Applied Soft Computing Journal 91 (2020) 106208

Contents lists available at ScienceDirect

Applied Soft Computing Journal

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

Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning

Shu Luo

National Engineering Research Center for Computer Integrated Manufacturing Systems, Department of Automation, Tsinghua University, Beijing 100084, China

ARTICLE INFO

Article history:
Received 19 June 2019
Received in revised form 30 December 2019
Accepted 2 March 2020
Available online 6 March 2020

Keywords:
Flexible job shop scheduling
New job insertion
Dispatching rules
Deep reinforcement learning
Deep Q network

ABSTRACT

In modern manufacturing industry, dynamic scheduling methods are urgently needed with the sharp increase of uncertainty and complexity in production process. To this end, this paper addresses the dynamic flexible job shop scheduling problem (DFJSP) under new job insertions aiming at minimizing the total tardiness. Without lose of generality, the DFJSP can be modeled as a Markov decision process (MDP) where an intelligent agent should successively determine which operation to process next and which machine to assign it on according to the production status of current decision point, making it particularly feasible to be solved by reinforcement learning (RL) methods. In order to cope with continuous production states and learn the most suitable action (i.e. dispatching rule) at each rescheduling point, a deep Q-network (DQN) is developed to address this problem. Six composite dispatching rules are proposed to simultaneously select an operation and assign it on a feasible machine every time an operation is completed or a new job arrives. Seven generic state features are extracted to represent the production status at a rescheduling point. By taking the continuous state features as input to the DQN, the state–action value (Q-value) of each dispatching rule can be obtained. The proposed DQN is trained using deep Q-learning (DQL) enhanced by two improvements namely double DQN and soft target weight update. Moreover, a "softmax" action selection policy is utilized in real implementation of the trained DQN so as to promote the rules with higher Q-values while maintaining the policy entropy. Numerical experiments are conducted on a large number of instances with different production configurations. The results have confirmed both the superiority and generality of DQN compared to each composite rule, other well-known dispatching rules as well as the stand Q-learning-based agent.

© 2020 Elsevier B.V. All rights reserved.

1. Introduction

The flexible job shop scheduling problem (FJSP), acting as a high abstraction of modern production environment such as semiconductor manufacturing process, automobile assembly process and mechanical manufacturing systems [1], has been intensively studied over the past decades. Compared to the classical job shop scheduling problem which has been proved to be NP-hard [2], the FJSP is more intractable since each operation can be assigned on one or more available machines. To date, most of existing methods for solving the FJSP have assumed a static manufacturing environment where the information of shopfloor is completely known in advance, hence outputting a deterministic scheduling scheme without any modification during the entire working process. However, in today's complex and varying manufacturing systems, dynamic events such as the insertions, cancellations or modifications of orders, machine breakdowns, variations in processing times and so on, are inevitable to be considered [3]. These disturbances distract the realized execution of a static schedule far from its expected outcome and deteriorate the production efficiency seriously. Therefore, it is of remarkable importance to develop on-line scheduling methods for the dynamic FJSP (DFJSP) so as to handle uncertain events in real time.

As an attractive research field in both academia and industry, dynamic scheduling has been intensively researched over the past decades. Various of methods have been presented, the most widely used ones among which are dispatching rules and metaheuristics [4]. Dispatching rules immediately react to dynamic events thus achieving the best time efficiency. However, they fail to guarantee even a local optimum, much less a global optimum. Meanwhile, since different rules are suitable for different scenarios, it is hard for the decision maker to select the best rule at a specific time point. Metaheuristics always decompose dynamic scheduling problems into a series of static sub-problems and solve them separately by some intelligent optimization algorithms such as genetic algorithm (GA) [5] and particle swarm

E-mail address: luos17@mails.tsinghua.edu.cn.

https://doi.org/10.1016/j.asoc.2020.106208
1568-4946/© 2020 Elsevier B.V. All rights reserved.

optimization (PSO) [6]. They acquire higher solution quality but may be time-consuming and infeasible for real-time scheduling. On this account, if one can choose the most appropriate dispatching rule at each rescheduling point, then both the timeliness and shop performance can be guaranteed, which serves as the primary motivation of this paper.

To find the most suitable dispatching rule at each decision point, the DFJSP can be regarded as a Markov decision process (MDP) where an intelligent agent should determine the optimal action, i.e., which rule to choose, after the occurrence of a disturbance by comprehensively utilizing the information from current production state. In order to solve a MDP, previous work often resorted to dynamic programming (DP) [7]. However, in today's complex manufacturing system, the state transition probability of each state–action pair cannot be accurately modeled in advance, which restricts the practical application of DP. Meanwhile, other methods may be either too myopic (dispatching rules) or time-consuming (metaheuristics).

In recent years, reinforcement learning (RL) has emerged as a powerful way to deal with MDP [8]. Due to the ability of RL to learn the best action at each decision point and react to dynamic events completely in real time, many RL-based methods have been applied to different kinds of dynamic scheduling problems. One of the earliest work was from Riedmiller and Riedmiller [9], they proposed a RL approach to learn local dispatching policies in a job shop with the aim of reducing the summed tardiness. A neural network based agent was associated to each resource and trained by Q-learning. It demonstrated better performance than common heuristic dispatching rules. Later, Aydin and Öztemel [10] developed an improved Q-learning based algorithm named as Q-III to train an agent to select the most appropriate dispatching rule in real time for a dynamic job-shop with new job insertions. Wang and Usher [11] used the Q-learning to train a single machine agent which selected the optimal dispatching rule among three given rules so as to minimize mean tardiness. Chen et al. [12] suggested a rule driven method to develop composite dispatching rule for multi-objective dynamic job shop scheduling. An intelligent agent was trained by Q-learning to obtain the appropriate weights of elementary rules for building the single composite rule. In their work, only several predefined weight vectors are available, thus the optimal construction of composite rules cannot be guaranteed. Gabel and Riedmiller [13] proposed a multi-agent system with autonomous dispatching agents for job-shop scheduling to minimize makespan. Each machine was attached to an agent which employed probabilistic dispatching policies to decide which operation waiting currently to be processed next. The parameters of policies were adapted by Policy gradient. Bouazza et al. [14] utilized the Q-learning algorithm to determine the most suitable machine selection rule and dispatching rule in a dynamic flexible job shop scheduling problem with new job insertions. Two Q matrices were used to maintain the probabilities of choosing respectively a machine selection rule and a particular dispatching rule. Shahrabi et al. [15] used Q-learning to find the optimal parameters of variable neighborhood search (VNS) at any rescheduling point for a dynamic job shop scheduling problem considering random job arrivals and machine breakdowns. More recently, Wang [16] proposed a multi-agent system containing machine, buffer, state and job agents for dynamic job shop scheduling to minimize earliness and tardiness punishment. A weighted Q-learning algorithm based on dynamic greedy search was adopted to determine the optimal scheduling rules. Table 1 summarizes the differences between the aforementioned work and our work.

Without lose of generality, most of the RL-based methods mentioned above adopt Q-learning in discrete and finite state space, where a lookup Q table is maintained with each element denoting the estimated Q function value of a state–action pair. However, for a practical problem with continuous state features, the total number of states may be infinite thus maintaining a huge Q table is impossible. To address this issue, one of the straightforward ways is by discretizing the continuous state space into to some separate values with certain sacrifice of model accuracy. For instance, in the work of Shahrabi et al. [15], two variables including number of jobs in shop floor and mean processing time of current operations are classified into several numerical intervals so that each interval denotes a state. In addition, Shiue et al. [17] used self-organizing map (SOM) to determine the system states for a RL agent in a smart factory. The shortcoming of compulsive state discretization is obvious since there is no efficient guidance on how to choose the proper number of states so as to compromise the computational complexity and model accuracy.

In order to thoroughly overcome the limitation of traditional RL techniques in handling continuous state space, the breakthroughs of deep reinforcement learning (DRL) have set a good example recently [18]. The biggest innovation of DRL is the use of deep neural networks as function approximator to estimate the value function or selection probability of each candidate action instead of some shallow models such as linear function and decision trees [19]. The existing DRL methods can be divided into value-based ones (output the Q-value of each action) and policy-based ones (output the selection probability of each action). Compared to the policy-based methods, the value based methods are more intuitive which directly reflect the pros and cons among different actions. Meanwhile, they are easier to implement and have less parameters to be fine-tuned. On this account, we resort to a classical value-based DRL method named as deep Q network (DQN) [20] in this paper. By adopting a DQN, the Q-value of each state–action pair can be directly obtained with the continuous state features as input, thus no extra space is needed to store a cumbersome Q-table. Due to the superiority of DRL over stand Q-learning, it has been extended to various kinds of scheduling problems. Recently, Cao et al. [21] developed a DRL approach for multi-component job scheduling in edge computing. Waschneck et al. [22] designed some cooperative DQN-based agents for a job shop where each agent optimized the dispatching rules at one work center while monitoring the actions of other agents and optimizing a global reward. Khalil et al. [23] proposed a fitted Q-learning based on a deep learning architecture over graphs to learn greedy policies for a diverse range of combinatorial optimization problems. The learned greedy policy behaves like a meta-algorithm that constructs a solution incrementally. Mirhoseini et al. [24] developed a sequence-to-sequence deep model with LSTM and a content-based attention mechanism to predict which subsets of operations in a TensorFlow graph should run on which of the available devices. The model was trained by Policy gradients. However, few literature have applied DRL for solving dynamic flexible job shop scheduling problem.

With the motivations above, in this paper, we utilize DRL to solve the DFJSP under new job insertions aiming at minimizing the total tardiness. The contributions of this paper can be listed as follows: (1) Seven generic features taking values in $[0, 1]$ are extracted to represent a state at each rescheduling point. (2) Six composite rules (actions) are designed to simultaneously determine which operation to process next and which machine to assign it on. (3) A deep Q network (DQN) is developed to obtain the state–action value of each rule, based on which the most suitable dispatching rules at different decision points can be selected. Meanwhile, a "softmax" action selection policy is designed for the trained DQN to emphasize more on the rules with higher Q-values while maintaining the policy entropy. (4) Numerical experiments on different production configurations have demonstrated that the DQN is more effective than the

Table 1
Existing methods on RL-based dynamic job shop scheduling.

Work	State space	Algorithm	Agent	Objective	Dynamic events	Problem
Riedmiller and Riedmiller [9]	Continuous	Q-learning	Multi-agent	Summed tardiness	None	Job shop scheduling
Aydin and Öztemel [10]	Discrete	Q-learning	Single-agent	Mean tardiness	New job insertions	Job shop scheduling
Wang and Usher [11]	Discrete	Q-learning	Single-agent	Mean tardiness	New job insertions	Job shop scheduling
Chen et al. [12]	Discrete	Q-learning	Single-agent	Mean flow time; Mean tardiness	Fluctuation of work in process	Job shop scheduling
Gabel and Riedmiller [13]	Discrete	Policy gradient	Multi-agent	Makespan	None	Job shop scheduling
Bouazza et al. [14]	Discrete	Q-learning	Multi-agent	Makespan; Total weighted completion time; Weighted average waiting time	New job insertions	Flexible job shop scheduling
Shahrabi et al. [15]	Discrete	Q-learning	Single-agent	Mean flow time	New job insertions Machine breakdowns	Job shop scheduling
Wang [16]	Discrete	Q-learning	Multi-agent	Earliness and tardiness punishment	New job insertions	Job shop scheduling
Our method	Continuous	Deep Q-learning	Single-agent	Total tardiness	New job insertions	Flexible job shop scheduling

composite dispatching rules used in this paper, other well-known dispatching rules as well as the stand Q-learning agent. Moreover, it can be well generalized to untrained situations.

The remainder of this paper is organized as follows. Section 2 gives a brief review of dynamic scheduling methods. Section 3 presents the background of Q-learning and deep Q-learning. The mathematical model of DFJSP is established in Section 4. Section 5 provides the implementation details of the proposed DQN. The results of numerical experiments are given in Section 6. Finally, conclusions are drawn in Section 7.

2. Literature review

Dynamic scheduling can be mainly divided into 3 categories namely completely reactive scheduling, predictive–reactive scheduling, and robust pro-active scheduling [25].

In completely reactive approaches, no firm schedule is generated in advance and decisions are made locally in real time after the occurrence of dynamic events. This approach can immediately respond to dynamic events but may be myopic as decisions are made locally. The most frequently used reactive approach is priority dispatching rules. Rajendran and Holthaus [26] conducted a comparative study on the performance of different dispatching rules in the dynamic flow shops and job shops and observed that the relative performance of dispatching rules could be influenced by the routing of jobs and shopfloor configurations. Lawrence and Sewell [27] found that simple dispatching heuristics could provide comparable or even superior performance to that of optimum seeking methods as the uncertainty in processing times increases. Subramaniam et al. [28] studied the dynamic FJSP under machine breakdowns considering mean job cost and mean job tardiness. They demonstrated that the performance of simple dispatching rules could be significantly enhanced when used with machine selection rules. Gabel and Riedmiller [29] proposed an adaptive reactive job shop scheduling method where each resource was attached to an adaptive agent that made its job dispatching decisions independently of the other agents. The agents were trained by trial and error employing a neural reinforcement learning algorithm. Zandieh and Adibi [30] proposed a reactive scheduling method based on variable neighborhood search (VNS) for a dynamic job shop with random job arrivals and machine breakdowns. An artificial neural network (ANN) was used to update parameters of the VNS at any rescheduling point. Nie et al. [31] developed an approach based on gene expression programming (GEP) for reactive flexible job shop scheduling problem with new job insertions, where the proper dispatching rules for machine assignment and operation sequencing were automatically constructed and evolved by genetic operators. Xiong et al. [32] presented four new dispatching rules for a dynamic job shop with batch release and extended technical precedence constraints to minimize the total tardiness. They found that the relative performance of dispatching rules can be affected by both the level of extended technical precedence constraints and due date tightness.

In predictive–reactive scheduling, different from completely reactive scheduling, a predictive schedule is generated in advance to optimize shop performance without considering the possible disruptions in future. When disruptions occur during execution, the predictive schedule is modified (rescheduled) to maintain feasibility or improve performance. Based on the particular time point to perform rescheduling, this approach can be further divided into two kinds: periodic rescheduling, where rescheduling takes place at the beginning of regular time intervals, and event driven rescheduling, where rescheduling is invoked every time a disruption occurs. Nelson et al. [33] conducted the first study on predictive–reactive scheduling of dynamic job shop scheduling problem under intermittent job arrivals and stochastic processing times, where a multi-pass heuristic was proposed to modify the schedule periodically. Tao et al. [6] developed a hybrid multi-phase quantum particle swarm optimization (HMQPSO) algorithm together with a periodic rescheduling strategy for a dynamic flexible job shop with new job insertions, breakdowns of the machine, and delivery changes. They observed that scheduling efficiency and scheduling stability could be affected by different rescheduling cycles. Baykasoğlu and Karaslan [34] proposed a greedy randomized adaptive search procedure (GRASP) for dynamic job shop rescheduling under new order arrivals, machine breakdowns, changes of the due dates and order cancellations. They showed that the event-driven rescheduling policy usually provided better results than the periodical rescheduling policy for the studied problems. Lou et al. [4] suggested a multi-agent-based proactive–reactive scheduling method for a job shop with machine breakdowns and rush jobs. In the proactive scheduling stage, a robust predictive schedule was built on the basis of a stochastic scheduling model where the processing times were assumed to follow normal distribution. In the reactive scheduling stage, different agents including task management agent, scheduling management agent, and machine agent dynamically rectified the predictive schedule when unexpected events occurred. Gao et al. [1] developed a two-stage artificial bee colony (TABC) algorithm for the FJSP with new job insertions to minimize makespan. An initial schedule was created at start stage. After new job arrived, the new operations together with the remaining operations of existing jobs were rescheduled in the second stage. Shen and Yao [35] proposed a multi-objective evolutionary algorithm (MOEA) based proactive–reactive method for dynamic

flexible job shop scheduling problem with random new job arrivals and machine breakdowns. At the initial time, a predictive schedule was generated by an MOEA considering makespan, tardiness and the maximal machine workload. At each rescheduling point, a new schedule was constructed by considering stability together with the other three objectives. Kundakcı and Kulak [5] developed a hybrid genetic algorithm (HGA) for minimizing makespan in dynamic job shop scheduling problem with new job insertions, machine breakdowns and changes in processing time. Some well-known dispatching rules were combined with GA to generate an efficient initial population then tabu search (TS) was used to reschedule every time a dynamic event occurred. Shahgholi Zadeh et al. [36] presented an artificial bee colony (ABC) algorithm for dynamic flexible job-shop scheduling problem to minimize makespan. An initial schedule was generated at first according to the estimated processing times and rescheduling was performed after any change in processing times. Nouiri et al. [37] proposed a predictive–reactive approach based on particle swarm optimization (PSO) for energy efficient scheduling and rescheduling in a dynamic flexible job shop under machine breakdowns, where makespan and global energy consumption were considered to be minimized.

