Computers & Industrial Engineering 159 (2021) 107489

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

Dynamic multi-objective scheduling for flexible job shop by deep reinforcement learning

Shu Luo, Linxuan Zhang *, Yushun Fan

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

ARTICLE INFO

Keywords:
Flexible job shop scheduling
Multi-objective
New job insertion
Dispatching rules
Deep reinforcement learning

ABSTRACT

In modern volatile and complex manufacturing environment, dynamic events such as new job insertions and machine breakdowns may randomly occur at any time and different objectives in conflict with each other should be optimized simultaneously, leading to an urgent requirement of real-time multi-objective rescheduling methods that can achieve both time efficiency and solution quality. In this regard, this paper proposes an on-line rescheduling framework named as two-hierarchy deep Q network (THDQN) for the dynamic multi-objective flexible job shop scheduling problem (DMOFJSP) with new job insertions. Two practical objectives including total weighted tardiness and average machine utilization rate are optimized. The THDQN model contains two deep Q network (DQN) based agents. The higher-level DQN is a controller determining the temporary optimization goal for the lower DQN. At each rescheduling point, it takes the current state features as input and chooses a feasible goal to guide the behaviour of the lower DQN. Four different goals corresponding to four different forms of reward functions are suggested, each of which optimizes an indicator of tardiness or machine utilization rate. The lower-level DQN acts as an actuator. It takes the current state features together with the higher optimization goal as input and chooses a proper dispatching rule to achieve the given goal. Six composite dispatching rules are developed to select an available operation and assign it on a feasible machine, which serve as the candidate action set for the lower DQN. A novel training framework based on double DQN (DDQN) is designed. The trained THDQN is compared with each proposed composite dispatching rule, existing well-known dispatching rules as well as other reinforcement learning based scheduling methods on a wide range of test instances. Results of numerical experiments have confirmed both the effectiveness and generality of the proposed THDQN.

1. Introduction

Nowadays, with the dramatic growth of complexity and uncertainty in manufacturing systems, there has been an urgent need of dynamic multi-objective rescheduling methods with the ability of handling random disturbances such as dynamic demands, machine breakdowns and uncertain processing times in real time while simultaneously considering different objectives like makespan and total tardiness. Without lose of generality, most of scheduling problems existing in modern multi-type and low-volume discrete manufacturing systems can be regarded as dynamic multi-objective flexible job shop scheduling problem (DMOFJSP). It has been proved to be NP-hard (Garey, Johnson, & Sethi, 1976) and is more intractable than classical job shop scheduling problem (JSP) since each operation can be processed on one or more compatible machines. Due to its high complexity and universality, the DMOFJSP is of great significance to be researched for both academia and industry.

To address the dynamic events, traditional scheduling methods for the DMOFJSP can be mainly divided into two kinds, namely metaheuristics and dispatching rules. Metaheuristics, such as genetic algorithm (GA) (Kundakcı & Kulak, 2016), particle swarm optimization (PSO) (Tang, Dai, Salido, & Giret, 2016) and ant colony optimization algorithm (ACO) (Zhang, Li, Zhang, & Wang, 2020), always decompose a dynamic scheduling problem into a sequence of static scheduling problems and solve them separately. They can obtain near-optimal solutions but suffer from poor time efficiency. Dispatching rules, like first in first out (FIFO), earliest due date (EDD) and most remaining processing time (MRT) (Rajendran & Holthaus, 1999), react to the dynamic disturbances exactly in real time. They can adjust the schedule in the shortest time but the obtained results are far from optimal in the long

* Corresponding author.
E-mail addresses: luos17@mails.tsinghua.edu.cn (S. Luo), lxzhang@mail.tsinghua.edu.cn (L. Zhang), fanyus@tsinghua.edu.cn (Y. Fan).

https://doi.org/10.1016/j.cie.2021.107489
Received 19 August 2020; Received in revised form 4 May 2021; Accepted 15 June 2021
Available online 18 June 2021

run since most of the dispatching rules are myopic (Holthaus & Rajendran, 2000; Mohan, Lanka, & Rao, 2019). Meanwhile, it is extremely hard to select a single rule to simultaneously optimize over all objectives since different rules are suitable for different objectives and production environments (Nie, Gao, Li, & Li, 2013). In order to achieve both time efficiency and solution quality, an advisable way is to intelligently select the proper dispatching rule at each rescheduling point so that each rule optimizes over a short period and thereby a good long-term performance among different objectives can be guaranteed.

In essence, the rescheduling process of DMOFJSP can be modeled as a Markov decision process (MDP) where the decision maker should successively determine the right actions, i.e., the feasible dispatching rules based on the production status of different rescheduling points so as to optimize the predefined long-term objectives. Earlier work for solving the MDP always resorted to dynamic programming (DP) (Howard, 1960), which is a model-based method requiring accurate modelling of state transition function. However, in modern manufacturing systems with various of uncertainties, the state transition function could not be exactly modeled in advance. In recent years, reinforcement learning (RL) (Sutton & Barto, 2018) has arose as an effective way to deal with MDP. Due to its ability to learn the optimal behavioral strategy in a model-free manner solely by trial and error, RL has achieved many successful applications in real-world dynamic scheduling problems (Chen, Hao, Lin, & Murata, 2010; Gabel & Riedmiller, 2012; Li, Wang, & Sawhney, 2012; Bouazza, Sallez, & Beldjilali, 2017).

