	1.0	0.8	0.8
	200	1	**36.0**	38.0	38.0	39.0	40.1	39.0	50.0	50.3	51.4	50.4	49.9	51.0	52.0	51.0
		2	17.0	17.0	17.0	17.3	17.0	**16.8**	19.7	19.9	19.7	19.3	19.6	20.4	19.9	18.8
		3	**1.8**	3.0	6.0	5.7	4.7	5.0	4.9	4.7	3.9	3.7	4.6	4.9	3.7	3.4
		4	0.6	0.8	1.7	0.9	0.5	0.6	0.9	1.1	0.6	0.6	0.9	0.7	**0.4**	**0.4**

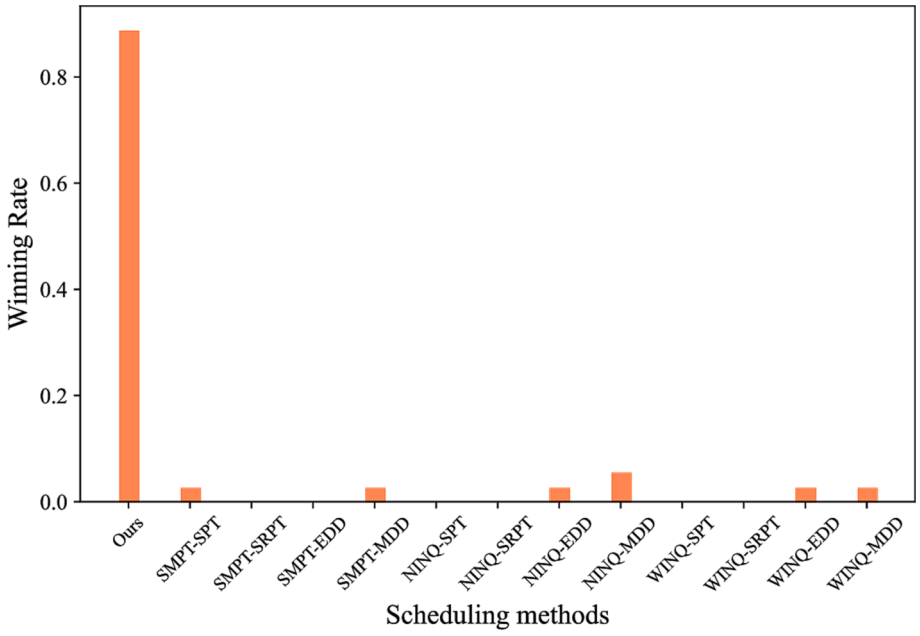


Fig. 8. The winning rates of the 12 SDRs and our method under 36 shop configuration conditions.

and decreases with a higher number of new jobs, longer arrival intervals of new jobs, and looser due dates. Based on the findings in the Appendix, none of the 12 SDRs can attain the best performance under all shop configuration conditions or even one-third of the shop configuration conditions. To test the generality of our method compared with the SDRs, the comparisons with the SDRs are provided as follows.

Based on the results listed in Table 10, the winning rate of the 12 SDRs and our method under 36 shop configuration conditions are shown

Table 11
The performance rankings of the12 SDRs and our method under each shop configuration condition.