As for robust scheduling, it takes future disruptions into account and mainly focuses on building the predictive schedule to minimize (absorb) the effects of predictable disruptions so that the practical execution is less likely to deviate from the ideal schedule when disruptions occur. Mehta and Uzsoy [38] presented a robust scheduling method for a job shop to minimize maximum lateness by inserting additional idle time into the schedule to absorb the impacts of machine breakdowns. The amount and location of the additional idle time are determined from the probability distributions of time between breakdowns and time to repair. Inserting redundant idle times is easy to implement but may result in inactive schedule [39] and reduce the resource utilization. To this end, Al-Hinai and ElMekkawy [40] proposed a hybrid genetic algorithm (HGA) for robust and stable flexible job shop scheduling with random machine breakdowns. Instead of adding redundant idle times, the algorithm searched for a predictive schedule that could work around the expected breakdowns by simultaneously integrating the knowledge of machine breakdown probability distribution along with the available flexible routing of machines. Buddala and Mahapatra [41] proposed a two-stage teaching-learning-based optimization (2S-TLBO) method for the FJSP under machine breakdowns. The makespan was optimized without considering any breakdown in the first stage. In the second stage, a bi-objective function containing robustness and stability of the schedule was optimized under uncertainty of machine breakdowns. Similar to this work, Sajadi et al. [42] developed a two-stage multi-objective genetic algorithm (2S-MOGA) for robust and stable scheduling in a flexible job-shop. The first stage minimized the makespan with no expected disruptions. The second stage optimized both makespan and stability in the presence of random machine breakdowns. Moreover, Wang et al. [43] presented a NSGA-II algorithm hybrid with local simulated-annealing (SA) operators for a bi-criteria robust job-shop scheduling problem with uncertain processing times described by discrete scenario set. The mean makespan and the worst-case makespan among all scenarios were minimized to realize solution optimality and robustness, respectively.

3. Background of Q-learning and deep Q-learning

3.1. RL and Q-learning

In general, RL can be modeled as a MDP with a 5-tuple representation $(\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{P}, \gamma, \boldsymbol{R})$. In a MDP model, an intelligent agent interacts with the surrounding environment following a specific policy π. At each decision point t, the agent observes the current state $s_t \in \boldsymbol{S}$ and chooses an action $a_t \in \boldsymbol{A}$ according to the policy $\pi(\boldsymbol{S} \rightarrow \boldsymbol{A})$, after which it gets into a new state s_{t+1} with the transition probability $p(s_{t+1}|s_t, a_t) \in \boldsymbol{P}(\boldsymbol{S} \times \boldsymbol{A} \rightarrow \boldsymbol{S})$ and receives an immediate reward $r_t \in \boldsymbol{R}$.

The objective of a RL agent is to find the optimal policy π^* which maximizes the expected sum of long-term rewards when taking an action a in state s and following a specific policy π thereafter, as defined in Eq. (1):

$$\begin{aligned} Q_{\pi^*}(s,a) &= \max_{\pi} Q_{\pi}(s,a) \\ &= \max_{\pi} \mathbb{E}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots \middle| s_t = s, a_t = a, \pi\right] \end{aligned} \tag{1}$$

where $\gamma \in (0, 1]$ is the discount factor which differentiates the relative importance of short-term reward and long-term reward. $Q_{\pi}(s, a)$ is also called as Q function or action–value function. Bellman [44] has proved that the optimal value of action–value function must satisfy the Bellman optimality equation in Eq. (2):

$$Q_{\pi^*}(s,a) = \sum_{s'} p(s'|s,a)\left[r(s,a,s') + \gamma \max_{a'} Q_{\pi^*}(s',a')\right] \tag{2}$$

Based on the Bellman optimality equation, the stand Q-learning algorithm [8] can be derived.

3.2. Deep Q network and deep Q-learning

In order to address the curse of dimensionality existing in the standard Q-learning, the concept of deep Q network (DQN) was first proposed by Mnih et al. [20], which can be regarded as a neural network Q function approximator with weights θ. By directly taking the raw data (state features) as input and the Q function value of each state–action pair as output, DQN can handle complicated decision process with large and continuous state space.

Deep Q-learning (DQL) [45] is an effective way to train a DQN. The advancements of DQL are mainly presented in two aspects. First, in order to eliminate the correlation between consecutive transitions which may result in high variance of parameter updating and make the training process unstable, a replay memory D with capacity N is established where the agent's experience at each time-step t, i.e. (s_t, a_t, r_t, s_{t+1}) is stored. The updates of parameters are based on minibatch of samples randomly drawn from D. Once the capacity N is exceeded, old experiences are substituted with the new ones. Since each transition can be used multiple times to update the parameters, better data efficiency can also be achieved. The second improvement is the use of a separate target network $\hat{Q}$. Every C updates, the weights θ^- of the target network $\hat{Q}$ are replaced by the online network Q and kept fixed for the following C steps. At each time step t, the parameters θ of the online network are updated according to the target values calculated as $y_t^{DQN} = r_t + \gamma \max_{a'} \hat{Q}(s_{t+1}, a'; \theta^-)$. This procedure adds a delay between the time Q is updated and the time the update affects the training targets, thus making the training process more stable.

3.3. Double DQN

The standard DQL encounters an inevitable problem, that is, the max operator uses the same values to both select and evaluate an action, which may easily lead to overoptimistic value estimates [46]. To relieve this problem, Van Hasselt et al. [47] proposed a technique named as double DQN (DDQN). DDQN is

the same as DQN except for the way of calculating the training target y_t, as shown in Eq. (3).

$$y_t^{DDQN} = r_t + \gamma \hat{Q}(s_{t+1}, \arg\max_{a'} Q(s_{t+1}, a'; \theta); \theta^-) \tag{3}$$

In DDQN, the greedy policy is evaluated according to the online network Q but its value is estimated using the target network $\hat{Q}$. By adopting DDQN, the selection is decoupled from the evaluation, which reduces overoptimism and results in more stable and reliable learning. The procedure of DDQN is given in Algorithm 1.

Algorithm 1 Procedure of double DQN

Initialize replay memory D to capacity N
Initialize online network Q with random weights θ
Initialize target network $\hat{Q}$ with weights $\theta^- = \theta$
for episode = $1 : L$ **do**
  Observe initial state s_1 and extract the feature vector ϕ_1 of state s_1
  for $t = 1 : T$ (T is the terminal time) **do**
    With probability ε select a random action a_t
    otherwise select $a_t = \arg\max_a Q(\phi_t, a; \theta)$
    Execute action a_t, observe reward r_t and next state s_{t+1}
    Extract feature vector ϕ_{t+1} of state s_{t+1}
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D

Set $y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \hat{Q}(\phi_{j+1}, \arg\max_{a'} Q(\phi_{j+1}, a'; \theta); \theta^-) & \text{otherwise} \end{cases}$

    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ with respect to the network parameters θ
    Every C steps reset $\hat{Q} = Q$
  end for
end for

3.4. Soft target weight update

In the classical DQN and DDQN, the weights of target network $\hat{Q}$ are updated periodically every C steps, as shown in Algorithm 1. However, it is difficult to choose a proper value of C so as to make the training process efficient and stable. On this account, Lillicrap et al. [48] proposed a soft target update strategy. Instead of directly copying the weights from online network Q to the target network $\hat{Q}$ every C steps, the weights of target network are updated at each training step by having them slowly track the online network: $\hat{Q} = \tau Q + (1-\tau)\hat{Q}$ with the soft parameter $\tau \in (0, 1)$. This simple change forces the target values to change smoothly, greatly improving the stability of learning.

4. Problem formulation

The DFJSP with new job insertions considered in this paper can be defined as follows. There are n successively arriving jobs $\boldsymbol{J} = \{J_1, J_2, \ldots, J_n\}$ to be processed on m machines $\boldsymbol{M} = \{M_1, M_2, \ldots, M_m\}$. Each job J_i consists of n_i operations where $O_{i,j}$ is the jth operation of job J_i. Each operation $O_{i,j}$ can be processed on any machine M_k selected from a compatible machine set $\boldsymbol{M_{i,j}}$ ($\boldsymbol{M_{i,j}} \subseteq \boldsymbol{M}$). The processing time of operation $O_{i,j}$ on machine M_k is denoted by $t_{i,j,k}$. The arrival time and due date of a job J_i is A_i and D_i, respectively. $C_{i,j}$ represents the actual completion time of operation $O_{i,j}$. The objective is to minimize the total tardiness of all jobs. To simplify the problem at hand, several predefined constraints should be satisfied as follows.

(1) Each machine can process at most one operation at a time (capacity constraint).
(2) All operations belonging to the same job should be processed one after another in a fixed order (precedence constraint).
(3) Each operation should be processed nonpreemptively without interruption.
(4) Transportation times and setup times are negligible.

The notations used for problem formulation are listed below.

(1) Parameters:
n: total number of jobs
m: total number of machines
J_i: the ith job
n_i: total number of operations belonging to job J_i
M_k: the kth machine
$O_{i,j}$: the jth operation of job J_i
$\boldsymbol{M_{i,j}}$: the available machine set for operation $O_{i,j}$
$t_{i,j,k}$: the processing time of operation $O_{i,j}$ on machine M_k
A_i: the arrival time of job J_i
D_i: the due date of job J_i
i, h: index of jobs, $i, h = 1, 2, \ldots, n$
j, g: index of operations belonging to job J_i and J_h, $j = 1, 2, \ldots, n_i$, $g = 1, 2, \ldots, n_h$
k: index of machines, $k = 1, 2, \ldots, m$

(2) Decision variables:
$C_{i,j}$: the completion time of operation $O_{i,j}$.

$$X_{i,j,k} = \begin{cases} 1 & \text{if } O_{i,j} \text{ is assigned on machine } M_k \\ 0 & \text{otherwise} \end{cases}$$

$$Y_{i,j,h,g} = \begin{cases} 1 & \text{if } O_{i,j} \text{ is a predecessor of } O_{h,g} \\ -1 & \text{if } O_{i,j} \text{ is a successor of } O_{h,g} \end{cases}$$

$X_{i,j,k}$ determines which machine an operation is assigned on, while $Y_{i,j,h,g}$ determines the relative processing priority between any two operations.

Based on the notations above and the model developed in Lu et al. [49], the DFJSP addressed in this paper can be described mathematically as follows.

$$\text{Minimize} \sum_{i=1}^{n} \max\{C_{i,n_i} - D_i, 0\} \tag{4}$$

$$s.t. \begin{cases} C_{i,0} = 0, \quad C_{i,j} > 0, \quad \forall i, j & \text{(a)} \\ \sum_{k \in \boldsymbol{M_{i,j}}} X_{i,j,k} = 1, \quad \forall i, j & \text{(b)} \\ (C_{i,1} - t_{i,1,k} - A_i) X_{i,1,k} \geq 0, \quad \forall i, k & \text{(c)} \\ (C_{i,j} - t_{i,j,k} - C_{i,j-1}) X_{i,j,k} \geq 0, \quad \forall i, j, k & \text{(d)} \\ (C_{h,g} - t_{h,g,k} - C_{i,j}) X_{i,j,k} X_{h,g,k} (Y_{i,j,h,g} + 1) \\ + (C_{i,j} - t_{i,j,k} - C_{h,g}) X_{i,j,k} X_{h,g,k} (1 - Y_{i,j,h,g}) \geq 0, \\ \quad \forall i, j, h, g, k & \text{(e)} \end{cases} \tag{5}$$

Objective (4) is total tardiness of all jobs. Eq. (5)(a) indicates that the completion time of each operation must be non-negative. Eq. (5)(b) suggested that each operation can be assigned on only one machine. Eq. (5)(c) makes sure that a job can only be processed after its arrival time. Precedence constraint is ensured in Eq. (5)(d). Capacity constraint is guaranteed in Eq. (5)(e).

5. Proposed methods for the DFJSP

In this section, the definition of state features is provided at first. Then the candidate dispatching rules (actions) and definition of reward at each rescheduling point are successively given. Finally, the network structure together with the training method of DQN are presented.

5.1. Definition of state features

In most RL-based scheduling methods, state features are defined as some indicators of production status, i.e. the number of machines/jobs/operations in shop floor, the remaining processing time of uncompleted jobs, the current workload/queue length of each machine and so on [15,17]. However, in real world application, the number of machines/jobs/operations are unlimited and can be extremely large. If these indicators are directly taken as state features, the input of DQN may vary in a wide range. This may deteriorate the performance and generality of DQN in untrained situations since only particular production configurations are experienced in the training process. To address this issue, seven elaborately-designed state features with each one taking values in the range of [0, 1] are extracted to serve as the input of DQN. By limiting all the state features in [0, 1], the DQN can be easily extended to different untrained production environments.

Before presenting the details of state features, some notations should be given in advance. First, define $CT_k(t)$ as the completion time of the last operation having been assigned on machine M_k at rescheduling point t, and $OP_i(t)$ as the current number of completed operations of job J_i. Then the utilization rate of machine M_k at time t, denoted by $U_k(t)$ can be calculated as $U_k(t) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{OP_i(t)} t_{i,j,k}X_{i,j,k}}{CT_k(t)}$. Meanwhile, the completion rate of job J_i at t, denoted by $CRJ_i(t)$ can be calculated as $CRJ_i(t) = \frac{OP_i(t)}{n_i}$. Based on the notations above, the state features at each rescheduling point t can be listed as follows.

(1) Average utilization rate of machines $U_{ave}(t)$, as defined in Eq. (6):

$$U_{ave}(t) = \frac{\sum_{k=1}^{m} U_k(t)}{m} \tag{6}$$

(2) The standard deviation of machine utilization rate $U_{std}(t)$, as defined in Eq. (7):

$$U_{std}(t) = \sqrt{\frac{\sum_{k=1}^{m} (U_k(t) - U_{ave}(t))^2}{m}} \tag{7}$$

(3) Average completion rate of operations $CRO_{ave}(t)$, as defined in Eq. (8):

$$CRO_{ave}(t) = \frac{\sum_{i=1}^{n} OP_i(t)}{\sum_{i=1}^{n} n_i} \tag{8}$$

(4) Average completion rate of jobs $CRJ_{ave}(t)$, as defined in Eq. (9):

$$CRJ_{ave}(t) = \frac{\sum_{i=1}^{n} CRJ_i(t)}{n} \tag{9}$$

(5) The standard deviation of job completion rate $CRJ_{std}(t)$, as defined in Eq. (10):

$$CRJ_{std}(t) = \sqrt{\frac{\sum_{i=1}^{n} (CRJ_i(t) - CRJ_{ave}(t))^2}{n}} \tag{10}$$

(6) Estimated tardiness rate $Tard_e(t)$

Define T_{cur} as the average completion time of the last operations on all machines at current time t, then the estimated tardy jobs can be regarded as the ones whose remaining processing times are longer than the slack times from T_{cur} to their due dates. The estimated tardiness rate $Tard_e(t)$ is defined as the number of estimated tardy operations divided by the number of uncompleted operations belonging to all remaining jobs, the calculating method of which is given in Algorithm 2.

Algorithm 2 Procedure to calculate the estimated tardiness rate $Tard_e(t)$

Input: $CT_k(t)$, $OP_i(t)$, D_i
Output: $Tard_e(t)$

$T_{cur} \leftarrow \frac{\sum_{k=1}^{m} CT_k(t)}{m}$
$N_{tard} \leftarrow 0$
$N_{left} \leftarrow 0$
for $i = 1 : n$ **do**
  if $OP_i(t) < n_i$ **then**
    $N_{left} \leftarrow N_{left} + n_i - OP_i(t)$
    $T_{left} \leftarrow 0$
    for $j = OP_i(t) + 1 : n_i$ **do**
      $\bar{t}_{i,j} = \text{mean}_{k \in \boldsymbol{M_{i,j}}} t_{i,j,k}$
      $T_{left} \leftarrow T_{left} + \bar{t}_{i,j}$
      if $T_{cur} + T_{left} > D_i$ **then**
        $N_{tard} \leftarrow N_{tard} + n_i - j + 1$
        break
      end if
    end for
  end if
end for
$Tard_e(t) \leftarrow \frac{N_{tard}}{N_{left}}$
Return $Tard_e(t)$

(7) Actual tardiness rate $Tard_a(t)$

The actual tardy jobs are defined as the ones who have not been completed until their due dates. The actual tardiness rate $Tard_a(t)$ is defined as the number of actual tardy operations divided by the number of uncompleted operations belonging to all remaining jobs, the calculating method of which is given in Algorithm 3.

Algorithm 3 Procedure to calculate the actual tardiness rate $Tard_a(t)$

Input: $CT_k(t)$, $OP_i(t)$, D_i
Output: $Tard_a(t)$

$N_{tard} \leftarrow 0$
$N_{left} \leftarrow 0$
for $i = 1 : n$ **do**
  if $OP_i(t) < n_i$ **then**
    $N_{left} \leftarrow N_{left} + n_i - OP_i(t)$
    if $C_{i,OP_i(t)} > D_i$ **then**
      $N_{tard} \leftarrow N_{tard} + n_i - OP_i(t)$
    end if
  end if
end for
$Tard_a(t) \leftarrow \frac{N_{tard}}{N_{left}}$
Return $Tard_a(t)$

5.2. The proposed dispatching rules

It is widely acknowledged that no rule has been found to perform well for all shop configurations, operation conditions and performance objectives [31], thus different rules should be utilized corresponding to different production status. Moreover, note that the DFJSP considered in this paper contains both the operation sequence and machine assignment sub-problems, the decision maker should determine not only an operation selection rule but also a machine assignment rule at each decision point. On this account, we present six composite dispatching rules to first select a feasible operation and then assign it on a proper machine. Since these rules may utilize the processing time of an operation which changes in different available machines, we approximate the processing time $\bar{t}_{i,j}$ of each operation $O_{i,j}$ as

Table 2
Parameter settings of different production configurations.