Despite that a large number of breakthroughs have been made, there are two problems remained to be solved in classical RL based dynamic scheduling methods. First, most of the previous work (Wang & Usher, 2004; Yingzi & Mingyang, 2004; Shahrabi, Adibi, & Mahootchi, 2017) use stand Q-learning or SARSA where a look-up Q table is adopted to store the Q-value of each state-action pair. However, in real manufacturing systems, most state features are continuous thus there are countless different states, making it impossible to maintain a cumbersome Q-table in memory. An intuitive way to handle this problem is to discretize the continuous state space into several numerical intervals so that each interval corresponds to a state (Yingzi & Mingyang, 2004; Shiue, Lee, & Su, 2018; Chen, Yang, Li, & Wang, 2020). However, this may reduce the model accuracy and the number of numerical intervals (states) is hard to determine. Recently, deep reinforcement learning (DRL) (Li, 2017; Cunha, Madureira, Fonseca, & Coelho, 2018), which uses deep neural networks as the Q-function approximators, have demonstrated great advantages over traditional RL methods for handling continuous state space. DRL based methods directly take the raw state features as input and the Q-values of different state-action pairs as output without the need of storing a huge Q-table, thus are more convenient and efficient to implement in practical applications. Second, for multi-objective rescheduling problems, it is difficult to design a single RL agent to simultaneously optimize all objectives since different objectives lead to different reward functions and behavioral strategies, which are hard to compromise by a single agent. Nowadays, hierarchical reinforcement learning (HRL) (Nachum, Gu, Lee, & Levine, 2018a; Li, Florensa, Clavera, & Abbeel, 2019; Rafati & Noelle, 2019) can be regarded as a promising approach to address this issue. In general, HRL learns to operate over different levels of spatio-temporal abstraction where a higher-level controller learns a policy over different objectives at a slower time-scale, and a lower-level actuator learns a policy over atomic actions to satisfy the given objectives in a real-time manner. By adaptively adjusting the optimization goal (corresponding to different reward functions) through the higher controller and choosing the appropriate dispatching rules to achieve the given goal through the lower actuator, a good compromise among different objectives can be made during the long-term schedule.

These factors have inspired us to combine both the merits of DRL and HRL to deal with the two dilemmas. To the best of our knowledge, there is no previous work adopting a hierarchical deep reinforcement learning based method for solving the DMOFJSP.

With the motivations above, in this paper, a two-hierarchy deep reinforcement learning model named as two-hierarchy deep Q network (THDQN) is developed to address the DMOFJSP with new job insertions. The novelties are mainly presented as follows: (1) The proposed model contains two DQN based agents, a higher controller and a lower actuator. The higher DQN is a controller determining the optimization goal for the lower agent. At each rescheduling point, it takes the state features as input and the Q-value of each goal as output, through which the higher optimization goal (i.e. the specific form of reward function) at current rescheduling point can be chosen. The lower DQN serves as an actuator taking both the state features and the optimization goal delivered from the higher DQN as input and the Q-value of each dispatching rule as output. Based on the obtained Q-values, the most feasible rule at each rescheduling point can be selected. (2) A double DQN (DDQN) based training framework is developed to train the two DQN agents. The training process is conducted on various of different production configurations, making the proposed model more effective and generic to be applied in real-world manufacturing systems. (3) Ten state features are extracted to comprehensively represent the production status. (4) Two objectives including total weighted tardiness (*TWT*) and average machine utilization rate (U_{ave}) are considered to be optimized. In order to achieve a compromise between the two objectives, four different higher goals are developed, which correspond to four different forms of reward function in the training process. (5) Six problem-specific composite dispatching rules are developed to simultaneously select an operation and assign it on a feasible machine with the aim of reducing total tardiness or improving machine utilization rate.

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

2. Literature review

RL based dynamic scheduling methods for job shop scheduling problems have been extensively studied by researchers over the past decades. The basic idea of these methods is using intelligent RL agents to adaptively select feasible actions (usually dispatching rules) at different rescheduling points so as to minimize the impacts of uncertain disturbances and optimize the long-term objectives. To summarize, the existing approaches can be divided into classical RL based methods and deep RL based methods. The first ones solve a MDP by some traditional RL algorithms such as Q-learning and SARSA (Sutton & Barto, 2018), where a look-up Q table is always used to store the Q-value of each state-action pair.

Aydin and Öztemel (2000) developed an intelligent agent based dynamic scheduling system to minimize mean tardiness in a dynamic job-shop with new job insertions. The agent is trained by an improved Q-learning algorithm so as to select the most appropriate dispatching rules in real time. Wei and Mingyang (2005) designed several composite dispatching rules considering both job selection and machine selection for a dynamic job shop with random job arrivals. A Q-learning based agent is applied to intelligently select the dispatching rules at each rescheduling moment. Shahrabi et al. (2017) proposed a Q-learning based method to find the optimal parameters of variable neighborhood search (VNS) at each rescheduling point for a dynamic job shop with random job arrivals and machine breakdowns. Bouazza et al. (2017) developed a Q-learning agent which determines the most suitable machine selection rules and operation dispatching rules so as to minimize the weighted average waiting time in a dynamic flexible job shop with new job insertions. Méndez-Hernández, Rodríguez-Bazan, Martinez-Jimenez, Libin, and Nowé (2019) suggested a multi-objective multi-

Table 1
Existing RL based methods for dynamic job shop scheduling problem.