N_a	θ_a	*DDT*	Ours	SMPT				NINQ				WINQ			
				SPT	SRPT	EDD	MDD	SPT	SRPT	EDD	MDD	SPT	SRPT	EDD	MDD
20	50	1	**1**	**2**	4	**5**	**3**	6	7	12	7	7	11	**13**	**10**
		2	**1**	5	4	3	2	8	10	9	6	12	13	11	7
		3	**1**	9	8	2	**3**	11	10	4	5	13	**12**	6	7
		4	**1**	9	10	**5**	4	11	**8**	**2**	3	13	**12**	6	7
	100	1	2	1	4	**5**	**3**	7	**8**	12	8	**6**	11	**13**	**10**
		2	**1**	4	5	3	2	8	9	7	6	11	**12**	**13**	**10**
		3	**1**	11	7	4	5	**9**	6	**2**	**2**	13	**12**	10	7
		4	**1**	10	8	**5**	5	7	4	**2**	**2**	13	**12**	9	**10**
	200	1	**1**	**2**	4	**5**	**3**	7	**8**	12	8	**6**	11	**13**	**10**
		2	**1**	**2**	5	3	**3**	10	11	8	6	7	13	12	9
		3	**1**	10	9	4	5	8	6	**2**	**2**	12	**12**	11	7
		4	**1**	9	8	**5**	6	7	4	**2**	**2**	13	**12**	11	**10**
50	50	1	**1**	**2**	4	**5**	**3**	**9**	10	13	11	**6**	7	12	7
		2	**1**	3	9	4	2	11	13	12	8	**6**	7	10	5
		3	**1**	11	13	7	6	10	12	5	4	9	8	2	3
		4	**1**	13	**12**	6	7	10	11	**2**	3	9	8	4	5
	100	1	**1**	**2**	3	**5**	**3**	7	10	13	11	**6**	8	12	8
		2	**1**	3	5	8	2	10	11	13	7	9	11	6	4
		3	**1**	13	**12**	11	10	6	7	4	4	9	8	2	3
		4	4	13	**12**	10	10	7	3	**2**	1	8	8	4	6
	200	1	**1**	**2**	3	**5**	**3**	**9**	10	13	12	**6**	7	11	7
		2	**1**	**2**	5	8	**3**	12	12	11	10	4	6	9	6
		3	**1**	13	**12**	10	11	**9**	7	**2**	**2**	8	6	5	4
		4	**1**	13	**12**	9	7	7	4	**2**	**2**	11	10	5	6
100	50	1	**1**	**2**	4	**5**	**3**	8	6	12	7	11	9	**13**	9
		2	**1**	5	6	8	2	11	7	10	3	13	9	12	4
		3	**1**	13	**12**	6	7	11	9	**2**	3	10	8	4	5
		4	**1**	13	**12**	6	7	10	**8**	5	4	10	9	2	2
	100	1	**1**	**2**	3	**5**	**3**	6	**8**	12	9	7	10	**13**	**10**
		2	**1**	3	5	4	2	**9**	**8**	12	7	9	11	**13**	6
		3	**1**	13	**12**	8	10	11	7	3	**2**	**6**	9	4	5
		4	**1**	13	**12**	7	7	11	10	1	1	**6**	7	4	4
	200	1	**1**	**2**	3	**5**	**3**	7	**8**	12	9	**6**	10	**13**	**10**
		2	2	**2**	5	2	1	**9**	11	9	7	8	13	11	6
		3	**1**	13	**12**	7	11	**9**	7	5	3	**6**	9	3	2
		4	4	13	9	3	4	**9**	12	4	4	9	8	1	1

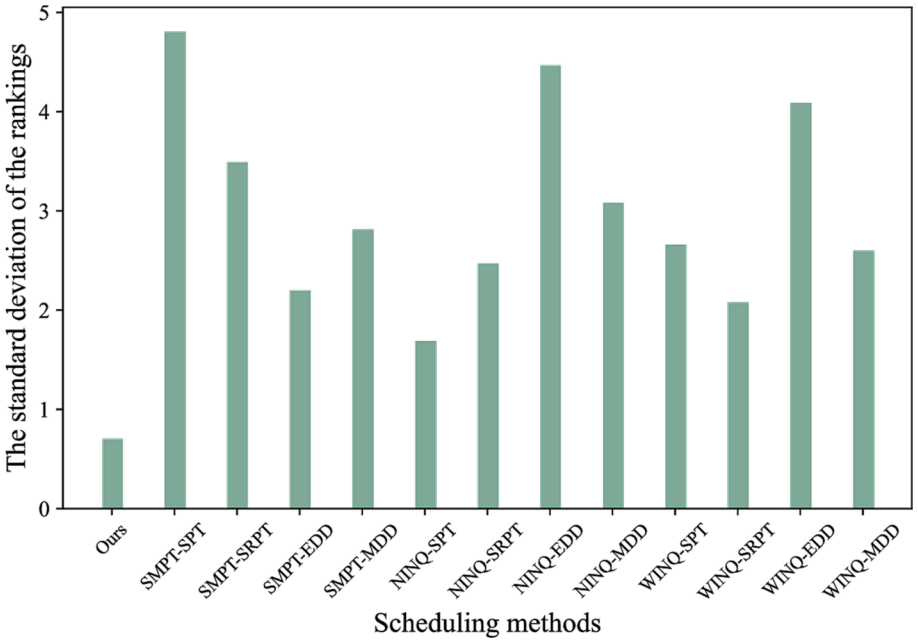


Fig. 9. The standard deviation of the performance rankings under 36 shop configuration conditions.

in Fig. 8. Here, the winning rate of a method is defined as the number of shop configuration conditions in which the method obtains the best performance divided by 36. Fig. 8 reveals that our method achieves the best results under more than 85 % of the shop configuration conditions. In contrast, each SDR attains the best results under less than 0.06 % of the shop configuration conditions. This indicates that our method can achieve better results than the SDRs under most shop configuration conditions.