Parameter	Value
Total number of machines (m)	{10, 20, 30, 40, 50}
Number of available machines of each operation	Unif[0, M]
Number of initial jobs at beginning (n_{ini})	20
Total number new inserted jobs (n_{add})	{50, 100, 200}
Due date tightness (DDT)	{0.5, 1.0, 1.5}
Number of operations belonging to a job	Unif[0, 20]
Processing time of an operation on an available machine	Unif[0, 50]
Average value of exponential distribution Between two successive new job arrivals (E_{ave})	{50, 100, 200}

the average value of its processing times among all available machines: $\bar{t}_{i,j} = \frac{\sum_{k \in \boldsymbol{M_{i,j}}} t_{i,j,k}}{|\boldsymbol{M_{i,j}}|}$.

The six composite dispatching rules are described in details in the following part. All the rules are designed for the purpose of reducing total tardiness. In each composite rule, the candidate operations are sorted and selected according to a specific indicator of tardiness. While the feasible machines are selected according to their earliest available time, total workload or utilization rate.

5.2.1. Composite dispatching rule 1

For dispatching rule 1, we first define a reference time T_{cur} as the average completion time of the last operation already assigned on each machine at current decision point t. Then an approximate set of tardy jobs, $Tard_{job}(t)$, can be defined as the uncompleted ones whose due dates are earlier than T_{cur}. Meanwhile, the set of all uncompleted jobs at t is denoted by $UC_{job}(t)$. If $Tard_{job}(t)$ is empty, i.e., no tardy jobs exist, we rank each job $J_i \in UC_{job}(t)$ by the average slack time of its remaining operations defined as $\frac{D_i - T_{cur}}{n_i - OP_i(t)}$. Then the next operation of the job with minimum average slack time is selected. If $Tard_{job}(t)$ is not empty, we estimate the tardiness of each job $J_i \in Tard_{job}(t)$ as $T_{cur} + \sum_{j=OP_i(t)+1}^{n_i} \bar{t}_{i,j} - D_i$. Then the next operation of the job with the biggest estimated tardiness is chosen. After determining the operation $O_{i,j}$ to be processed later, it is assigned on the earliest available machine M_k where $k = \arg\min_{k \in \boldsymbol{M_{i,j}}} \left(\max(CT_k(t), C_{i,j-1}, A_i)\right)$. If there are several machines with same earliest available time, the selected operation is assigned on one of them at random. The procedure of dispatching rule 1 is shown in Algorithm 4.

Algorithm 4 Procedure of dispatching rule 1

$T_{cur} \leftarrow \text{mean}_k\ CT_k(t)$
$Tard_{job}(t) \leftarrow \{i | OP_i(t) < n_i \ \&\&\ D_i < T_{cur}\}$
$UC_{job}(t) \leftarrow \{i | OP_i(t) < n_i\}$
if isempty($Tard_{job}(t)$) **then**
  $J_i \leftarrow \arg\min_{i \in UC_{job}(t)} \frac{D_i - T_{cur}}{n_i - OP_i(t)}$
else
  $J_i \leftarrow \arg\max_{i \in Tard_{job}(t)} \left(T_{cur} + \sum_{j=OP_i(t)+1}^{n_i} \bar{t}_{i,j} - D_i\right)$
end if
$j \leftarrow OP_i(t) + 1$
$M_k \leftarrow \arg\min_{k \in \boldsymbol{M_{i,j}}} \left(\max(CT_k(t), C_{i,j-1}, A_i)\right)$
assign $O_{i,j}$ on M_k

5.2.2. Composite dispatching rule 2

For dispatching rule 2, we first obtain the estimated set of tardy jobs $Tard_{job}(t)$ as in rule 1. If no tardy job exists, we rank each job by the ratio of its slack time to its remaining processing time defined as $\frac{D_i - T_{cur}}{\sum_{j=OP_i(t)+1}^{n_i} \bar{t}_{i,j}}$ (also known as critical ratio). Then we choose the next operation of the job with the minimum ratio. Otherwise, we select the job in $Tard_{job}(t)$ with the maximum estimated tardiness. Finally, the selected operation is assigned on the earliest available machine. The procedure of rule 2 is provided in Algorithm 5.

Algorithm 5 Procedure of dispatching rule 2

$T_{cur} \leftarrow \text{mean}_k\ CT_k(t)$
$Tard_{job}(t) \leftarrow \{i | OP_i(t) < n_i \ \&\&\ D_i < T_{cur}\}$
$UC_{job}(t) \leftarrow \{i | OP_i(t) < n_i\}$
if isempty($Tard_{job}(t)$) **then**
  $J_i \leftarrow \arg\min_{i \in UC_{job}(t)} \frac{D_i - T_{cur}}{\sum_{j=OP_i(t)+1}^{n_i} \bar{t}_{i,j}}$
else
  $J_i \leftarrow \arg\max_{i \in Tard_{job}(t)} \left(T_{cur} + \sum_{j=OP_i(t)+1}^{n_i} \bar{t}_{i,j} - D_i\right)$
end if
$j \leftarrow OP_i(t) + 1$
$M_k \leftarrow \arg\min_{k \in \boldsymbol{M_{i,j}}} \left(\max(CT_k(t), C_{i,j-1}, A_i)\right)$
assign $O_{i,j}$ on M_k

5.2.3. Composite dispatching rule 3

For dispatching rule 3, The next operation of the job with the biggest estimated tardiness is selected. Then with probability 0.5 it is assigned on the machine with the lowest utilization rate, otherwise it is assigned on the machine with the lowest workload. The procedure of dispatching rule 3 is shown in Algorithm 6.

Algorithm 6 Procedure of dispatching rule 3

$T_{cur} \leftarrow \text{mean}_k\ CT_k(t)$
$UC_{job}(t) \leftarrow \{i | OP_i(t) < n_i\}$
$J_i \leftarrow \arg\max_{i \in UC_{job}(t)} \left(T_{cur} + \sum_{j=OP_i(t)+1}^{n_i} \bar{t}_{i,j} - D_i\right)$
$j \leftarrow OP_i(t) + 1$
Generate a random number r in [0, 1]
if $r < 0.5$ **then**
  $M_k \leftarrow \arg\min_{k \in \boldsymbol{M_{i,j}}} \frac{\sum_{i=1}^{n} \sum_{j=1}^{OP_i(t)} t_{i,j,k} X_{i,j,k}}{CT_k(t)}$
else
  $M_k \leftarrow \arg\min_{k \in \boldsymbol{M_{i,j}}} \sum_{i=1}^{n} \sum_{j=1}^{OP_i(t)} t_{i,j,k} X_{i,j,k}$
end if
assign $O_{i,j}$ on M_k

5.2.4. Composite dispatching rule 4

For dispatching rule 4, an uncompleted operation is randomly chosen and assigned on the earliest available machine, as shown in Algorithm 7.

Algorithm 7 Procedure of dispatching rule 4

$UC_{job}(t) \leftarrow \{i | OP_i(t) < n_i\}$
Randomly choose an uncompleted job J_i from $UC_{job}(t)$
$j = OP_i(t) + 1$
$M_k \leftarrow \arg\min_{k \in \boldsymbol{M_{i,j}}} \left(\max(CT_k(t), C_{i,j-1}, A_i)\right)$
assign $O_{i,j}$ on M_k

5.2.5. Composite dispatching rule 5

For dispatching rule 5, we first find the estimated tardy set $Tard_{job}(t)$. If no tardy job exists, we rank each job J_i by the product of its completion rate and slack time defined as $\frac{OP_i(t)}{n_i} \cdot (D_i - T_{cur})$, then the next operation of the job with the minimum product is selected. Otherwise, we rank each job $J_i \in Tard_{job}(t)$ by the product of its inverse completion rate and estimated tardiness defined as $\frac{n_i}{OP_i(t)} \cdot (T_{cur} + \sum_{j=OP_i(t)+1}^{n_i} \bar{t}_{i,j} - D_i)$, then we select the next operation of the job with the maximum product. Finally, the selected operation is assigned on the earliest available machine. The procedure of rule 5 is given in Algorithm 8.

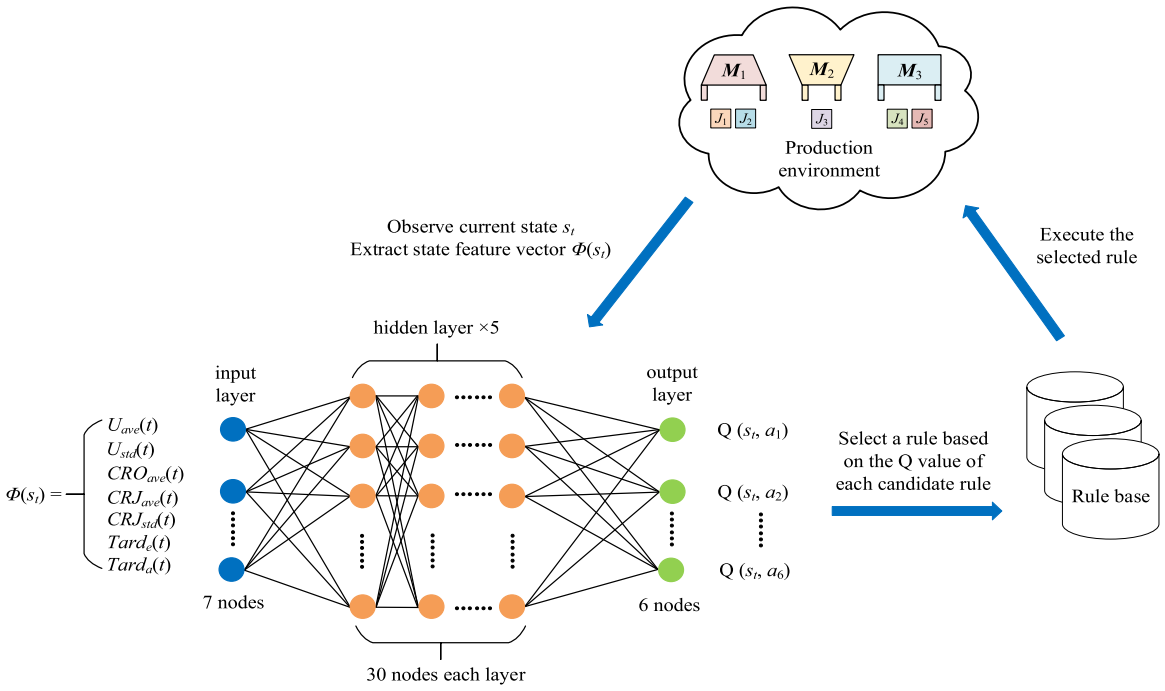

Fig. 1. Structure of the proposed DQN.

Algorithm 8 Procedure of dispatching rule 5

$T_{cur} \leftarrow \text{mean}_k\ CT_k(t)$
$Tard_{job}(t) \leftarrow \{i | OP_i(t) < n_i \ \&\&\ D_i < T_{cur}\}$
$UC_{job}(t) \leftarrow \{i | OP_i(t) < n_i\}$
if isempty($Tard_{job}(t)$) **then**
  $J_i \leftarrow \arg\min_{i \in UC_{job}(t)} \frac{OP_i(t)}{n_i} \cdot (D_i - T_{cur})$
else
  $J_i \leftarrow \arg\max_{i \in Tard_{job}(t)} \frac{n_i}{OP_i(t)} \cdot (T_{cur} + \sum_{j=OP_i(t)+1}^{n_i} \bar{t}_{i,j} - D_i)$
end if
$j \leftarrow OP_i(t) + 1$
$M_k \leftarrow \arg\min_{k \in \mathbf{M_{i,j}}} \left(\max(CT_k(t), C_{i,j-1}, A_i)\right)$
assign $O_{i,j}$ on M_k

5.2.6. Composite dispatching rule 6

For dispatching rule 6, we select the next operation of job J_i with maximum estimated tardiness. Then the selected operation is assigned on the machine with the earliest available time. The procedure of rule 6 is given in Algorithm 9.

Algorithm 9 Procedure of dispatching rule 6

$T_{cur} \leftarrow \text{mean}_k\ CT_k(t)$
$UC_{job}(t) \leftarrow \{i | OP_i(t) < n_i\}$
$J_i \leftarrow \arg\max_{i \in UC_{job}(t)} \left(T_{cur} + \sum_{j=OP_i(t)+1}^{n_i} \bar{t}_{i,j} - D_i\right)$
$M_k \leftarrow \arg\min_{k \in \mathbf{M_{i,j}}} \left(\max(CT_k(t), C_{i,j-1}, A_i)\right)$
assign $O_{i,j}$ on M_k

5.3. Definition of rewards

Since the objective is to minimize the total tardiness, the reward r_t for the state–action pair (s_t, a_t) is defined by successively considering the values of three critical state features at current state s_t and the next state s_{t+1}, including actual tardiness rate $Tard_a$, estimated tardiness rate $Tard_e$ and average machine utilization rate U_{ave}. The procedure to calculate r_t is given in Algorithm 10.

Algorithm 10 Definition of the reward r_t for the state–action pair (s_t, a_t) at each decision point t

if $Tard_a(t+1) < Tard_a(t)$ **then**
  $r_t \leftarrow 1$
else
  if $Tard_a(t+1) > Tard_a(t)$ **then**
    $r_t \leftarrow -1$
  else
    if $Tard_e(t+1) < Tard_e(t)$ **then**
      $r_t \leftarrow 1$
    else
      if $Tard_e(t+1) > Tard_e(t)$ **then**
        $r_t \leftarrow -1$
      else
        if $U_{ave}(t+1) > U_{ave}(t)$ **then**
          $r_t \leftarrow 1$
        else
          if $U_{ave}(t+1) > U_{ave}(t) \cdot 0.95$ **then**
            $r_t \leftarrow 0$
          else
            $r_t \leftarrow -1$
          end if
        end if
      end if
    end if
  end if
end if

5.4. Network structure

The DQN used in this paper 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 numbers of state features and available actions, respectively. The

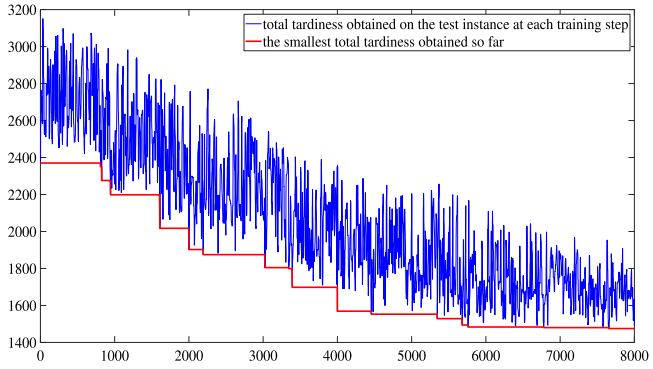

Fig. 2. Total tardiness obtained by the DQN at each training step. X axis: number of training step, Y axis: total tardiness.

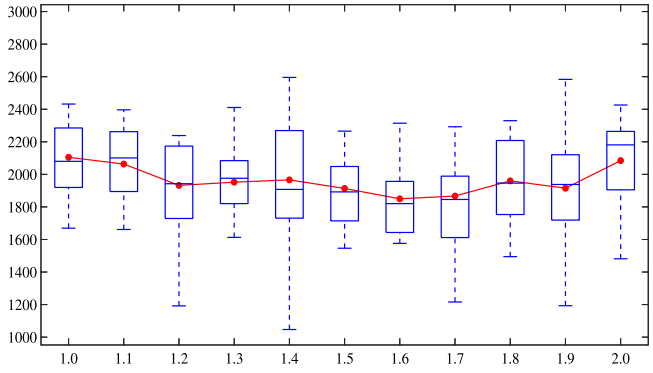

Fig. 3. Box plots of total tardiness under different levels of μ. X axis: μ, Y axis: total tardiness.

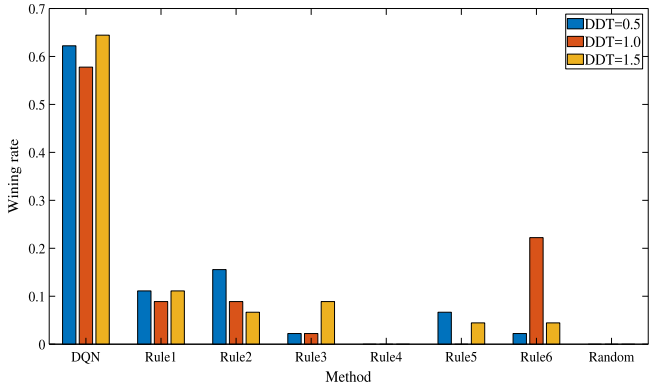

Fig. 4. Winning rate of DQN and other composite dispatching rules under different due date tightness.

number of nodes in each hidden layer is 30. We use "tansig" activation function for the input and hidden layers, and "purelin" for the output layer. Fig. 1 gives an illustration of the proposed DQN.

5.5. Overall framework of the training method

The training method is based on the framework of DDQN [47]. In the training process, the decision point t is defined as every time a new job arrives or an operation is completed. The overall framework of the DDQN-based training method is provided in Algorithm 11, it integrates the technique of soft target update mentioned in Section 3.4.

Table 3
Parameter settings of the training method.

Parameter	Value
Number of training episodes L	10
Replay buffer size N	10^3
Batch size of samples to perform gradient descent	32
ε in the action selection policy	Linearly decreased from 0.5 to 0.1
Discount factor γ	0.9
τ in the soft target update strategy	0.01

Table 4
Correlation coefficients between each pair of state features.