Work	State space	Algorithm	Agent	Objective	Dynamic events	Problem
Zhang and Dietterich (1995)	Continuous	Temporal difference algorithm	Single−agent	Makespan	None	Job shop scheduling
Riedmiller and Riedmiller (1999)	Continuous	Q-learning	Multi-agent	Summed tardiness	None	Job shop scheduling
Aydin and Öztemel (2000)	Discrete	Q-learning	Single−agent	Mean tardiness	New job insertions	Job shop scheduling
Wang and Usher (2004)	Discrete	Q-learning	Single−agent	Mean tardiness	New job insertions	Job shop scheduling
Yingzi and Mingyang (2004)	Discrete	Q-learning	Single−agent	Mean tardiness	New job insertions	Job shop scheduling
Chen et al. (2010)	Discrete	Q-learning	Single−agent	Mean flow time; Mean tardiness	Fluctuation of work in process	Job shop scheduling
Gabel and Riedmiller (2012)	Discrete	Policy gradient	Multi-agent	Makespan	None	Job shop scheduling
Bouazza et al. (2017)	Discrete	Q-learning	Multi-agent	Makespan; Total weighted completion time; Weighted average waiting time	New job insertions	Flexible job shop scheduling
Shahrabi et al. (2017)	Discrete	Q-learning	Single−agent	Mean flow time	New job inserions; Machine breakdowns	Job shop scheduling
Wang (2018)	Discrete	Q-learning	Multi-agent	Earliness and tardiness punishment	New job insertions	Job shop scheduling
Waschneck et al. (2018)	Continuous	Deep Q-learning	Multi-agent	Uptime utilization	Machine breakdowns	Flexible job shop scheduling
Kuhnle et al. (2019)	Continuous	Trust region policy optimization	Single−agent	Machine utilization; Lead time of orders	None	Job shop scheduling
Liu et al. (2020)	Continuous	Deep deterministic policy gradient	Multi-agent	Makespan	Processing time variations	Job shop scheduling
Altenmüller et al. (2020)	Continuous	Deep Q-learning	Single−agent	Time constraint violations	Machine breakdowns; New job insertions	Job shop scheduling
Our method	Continuous	Hierarchical deep Q-learning	Single−agent	Total weighted tardiness; Average machine utilization rate	New job insertions	Flexible job shop scheduling

agent reinforcement learning algorithm for job shop scheduling considering makespan and tardiness to be minimized. Each resource is associated with an agent determining which is the next operation to process. The agents are trained by a two-phase Q-learning algorithm where each agent acts independently optimizing its own objective at first and then collaborates with others to find the Pareto-optimal solution for all objectives. Wang (2020) developed a dynamic multi-agent scheduling model to minimize earliness and tardiness punishment in a job shop with new job insertions. Multiple agents including machine, buffer, state and job agents are used and trained by a weighted Q-learning algorithm based on clustering and dynamic search.

Despite the remarkable success made by the classical RL models, the stand Q-learning confronts with an intractable dilemma, that is, the state explosion problem. Since the number of different states might be infinite in real manufacturing systems with continuous state space, it is impossible to maintain such a huge look-up Q table, making the stand Q-learning based methods infeasible for practical implementations. Recently, deep reinforcement learning (DRL) (Mnih et al., 2013; Mnih et al., 2015; Li, 2017) has achieved great breakthroughs for solving sequential decision problems with complex and continuous state space. Instead of the shallow models such as linear function and decision trees, DRL uses deep neural network (DNN) as the Q-function or policy function estimator. By directly taking the state feature vectors as the DNN's input, the Q-value or selection probability of each candidate action can be obtained and thereby the state explosion problem is solved.

Up to date, a variety of DRL based methods have been applied to dynamic job shop scheduling problems. The earliest work combining neural network with RL dates back to Zhang and Dietterich (1995) where an artificial neural network (ANN) is utilized to learn domain-specific heuristics for job shop scheduling and trained by temporal difference algorithm TD(λ). Later, Riedmiller and Riedmiller (1999) proposed a neural network based agent to learn local dispatching policies in a static job shop. The neural agent is trained by Q-learning to minimize the summed tardiness. Waschneck et al. (2018) developed a multi-agent dynamic scheduling method for a flexible job shop with re-entrant flows of jobs and machine breakdowns. Each agent is represented by a deep Q network (DQN) and trained by deep Q-learning (DQL). The agents optimize their own dispatching rules at one work center while monitoring the actions of other agents so as to maximize the uptime utilization. Altenmüller, Stüker, Waschneck, Kuhnle, and Lanza (2020) also proposed a DQN based agent for solving the decision problem of order dispatching in a complex job shop with strict time constraints. In Hu et al. (2020), a graph convolutional network (GCN) based DQN is designed to solve the dynamic scheduling problem of flexible manufacturing systems (FMSs) with route flexibility and stochastic arrivals of products. A graph convolution layer called Petri-net convolution (PNC) layer is developed to approximate the DQN action-value function. Zhu, Li, Tang, and Sun (2020) suggested a DRL based online learning algorithm for real-time scheduling in cloud manufacturing. A deep neural network (DNN) is used as the policy approximator which maps the state features observed from the environment to the probability distribution over candidate actions. The policy parameters of DNN is optimized by policy gradient. Liu, Chang, and Tseng (2020) developed an actor-critic deep reinforcement learning model for job shop scheduling problem aiming at minimizing the makespan. Both the actor and critic are deep network including convolution layers and fully connected layers. The model is trained by a combination of asynchronous update as well as deep deterministic policy gradient (DDPG). Kuhnle, Schäfer, Stricker, and Lanza (2019) proposed a trust region policy optimization (TRPO) based DRL algorithm for adaptive order dispatching in job shop manufacturing systems. A dense two layered net is used as policy approximator to dispatch orders to feasible machines. For interested readers, a survey of DRL based methods for job shop scheduling problems is provided in Cunha et al. (2018). Table 1 summarizes the differences between our method and other existing RL based dynamic job shop scheduling methods.