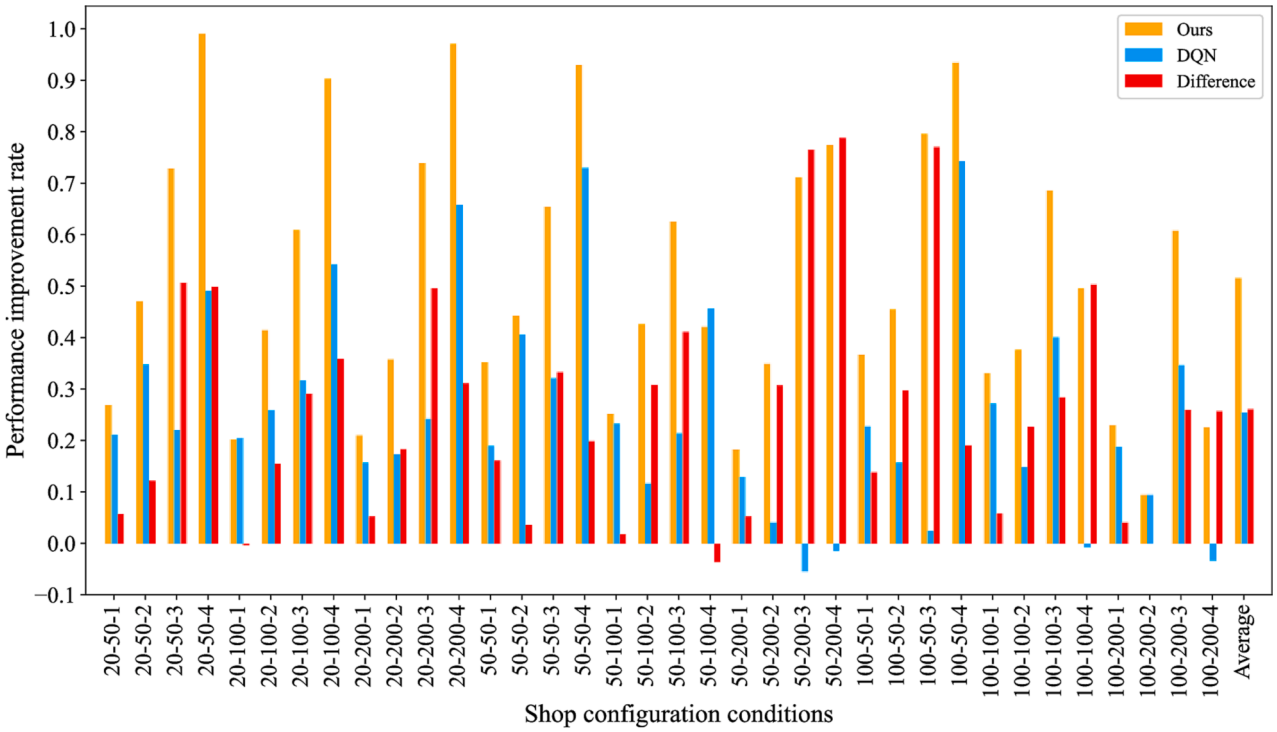


Fig. 10. The performance improvement rates of our method and the DQN-based method compared with $P_{\overline{R}}$.

Table 10 also reveals that the shop configuration conditions where our method failed to obtain the best results include 20–100-1, 50–100-4, 100–200-2, and 100–200-4. These configuration conditions have a common feature, which is that they all have a long arrival interval for new jobs. This indicates that our method may exhibit poorer performance than an SDR under certain shop configuration conditions with a long arrival interval of new jobs.