	U_{ave}	U_{std}	CRO_{ave}	CRJ_{ave}	CRJ_{std}	$Tard_e$	$Tard_a$
U_{ave}	1.0000	0.7214	−0.6456	−0.5498	0.8585	0.4606	0.0621
U_{std}	–	1.0000	−0.8329	−0.8064	0.6164	0.0985	−0.0533
CRO_{ave}	–	–	1.0000	0.9858	−0.6533	−0.1143	0.1118
CRJ_{ave}	–	–	–	1.0000	−0.5694	−0.0776	0.1131
CRJ_{std}	–	–	–	–	1.0000	0.4966	−0.0023
$Tard_e$	–	–	–	–	–	1.0000	0.4169
$Tard_a$	–	–	–	–	–	–	1.0000

Algorithm 11 The DDQN-based training method

Initialize replay memory D to capacity N
Initialize online network Q with random weights θ
Initialize target network $\hat{Q}$ with weights $\theta^- = \theta$
for episode = 1 : L **do**
  Generate the initial state s_1 with feature vector $\phi_1 = \{U_{ave}(1), U_{std}(1), CRO_{ave}(1), CRJ_{ave}(1), CRJ_{std}(1), Tard_e(1), Tard_a(1)\} = \{0, 0, 0, 0, 0, 0, 0\}$
  for $t = 1 : T$ (t is the decision point at which an operation has been completed or a new job arrives, T is the terminal time when all operations have been scheduled) **do**
    With probability ε select a random action a_t
    otherwise select $a_t = \arg\max_a Q(\phi_t, a; \theta)$
    Execute action a_t, observe the next state s_{t+1}, calculate the immediate reward r_t by Algorithm 10
    Obtain the new feature vector $\phi_{t+1} = \{U_{ave}(t+1), U_{std}(t+1), CRO_{ave}(t+1), CRJ_{ave}(t+1), CRJ_{std}(t+1), Tard_e(t+1), Tard_a(t+1)\}$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D

$$\text{Set } y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \hat{Q}(\phi_{j+1}, \arg\max_{a'} Q(\phi_{j+1}, a'; \theta); \theta^-) & \text{otherwise} \end{cases}$$

    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ with respect to the parameters θ of online network Q
    $\hat{Q} = \tau Q + (1-\tau)\hat{Q}$
  end for
end for

5.6. The action selection policy in real implementation

Note that in the training stage, in order to enhance exploration, the action at each rescheduling point is randomly selected with probability ε. However, when applying the trained DQN to the actual production process, the action with higher Q value should be chosen with a higher probability. On the other hand, if we always choose the action with the highest Q-value, the obtained schedule may easily fall into local optimum. In this

Table 5
Mean value and standard deviation of tardiness by DQN and each composite dispatching rule when $DDT = 0.5$ (mean/std).

E_{ave}	m	n_{add}	DQN	$Rule_1$	$Rule_2$	$Rule_3$	$Rule_4$	$Rule_5$	$Rule_6$	Random
50	10	50	2.27e+04/9.38e+02	2.32e+04/0	**2.17e+04*/0**	2.67e+04/0	2.54e+04/1.41e+03	2.31e+04/0	2.21e+04/0	2.35e+04/1.33e+03
		100	**3.08e+04*/1.03e+03**	3.20e+04/0	3.17e+04/0	3.79e+04/0	3.44e+04/1.92e+03	3.22e+04/0	3.36e+04/0	3.31e+04/1.47e+03
		200	**4.97e+04/1.33e+03**	5.12e+04/0	5.01e+04/0	7.49e+04/0	6.07e+04/2.02e+03	5.85e+04/0	5.31e+04/0	5.70e+04/2.64e+03
	20	50	**0.91e+04*/3.35e+02**	1.09e+04/0	1.02e+04/0	1.49e+04 /0	1.10e+04/6.11e+02	1.11e+04/0	1.14e+04/0	1.05e+04/4.01e+02
		100	**2.01e+04/4.31e+02**	2.08e+04/0	2.14e+04/0	2.65e+04/0	2.17e+04/7.89e+02	2.03e+04/0	2.06e+04/0	2.07e+04/6.42e+02
		200	3.48e+04/5.41e+02	3.46e+04/0	**3.34e+04*/0**	4.20e+04/0	3.64e+04/8.95e+02	3.45e+04/0	3.45e+04/0	3.47e+04/8.45e+02
	30	50	**0.90e+04/2.55e+02**	0.93e+04/0	0.95e+04/0	1.22e+04/0	0.91e+04/3.53e+02	0.95e+04/0	0.92e+04/0	0.91e+04/2.91e+02
		100	1.70e+04/4.69e+02	1.62e+04/0	**1.54e+04*/0**	1.92e+04/0	1.65e+04/4.99e+02	1.57e+04/0	1.65e+04/0	1.64e+04/5.57e+02
		200	3.27e+04/5.99e+02	3.25e+04/0	**3.22e+04/0**	3.85e+04/0	3.24e+04/5.91e+02	3.36e+04/0	3.30e+04/0	3.25e+04/5.59e+02
	40	50	**0.85e+04*/2.37e+02**	0.91e+04/0	0.94e+04/0	1.04e+04/0	0.90e+04/2.99e+02	0.89e+04/0	0.92e+04/0	0.91e+04/3.30e+02
		100	1.62e+04/4.47e+02	1.60e+04/0	1.55e+04/0	1.92e+04/0	1.53e+04/4.50e+02	**1.51e+04/0**	1.57e+04/0	1.54e+04/4.57e+02
		200	**2.69e+04/5.27e+02**	2.71e+04/0	2.83e+04/0	3.08e+04/0	2.72e+04/7.30e+02	2.75e+04/0	2.82e+04/0	2.76e+04/6.79e+02
	50	50	**0.92e+04/3.88e+02**	0.96e+04/0	1.03e+04/0	1.14e+04/0	1.00e+04/3.27e+02	0.98e+04/0	0.94e+04/0	0.98e+04/2.85e+02
		100	1.66e+04/5.09e+02	1.56e+04/0	**1.49e+04*/0**	1.69e+04/0	1.59e+04/4.55e+02	1.67e+04/0	1.64e+04/0	1.59e+04/4.82e+02
		200	2.96e+04/5.63e+02	**2.85e+04*/0**	2.91e+04/0	3.17e+04/0	2.93e+04/6.87e+02	2.89e+04/0	2.97e+04/0	2.90e+04/4.01e+02
100	10	50	**2.06e+04*/5.57e+02**	2.24e+04/0	2.14e+04/0	2.44e+04/0	2.27e+04/8.59e+02	2.21e+04/0	2.24e+04/0	2.19e+04/7.95e+02
		100	2.10e+04/5.79e+02	**1.99e+04*/0**	2.07e+04/0	2.52e+04/0	2.21e+04/9.58e+02	2.02e+04/0	2.19e+04/0	2.11e+04/6.75e+02
		200	3.99e+04/6.93e+02	3.97e+04/0	**3.75e+04*/0**	4.52e+04/0	4.04e+04/1.22e+03	3.86e+04/0	4.10e+04/0	4.01e+04/8.07e+02
	20	50	**1.05e+04*/4.08e+02**	1.24e+04/0	1.23e+04/0	1.47e+04/0	1.23e+04/4.22e+02	1.20e+04/0	1.27e+04/0	1.19e+04/4.10e+02
		100	**1.87e+04/4.05e+02**	1.93e+04/0	1.89e+04/0	2.15e+04/0	1.92e+04/6.07e+02	1.89e+04/0	2.00e+04/0	1.90e+04/4.27e+02
		200	**3.02e+04*/5.17e+02**	3.17e+04/0	3.11e+04/0	3.49e+04/0	3.12e+04/6.10e+02	3.14e+04/0	3.20e+04/0	3.13e+04/6.68e+02
	30	50	**0.87e+04/3.76e+02**	0.91e+04/0	0.89e+04/0	1.01e+04/0	0.90e+04/2.55e+02	0.88e+04/0	0.89e+04/0	0.92e+04/3.32e+02
		100	**1.61e+04*/5.10e+02**	1.80e+04/0	1.74e+04/0	1.97e+04/0	1.71e+04/4.71e+02	1.81e+04/0	1.69e+04/0	1.77e+04/4.69e+02
		200	3.07e+04/5.24e+02	**3.01e+04*/0**	3.03e+04/0	3.35e+04/0	3.10e+04/5.22e+02	3.03e+04/0	3.16e+04/0	3.08e+04/5.42e+02
	40	50	**0.92e+04/2.97e+02**	0.97e+04/0	0.94e+04/0	1.10e+04/0	0.94e+04/3.73e+02	0.95e+04/0	0.95e+04/0	0.94e+04/2.48e+02
		100	1.70e+04/3.93e+02	1.69e+04/0	1.66e+04/0	**1.64e+04*/0**	1.72e+04/4.85e+02	1.71e+04/0	1.67e+04/0	1.71e+04/3.79e+02
		200	**2.95e+04*/5.77e+02**	3.03e+04/0	3.09e+04/0	3.06e+04/0	3.05e+04/6.39e+02	3.10e+04/0	3.04e+04/0	3.07e+04/6.79e+02
	50	50	**0.95e+04*/3.34e+02**	1.03e+04/0	1.02e+04/0	1.15e+04/0	1.02e+04/4.47e+02	1.04e+04/0	1.04e+04/0	1.02e+04/3.18e+02
		100	1.63e+04/3.39e+02	1.61e+04/0	**1.56e+04*/0**	1.84e+04/0	1.64e+04/3.49e+02	1.67e+04/0	1.69e+04/0	1.64e+04/3.79e+02
		200	**2.85e+04/5.69e+02**	2.87e+04/0	2.88e+04/0	2.88e+04/0	2.87e+04/4.55e+02	2.89e+04/0	2.96e+04/0	2.86e+04/6.17e+02
200	10	50	**1.55e+04/4.09e+02**	1.65e+04/0	1.57e+04/0	1.62e+04/0	1.63e+04/7.44e+02	1.67e+04/0	1.77e+04/0	1.69e+04/6.04e+02
		100	1.85e+04/5.54e+02	1.82e+04/0	1.79e+04/0	2.05e+04/0	1.84e+04/5.30e+02	**1.76e+04*/0**	1.88e+04/0	1.85e+04/4.41e+02
		200	3.45e+04/5.35e+02	**3.39e+04*/0**	3.54e+04/0	3.79e+04/0	3.47e+04/6.75e+02	3.41e+04/0	3.42e+04/0	3.49e+04/6.69e+02
	20	50	**1.06e+04*/3.23e+02**	1.17e+04/0	1.18e+04/0	1.30e+04/0	1.12e+04/3.22e+02	1.15e+04/0	1.18e+04/0	1.14e+04/4.99e+02
		100	**1.72e+04*/3.99e+02**	1.80e+04/0	1.77e+04/0	1.99e+04/0	1.82e+04/3.15e+02	1.79e+04/0	1.86e+04/0	1.81e+04/4.20e+02
		200	**3.09e+04/5.47e+02**	3.25e+04/0	3.15e+04/0	3.23e+04/0	3.17e+04/5.57e+02	3.10e+04/0	3.23e+04/0	3.12e+04/5.33e+02
	30	50	**0.91e+04*/2.16e+02**	0.97e+04/0	0.95e+04/0	1.08e+04/0	0.98e+04/2.19e+02	0.99e+04/0	1.02e+04/0	0.99e+04/4.29e+02
		100	**1.59e+04*/3.54e+02**	1.69e+04/0	1.72e+04/0	1.82e+04/0	1.68e+04/2.75e+02	1.67e+04/0	1.74e+04/0	1.71e+04/4.09e+02
		200	2.93e+04/4.22e+02	**2.85e+04*/0**	2.95e+04/0	3.02e+04/0	2.92e+04/3.69e+02	2.90e+04/0	2.96e+04/0	2.94e+04/5.01e+02
	40	50	8.97e+03/2.89e+02	8.92e+03/0	8.69e+03/0	9.87e+03/0	8.72e+03/3.22e+02	8.97e+03/0	**8.61e+03/0**	8.69e+03/2.20e+02
		100	**1.57e+04*/3.92e+02**	1.66e+04/0	1.61e+04/0	1.82e+04/0	1.65e+04/3.17e+02	1.69e+04/0	1.63e+04/0	1.67e+04/4.43e+02
		200	**2.94e+04/5.31e+02**	3.00e+04/0	2.99e+04/0	3.21e+04/0	2.96e+04/2.77e+02	2.99e+04/0	2.98e+04/0	2.97e+04/4.62e+02
	50	50	**9.11e+03*/3.85e+02**	9.52e+03/0	9.81e+03/0	9.59e+03/0	9.79e+03/2.11e+02	9.62e+03/0	9.71e+03/0	9.74e+03/2.10e+02
		100	**1.75e+04/5.25e+02**	1.79e+04/0	1.77e+04/0	1.85e+04/0	1.76e+04/2.82e+02	1.77e+04/0	1.77e+04/0	1.76e+04/2.99e+02
		200	2.84e+04/5.57e+02	2.81e+04/0	2.89e+04/0	2.91e+04/0	2.85e+04/3.27e+02	**2.77e+04*/0**	2.79e+04/0	2.87e+04/3.99e+02

regard, we utilize a "softmax" policy to select an action after the DQN has been trained. In this policy, the probability $P(a_i)$ of adopting an action a_i at each rescheduling point t is calculated by Eq. (11).

$$P(a_i) = \frac{\exp(\mu \cdot Q(\phi_t, a_i; \theta))}{\sum_{j=1}^{|\boldsymbol{A}|} \exp(\mu \cdot Q(\phi_t, a_j; \theta))} \tag{11}$$

where μ is a hyperparameter to control the entropy in the softmax policy. It implies how much we can trust the trained agent's estimation over all execution states and makes the trained policy more robust when unseen states are observed in real execution.

6. Numerical experiments

In this section, the details of training process are provided at first. Then a sensitivity study on the control parameter μ of the "softmax" action selection policy is conducted. To confirm the effectiveness and generality of the trained DQN, we compare the performance of DQN with each composite dispatching rule under different production configurations. To show the superiority of DQN over continuous state space, we compare the DQN with stand Q-learning agent where a Q-table is maintained by compulsively discretizing the state space. Last but not the least, the superiority of DQN is further validated by taking other well-known dispatching rules as comparisons.

The test benchmarks used in this paper are generated at random. It is assumed that there are several jobs existing in the shopfloor at the very beginning, after which the new jobs arrive following Poisson distribution, hence the interarrival time between two successive new job insertions is subjected to exponential distribution. Each instance used in this study can be described by the parameters listed in Table 2.

In Table 2, the due date tightness (DDT) is an indicator of job urgency. For a job J_i with n_i operations arriving at time point A_i, its due date D_i can be calculated as $D_i = A_i + \left(\sum_{j=1}^{n_i} \bar{t}_{i,j}\right) \cdot DDT$. A small DDT leads to less slack time between the arriving time of a job and its due date, which means the job is more urgent than others.

Table 6
Mean value and standard deviation of tardiness by DQN and each composite dispatching rule when $DDT = 1.0$ (mean/std).