3. Background of RL and Deep Q-learning

3.1. Definition of RL

In general, RL deals with the MDP where an intelligent agent interacts with its surrounding environment trying to maximize the expected sum of long-term reward. The training process could be described by a 5-tuple representation $(\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{P}, \gamma, \boldsymbol{R})$. At each decision point t, the agent observes the current state $s_t \in \boldsymbol{S}$ and takes a feasible action $a_t \in \boldsymbol{A}$ according to the policy $\pi(\boldsymbol{S}\rightarrow\boldsymbol{A})$, after which it enters a new state s_{t+1} with transition probability $p(s_{t+1}|s_t, a_t) \in \boldsymbol{P}(\boldsymbol{S}\times\boldsymbol{A}\rightarrow\boldsymbol{S})$. Meanwhile, an immediate reward $r_t \in \boldsymbol{R}(\boldsymbol{S}\times\boldsymbol{A}\times\boldsymbol{S}\rightarrow\mathbb{R})$ is obtained as a result of the state transition (s_t, a_t, s_{t+1}). The objective of a RL agent is to find the optimal policy π^* maximizing the expected discounted future reward starting from state s and following a specific policy π thereafter (which is known as the state-value function), as defined in Eq. (1):

$$V_\pi(s) = \mathbb{E}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots \middle| s_t = s, \pi\right] \tag{1}$$

where $\gamma \in (0, 1]$ is the discount factor differentiating the relative importance of short-term reward and long-term reward. Similarly, we can define the action-value function as the expected discounted future reward upon taking action a in state s and following a specific policy π thereafter, as defined in Eq. (2):

$$Q_\pi(s, a) = \mathbb{E}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots \middle| s_t = s, a_t = a, \pi\right] \tag{2}$$

Bellman (1957) has proved that the optimal action-value function under the optimal policy π^* must satisfy the Bellman optimality equation in Eq. (3), which is particularly important for many learning methods.

$$\begin{aligned} Q_{\pi^*}\left(s, a\right) &= \max_\pi Q_\pi\left(s, a\right) \\ &= \sum_{s'} p\left(s'|s, a\right)\left[r\left(s, a, s'\right) + \gamma \max_{a'} Q_{\pi^*}\left(s', a'\right)\right] \end{aligned} \tag{3}$$

3.2. Deep Q-learning and Deep Q network

Deep Q-learning (DQL) is proposed to address the curse of dimensionality that can not be solved in standard Q-learning. The core idea of DQL is to use a deep Q network (DQN) (Mnih et al., 2013) as the Q function approximator. By directly taking the raw data (state features, usually in continuous values) as input and the Q function value of each state-action pair as output, the DQN can deal with real-world decision process with infinite state space. The stand DQN has two major advancements. First, an experience replay memory D is used to eliminate the correlation between consecutive transitions, where the updates of parameters are based on minibatch of samples randomly drawn from D. Second, a separate target network $\widehat{Q}(\theta^-)$ whose weights θ^- are periodically replaced by the online network $Q(\theta)$ is utilized to stabilize the training process. At each training step t, the training target y_t of the online network $Q(\theta)$ are calculated by Eq. (4).

$$y_t = r_t + \gamma \max_{a'} \widehat{Q}\left(s_{t+1}, a'; \theta^-\right) \tag{4}$$

Note that in the standard DQN, the max operator uses the same values for both action selection and evaluation, which may easily lead to overoptimistic value estimates (Hasselt, 2010). To address this problem, a technique called double DQN (DDQN) (Van Hasselt, Guez, & Silver, 2016) is developed. The training procedure of DDQN is the same as DQN except for the way of calculating the training target y_t, as shown in Eq. (5).

$$y_t = r_t + \gamma \widehat{Q}\left(s_{t+1}, \operatorname{argmax}_{a'} Q\left(s_{t+1}, a'; \theta\right); \theta^-\right) \tag{5}$$

DDQN decouples the selection of actions from the evaluation, greatly reducing the overoptimism and resulting in more stable learning. The procedure of DDQN is given in Algorithm 1, which serves as the basic training framework in this paper.