To test the performance stability of our method and the SDRs under various shop configuration conditions, the performances of these methods are ranked under each shop configuration condition, where a lower number represents better performance. The ranking results displayed in Table 11 suggest that none of these methods consistently maintains a stable performance ranking under all shop configuration conditions. By highlighting in bold the most frequently occurring ranking for each method in Table 11, we observe that our method maintains a stable performance ranking under 32 of the 36 shop configuration conditions. Conversely, each of the SDRs maintains a stable performance ranking under no more than 12 of the 36 shop configuration conditions. This confirms that there does not exist any SDR, as our method, can maintain stable performance under most shop configuration conditions. Moreover, Fig. 9 presents the standard deviation of the performance rankings for each method under the 36 shop configuration conditions. It reveals that our method has the smallest standard deviation of all the methods, which indicates that our method can achieve more stable performance than the SDRs under various shop configuration conditions.

5.3.2. *Comparisons with the DQN-based method*

According to the performance results of our method and the DQN-based method in Table 10, we can see that our method outperforms the DQN-based method in 34 of the 36 shop configuration conditions. This confirms that our method can obtain better performance than the DQN-based method under most shop configuration conditions.

Additionally, to test the performance improvement of our method and the DQN-based method compared with the SDR, the average performance $P_{\overline{R}}$ of the 12 SDRs is used as the reference standard. The performance improvement rates of our method and the DQN-based method compared with $P_{\overline{R}}$ under each shop configuration condition are shown in Fig. 10. The performance improvement rate R_m of a method is calculated by $R_m = \frac{P_m - P_{\overline{R}}}{P_{\overline{R}}}$, where P_m is the performance of the method. Meanwhile, the difference between the performance improvement rates of our method and the DQN-based method is also shown in Fig. 10.

It reveals that our method exhibits superior performance improvement rates under 34 shop configuration conditions and poorer rates under two shop configuration conditions, compared with the DQN-based method. Also, the difference between the performance improvement rates of our method and the DQN-based method exceeds 10 % under 26 of the 34 shop configuration conditions. However, under the two shop configuration conditions where the improvement rates of our method are poorer, the difference is less than 5 %. This suggests that the performance improvement using our method is significantly higher than that of the DQN-based method under most shop configuration conditions and only slightly lower under a few conditions. Besides, from the averaging of differences under all shop configuration conditions in Fig. 10, we observe that the performance improvement rate of our method exceeds that of the DQN-based method by more than 20 %. This also reflects the more significant performance improvement of our method compared with the DQN-based method overall.

6. Conclusion and future work

In this paper, we study a scheduling method based on deep reinforcement learning for the DFJSP that aims to minimize mean tardiness. Here, the DDPG algorithm is used to train the policy network to select the most appropriate weights at each decision point, thereby aggregating SDRs into a better rule.

Based on the results of the numerical experiments, we can reach the following conclusions.

(1) None of the SDRs for the DFJSP can attain better scheduling results than other rules under various shop configuration conditions.

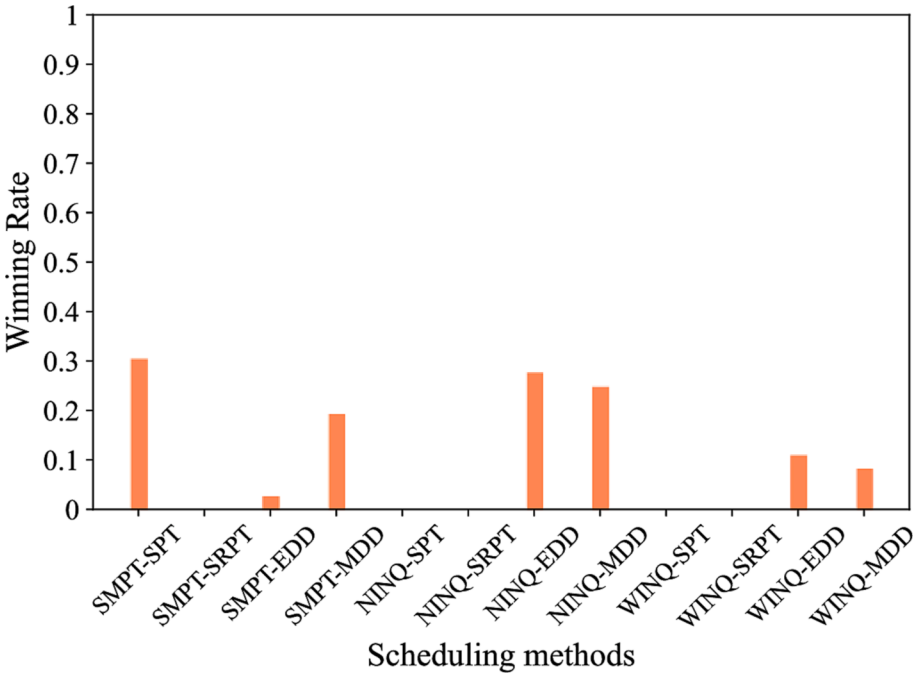


Fig. 11. The winning rates of the 12 SDRs under 36 shop configuration conditions.