E_{ave}	m	n_{add}	DQN	$Rule_1$	$Rule_2$	$Rule_3$	$Rule_4$	$Rule_5$	$Rule_6$	Random
50	10	50	**1.01e+04*/1.03e+03**	1.37e+04/0	1.55e+04/0	1.68e+04/0	1.82e+04/1.89e+03	1.62e+04/0	1.12e+04/0	1.59e+04/1.39e+03
		100	1.68e+04/1.71e+03	1.67e+04/0	1.59e+04/0	2.22e+04/0	2.49e+04/1.88e+03	2.36e+04/0	**1.20e+04*/0**	2.07e+04/1.77e+03
		200	3.08e+04/2.40e+03	2.91e+04/0	2.53e+04/0	5.61e+04/0	4.51e+04/2.55e+03	4.79e+04/0	**2.42e+04*/0**	3.82e+04/2.19e+03
	20	50	**1.58e+03*/4.21e+02**	2.31e+03/0	2.01e+03/0	4.03e+03/0	4.52e+03/9.39e+02	3.66e+03/0	1.97e+03/0	3.49e+03/6.11e+02
		100	**3.52e+03/6.26e+02**	4.23e+03/0	3.75e+03/0	6.87e+03/0	7.64e+03/1.09e+03	5.66e+03/0	4.61e+03/0	5.88e+03/6.60e+02
		200	6.69e+03/5.79e+02	6.23e+03/0	5.31e+03/0	8.52e+03/0	7.39e+03/6.74e+02	6.10e+03/0	**4.29e+03*/0**	6.33e+03/6.85e+02
	30	50	**1.40e+03/3.70e+02**	2.09e+03/0	1.45e+03/0	2.50e+03/0	1.97e+03/3.19e+02	1.49e+03/0	1.86e+03/0	1.57e+03/2.03e+02
		100	2.73e+03/2.93e+02	**1.93e+03*/0**	2.22e+03/0	3.85e+03/0	3.07e+03/4.46e+02	2.38e+03/0	2.51e+03/0	2.45e+03/2.35e+02
		200	4.97e+03/3.12e+02	4.17e+03/0	**4.02e+03*/0**	4.95e+03/0	4.92e+03/5.59e+02	4.46e+03/0	4.75e+03/0	4.67e+03/3.67e+02
	40	50	**1.02e+03*/2.27e+02**	1.35e+03/0	1.64e+03/0	2.27e+03/0	1.41e+03/2.05e+02	1.69e+03/0	1.29e+03/0	1.39e+03/1.55e+02
		100	**2.17e+03/2.53e+02**	2.21e+03/0	2.32e+03/0	2.72e+03/0	2.38e+03/2.41e+02	2.67e+03/0	2.22e+03/0	2.20e+03/2.05e+02
		200	4.76e+03/3.15e+02	4.32e+03/0	4.39e+03/0	5.45e+03/0	4.24e+03/3.47e+02	3.93e+03/0	**3.81e+03*/0**	4.25e+03/4.25e+02
	50	50	**0.91e+03/1.57e+02**	1.12e+03/0	1.15e+03/0	1.15e+03/0	1.14e+03/1.94e+02	0.93e+03/0	1.29e+03/0	1.14e+03/1.60e+02
		100	2.87e+03/2.75e+02	2.85e+03/0	3.00e+03/0	2.57e+03/0	2.46e+03/3.69e+02	2.29e+03/0	**2.11e+03*/0**	2.39e+03/2.15e+02
		200	**3.20e+03*/3.21e+02**	4.71e+03/0	3.97e+03/0	4.14e+03/0	4.01e+03/3.31e+02	4.17e+03/0	4.16e+03/0	4.15e+03/3.77e+02
100	10	50	5.90e+03/6.37e+02	5.53e+03/0	7.65e+03/0	8.12e+03/0	7.87e+03/6.45e+02	6.52e+03/0	**5.31e+03*/0**	6.77e+03/6.41e+02
		100	**0.61e+04*/6.15e+02**	0.74e+04/0	0.77e+04/0	0.85e+04/0	1.00e+04/7.05e+02	0.90e+04/0	0.71e+04/0	0.82e+04/6.18e+02
		200	0.93e+04/6.57e+02	0.85e+04/0	**0.77e+04*/0**	1.07e+04/0	1.11e+04/6.94e+02	1.04e+04/0	0.91e+04/0	0.94e+04/4.99e+02
	20	50	**1.71e+03*/1.79e+02**	2.01e+03/0	2.13e+03/0	3.15e+03/0	2.89e+03/4.15e+02	2.58e+03/0	1.92e+03/0	2.15e+03/2.10e+02
		100	**2.82e+03/2.24e+02**	2.97e+03/0	2.89e+03/0	3.99e+03/0	3.58e+03/3.49e+02	3.27e+03/0	3.25e+03/0	3.17e+03/2.51e+02
		200	4.74e+03/3.45e+02	4.14e+03/0	4.32e+03/0	5.40e+03/0	4.77e+03/4.59e+02	5.27e+03/0	**3.87e+03*/0**	4.57e+03/3.37e+02
	30	50	**0.79e+03*/2.11e+02**	0.99e+03/0	1.34e+03/0	1.67e+03/0	1.10e+03/1.57e+02	1.07e+03/0	0.97e+03/0	1.14e+03/1.90e+02
		100	**1.72e+03*/2.07e+02**	2.37e+03/0	1.97e+03/0	2.90e+03/0	2.13e+03/2.52e+02	2.19e+03/0	2.01e+03/0	2.15e+03/2.41e+02
		200	**3.77e+03/4.45e+02**	4.12e+03/0	3.92e+03/0	5.24e+03/0	4.18e+03/3.74e+02	4.12e+03/0	4.24e+03/0	4.03e+03/2.71e+02
	40	50	**1.17e+03*/1.53e+02**	1.41e+03/0	1.51e+03/0	2.15e+03/0	1.51e+03/2.24e+02	1.45e+03/0	1.69e+03/0	1.49e+03/1.77e+02
		100	**2.04e+03/2.25e+02**	2.36e+03/0	2.10e+03/0	2.41e+03/0	2.23e+03/2.40e+02	2.11e+03/0	2.57e+03/0	2.09e+03/2.42e+02
		200	3.67e+03/2.55e+02	4.17e+03/0	3.85e+03/0	4.72e+03/0	3.44e+03/2.69e+02	3.95e+03/0	**3.38e+03/0**	3.64e+03/3.47e+02
	50	50	**1.06e+03/1.52e+02**	1.15e+03/0	1.12e+03/0	1.16e+03/0	1.25e+03/1.40e+02	1.10e+03/0	1.17e+03/0	1.19e+03/2.06e+02
		100	1.98e+03/2.75e+02	1.83e+03/0	**1.67e+03*/0**	2.44e+03/0	1.91e+03/2.45e+02	2.07e+03/0	1.76e+03/0	1.97e+03/1.65e+02
		200	**3.36e+03*/2.95e+02**	3.99e+03/0	4.14e+03/0	4.87e+03/0	3.90e+03/2.20e+02	3.93e+03/0	4.42e+03/0	3.86e+03/2.49e+02
200	10	50	4.96e+03/3.47e+02	4.55e+03/0	4.77e+03/0	4.89e+03/0	5.24e+03/5.32e+02	4.17e+03/0	**3.86e+03*/0**	4.55e+03/3.32e+02
		100	6.60e+03/4.19e+02	**5.71e+03*/0**	5.96e+03/0	6.77e+03/0	7.16e+03/5.39e+02	6.85e+03/0	6.14e+03/0	6.47e+03/5.00e+02
		200	**0.77e+04*/4.81e+02**	0.95e+04/0	0.97e+04/0	1.12e+04/0	1.05e+04/6.11e+02	1.08e+04/0	0.85e+04/0	0.91e+04/4.89e+02
	20	50	1.36e+03/2.26e+02	1.30e+03/0	1.03e+03/0	1.62e+03/0	1.58e+03/2.36e+02	1.14e+03/0	**0.89e+03*/0**	1.30e+03/2.64e+02
		100	**2.31e+03*/2.45e+02**	2.72e+03/0	2.99e+03/0	3.61e+03/0	3.37e+03/3.02e+02	3.15e+03/0	2.80e+03/0	2.98e+03/3.11e+02
		200	**3.92e+03*/4.12e+02**	4.56e+03/0	4.49e+03/0	6.52e+03/0	5.04e+03/3.17e+02	4.85e+03/0	4.42e+03/0	4.65e+03/3.77e+02
	30	50	**1.10e+03/1.70e+02**	1.43e+03/0	1.14e+03/0	1.46e+03/0	1.41e+03/1.89e+02	1.34e+03/0	1.16e+03/0	1.28e+03/1.99e+02
		100	2.42e+03/2.87e+02	**1.71e+03*/0**	1.74e+03/0	2.71e+03/0	2.25e+03/2.00e+02	2.38e+03/0	1.96e+03/0	2.17e+03/2.24e+02
		200	3.95e+03/3.15e+02	**3.22e+03*/0**	3.31e+03/0	4.98e+03/0	3.51e+03/2.36e+02	3.57e+03/0	3.25e+03/0	3.61e+03/2.85e+02
	40	50	**1.06e+03/2.12e+02**	1.08e+03/0	1.13e+03/0	1.29e+03/0	1.09e+03/1.61e+02	1.37e+03/0	1.07e+03/0	1.07e+03/1.31e+02
		100	**1.96e+03/2.60e+02**	2.01e+03/0	1.98e+03/0	2.75e+03/0	2.09e+03/1.39e+02	1.99e+03/0	2.09e+03/0	2.02e+03/2.39e+02
		200	3.80e+03/4.05e+02	3.51e+03/0	**3.11e+03/0**	4.60e+03/0	3.27e+03/2.67e+02	3.52e+03/0	3.17e+03/0	3.54e+03/3.31e+02
	50	50	**1.02e+03*/1.17e+02**	1.15e+03/0	1.47e+03/0	1.52e+03/0	1.27e+03/1.47e+02	1.29e+03/0	1.19e+03/0	1.37e+03/1.80e+02
		100	**1.94e+03/2.22e+02**	2.25e+03/0	2.03e+03/0	1.99e+03/0	2.41e+03/1.41e+02	2.15e+03/0	2.39e+03/0	2.20e+03/1.79e+02
		200	4.07e+03/3.51e+02	3.81e+03/0	3.74e+03/0	**3.68e+03*/0**	4.47e+03/1.59e+02	3.78e+03/0	4.37e+03/0	4.24e+03/3.02e+02

6.1. The training process of DQN

The proposed DQN is implemented in MATLAB® 2016b [50] on a PC with Intel Core i7-6700 @ 4.0 GHz CPU and 8 GB RAM. It is trained in a simulated flexible job shop with 30 machines, 20 initial jobs and 100 new job insertions. The average value of exponential distribution between two successive job insertions (E_{ave}) is 100. The *DDT* is set as 1.0. The parameter settings of the training method are listed in Table 3. After each training step, the DQN is tested on another predefined instance containing 30 machines, 100 new job insertions with E_{ave} set as 100 and *DDT* set as 1.0. The results of total tardiness obtained by the DQN in the first 8000 training steps are shown in Fig. 2. It can be seen that the curve of total tardiness drops smoothly with the increase of training steps, indicating that the proposed DQN has learned the proper dispatching rules for different situations in an efficient way.

Meanwhile, we have conducted a correlation analysis between the state features obtained in the training process, the results are given in Table 4.

6.2. Sensitivity study on the control parameter μ

The entropy control parameter μ in Eq. (11) plays an important role in affecting the performance of the DQN in practical implementation. More precisely, a large value of μ tends to choose the action with the highest Q value but might be too arbitrary, while a lower value of μ may lead to a completely random strategy without leveraging the learned knowledge. To determine a proper value of μ, we increase it from 1.0 to 2.0 with the step size of 0.1. Under each parameter level, the trained DQN is tested independently for 20 times on a instance with 30 machines and 100 new job insertions. The *DDT* and E_{ave} are set as 1.0 and 100, respectively. The box plots of total tardiness from 20 runs under different levels of μ are provided in Fig. 3 with the average values marked as red dots. It can be observed that $\mu = 1.6$ achieves a lower degree in both terms of distribution range and average value of total tardiness. Thus it can be concluded that $\mu = 1.6$ is recommended in this paper.

Table 7
Mean value and standard deviation of tardiness by DQN and each composite dispatching rule when $DDT = 1.5$ (mean/std).

E_{ave}	m	n_{add}	DQN	$Rule_1$	$Rule_2$	$Rule_3$	$Rule_4$	$Rule_5$	$Rule_6$	Random
50	10	50	**2.69e+03*/6.87e+02**	3.31e+03/0	4.44e+03/0	7.25e+03/0	7.94e+03/9.06e+02	7.17e+03/0	4.00e+03/0	5.61e+03/1.01e+03
		100	**0.17e+04*/5.97e+02**	0.25e+04/0	0.23e+04/0	1.07e+04/0	1.05e+04/1.53e+03	1.13e+04/0	0.31e+04/0	0.70e+04/1.54e+03
		200	0.28e+04/5.71e+02	**0.19e+04*/0**	0.25e+04/0	0.91e+04/0	1.06e+04/1.32e+03	1.34e+04/0	0.21e+04/0	0.54e+04/9.95e+02
	20	50	**0.01e+03/0.22e+02**	0.02e+03/0	0.05e+03/0	0.49e+03/0	1.11e+03/4.29e+02	0.20e+03/0	0.03e+03/0	0.14e+03/0.70e+02
		100	**0.23e+02*/0.25e+02**	2.35e+02/0	1.23e+02/0	2.67e+02/0	4.64e+02/2.31e+02	1.71e+02/0	0.37e+02/0	0.95e+02/0.52e+02
		200	**0.69e+02/0.35e+02**	2.92e+02/0	2.02e+02/0	0.79e+02/0	2.96e+02/1.23e+02	1.42e+02/0	1.35e+02/0	1.22e+02/0.64e+02
	30	50	**0.12e+02/0.10e+02**	0.19e+02/0	0.21e+02/0	0.14e+02/0	0.60e+02/0.57e+02	0.31e+02/0	0.15e+02/0	0.19e+02/0.10e+02
		100	0.66e+02/0.35e+02	1.97e+02/0	1.13e+02/0	0.36e+02/0	0.91e+02/0.72e+02	1.32e+02/0	**0.20e+02*/0**	1.46e+02/0.42e+02
		200	0.54e+02/0.27e+02	0.61e+02/0	0.55e+02/0	0.47e+02/0	1.22e+02/0.95e+02	**0.06e+02*/0**	0.87e+02/0	0.38e+02/0.27e+02
	40	50	0.20e+02/0.10e+02	**0.00e+02/0**	0.00e+02/0	0.10e+02/0	0.21e+02/0.19e+02	0.17e+02/0	0.09e+02/0	0.19e+02/0.10e+02
		100	**0.14e+02/0.12e+02**	0.79e+02/0	0.22e+02/0	0.27e+02/0	0.17e+02/0.15e+02	0.85e+02/0	0.16e+02/0	0.19e+02/0.19e+02
		200	**0.31e+02*/0.15e+02**	0.44e+02/0	0.64e+02/0	0.65e+02/0	0.75e+02/0.22e+02	0.62e+02/0	0.44e+02/0	0.66e+02/0.20e+02
	50	50	**0.05e+02*/0.04e+02**	0.12e+02/0	0.25e+02/0	0.08e+02/0	0.34e+02/0.20e+02	0.20e+02/0	0.47e+02/0	0.35e+02/0.15e+02
		100	0.27e+02/0.15e+02	0.25e+02/0	0.47e+02/0	**0.00e+02*/0**	0.47e+02/0.44e+02	0.25e+02/0	0.37e+02/0	0.30e+02/0.29e+02
		200	**0.47e+02/0.24e+02**	1.19e+02/0	0.56e+02/0	0.55e+02/0	1.29e+02/0.72e+02	0.77e+02/0	0.81e+02/0	0.85e+02/0.30e+02
100	10	50	0.73e+03/2.56e+02	**0.23e+03*/0**	0.38e+03/0	1.78e+03/0	2.12e+03/4.65e+02	0.52e+03/0	0.29e+03/0	0.82e+03/2.45e+02
		100	**0.52e+03*/3.35e+02**	0.95e+03/0	0.87e+03/0	1.37e+03/0	3.22e+03/6.13e+02	1.37e+03/0	0.77e+03/0	1.37e+03/5.40e+02
		200	2.46e+03/5.65e+02	**0.93e+03*/0**	2.14e+03/0	2.57e+03/0	3.98e+03/6.33e+02	3.08e+03/0	1.41e+03/0	2.69e+03/4.52e+02
	20	50	0.20e+02/0.12e+02	0.05e+02/0	**0.00e+02*/0**	0.17e+02/0	1.21e+02/1.06e+02	0.24e+02/0	0.31e+02/0	0.21e+02/0.15e+02
		100	**0.19e+02/0.15e+02**	1.29e+02/0	0.42e+02/0	0.33e+02/0	1.79e+02/1.02e+02	0.22e+02/0	0.20e+02/0	0.27e+02/0.14e+02
		200	**0.45e+02/0.25e+02**	2.45e+02/0	2.39e+02/0	0.57e+02/0	2.85e+02/1.88e+02	0.66e+02/0	0.49e+02/0	0.47e+02/0.14e+02
	30	50	0.15e+02/0.11e+02	0.11e+02/0	0.42e+02/0	**0.02e+02*/0**	0.47e+02/0.45e+02	0.05e+02/0	0.22e+02/0	0.14e+02/0.09e+02
		100	0.24e+02/0.12e+02	**0.01e+02*/0**	0.35e+02/0	0.59e+02/0	0.41e+02/0.37e+02	0.19e+02/0	0.25e+02/0	0.20e+02/0.17e+02
		200	**0.45e+02/0.17e+02**	0.57e+02/0	0.49e+02/0	0.62e+02/0	0.67e+02/0.31e+02	0.47e+02/0	0.67e+02/0	0.51e+02/0.20e+02
	40	50	**0.11e+02*/0.10e+02**	0.20e+02/0	0.19e+02/0	0.21e+02/0	0.33e+02/0.12e+02	0.27e+02/0	0.19e+02/0	0.23e+02/0.07e+02
		100	0.23e+02/0.17e+02	0.47e+02/0	0.61e+02/0	**0.06e+02*/0**	0.79e+02/0.29e+02	0.51e+02/0	0.61e+02/0	0.62e+02/0.24e+02
		200	**0.40e+02/0.19e+02**	0.42e+02/0	0.44e+02/0	0.51e+02/0	0.62e+02/0.37e+02	0.66e+02/0	0.77e+02/0	0.43e+02/0.15e+02
	50	50	**0.01e+02/0.02e+02**	0.02e+02/0	0.10e+02/0	0.02e+02/0	0.07e+02/0.06e+02	0.06e+02/0	0.04e+02/0	0.11e+02/0.10e+02
		100	0.15e+02/0.07e+02	0.18e+02/0	0.10e+02/0	**0.00e+02/0**	0.08e+02/0.06e+02	0.00e+02/0	0.04e+02/0	0.09e+02/0.09e+02
		200	**0.14e+02/0.13e+02**	0.67e+02/0	0.57e+02/0	0.19e+02/0	0.35e+02/0.12e+02	0.42e+02/0	0.32e+02/0	0.39e+02/0.24e+02
200	10	50	**0.10e+03/1.01e+02**	0.12e+03/0	0.29e+03/0	0.95e+03/0	1.52e+03/3.92e+02	1.10e+03/0	0.75e+03/0	0.61e+03/2.55e+02
		100	**0.17e+03/1.81e+02**	0.21e+03/0	0.25e+03/0	1.51e+03/0	1.79e+03/5.39e+02	0.51e+03/0	0.37e+03/0	0.76e+03/3.59e+02
		200	3.80e+03/3.05e+02	3.51e+03/0	**3.02e+03*/0**	4.60e+03/0	3.27e+03/2.27e+02	3.52e+03/0	3.17e+03/0	3.54e+03/3.31e+02
	20	50	**0.03e+02*/0.02e+02**	0.06e+02/0	0.13e+02/0	0.11e+02/0	1.14e+02/0.72e+02	0.09e+02/0	0.13e+02/0	0.19e+02/0.14e+02
		100	0.32e+02/0.12e+02	0.52e+02/0	0.36e+02/0	0.23e+02/0	1.99e+02/1.01e+02	0.13e+02/0	**0.04e+02*/0**	0.57e+02/0.42e+02
		200	**0.71e+02*/0.14e+02**	1.14e+02/0	1.17e+02/0	1.57e+02/0	2.24e+02/1.02e+02	0.85e+02/0	1.25e+02/0	0.99e+02/0.16e+02
	30	50	**0.09e+02*/0.05e+02**	0.42e+02/0	0.29e+02/0	0.15e+02/0	0.45e+02/0.32e+02	0.53e+02/0	0.29e+02/0	0.26e+02/0.11e+02
		100	**0.19e+02*/0.07e+02**	0.91e+02/0	0.86e+02/0	0.27e+02/0	0.93e+02/0.24e+02	0.81e+02/0	0.74e+02/0	0.62e+02/0.19e+02
		200	0.30e+02/0.12e+02	0.32e+02/0	**0.03e+02*/0**	0.50e+02/0	0.10e+02/0.06e+02	0.08e+02/0	0.07e+02/0	0.15e+02/0.08e+02
	40	50	**0.05e+02/0.06e+02**	0.18e+02/0	0.17e+02/0	0.07e+02/0	0.43e+02/0.32e+02	0.17e+02/0	0.17e+02/0	0.15e+02/0.05e+02
		100	**0.12e+02/0.07e+02**	0.14e+02/0	0.17e+02/0	0.36e+02/0	0.22e+02/0.19e+02	0.64e+02/0	0.18e+02/0	0.24e+02/0.15e+02
		200	**0.41e+02/0.13e+02**	0.66e+02/0	0.81e+02/0	0.61e+02/0	0.77e+02/0.17e+02	0.57e+02/0	0.81e+02/0	0.52e+02/0.17e+02
	50	50	0.04e+02/0.05e+02	0.27e+02/0	0.24e+02/0	0.72e+02/0	0.17e+02/0.11e+02	**0.00e+02*/0**	0.09e+02/0	0.12e+02/0.10e+02
		100	**0.11e+02/0.07e+02**	0.23e+02/0	0.17e+02/0	0.20e+02/0	0.32e+02/0.17e+02	0.19e+02/0	0.24e+02/0	0.15e+02/0.14e+02
		200	**0.12e+02/0.08e+02**	0.15e+02/0	0.14e+02/0	0.27e+02/0	0.36e+02/0.12e+02	0.18e+02/0	0.27e+02/0	0.35e+02/0.20e+02