Algorithm 1. Training procedure of DDQN

1: Initialize replay memory D to capacity N
2: Initialize online network Q with random weights θ
3: Initialize target network $\widehat{Q}$ with weights $\theta^- = \theta$
4: **for** epoch $= 1 : L$ **do**
5: Observe initial state s_1 and extract the feature vector ϕ_1 of state s_1
6: **for** $t = 1 : T$ (T is the terminal time) **do**
7: With probability ε select a random action a_t
8: otherwise select $a_t = \operatorname{argmax}_a Q(\phi_t, a; \theta)$
9: Execute action a_t, observe reward r_t and next state s_{t+1}
10: Extract feature vector ϕ_{t+1} of state s_{t+1}
11: Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
12: Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D
13:

$$\text{Set} y_j = \begin{cases} r_j & \text{if epoch terminates at step} j+1 \\ r_j + \gamma \widehat{Q}\left(\phi_{j+1}, \operatorname{argmax}_{a'} Q\left(\phi_{j+1}, a'; \theta\right); \theta^-\right) & \text{otherwise} \end{cases}$$

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

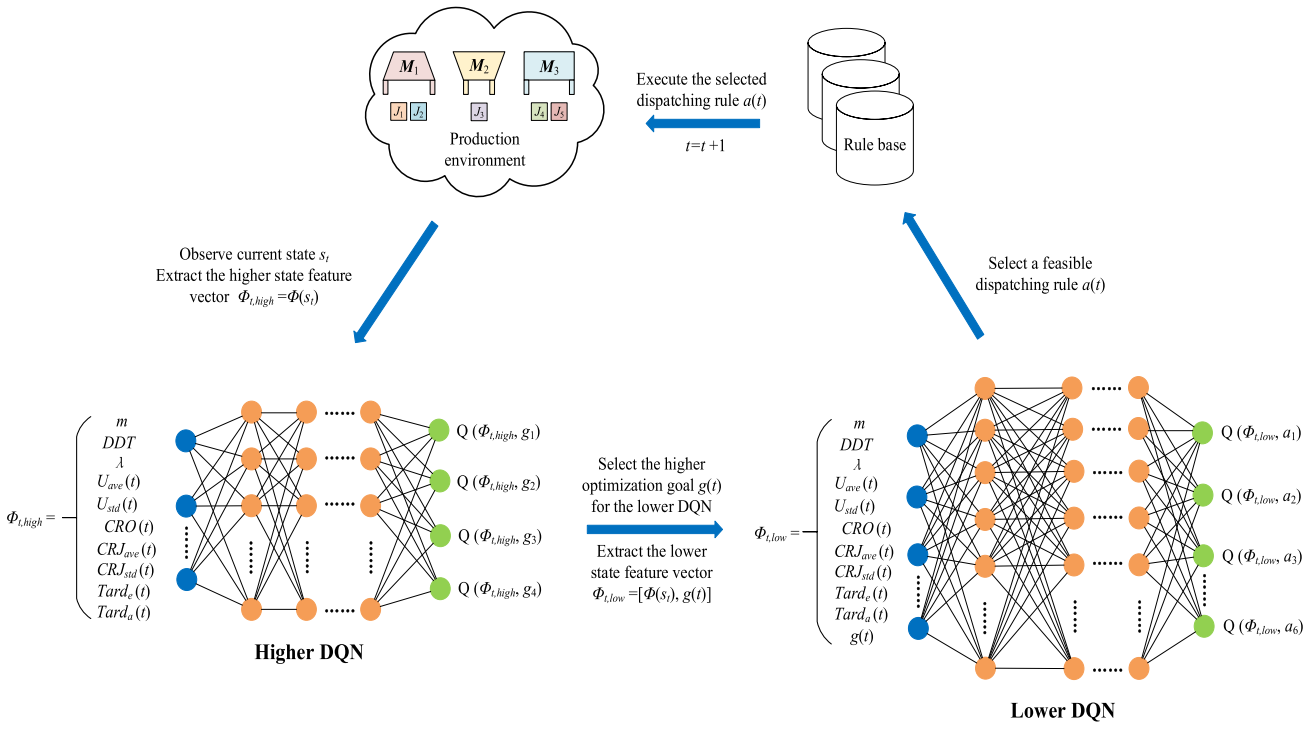


Fig. 1. Structure of the proposed THDQN.

Table 2
Parameter settings for training and testing benchmarks.

Parameter	Value
total number of machines (m)	randi[10, 50]
number of available machines per operation	randi[1, m]
number of initial jobs at beginning	randi[1, 20]
total number new inserted jobs	randi[50, 200]
number of operations per job	randi[1, 20]
urgency degree Pr_i of each job	randi[1, 5]
processing time of an operation on each available machine	randf[1, 50]
due date tightness (DDT)	randf[0.5, 1.5]
mean value λ of the interarrival time $exp(1/\lambda)$	randf[50, 200]

Table 3
Parameter settings of the training algorithm.

Parameter	Value
number of training epochs L	1000
higher replay memory size N_{high}	32
lower replay memory size N_{low}	1000
minibatch size for gradient descent	32
ε-*greedy* parameter ε	linearly decreased from 0.9 to 0.1
step length C for training target replacement	100
discount factor γ	0.9
optimizer	Adam

4. Problem formulation

The DMOFJSP 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 job J_i is A_i and D_i, respectively. $C_{i,j}$ represents the actual completion time of operation $O_{i,j}$. The urgency degree of job J_i is denoted by Pr_i, where a higher urgency degree indicates a higher punishment on tardiness. The objective is to minimize the total weighted tardiness and maximize the average machine utilization rate simultaneously. 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.
Pr_i: the urgency degree 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:

Table 4
GD values for the Pareto-optimal fronts obtained by the THDQN and each composite dispatching rule.