(2) The method based on the DDPG algorithm can select the most appropriate weights at each decision point from continuous weights space to aggregate SDRs into a better rule for the DFJSP.
(3) Our method can not only achieve better scheduling results than the SDR and the DQN-based method under most shop configuration conditions but also improve the performance more significantly than the DQN-based method under most shop configuration conditions.

In future work, research on the following problems will be carried out.

(1) In this paper, only changes in due date, order size, and arrival interval of new jobs in the shop environment are considered. In future research, we will consider more dynamic factors, such as machine failure.
(2) For the DFJSP, when multiple machines or jobs need to make decisions simultaneously at a scheduling decision point, the method in this paper gives them the same dispatching rules. This may not exhibit better performance than independent decision-making rules for each machine or job. Thus, we will study the DFJSP with multi-agent deep reinforcement learning, thereby training each machine or job to independently select rules at each decision point.
(3) In this paper, we only study the weighted aggregation of SDRs. Some researchers (Nie et al., 2010; Pickardt et al., 2013; Su et al., 2013) use genetic programming (GP) for the aggregation of multiple SDRs, which widely explores the solution space of new rules and achieves significant improvement compared with the SDRs. Considering that the GP-based aggregation method has not been used in the dynamic environment nor been compared with the weighted aggregation method in existing related works, we will adopt the GP-based aggregation method in the action design.

CRediT authorship contribution statement

Yong Gui: Conceptualization, Methodology, Software, Investigation, Data curation, Formal analysis, Resources, Validation, Visualization, Writing – original draft, Writing – review & editing. **Dunbing Tang:** Project administration, Funding acquisition, Writing – review & editing. **Haihua Zhu:** Supervision, Funding acquisition. **Yi Zhang:** Conceptualization, Visualization. **Zequn Zhang:** Conceptualization.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgements

We thank the editors and the anonymous reviewers for their valuable and helpful comments in enhancing the quality of the paper. This work was supported by the National Natural Science Foundation of China [grant number 52075257], Key Research and Development Program of Jiangsu Province [grant number BE2021091] and the National Key Research and Development Program of China [grant number 2020YFB1710500].

Appendix

To verify the limitation of an SDR for the DFJSP under different shop configuration conditions and reflect the research motivation of this paper, the proposed 12 SDRs are compared and analyzed. Based on the results presented in Table 10, the winning rates of the 12 SDRs under 36 shop configuration conditions are shown in Fig. 11. Here, the winning rate of an SDR is defined as the number of shop configuration conditions in which the SDR obtains the best performance divided by 36. It can be seen that none of the 12 SDRs attains the best performance under all shop configuration conditions or even under one-third of the shop configuration conditions. This confirms that none of the SDRs consistently attains better performance than other rules under various shop configuration conditions.

References

Antonio Cruz-Chavez, M., Martinez-Rangel, M. G., & Cruz-Rosales, M. H. (2017). Accelerated simulated annealing algorithm applied to the flexible job shop scheduling problem. *International Transactions in Operational Research, 24*(5), 1119–1137.

Aydin, M. E., & Oztemel, E. (2000). Dynamic job-shop scheduling using reinforcement learning agents. *Robotics and Autonomous Systems, 33*(2–3), 169–178.

Baker, K. R. (1984). Sequencing rules and due-date assignments in a job shop. *Management Science, 30*(9), 1093–1104.

Blackstone, J. H., Phillips, D. T., & Hogg, G. L. (1982). A state-of-the-art survey of dispatching rules for manufacturing job shop operations. *International Journal of Production Research, 20*(1), 27–45.

Chen, C., Hu, Z.-H., & Wang, L. (2021). Scheduling of AGVs in Automated Container Terminal Based on the Deep Deterministic Policy Gradient (DDPG) Using the Convolutional Neural Network (CNN). *Journal of Marine Science and Engineering, 9*(12).

Chen, C. C., Yih, Y., & Wu, Y. C. (1999). Auto-bias selection for developing learning-based scheduling systems. *International Journal of Production Research, 37*(9), 1987–2002.