6.3. Comparisons with the proposed composite dispatching rules

To verify the effectiveness of the proposed DQN, we compare the performance of DQN with each composite rule used in this paper. In order to obtain convincing results, a random action selection strategy, i.e., randomly select a rule at each rescheduling point, is also taken into account. A wide range of production configurations in different parameter settings of DDT, E_{ave}, m and n_{add} are generated so as to investigate the generality of DQN. On each instance, the DQN and each composite rule are repeated independently for 20 times. The mean values and standard deviations of total tardiness obtained by each method are provided in Tables 5–7 with the best results highlighted in bold font. Meanwhile, for each test instance, we have run pair-wise t tests with 5% significance level to determine if there exists significant difference between the best result and the inferior results. The null hypothesis is that the two data vectors of total tardiness obtained by two comparative methods among 20 independent runs are from populations with equal means. The superscript "*" denotes that the corresponding best result is significantly better than (i.e. the mean value of population where it comes from is significantly lower than) all of the results obtained by other methods.

It can be seen from the results that the performance of the DQN and other dispatching rules can be affected by different production configurations. As can be expected, the tardiness increases with tighter due date, more frequent arrivals of new jobs, as well as more number of new job insertions. On the other hand, the tardiness can be reduced by placing more machines in the shop floor.

Compared with the random action selection policy, the DQN can obtain lower total tardiness in almost all instances, meaning that the DQN has learned the correct strategy to determine the proper dispatching rules at different rescheduling points. Compared with each single dispatching rule, the DQN demonstrates better performance in most instances. Only for some special cases, the DQN is outperformed by other rules. However, there does not exist any single dispatching rule, as like DQN, can achieve the best performance in most instances under different production environments. This confirms the generality of DQN in

various of untrained situations. Moreover, note that the rule 4, which is the simplest one among all candidate composite rules, can never obtain the best result. Thus it can be concluded that the good design and reasonable choice of each dispatching rule are the dominant factors in affecting the performance of a schedule. Fig. 4 illustrates the winning rate of each method under different due date tightness, where the winning rate is defined as the number of instances in which the method achieves the best result divided by the total number of instances. It can be seen that the DQN outperforms other composite rules for 61.5% of the test instances on average.

To summary, the DQN can select the most suitable dispatching rule at each rescheduling point by comprehensively investigating the current production status, due to which it is more effective and generic than a single dispatching rule.

6.4. Comparisons with stand Q-learning agent

Next, in order to verify the superiority of DQN compared to stand Q-learning agent over discrete state space, we use a neural network with self-organizing map (SOM) layer to partition the features into 9 discrete states. A 9×6 Q table is maintained to store the Q-values of all state–action pairs and a RL agent is trained by stand Q-learning. The Q-learning agent possesses the same set of available rules used in this paper at each state. We compare the performance of stand Q-learning agent and DQN agent over different production configurations with $DDT = 1.0$, the results are given in Table 8.

It can be seen that the DQN agent outperforms the stand Q-learning agent for 38 out of the 45 test instances. This indicates that the compulsive discretization of continuous state features is too rough which fails to accurately distinguish different production statuses of DFJSP thus inevitably deteriorating the overall performance of a schedule. In conclusion, a DQN agent is more reasonable and effective compared to stand Q-learning agent in handling continuous state space.

6.5. Comparisons with other well-known dispatching rules

In order to further confirm the superiority of DQN, we compare the DQN with other five well-known dispatching rules, including first in first out (FIFO), earliest due date (EDD), most remaining processing time (MRT), shortest processing time (SPT) and longest processing time (LPT). FIFO chooses the next operation of the earliest arriving job. EDD chooses the next operation of the job with the earliest due date. MRT chooses the next operation of the job with most remaining processing time. SPT chooses the next operation among the existing jobs with the shortest processing time. LPT chooses the next operation among the existing jobs with the longest processing time. In addition, a totally random rule, i.e., randomly choose an unprocessed operation and assign it on an available machine, is also taken as comparison. Note that the aforementioned dispatching rules (except random rule) do not explicitly determine a processing machine hence are inadequate to handle the FJSP in this paper. To address this issue, for the five well-known dispatching rules, we assign the selected operation on the earliest available machine so as to reduce total tardiness and make fair comparisons. Moreover, for MRT, SPT and LPT, the processing time of an operation is approximated by taking the average value of its processing times among all available machines.

We test the DQN and other comparative rules on different instances with 20 independent replications for each instance. The mean values and standard deviations of total tardiness obtained by each method are given in Tables 9–11.

Table 8
Mean value and standard deviation of tardiness by DQN and stand Q-learning agent when $DDT = 1.0$ (mean/std).

E_{ave}	m	n_{add}	DQN	Stand Q-learning
50	10	50	**1.23e+04*/1.27e+03**	1.34e+04/0
		100	**1.90e+04*/2.22e+03**	2.87e+04/0
		200	**2.31e+04*/1.74e+03**	4.22e+04/0
	20	50	**3.27e+03*/4.19e+02**	4.67e+03/0
		100	**3.23e+03*/2.88e+02**	3.56e+03/0
		200	**6.99e+03*/5.57e+02**	8.19e+03/0
	30	50	**1.77e+03*/2.98e+02**	2.13e+03/0
		100	**3.01e+03*/4.08e+02**	3.44e+03/0
		200	**4.75e+03*/5.30e+02**	5.72e+03/0
	40	50	**1.78e+03/3.14e+02**	1.83e+03/0
		100	**2.36e+03*/4.07e+02**	3.29e+03/0
		200	4.76e+03/4.23e+02	**4.32e+03*/0**
	50	50	**1.59e+03*/1.67e+02**	1.75e+03/0
		100	**2.32e+03*/1.77e+02**	3.00e+03/0
		200	4.89e+03/5.25e+02	**4.88e+03/0**
100	10	50	**6.56e+03*/2.94e+02**	6.82e+03/0
		100	5.81e+03/5.23e+02	**5.72e+03/0**
		200	**1.13e+04*/7.60e+02**	1.27e+04/0
	20	50	**2.04e+03*/3.29e+02**	3.06e+03/0
		100	**2.84e+03*/3.49e+02**	3.32e+03/0
		200	**4.47e+03*/4.45e+02**	4.87e+03/0
	30	50	**1.79e+03/2.43e+02**	1.84e+03/0
		100	2.91e+03/3.92e+02	**2.54e+03*/0**
		200	**4.43e+03*/4.66e+02**	5.94e+03/0
	40	50	**1.30e+03*/1.97e+02**	1.60e+03/0
		100	**2.41e+03/2.94e+02**	2.45e+03/0
		200	**4.16e+03/4.08e+02**	4.28e+03/0
	50	50	**1.31e+03*/2.15e+02**	1.50e+03/0
		100	**2.32e+03*/2.81e+02**	2.60e+03/0
		200	**4.34e+03/3.04e+02**	4.45e+03/0
200	10	50	**3.64e+03*/5.28e+02**	4.29e+03/0
		100	**5.59e+03*/4.74e+02**	6.47e+03/0
		200	7.96e+03/3.15e+02	**7.77e+03*/0**
	20	50	**1.67e+03*/2.82e+02**	1.98e+03/0
		100	**2.39e+03*/3.34e+02**	2.70e+03/0
		200	4.27e+03/4.23e+02	**3.81e+03*/0**
	30	50	**1.07e+03*/2.13e+02**	1.26e+03/0
		100	**1.76e+03*/1.62e+02**	2.08e+03/0
		200	**4.12e+03*/3.26e+02**	4.53e+03/0
	40	50	1.25e+03/2.26e+02	**1.22e+03/0**
		100	**2.06e+03*/2.70e+02**	2.39e+03/0
		200	**3.76e+03*/3.59e+02**	4.15e+03/0
	50	50	**1.29e+03/2.22e+02**	1.36e+03/0
		100	**2.29e+03*/2.21e+02**	2.50e+03/0
		200	**3.53e+03*/3.11e+02**	3.90e+03/0

It can be first observed that the totally random rule can never achieve the best result among the total 45 instances, which confirms the necessity in adopting the DQN. Meanwhile, the results demonstrate that the proposed DQN is superior to other well-known dispatching rules for most instances. Particularly, the DQN achieves the best performance when DDT is 1.5. Note that in real manufacturing enterprises, the due dates are always not that tight so as to guarantee on-time delivery. Thus it can be concluded that the DQN is more effective compared with other common dispatching rules in real-world application.

Fig. 5 illustrates the winning rate of each algorithm under different due date tightness, from which we can see that the DQN outperforms other well-known dispatching rules for 83.7% of the test instances on average.

7. Conclusion

In this paper, a deep Q network (DQN) is proposed to address the DFJSP with new job insertions. Six composite dispatching

Table 9
Mean value and standard deviation of tardiness by DQN and other well-known dispatching rules when DDT = 0.5.

E_{ave}	m	n_{add}	DQN	FIFO	MRT	EDD	SPT	LPT	Random
50	10	50	**2.29e+04*/5.05e+02**	3.29e+04/2.37e+03	3.69e+04/0	3.45e+04/0	5.47e+04/0	6.51e+04/0	4.44e+04/2.70e+03
		100	**0.33e+05*/7.32e+02**	0.55e+05/3.97e+03	0.70e+05/0	0.62e+05/0	1.59e+05/0	1.67e+05/0	0.67e+05/3.29e+03
		200	**0.55e+05*/1.06e+03**	0.81e+05/4.52e+03	0.88e+05/0	0.84e+05/0	3.70e+05/0	4.24e+05/0	1.12e+05/6.41e+03
	20	50	**1.22e+04*/5.33e+02**	1.35e+04/7.29e+02	1.90e+04/0	1.42e+04/0	5.61e+04/0	6.42e+04/0	2.34e+04/1.45e+03
		100	**1.71e+04*/6.42e+02**	1.86e+04/6.92e+02	2.25e+04/0	2.03e+04/0	9.42e+04/0	9.94e+04/0	3.31e+04/2.52e+03
		200	**0.32e+05*/6.71e+02**	0.34e+05/3.86e+02	0.35e+05/0	0.35e+05/0	2.79e+05/0	3.29e+05/0	0.57e+05/2.65e+03
	30	50	**0.90e+04*/2.96e+02**	0.97e+04/2.75e+02	1.03e+04/0	0.94e+04/0	2.11e+04/0	2.19e+04/0	1.52e+04/1.09e+03
		100	**1.62e+04*/4.41e+02**	1.86e+04/2.75e+02	1.92e+04/0	1.71e+04/0	9.42e+04/0	8.64e+04/0	2.76e+04/1.37e+03
		200	**0.23e+05*/5.65e+02**	0.27e+05/2.85e+02	0.28e+05/0	0.27e+05/0	2.26e+05/0	1.87e+05/0	0.43e+05/1.20e+03
	40	50	**1.02e+04*/3.92e+02**	1.09e+04/3.32e+02	1.19e+04/0	1.07e+04/0	2.23e+04/0	2.47e+04/0	1.67e+04/8.15e+02
		100	1.63e+04/4.50e+02	1.59e+04/3.59e+02	1.65e+04/0	**1.51e+04*/0**	5.22e+04/0	4.55e+04/0	2.42e+04/1.47e+03
		200	**0.22e+05*/5.31e+02**	0.27e+05/3.75e+02	0.29e+05/0	0.27e+05/0	2.17e+05/0	1.44e+05/0	0.40e+05/1.49e+03
	50	50	**0.97e+04*/3.19e+02**	1.04e+04/2.81e+02	1.07e+04/0	1.10e+04/0	1.91e+04/0	1.45e+04/0	1.53e+04/1.12e+03
		100	1.61e+04/5.19e+02	1.55e+04/2.45e+02	**1.54e+04/0**	1.65e+04/0	4.77e+04/0	8.36e+04/0	2.19e+04/7.33e+02
		200	**0.23e+05*/5.71e+02**	0.27e+05/1.92e+02	0.28e+05/0	0.29e+05/0	1.79e+05/0	1.41e+05/0	0.39e+05/1.12e+03
100	10	50	1.60e+04/6.10e+02	1.64e+04/7.91e+02	1.83e+04/0	**1.49e+04*/0**	4.97e+04/0	3.70e+04/0	2.52e+04/1.96e+03
		100	**2.12e+04*/6.09e+02**	2.36e+04/5.15e+02	2.71e+04/0	2.52e+04/0	4.55e+04/0	5.42e+04/0	3.62e+04/1.45e+03
		200	**0.37e+05*/6.97e+02**	0.42e+05/8.39e+02	0.45e+05/0	0.47e+05/0	0.89e+05/0	1.22e+05/0	0.62e+05/1.61e+03
	20	50	**1.02e+04*/3.69e+02**	1.12e+04/3.62e+02	1.25e+04/0	1.17e+04/0	1.54e+04/0	2.01e+04/0	1.64e+04/8.59e+02
		100	**1.75e+04/4.22e+02**	1.79e+04/3.15e+02	1.98e+04/0	1.77e+04/0	2.61e+04/0	3.04e+04/0	2.47e+04/9.62e+02
		200	3.16e+04/5.32e+02	**3.14e+04/4.09e+02**	3.43e+04/0	3.27e+04/0	8.97e+04/0	7.58e+04/0	4.22e+04/1.22e+03.
	30	50	**0.98e+04/4.17e+02**	0.99e+04/3.17e+02	1.14e+04/0	1.00e+04/0	1.83e+04/0	1.35e+04/0	1.36e+04/7.02e+02
		100	**1.81e+04/4.05e+02**	1.83e+04/2.37e+02	1.93e+04/0	1.87e+04/0	4.75e+04/0	2.49e+04/0	2.35e+04/1.31e+03
		200	**2.75e+04*/4.78e+02**	2.95e+04/2.92e+02	3.10e+04/0	2.97e+04/0	3.19e+04/0	3.68e+04/0	3.67e+04/1.38e+03
	40	50	**0.92e+04*/3.46e+02**	1.04e+04/2.57e+02	1.15e+04/0	0.99e+04/0	1.15e+04/0	1.10e+04/0	1.42e+04/9.05e+02
		100	1.60e+04/4.52e+02	1.56e+04/2.41e+02	1.65e+04/0	**1.50e+04*/0**	1.67e+04/0	1.74e+04/0	2.02e+04/9.42e+02
		200	**0.25e+05*/5.87e+02**	0.29e+05/1.90e+02	0.29e+05/0	0.30e+05/0	0.29e+05/0	1.36e+05/0	0.35e+05/6.76e+02
	50	50	**0.99e+04*/2.90e+02**	1.07e+04/2.18e+02	1.11e+04/0	1.07e+04/0	1.49e+04/0	1.18e+04/0	1.36e+04/7.52e+02
		100	**1.62e+04/5.72e+02**	1.65e+04/1.84e+02	1.67e+04/0	1.64e+04/0	1.79e+04/0	1.64e+04/0	1.97e+02/7.67e+02
		200	2.81e+04/5.85e+02	2.85e+04/1.62e+02	2.91e+04/0	2.81e+04/0	**2.79e+04/0**	2.84e+04/0	3.24e+04/1.03e+03
200	10	50	**1.41e+04*/4.49e+02**	1.59e+04/6.49e+02	1.77e+04/0	1.67e+04/0	1.83e+04/0	1.76e+04/0	2.05e+04/1.07e+03
		100	**2.05e+04/6.40e+02**	2.07e+04/2.99e+02	2.34e+04/0	2.12e+04/0	2.44e+04/0	2.47e+04/0	2.64e+04/1.05e+03
		200	**3.72e+04*/5.31e+02**	3.85e+04/6.20e+02	4.41e+04/0	3.97e+04/0	4.89e+04/0	4.77e+04/0	4.72e+04/1.17e+03
	20	50	**1.03e+04/5.69e+02**	1.04e+04/2.69e+02	1.25e+04/0	1.11e+04/0	1.12e+04/0	1.13e+04/0	1.39e+04/6.61e+02
		100	**1.71e+04*/5.31e+02**	1.79e+04/3.11e+02	2.01e+04/0	1.85e+04/0	1.79e+04/0	1.77e+04/0	2.03e+04/9.24e+02
		200	**3.00e+04/5.84e+02**	3.02e+04/2.75e+02	3.14e+04/0	3.13e+04/0	3.22e+04/0	3.18e+04/0	3.45e+04/9.65e+02
	30	50	**0.81e+04*/4.35e+02**	0.87e+04/1.42e+02	0.91e+04/0	0.88e+04/0	0.89e+04/0	0.91e+04/0	1.10e+04/5.67e+02
		100	**1.52e+04*/3.35e+02**	1.60e+04/2.62e+02	1.71e+04/0	1.63e+04/0	1.64e+04/0	1.69e+04/0	1.87e+04/5.62e+02
		200	**3.01e+04*/4.07e+02**	3.09e+04/2.75e+02	3.17e+04/0	3.10e+04/0	3.17e+04/0	3.22e+04/0	3.59e+04/7.10e+02
	40	50	0.95e+04/2.57e+02	0.93e+04/1.90e+02	0.94e+04/0	0.97e+04/0	**0.92e+04/0**	0.96e+04/0	1.12e+04/6.32e+02
		100	**1.55e+04*/4.37e+02**	1.64e+04/2.32e+02	1.70e+04/0	1.65e+04/0	1.64e+04/0	1.63e+04/0	1.84e+04/5.24e+02
		200	**3.03e+04/4.64e+02**	3.06e+04/2.26e+02	3.10e+04/0	3.05e+04/0	3.06e+04/0	3.12e+04/0	3.25e+04/8.47e+02
	50	50	**0.89e+04/3.22e+02**	0.92e+04/1.92e+02	0.95e+04/0	0.90e+04/0	0.96e+04/0	0.95e+04/0	1.08e+04/4.52e+02
		100	1.57e+04/4.20e+02	**1.54e+04/1.72e+02**	1.55e+04/0	1.56e+04/0	1.60e+04/0	1.60e+04/0	1.71e+04/4.97e+02
		200	**2.82e+04*/5.71e+02**	2.97e+04/2.12e+02	3.00e+04/0	2.99e+04/0	2.95e+04/0	2.94e+04/0	3.09e+04/8.55e+02