DDT	*m*	λ	THDQN	Random	$Rule_1$	$Rule_2$	$Rule_3$	$Rule_4$	$Rule_5$	$Rule_6$
0.5	10	50	**0.00e+00**	1.25e−01	1.12e+00	1.11e+00	1.91e−01	3.28e−01	9.98e−01	1.08e+00
		100	**0.00e+00**	1.52e−01	1.83e−01	1.57e−01	1.78e−01	2.02e−01	2.14e−01	1.58e−01
		200	**1.03e−01**	1.64e−01	2.37e−01	1.44e−01	1.12e−01	1.65e−01	2.62e−01	1.88e−01
	30	50	**0.00e+00**	1.03e−01	1.34e−01	1.60e−01	2.36e−01	1.19e−01	2.16e−01	1.61e−01
		100	**0.00e+00**	1.08e−01	1.15e−01	1.17e−01	1.80e−01	1.00e−01	9.65e−02	2.18e−01
		200	1.11e−01	1.05e−01	1.45e−01	1.31e−01	1.32e−01	**7.18e−02**	1.47e−01	1.77e−01
	50	50	**1.05e−01**	1.33e−01	1.25e−01	1.41e−01	1.66e−01	1.32e−01	1.09e−01	1.69e−01
		100	**1.12e−01**	1.58e−01	1.63e−01	2.41e−01	1.18e−01	1.15e−01	1.49e−01	1.23e−01
		200	1.20e−01	1.31e−01	1.17e−01	**1.09e−01**	1.26e−01	1.41e−01	1.51e−01	1.16e−01
1.0	10	50	**0.00e+00**	1.26e−01	4.82e−01	5.13e−01	2.44e−01	3.02e−01	2.60e−01	4.77e−01
		100	**0.00e+00**	1.91e−01	2.87e−01	2.84e−01	2.13e−01	2.32e−01	2.36e−01	1.71e−01
		200	9.83e−02	1.29e−01	3.04e−01	1.52e−01	2.34e−01	2.00e−01	1.42e−01	**0.00e+00**
	30	50	**0.00e+00**	1.01e−01	2.39e−01	2.01e−01	1.76e−01	1.76e−01	3.38e−01	2.04e−01
		100	**0.00e+00**	1.24e−01	1.45e−01	2.01e−01	1.66e−01	1.74e−01	1.43e−01	1.17e−01
		200	1.52e−01	**8.83e−02**	1.48e−01	1.31e−01	1.69e−01	1.56e−01	1.42e−01	1.83e−01
	50	50	1.30e−01	1.39e−01	2.37e−01	1.43e−01	1.79e−01	1.70e−01	1.57e−01	**1.10e−01**
		100	**1.24e−01**	2.04e−01	1.88e−01	1.68e−01	1.40e−01	1.52e−01	1.49e−01	1.34e−01
		200	9.64e−02	9.77e−02	1.33e−01	1.29e−01	8.32e−02	1.29e−01	**8.13e−02**	1.43e−01
1.5	10	50	**0.00e+00**	2.17e−01	1.70e−01	2.65e−01	2.37e−01	2.04e−01	3.10e−01	1.61e−01
		100	1.17e−01	1.78e−01	3.66e−01	2.84e−01	1.35e−01	2.52e−01	4.16e−01	**1.01e−01**
		200	1.85e−01	3.16e−01	1.88e−01	2.62e−01	2.78e−01	2.18e−01	5.27e−01	**0.00e+00**
	30	50	**0.00e+00**	2.94e−01	2.28e−01	2.62e−01	4.12e−01	5.87e−01	2.54e−01	2.12e−01
		100	**9.20e−02**	1.83e−01	3.19e−01	3.14e−01	2.15e−01	2.40e−01	2.59e−01	1.90e−01
		200	1.07e−01	2.47e−01	3.30e−01	3.43e−01	**1.03e−01**	2.13e−01	4.52e−01	4.20e−01
	50	50	**1.43e−01**	1.66e−01	2.01e−01	1.48e−01	2.77e−01	4.62e−01	1.47e−01	2.08e−01
		100	**0.00e+00**	3.29e−01	2.35e−01	5.49e−01	1.67e−01	2.69e−01	2.70e−01	1.97e−01
		200	2.21e−01	2.68e−01	4.05e−01	3.98e−01	2.33e−01	3.23e−01	**2.01e−01**	2.70e−01

Table 5
IGD values for the Pareto-optimal fronts obtained by the THDQN and each composite dispatching rule.