Chen, J. C., Chen, K. H., Wu, J. J., & Chen, C. W. (2008). A study of the flexible job shop scheduling problem with parallel machines and reentrant process. *International Journal of Advanced Manufacturing Technology, 39*(3–4), 344–354.

Chen, X., Hao, X., Lin, H. W., & Murata, T. (2010). Rule driven multi objective dynamic scheduling by data envelopment analysis and reinforcement learning. In *In: 2010 IEEE International Conference on Automation and Logistics* (pp. 396–401). IEEE.

Cho, H., & Wysk, R. A. (1993). A ROBUST ADAPTIVE SCHEDULER FOR AN INTELLIGENT WORKSTATION CONTROLLER. *International Journal of Production Research, 31*(4), 771–789.

Doh, H.-H., Yu, J.-M., Kim, J.-S., Lee, D.-H., & Nam, S.-H. (2013). A priority scheduling approach for flexible job shops with multiple process plans. *International Journal of Production Research, 51*(12), 3748–3764.

Fujimoto, H., Yasuda, K., Tanigawa, Y., Iwahashi, K. & Ieee. (1995). Applications of genetic algorithm and simulation to dispatching rule-based FMS scheduling. In: 1995 IEEE International Conference on Robotics and Automation (pp. 190-195). Nagoya, Japan.

Grabot, B., & Geneste, L. (1994). DISPATCHING RULES IN SCHEDULING - A FUZZY APPROACH. *International Journal of Production Research, 32*(4), 903–915.

Han, B.-A., & Yang, J.-J. (2020). Research on Adaptive Job Shop Scheduling Problems Based on Dueling Double DQN. *Ieee Access, 8*, 186474–186495.

Hanson, B. B., Hodgson, T. J., Kay, M. G., King, R. E., & Thoney-Barletta, K. A. (2015). On the economic lot scheduling problem: Stock-out prevention and system feasibility. *International Journal of Production Research, 53*(16), 4903–4916.

Hershauer, J. C., & Ebert, R. J. (1975). Search and simulation selection of a job-shop sequencing rule. *Management Science, 21*(7), 833–843.

Jeong, K. C., & Kim, Y. D. (1998). A real-time scheduling mechanism for a flexible manufacturing system: Using simulation and dispatching rules. *International Journal of Production Research, 36*(9), 2609–2626.

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research, 4*, 237–285.

Kanet, J. J., & Li, X. M. (2004). A weighted modified due date rule for sequencing to minimize weighted tardiness. *Journal of Scheduling, 7*(4), 261–276.

Kuczapski, A. M., Micea, M. V., Maniu, L. A., & Cretu, V. I. (2010). EFFICIENT GENERATION OF NEAR OPTIMAL INITIAL POPULATIONS TO ENHANCE GENETIC ALGORITHMS FOR JOB-SHOP SCHEDULING. *Information Technology and Control, 39*(1), 32–37.

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D. & Wierstra, D.J.a.p.a. (2015). Continuous control with deep reinforcement learning.

Lin, C.-C., Deng, D.-J., Chih, Y.-L., & Chiu, H.-T. (2019). Smart Manufacturing Scheduling With Edge Computing Using Multiclass Deep Q Network. *Ieee Transactions on Industrial Informatics, 15*(7), 4276–4284.

Liu, C.-L., Chang, C.-C., & Tseng, C.-J. (2020). Actor-Critic Deep Reinforcement Learning for Solving Job Shop Scheduling Problems. *Ieee Access, 8*, 71752–71762.

Luo, S. (2020). Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning. *Applied Soft Computing, 91*.

Marchesano, M.G., Guizzi, G., Santillo, L.C. & Vespoli, S. (2021). Dynamic Scheduling in a Flow Shop Using Deep Reinforcement Learning. In: A. Dolgui, A. Bernard, D. Lemoine, G. VonCieminski & D. Romero, Advances in Production Management Systems: Artificial Intelligence for Sustainable and Resilient Production Systems, Apms 2021, Pt I (Vol. 630, pp. 152-160).

Min, B., & Kim, C. O. (2022). State-Dependent Parameter Tuning of the Apparent Tardiness Cost Dispatching Rule Using Deep Reinforcement Learning. *Ieee Access, 10*, 20187–20198.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. & Riedmiller, M. (2013). Playing atari with deep reinforcement learning.

Montazeri, M., & Vanwassenhove, L. N. (1990). ANALYSIS OF SCHEDULING RULES FOR AN FMS. *International Journal of Production Research, 28*(4), 785–802.