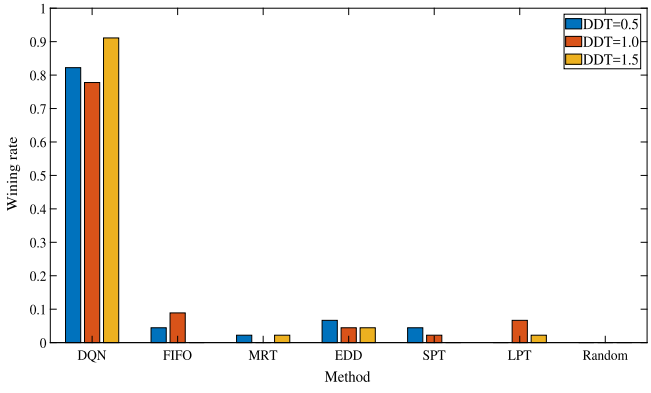

Fig. 5. Winning rate of DQN and other well-known dispatching rules under different due date tightness.

rules are developed to select an unprocessed operation and assign it on an available machine every time an operation is completed or a new job arrives. Seven elaborately-designed features normalized in the range of [0, 1] are utilized to represent a state at each rescheduling point. To determine the most suitable dispatching rule at each rescheduling point, the DQN is trained by deep Q-learning (DQL) combined with two improvements including double DQN and soft target weight update. Finally, a "softmax" action selection policy is used in real implementation of the DQN so that the action with higher Q-value can be chosen with higher probability.

Numerical experiments under different production environments are conducted to verify the effectiveness and generality of the proposed DQN. The results demonstrate that the DQN performs significantly better than the proposed composite dispatching rules and other well-known dispatching rules for both trained and untrained production configurations. Meanwhile, the comparisons between DQN and stand Q-learning agent has further verified the superiority of DQN in handling continuous state space.

Table 10
Mean value and standard deviation of tardiness by DQN and other well-known dispatching rules when DDT = 1.0.

E_{ave}	m	n_{add}	DQN	FIFO	MRT	EDD	SPT	LPT	Random
50	10	50	**1.32e+04*/1.02e+03**	2.74e+04/1.89e+03	3.79e+04/0	3.11e+04/0	6.21e+04/0	7.05e+04/0	3.84e+04/3.19e+03
		100	**0.14e+05*/1.09e+03**	0.25e+05/2.82e+03	0.42e+05/0	0.35e+05/0	0.89e+05/0	1.25e+05/0	0.45e+05/4.02e+03
		200	**0.19e+05*/1.59e+03**	0.50e+05/3.84e+03	0.52e+05/0	0.51e+05/0	2.20e+05/0	3.42e+05/0	0.69e+05/4.19e+03
	20	50	**0.19e+04*/4.37e+02**	0.29e+04/3.41e+02	0.59e+04/0	0.45e+04/0	4.43e+04/0	3.68e+04/0	1.10e+04/1.46e+03
		100	**0.05e+05*/4.94e+02**	0.07e+05/6.53e+02	0.11e+05/0	0.10e+05/0	1.15e+05/0	0.91e+05/0	0.22e+05/1.99e+03
		200	**0.06e+05*/5.39e+02**	0.08e+05/3.97e+02	0.12e+05/0	0.09e+05/0	2.37e+05/0	2.94e+05/0	0.29e+05/2.74e+03
	30	50	**0.18e+04/1.92e+02**	0.19e+04/1.71e+02	0.20e+04/0	0.23e+04/0	2.46e+04/0	1.64e+04/0	0.72e+04/8.87e+02
		100	**0.24e+04*/3.72e+02**	0.31e+04/2.49e+02	0.51e+04/0	0.36e+04/0	9.32e+04/0	8.75e+04/0	1.23e+04/1.52e+03
		200	**0.03e+05*/5.72e+02**	0.05e+05/1.68e+02	0.07e+05/0	0.06e+05/0	2.74e+05/0	3.84e+05/0	0.17e+05/1.90e+03
	40	50	**0.16e+04/2.40e+02**	0.17e+04/2.44e+02	0.25e+04/0	0.18e+04/0	1.04e+04/0	0.81e+04/0	0.65e+04/9.52e+02
		100	0.24e+04/3.17e+02	**0.22e+04/1.60e+02**	0.37e+04/0	0.28e+04/0	0.77e+04/0	4.26e+04/0	0.80e+04/9.69e+02
		200	**0.03e+05*/3.65e+02**	0.04e+05/2.05e+02	0.06e+05/0	0.05e+05/0	2.25e+05/0	1.57e+05/0	0.13e+05/1.15e+03
	50	50	**1.34e+03/2.12e+02**	1.51e+03/1.46e+02	1.63e+03/0	1.39e+03/0	8.89e+03/0	5.17e+03/0	4.39e+03/6.25e+02
		100	0.29e+04/3.72e+02	**0.24e+04/1.52e+02**	0.27e+04/0	0.25e+04/0	5.67e+04/0	5.59e+04/0	0.85e+04/8.45e+02
		200	**0.02e+05*/2.42e+02**	0.03e+05/2.08e+02	0.04e+05/0	0.03e+05/0	1.52e+05/0	0.84e+05/0	0.11e+05/1.19e+03
100	10	50	**0.68e+04*/4.05e+02**	0.91e+04/6.02e+02	1.14e+04/0	0.95e+04/0	2.64e+04/0	2.99e+04/0	1.62e+04/1.05e+03
		100	**0.57e+04*/3.91e+02**	0.71e+04/4.32e+02	0.99e+04/0	0.68e+04/0	1.54e+04/0	1.77e+04/0	1.53e+04/1.39e+03
		200	**0.81e+04*/5.97e+02**	0.97e+04/5.62e+02	1.22e+04/0	1.12e+04/0	4.17e+04/0	5.90e+04/0	2.32e+04/1.75e+03
	20	50	**0.21e+04*/3.04e+02**	0.29e+04/3.65e+02	0.53e+04/0	0.47e+04/0	0.75e+04/0	1.03e+04/0	0.84e+04/9.81e+02
		100	**0.30e+04/4.49e+02**	0.34e+04/2.82e+02	0.52e+04/0	0.31e+04/0	1.10e+04/0	0.65e+04/0	0.85e+04/8.57e+02
		200	**0.45e+04*/4.49e+02**	0.59e+04/2.69e+02	0.86e+04/0	0.67e+04/0	2.52e+04/0	5.95e+04/0	1.45e+04/1.31e+03
	30	50	1.67e+03/2.68e+02	1.86e+03/2.36e+02	2.96e+03/0	**1.42e+03*/0**	5.99e+03/0	2.47e+03/0	5.32e+03/4.42e+02
		100	**2.82e+03/4.10e+02**	2.97e+03/2.07e+02	5.45e+03/0	3.65e+03/0	6.65e+03/0	5.93e+03/0	8.43e+03/9.52e+02
		200	**0.39e+04*/4.02e+02**	0.47e+04/1.57e+02	0.55e+04/0	0.49e+04/0	1.75e+04/0	3.85e+04/0	0.96e+04/8.52e+02
	40	50	**1.17e+03/2.27e+02**	1.21e+03/1.47e+02	1.41e+03/0	1.72e+03/0	1.75e+03/0	1.32e+03/0	3.29e+03/4.29e+02
		100	**0.25e+04/2.62e+02**	0.27e+04/2.16e+02	0.32e+04/0	0.29e+04/0	2.03e+04/0	2.49e+04/0	0.61e+04/7.12e+02
		200	0.40e+04/3.47e+02	**0.32e+04*/1.72e+02**	0.37e+04/0	0.38e+04/0	0.75e+04/0	1.19e+04/0	0.81e+04/8.92e+02
	50	50	**1.50e+03/2.99e+02**	1.64e+03/2.72e+02	2.04e+03/0	1.75e+03/0	3.02e+03/0	1.72e+03/0	3.66e+03/5.51e+02
		100	**2.21e+03/3.57e+02**	2.35e+03/1.12e+02	2.55e+03/0	2.39e+03/0	2.69e+03/0	4.41e+03/0	4.89e+03/6.41e+02
		200	4.52e+03/4.14e+02	5.24e+03/1.61e+02	5.21e+03/0	5.29e+03/0	6.67e+03/0	**4.37e+03/0**	7.42e+03/6.91e+02
200	10	50	**3.99e+03/3.22e+02**	4.47e+03/3.64e+02	6.73e+03/0	4.05e+03/0	6.87e+03/0	5.72e+03/0	7.69e+03/7.16e+02
		100	**0.49e+04*/3.29e+02**	0.64e+04/4.57e+02	0.97e+04/0	0.72e+04/0	0.94e+04/0	0.80e+04/0	1.09e+04/8.92e+02
		200	**0.71e+04*/5.87e+02**	0.82e+04/3.99e+02	0.98e+04/0	0.87e+04/0	1.28e+04/0	1.29e+04/0	1.32e+04/1.18e+03
	20	50	**1.74e+03*/2.16e+02**	2.15e+03/2.15e+02	4.03e+03/0	2.29e+03/0	2.70e+03/0	1.97e+03/0	4.59e+03/5.20e+02
		100	**2.82e+03*/2.62e+02**	3.55e+03/2.55e+02	4.85e+03/0	4.38e+03/0	4.36e+03/0	4.51e+03/0	6.62e+03/6.04e+02
		200	**0.45e+04/2.20e+02**	0.47e+04/2.91e+02	0.71e+04/0	0.54e+04/0	0.59e+04/0	0.62e+04/0	1.01e+04/8.22e+02
	30	50	1.41e+03/2.10e+02	1.51e+03/1.10e+02	2.82e+03/0	1.49e+03/0	1.80e+03/0	**1.21e+03*/0**	4.12e+03/4.59e+02
		100	2.29e+03/2.02e+02	2.17e+03/1.00e+02	2.65e+03/0	2.32e+03/0	**2.07e+03*/0**	2.35e+03/0	4.41e+03/5.36e+02
		200	**3.79e+03*/3.14e+02**	4.37e+03/2.05e+02	4.85e+03/0	4.21e+03/0	5.05e+03/0	4.99e+03/0	7.54e+03/6.25e+02
	40	50	**1.19e+03*/1.60e+02**	1.56e+03/1.65e+02	1.67e+03/0	1.37e+03/0	1.55e+03/0	1.56e+03/0	2.51e+03/3.34e+02
		100	**2.23e+03/2.31e+02**	2.27e+03/1.03e+02	2.79e+03/0	2.35e+03/0	3.11e+03/0	2.67e+03/0	4.45e+03/4.55e+02
		200	3.71e+03/3.20e+02	3.92e+03/0.77e+02	4.05e+03/0	**3.63e+03/0**	3.75e+03/0	3.82e+03/0	6.80e+03/6.76e+02
	50	50	1.23e+03/2.20e+02	1.11e+03/0.79e+02	1.07e+03/0	1.08e+03/0	1.11e+03/0	**0.95e+03*/0**	2.38e+03/4.85e+02
		100	**1.91e+03/2.72e+02**	2.15e+03/1.35e+02	2.04e+03/0	1.97e+03/0	2.06e+03/0	2.36e+03/0	3.49e+03/3.69e+02
		200	4.37e+03/2.92e+02	**4.07e+03/1.29e+02**	4.31e+03/0	4.13e+03/0	4.35e+03/0	4.08e+03/0	6.02e+03/5.10e+02

Table 11
Mean value and standard deviation of tardiness by DQN and other well-known dispatching rules when DDT = 1.5.

E_{ave}	m	n_{add}	DQN	FIFO	MRT	EDD	SPT	LPT	Random
50	10	50	**1.23e+04*/1.14e+03**	2.55e+04/2.73e+03	4.02e+04/0	3.03e+04/0	6.49e+04/0	6.35e+04/0	3.75e+04/3.21e+03
		100	**0.11e+05*/2.08e+03**	0.42e+05/3.88e+03	0.43e+05/0	0.42e+05/0	1.25e+05/0	1.49e+05/0	0.41e+05/3.95e+03
		200	**0.06e+05*/2.15e+03**	0.25e+05/5.02e+03	0.42e+05/0	0.27e+05/0	2.57e+05/0	3.20e+05/0	0.51e+05/4.37e+03
	20	50	**0.01e+04*/0.29e+02**	0.07e+04/2.19e+02	0.15e+04/0	0.06e+04/0	3.00e+04/0	4.21e+04/0	0.62e+04/1.17e+03
		100	**0.00e+05*/0.29e+02**	0.03e+05/1.87e+02	0.02e+05/0	0.02e+05/0	1.12e+05/0	1.32e+05/0	0.08e+05/1.15e+03
		200	**0.00e+05*/0.25e+02**	0.05e+05/2.01e+02	0.04e+05/0	0.03e+05/0	2.49e+05/0	1.47e+05/0	0.08e+05/9.95e+02
	30	50	**0.00e+04*/0.11e+02**	0.02e+04/0.19e+02	0.03e+04/0	0.02e+04/0	1.56e+04/0	1.12e+04/0	0.24e+04/5.23e+02
		100	**0.00e+05*/0.12e+02**	0.02e+05/0.42e+02	0.01e+05/0	0.01e+05/0	1.27e+05/0	0.45e+05/0	0.04e+05/1.14e+03
		200	**0.00e+05*/0.22e+02**	0.04e+05/0.38e+02	0.03e+05/0	0.02e+05/0	2.11e+05/0	2.02e+05/0	0.04e+05/1.03e+03

(*continued on next page*)

In future work, more uncertain factors such as machine breakdowns and processing time variations will be studied. Other objectives like production costs and energy consumption are also worthy to be considered. Meanwhile, it should be noted that the DQN used in this paper is inherently a value based method, which cannot directly optimize over the policy. On this account, we will investigate other state-of-the-art policy based RL methods including A3C, TRPO, PPO and compare their performances with DQN.