DDT	*m*	λ	THDQN	Random	$Rule_1$	$Rule_2$	$Rule_3$	$Rule_4$	$Rule_5$	$Rule_6$
0.5	10	50	**1.84e−01**	1.73e+00	3.35e+00	3.16e+00	2.69e+00	2.49e−01	7.46e+00	3.48e+00
		100	**1.33e−01**	2.43e−01	2.97e−01	2.67e−01	2.77e−01	5.63e−01	1.14e+00	2.67e−01
		200	**5.91e−02**	2.01e−01	2.95e−01	2.55e−01	1.49e−01	3.27e−01	8.50e−01	3.21e−01
	30	50	**9.08e−02**	2.07e−01	2.50e−01	3.51e−01	3.86e−01	2.15e−01	2.29e−01	3.28e−01
		100	**2.28e−02**	2.53e−01	1.55e−01	2.52e−01	2.33e−01	1.70e−01	2.08e−01	3.04e−01
		200	8.63e−02	1.43e−01	1.58e−01	1.85e−01	**8.43e−02**	1.37e−01	1.60e−01	1.76e−01
	50	50	7.17e−02	1.69e−01	1.90e−01	2.64e−01	2.76e−01	2.28e−01	**6.69e−02**	2.98e−01
		100	**6.96e−02**	1.82e−01	1.98e−01	2.04e−01	1.44e−01	3.18e−01	8.74e−02	3.47e−01
		200	1.14e−01	1.55e−01	1.66e−01	**9.13e−02**	1.41e−01	1.72e−01	1.57e−01	1.58e−01
1.0	10	50	**3.07e−01**	4.17e−01	6.63e−01	9.22e−01	5.50e−01	3.12e+00	6.10e+00	4.78e−01
		100	**0.00e+00**	2.03e+00	2.14e+00	1.98e+00	3.95e+00	9.10e+00	1.31e+01	3.44e−01
		200	2.09e−01	2.23e−01	3.04e−01	2.33e−01	2.87e−01	4.69e−01	4.31e−01	**1.16e−01**
	30	50	**1.27e−01**	3.28e−01	3.59e−01	4.72e−01	1.03e+00	9.72e−01	6.70e−01	1.41e−01
		100	**2.03e−01**	2.35e−01	3.66e−01	3.83e−01	3.68e−01	3.69e−01	3.46e−01	2.31e−01
		200	1.33e−01	1.24e−01	1.82e−01	1.64e−01	1.67e−01	2.04e−01	**9.75e−02**	1.81e−01
	50	50	1.22e−01	2.59e−01	3.44e−01	3.25e−01	8.57e−01	5.93e−01	2.83e−01	**1.10e−01**
		100	**1.07e−01**	1.46e−01	2.59e−01	2.82e−01	1.50e−01	1.63e−01	1.12e−01	1.39e−01
		200	**8.74e−02**	1.25e−01	1.34e−01	1.69e−01	1.00e−01	1.53e−01	8.76e−02	1.22e−01
1.5	10	50	**1.07e−01**	7.61e−01	1.22e+00	8.24e−01	4.21e−01	2.82e+00	4.26e+00	3.17e−01
		100	1.16e−01	6.10e−01	1.51e+00	1.91e+00	1.85e+00	4.58e+00	4.11e+00	**1.03e−01**
		200	**1.85e−01**	4.23e−01	8.77e−01	1.00e+00	1.25e+00	9.65e−01	1.22e+00	2.40e−01
	30	50	**5.78e−02**	1.59e+01	3.31e+01	2.55e+01	3.93e+01	8.69e+01	7.61e+01	4.18e−01
		100	**1.55e−01**	2.15e−01	5.25e−01	6.35e−01	2.05e−01	4.42e−01	3.85e−01	1.79e−01
		200	**8.85e−02**	4.34e−01	7.32e−01	7.09e−01	1.39e−01	5.44e−01	3.78e−01	3.72e−01
	50	50	**2.07e−01**	3.79e−01	4.98e−01	5.35e−01	2.13e−01	4.66e−01	3.81e−01	2.21e−01
		100	**1.24e−01**	4.63e−01	1.45e+00	8.78e−01	1.81e−01	9.88e−01	4.54e−01	1.63e−01
		200	**2.81e−01**	4.62e−01	1.39e+00	5.14e−01	4.67e−01	3.86e−01	4.49e−01	4.50e−01

Table 6
Δ values for the Pareto-optimal fronts obtained by the THDQN and each composite dispatching rule.