Nie, L., Shao, X., Gao, L., & Li, W. (2010). Evolving scheduling rules with gene expression programming for dynamic single-machine scheduling problems. *International Journal of Advanced Manufacturing Technology, 50*(5–8), 729–747.

Nouiri, M., Bekrar, A., Jemai, A., Niar, S., & Ammari, A. C. (2018). An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem. *Journal of Intelligent Manufacturing, 29*(3), 603–615.

Ouelhadj, D., & Petrovic, S. (2009). A survey of dynamic scheduling in manufacturing systems. *Journal of Scheduling, 12*(4), 417–431.

Pachpor, P. S., Shrivastava, R. L., Seth, D., & Pokharel, S. (2017). Application of Petri nets towards improved utilization of machines in job shop manufacturing environments. *Journal of Manufacturing Technology Management, 28*(2), 169–188.

Park, I.-B., & Park, J. (2021). Scalable Scheduling of Semiconductor Packaging Facilities Using Deep Reinforcement Learning. Ieee Transactions on. *Cybernetics.*

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, 13*(4), 486–502.

Pezzella, F., Morganti, G., & Ciaschetti, G. (2008). A genetic algorithm for the Flexible Job-shop Scheduling Problem. *Computers & Operations Research, 35*(10), 3202–3212.

Pickardt, C. W., Hildebrandt, T., Branke, J., Heger, J., & Scholz-Reiter, B. (2013). Evolutionary generation of dispatching rule sets for complex dynamic scheduling problems. *International Journal of Production Economics, 145*(1), 67–77.

Pierreval, H. (1992). EXPERT SYSTEM FOR SELECTING PRIORITY RULES IN FLEXIBLE MANUFACTURING SYSTEMS. *Expert Systems with Applications, 5*(1–2), 51–57.

Pierreval, H., & Mebarki, N. (1997). Dynamic selection of dispatching rules for manufacturing system scheduling. *International Journal of Production Research, 35*(6), 1575–1591.

Priore, P., Ponte, B., Puente, J., & Gomez, A. (2018). Learning-based scheduling of flexible manufacturing systems using ensemble methods. *Computers & Industrial Engineering, 126*, 282–291.

Sabuncuoglu, I. (1998). A study of scheduling rules of flexible manufacturing systems: A simulation approach. *International Journal of Production Research, 36*(2), 527–546.

Sabuncuoglu, I., & Hommertzheim, D. L. (1992). EXPERIMENTAL INVESTIGATION OF FMS MACHINE AND AGV SCHEDULING RULES AGAINST THE MEAN FLOW-TIME CRITERION. *International Journal of Production Research, 30*(7), 1617–1635.

Shiue, Y.-R., Lee, K.-C., & Su, C.-T. (2018). Real-time scheduling for a smart factory using a reinforcement learning approach. *Computers & Industrial Engineering, 125*, 604–614.

Su, N., Zhang, M., Johnston, M., & Tan, K. C. (2013). A Computational Study of Representations in Genetic Programming to Evolve Dispatching Rules for the Job Shop Scheduling Problem. *Ieee Transactions on Evolutionary Computation, 17*(5), 621–639.

Wang, Y. C., & Usher, J. M. (2005). Application of reinforcement learning for agent-based production scheduling. *Engineering Applications of Artificial Intelligence, 18*(1), 73–82.

Watkins, C., & Dayan, P. (1992). Q-LEARNING. *Machine Learning, 8*(3–4), 279–292.

Wu, S. Y. D., & Wysk, R. A. (1989). An application of discrete-event simulation to on-line control and scheduling in flexible manufacturing. *International Journal of Production Research, 27*(9), 1603–1623.

Zhang, Y., Zhu, H., Tang, D., Zhou, T., & Gui, Y. (2022). Dynamic job shop scheduling based on deep reinforcement learning for multi-agent manufacturing systems. *Robotics and Computer-Integrated Manufacturing, 78*.

Zhang, Z., Zheng, L., & Weng, M. X. (2007). Dynamic parallel machine scheduling with mean weighted tardiness objective by Q-Learning. *International Journal of Advanced Manufacturing Technology, 34*(9–10), 968–980.

Zhao, Y., Wang, Y., Tan, Y., Zhang, J., & Yu, H. (2021). Dynamic Jobshop Scheduling Algorithm Based on Deep Q Network. *Ieee Access, 9*, 122995–123011.