Table 11 (*continued*).

E_{ave}	m	n_{add}	DQN	FIFO	MRT	EDD	SPT	LPT	Random
		50	**0.00e+04*/0.07e+02**	0.01e+04/0.08e+02	0.02e+04/0	0.01e+04/0	0.66e+04/0	1.08e+04/0	0.11e+04/4.55e+02
	40	100	**0.00e+04*/0.15e+02**	0.03e+04/0.37e+02	0.05e+04/0	0.02e+04/0	6.21e+04/0	1.32e+04/0	0.12e+04/3.08e+02
		200	**0.00e+05*/0.26e+02**	0.03e+05/0.22e+02	0.02e+05/0	0.01e+05/0	1.79e+05/0	1.08e+05/0	0.02e+05/8.15e+02
		50	**0.02e+03/0.14e+02**	0.03e+03/0.15e+02	0.06e+03/0	0.11e+03/0	4.29e+03/0	1.32e+03/0	0.60e+03/2.69e+02
	50	100	**0.00e+04*/0.17e+02**	0.02e+04/0.24e+02	0.01e+04/0	0.01e+04/0	8.24e+04/0	4.65e+04/0	0.10e+04/4.81e+02
		200	0.00e+05/0.27e+02	0.02e+05/0.12e+02	0.01e+05/0	**0.00e+05/0**	0.43e+05/0	1.31e+05/0	0.01e+05/4.94e+02
		50	**0.07e+04*/3.52e+02**	0.36e+04/5.45e+02	1.00e+04/0	0.45e+04/0	1.62e+04/0	3.45e+04/0	0.91e+04/1.17e+03
	10	100	**0.20e+04*/3.41e+02**	0.45e+04/5.25e+02	1.09e+04/0	0.44e+04/0	4.00e+04/0	3.07e+04/0	0.93e+04/9.39e+02
		200	**0.14e+04*/2.26e+02**	0.59e+04/8.65e+02	1.52e+04/0	0.90e+04/0	6.37e+04/0	5.11e+04/0	1.59e+04/1.57e+03
		50	**0.01e+03*/0.12e+02**	0.21e+03/0.71e+02	1.35e+03/0	0.26e+03/0	2.71e+03/0	5.61e+03/0	1.65e+03/4.71e+02
	20	100	**0.00e+04*/0.21e+02**	0.04e+04/2.69e+02	0.24e+04/0	0.09e+04/0	2.59e+04/0	3.69e+04/0	0.32e+04/8.61e+02
		200	**0.01e+04*/0.28e+02**	0.05e+04/1.87e+02	0.27e+04/0	0.11e+04/0	5.39e+04/0	4.04e+04/0	0.35e+04/9.70e+02
100		50	**0.01e+03*/0.21e+02**	0.09e+03/0.52e+02	0.72e+03/0	0.09e+03/0	1.65e+03/0	4.49e+03/0	1.22e+03/4.41e+02
	30	100	**0.01e+03/0.19e+02**	0.09e+03/0.51e+02	0.90e+03/0	0.02e+03/0	0.47e+03/0	2.75e+03/0	0.92e+03/3.07e+02
		200	**0.08e+03/0.15e+02**	0.10e+03/0.41e+02	0.90e+03/0	0.22e+03/0	7.38e+03/0	9.55e+03/0	0.81e+03/2.66e+02
		50	**0.01e+03/0.08e+02**	0.01e+03/0.10e+02	0.07e+03/0	0.02e+03/0	6.92e+03/0	0.02e+03/0	0.46e+03/2.54e+02
	40	100	**0.00e+04*/0.17e+02**	0.01e+04/0.12e+02	0.02e+04/0	0.01e+04/0	0.19e+04/0	1.06e+04/0	0.07e+04/2.42e+02
		200	**0.05e+03/0.24e+02**	0.07e+03/0.17e+02	0.10e+03/0	0.06e+03/0	5.83e+03/0	1.15e+03/0	0.42e+03/1.86e+02
		50	**0.13e+02/0.12e+02**	0.27e+02/0.23e+02	0.24e+02/0	1.75e+02/0	2.27e+02/0	0.52e+02/0	4.05e+02/2.36e+02
	50	100	**0.12e+02*/0.08e+02**	0.32e+02/0.08e+02	0.66e+02/0	0.25e+02/0	0.35e+02/0	0.37e+02/0	3.33e+02/1.69e+02
		200	**0.04e+03/0.25e+02**	0.08e+03/0.11e+02	0.09e+03/0	0.05e+03/0	4.99e+03/0	1.17e+03/0	0.29e+03/2.32e+02
		50	**0.05e+04*/1.85e+02**	0.29e+04/5.59e+02	0.86e+04/0	0.49e+04/0	0.79e+04/0	1.04e+04/0	0.89e+04/1.37e+03
	10	100	**0.41e+03*/2.11e+02**	2.12e+03/4.03e+02	4.34e+03/0	2.95e+03/0	3.01e+03/0	6.15e+03/0	5.69e+03/7.80e+02
		200	**0.72e+03*/1.97e+02**	2.17e+03/3.35e+02	5.15e+03/0	2.76e+03/0	9.07e+03/0	7.16e+03/0	6.64e+03/1.05e+03
		50	**0.01e+03*/0.12e+02**	0.31e+03/1.77e+02	2.06e+03/0	0.44e+03/0	1.36e+03/0	1.84e+03/0	2.03e+03/4.62e+02
	20	100	**0.03e+03*/0.22e+02**	0.14e+03/1.02e+02	0.70e+03/0	0.14e+03/0	0.18e+03/0	1.41e+03/0	0.99e+03/3.55e+02
		200	**0.07e+03*/0.24e+02**	0.25e+03/0.57e+02	0.95e+03/0	0.27e+03/0	1.24e+03/0	0.87e+03/0	1.97e+03/6.65e+02
200		50	**0.09e+02*/0.06e+02**	0.27e+02/0.10e+02	3.50e+02/0	0.15e+02/0	2.81e+02/0	0.20e+02/0	4.67e+02/2.42e+02
	30	100	**0.35e+02/0.17e+02**	0.47e+02/0.34e+02	1.39e+02/0	0.38e+02/0	5.06e+02/0	1.05e+02/0	3.89e+02/1.77e+02
		200	0.69e+02/0.23e+02	0.77e+02/0.49e+02	2.13e+02/0	0.46e+02/0	2.29e+02/0	**0.20e+02*/0**	7.75e+02/3.97e+02
		50	0.05e+02/0.04e+02	0.11e+02/0.12e+02	**0.00e+02*/0**	0.29e+02/0	1.13e+02/0	0.37e+02/0	4.23e+02/2.42e+02
	40	100	**0.08e+02/0.05e+02**	0.15e+02/0.06e+02	0.29e+02/0	0.10e+02/0	0.10e+02/0	1.24e+02/0	3.05e+02/1.65e+02
		200	**0.25e+02*/0.07e+02**	0.52e+02/0.05e+02	2.22e+02/0	0.36e+02/0	0.39e+02/0	0.54e+02/0	4.08e+02/1.65e+02
		50	**0.02e+02/0.03e+02**	0.04e+02/0.08e+02	0.19e+02/0	0.23e+02/0	0.09e+02/0	0.21e+02/0	1.89e+02/1.32e+02
	50	100	0.06e+02/0.04e+02	0.09e+02/0.05e+02	0.07e+02/0	**0.00e+02*/0**	0.07e+02/0	0.03e+02/0	3.34e+02/1.77e+02
		200	**0.21e+02*/0.05e+02**	0.59e+02/0.07e+02	1.45e+02/0	0.30e+02/0	0.33e+02/0	0.31e+02/0	2.45e+02/1.07e+02

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.asoc.2020.106208.

CRediT authorship contribution statement

Shu Luo: Conceptualization, Methodology, Software, Data curation, Writing - original draft, Visualization, Validation, Writing - review & editing.

Acknowledgments

We thank the editors and the anonymous reviewers for their fruitful comments and suggestions in improving the quality of this paper. This research is supported by the National Key Research and Development Program of China under Grant 2018YFB1703103.

References

[1] K.Z. Gao, P.N. Suganthan, T.J. Chua, C.S. Chong, T.X. Cai, Q.K. Pan, A two-stage artificial bee colony algorithm scheduling flexible job-shop scheduling problem with new job insertion, Expert Syst. Appl. 42 (21) (2015) 7652–7663.
[2] M.R. Garey, D.S. Johnson, R. Sethi, The complexity of flowshop and jobshop scheduling, Math. Oper. Res. 1 (2) (1976) 117–129.
[3] D. Ouelhadj, S. Petrovic, A survey of dynamic scheduling in manufacturing systems, J. Sched. 12 (4) (2009) 417–431.
[4] P. Lou, Q. Liu, Z. Zhou, H. Wang, S.X. Sun, Multi-agent-based proactive–reactive scheduling for a job shop, Int. J. Adv. Manuf. Technol. 59 (1–4) (2012) 311–324.
[5] N. Kundakcı, O. Kulak, Hybrid genetic algorithms for minimizing makespan in dynamic job shop scheduling problem, Comput. Ind. Eng. 96 (2016) 31–51.
[6] N. Tao, H. Ming, L. Xu, J. Hua, A novel dynamic scheduling strategy for solving flexible job-shop problems, J. Ambient Intell. Human. Comput. 7 (5) (2016) 721–729.
[7] R.A. Howard, Dynamic Programming and Markov Processes, John Wiley, 1960.
[8] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, 2018.
[9] S. Riedmiller, M. Riedmiller, A neural reinforcement learning approach to learn local dispatching policies in production scheduling, in: IJCAI, vol. 2, 1999, pp. 764–771.
[10] M. Aydin, E. Öztemel, Dynamic job-shop scheduling using reinforcement learning agents, Robot. Auton. Syst. 33 (2–3) (2000) 169–178.
[11] Y.-C. Wang, J.M. Usher, Learning policies for single machine job dispatching, Robot. Comput.-Integr. Manuf. 20 (6) (2004) 553–562.
[12] X. Chen, X. Hao, H.W. Lin, T. Murata, Rule driven multi objective dynamic scheduling by data envelopment analysis and reinforcement learning, in: 2010 IEEE International Conference on Automation and Logistics, IEEE, 2010, pp. 396–401.
[13] T. Gabel, M. Riedmiller, Distributed policy search reinforcement learning for job-shop scheduling tasks, Int. J. Prod. Res. 50 (1) (2012) 41–61.
[14] W. Bouazza, Y. Sallez, B. Beldjilali, A distributed approach solving partially flexible job-shop scheduling problem with a Q-learning effect, IFAC-PapersOnLine 50 (1) (2017) 15890–15895.
[15] J. Shahrabi, M.A. Adibi, M. Mahootchi, A reinforcement learning approach to parameter estimation in dynamic job shop scheduling, Comput. Ind. Eng. 110 (2017) 75–82.
[16] Y.-F. Wang, Adaptive job shop scheduling strategy based on weighted q-learning algorithm, J. Intell. Manuf. (2018) 1–16.
[17] Y.-R. Shiue, K.-C. Lee, C.-T. Su, Real-time scheduling for a smart factory using a reinforcement learning approach, Comput. Ind. Eng. (2018).

[18] H. Li, R. Cai, N. Liu, X. Lin, Y. Wang, Deep reinforcement learning: Algorithm, applications, and ultra-low-power implementation, Nano Commun. Netw. 16 (2018) 81–90.
[19] Y. Li, Deep reinforcement learning: An overview, 2017, arXiv preprint arXiv:1701.07274.
[20] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, Playing atari with deep reinforcement learning, 2013, arXiv preprint arXiv:1312.5602.
[21] Z. Cao, H. Zhang, B. Liu, A deep reinforcement learning approach to multi-component job scheduling in edge computing, 2019, arXiv preprint arXiv:1908.10290.
[22] B. Waschneck, A. Reichstaller, L. Belzner, T. Altenmüller, T. Bauernhansl, A. Knapp, A. Kyek, Optimization of global production scheduling with deep reinforcement learning, Proc. CIRP 72 (1) (2018) 1264–1269.
[23] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, L. Song, Learning combinatorial optimization algorithms over graphs, in: Advances in Neural Information Processing Systems, 2017, pp. 6348–6358.
[24] A. Mirhoseini, H. Pham, Q.V. Le, B. Steiner, R. Larsen, Y. Zhou, N. Kumar, M. Norouzi, S. Bengio, J. Dean, Device placement optimization with reinforcement learning, in: Proceedings of the 34th International Conference on Machine Learning-Volume 70, JMLR. org, 2017, pp. 2430–2439.
[25] S.V. Mehta, Predictable scheduling of a single machine subject to breakdowns, Int. J. Comput. Integr. Manuf. 12 (1) (1999) 15–38.
[26] C. Rajendran, O. Holthaus, A comparative study of dispatching rules in dynamic flowshops and jobshops, European J. Oper. Res. 116 (1) (1999) 156–170.
[27] S.R. Lawrence, E.C. Sewell, Heuristic, optimal, static, and dynamic schedules when processing times are uncertain, J. Oper. Manage. 15 (1) (1997) 71–82.
[28] V. Subramaniam, G. Lee, T. Ramesh, G. Hong, Y. Wong, Machine selection rules in a dynamic job shop, Int. J. Adv. Manuf. Technol. 16 (12) (2000) 902–908.
[29] T. Gabel, M. Riedmiller, Adaptive reactive job-shop scheduling with reinforcement learning agents, Int. J. Inf. Technol. Intell. Comput. 24 (4) (2008).
[30] M. Zandieh, M. Adibi, Dynamic job shop scheduling using variable neighbourhood search, Int. J. Prod. Res. 48 (8) (2010) 2449–2458.
[31] L. Nie, L. Gao, P. Li, X. Li, A GEP-based reactive scheduling policies constructing approach for dynamic flexible job shop scheduling problem with job release dates, J. Intell. Manuf. 24 (4) (2013) 763–774.
[32] H. Xiong, H. Fan, G. Jiang, G. Li, A simulation-based study of dispatching rules in a dynamic job shop scheduling problem with batch release and extended technical precedence constraints, European J. Oper. Res. 257 (1) (2017) 13–24.
[33] R. Nelson, C. Holloway, R.M.L. Wong, Centralized scheduling and priority implementation heuristics for a dynamic job shop model, AIIE Trans. 9 (1) (1977) 95–102.
[34] A. Baykasoğlu, F.S. Karaslan, Solving comprehensive dynamic job shop scheduling problem by using a GRASP-based approach, Int. J. Prod. Res. 55 (11) (2017) 3308–3325.
[35] X.-N. Shen, X. Yao, Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems, Inform. Sci. 298 (2015) 198–224.
[36] M. Shahgholi Zadeh, Y. Katebi, A. Doniavi, A heuristic model for dynamic flexible job shop scheduling problem considering variable processing times, Int. J. Prod. Res. (2018) 1–16.
[37] M. Nouiri, A. Bekrar, D. Trentesaux, Towards energy efficient scheduling and rescheduling for dynamic flexible job shop problem, IFAC-PapersOnLine 51 (11) (2018) 1275–1280.
[38] S.V. Mehta, R.M. Uzsoy, Predictable scheduling of a job shop subject to breakdowns, IEEE Trans. Robot. Autom. 14 (3) (1998) 365–378.
[39] P. Michael, Scheduling, theory, algorithms, and systems, Englewood Cli s, New Jersey, 1995.
[40] N. Al-Hinai, T.Y. ElMekkawy, Robust and stable flexible job shop scheduling with random machine breakdowns using a hybrid genetic algorithm, Int. J. Prod. Econ. 132 (2) (2011) 279–291.
[41] R. Buddala, S.S. Mahapatra, Two-stage teaching-learning-based optimization method for flexible job-shop scheduling under machine breakdown, Int. J. Adv. Manuf. Technol. (2018) 1–14.
[42] S.M. Sajadi, A. Alizadeh, M. Zandieh, F. Tavan, Robust and stable flexible job shop scheduling with random machine breakdowns: multi-objectives genetic algorithm approach, Int. J. Math. Oper. Res. 14 (2) (2019) 268–289.
[43] B. Wang, H. Xie, X. Xia, X.-X. Zhang, A NSGA-II algorithm hybridizing local simulated-annealing operators for a bicriteria robust job-shop scheduling problem under scenarios, IEEE Trans. Fuzzy Syst. (2018).
[44] R. Bellman, A Markovian decision process, J. Math. Mech. (1957) 679–684.
[45] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529.
[46] H.V. Hasselt, Double q-learning, in: Advances in Neural Information Processing Systems, 2010, pp. 2613–2621.
[47] H. Van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double Q-learning, in: AAAI, vol. 2, Phoenix, AZ, 2016, p. 5.
[48] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, 2015, arXiv preprint arXiv:1509.02971.
[49] C. Lu, X. Li, L. Gao, W. Liao, J. Yi, An effective multi-objective discrete virus optimization algorithm for flexible job-shop scheduling problem with controllable processing times, Comput. Ind. Eng. 104 (2017) 156–174.
[50] Matlab, version 9.1 (R2016b), 2016.