DDT	*m*	λ	THDQN	Random	$Rule_1$	$Rule_2$	$Rule_3$	$Rule_4$	$Rule_5$	$Rule_6$
0.5	10	50	**6.08e−01**	7.30e−01	6.34e−01	6.27e−01	7.36e−01	9.15e−01	9.78e−01	9.51e−01
		100	**7.42e−01**	7.87e−01	8.54e−01	8.00e−01	9.03e−01	7.94e−01	8.58e−01	8.90e−01
		200	**4.32e−01**	5.51e−01	5.90e−01	7.37e−01	6.62e−01	9.06e−01	8.21e−01	6.92e−01
	30	50	**7.08e−01**	7.34e−01	8.20e−01	7.76e−01	7.23e−01	8.99e−01	7.48e−01	8.11e−01
		100	**5.30e−01**	8.73e−01	7.66e−01	9.63e−01	7.53e−01	5.73e−01	7.40e−01	8.09e−01
		200	7.41e−01	8.85e−01	8.67e−01	8.51e−01	**5.19e−01**	7.72e−01	6.12e−01	7.86e−01
50	50	6.16e−01	6.88e−01	7.71e−01	7.62e−01	7.24e−01	7.69e−01	**5.93e−01**	8.31e−01	
		100	5.94e−01	8.35e−01	7.15e−01	7.90e−01	8.34e−01	7.63e−01	**4.22e−01**	7.33e−01
		200	**4.46e−01**	8.58e−01	8.32e−01	4.99e−01	9.00e−01	7.62e−01	7.34e−01	6.81e−01
1.0	10	50	**6.12e−01**	8.65e−01	7.91e−01	6.42e−01	8.99e−01	9.54e−01	9.67e−01	7.34e−01
		100	**4.08e−01**	9.44e−01	8.60e−01	8.82e−01	9.43e−01	9.59e−01	9.70e−01	7.51e−01
		200	8.25e−01	9.04e−01	8.56e−01	9.22e−01	7.88e−01	8.49e−01	9.65e−01	**7.11e−01**
	30	50	**6.09e−01**	9.20e−01	6.57e−01	8.39e−01	9.46e−01	8.61e−01	8.19e−01	8.98e−01
		100	6.28e−01	**5.98e−01**	8.62e−01	9.14e−01	9.84e−01	7.87e−01	7.41e−01	7.13e−01
		200	**4.94e−01**	7.84e−01	7.94e−01	9.37e−01	6.49e−01	6.73e−01	6.37e−01	7.95e−01
	50	50	**6.06e−01**	7.15e−01	7.02e−01	9.19e−01	9.18e−01	7.58e−01	9.65e−01	7.42e−01
		100	**4.31e−01**	8.77e−01	7.36e−01	7.16e−01	6.19e−01	6.03e−01	8.74e−01	6.62e−01
		200	**6.33e−01**	8.90e−01	7.85e−01	7.99e−01	7.08e−01	9.78e−01	8.00e−01	6.76e−01
1.5	10	50	**5.87e−01**	9.11e−01	9.03e−01	9.19e−01	7.12e−01	9.54e−01	9.48e−01	8.14e−01
		100	**8.09e−01**	9.05e−01	8.92e−01	9.09e−01	9.07e−01	9.62e−01	9.24e−01	8.21e−01
		200	**5.69e−01**	7.55e−01	9.62e−01	9.25e−01	8.51e−01	8.25e−01	7.33e−01	6.72e−01
	30	50	**4.95e−01**	9.33e−01	9.38e−01	9.61e−01	9.34e−01	1.05e+00	9.75e−01	8.32e−01
		100	**5.56e−01**	7.48e−01	8.50e−01	7.91e−01	7.88e−01	6.74e−01	7.55e−01	7.87e−01
		200	6.22e−01	8.20e−01	9.49e−01	8.59e−01	6.97e−01	8.44e−01	**5.95e−01**	7.51e−01
	50	50	**6.34e−01**	6.49e−01	8.07e−01	9.51e−01	6.64e−01	6.42e−01	9.54e−01	6.95e−01
		100	**6.23e−01**	7.80e−01	8.65e−01	7.67e−01	8.31e−01	8.61e−01	9.03e−01	7.19e−01
		200	6.99e−01	1.12e+00	1.10e+00	1.04e+00	9.77e−01	**6.54e−01**	8.76e−01	8.24e−01

$C_{i,j}$: 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, Li, Gao, Liao, and Yi (2017), the DMOFJSP addressed in this paper can be described mathematically as follows.

$$\text{Minimize} \begin{cases} TWT = \sum_{i=1}^{n} \max\left(C_{i,n_i} - D_i, 0\right) \cdot Pr_i & (6) \\ 1 \Big/ U_{ave} = 1 \Big/ \left(\frac{1}{m} \sum_{k=1}^{m} \frac{\sum_{i=1}^{n} \sum_{j=1}^{n_i} t_{i,j,k} X_{i,j,k}}{\max_i C_{i,n_i} \cdot X_{i,n_i,k}} \right) & (7) \end{cases}$$

$$s.t. \begin{cases} C_{i,0} = 0, \quad C_{i,j} > 0, \quad \forall i,j & (8) \\ \sum_{k \in M_{ij}} X_{i,j,k} = 1, \quad \forall i,j & (9) \\ \left(C_{i,1} - t_{i,1,k} - A_i\right) X_{i,1,k} \geqslant 0, \quad \forall i,k & (10) \\ \left(C_{i,j} - t_{i,j,k} - C_{i,j-1}\right) X_{i,j,k} \geqslant 0, \quad \forall i,j,k & (11) \\ \left(C_{h,g} - t_{h,g,k} - C_{i,j}\right) X_{i,j,k} X_{h,g,k} \left(Y_{i,j,h,g} + 1\right) \\ + \left(C_{i,j} - t_{i,j,k} - C_{h,g}\right) X_{i,j,k} X_{h,g,k} \left(1 - Y_{i,j,h,g}\right) \geqslant 0, \quad \forall i,j,h,g,k & (12) \end{cases}$$

Objective (6) is total weighted tardiness of all jobs, where the urgency degree is used as weight factor (i.e., penalty factor) of tardiness. Objective (7) is the reciprocal of average machine utilization rate. Eq. (8) indicates that the completion time of each operation must be non-negative. Eq. (9) suggested that each operation can be assigned on only one available machine. Eq. (10) makes sure that a job can only be processed after its arrival time. Precedence constraint is ensured in Eq. (11). Capacity constraint is guaranteed in Eq. (12).

5. Proposed methods for the DMOFJSP

In this section, the fundamental modules of the proposed THDQN are successively provided, including the definition of state features, the candidate dispatching rules (actions) at each rescheduling point, the definition of reward functions and the network structure of the proposed DQN agents. Finally, the overall training and implementation frameworks of the THDQN are derived.

5.1. Definition of state features

In order to improve the effectiveness and generality of the THDQN, ten generic state features are extracted to comprehensively represent a rescheduling point, containing three task-specific features and seven environment-specific features.

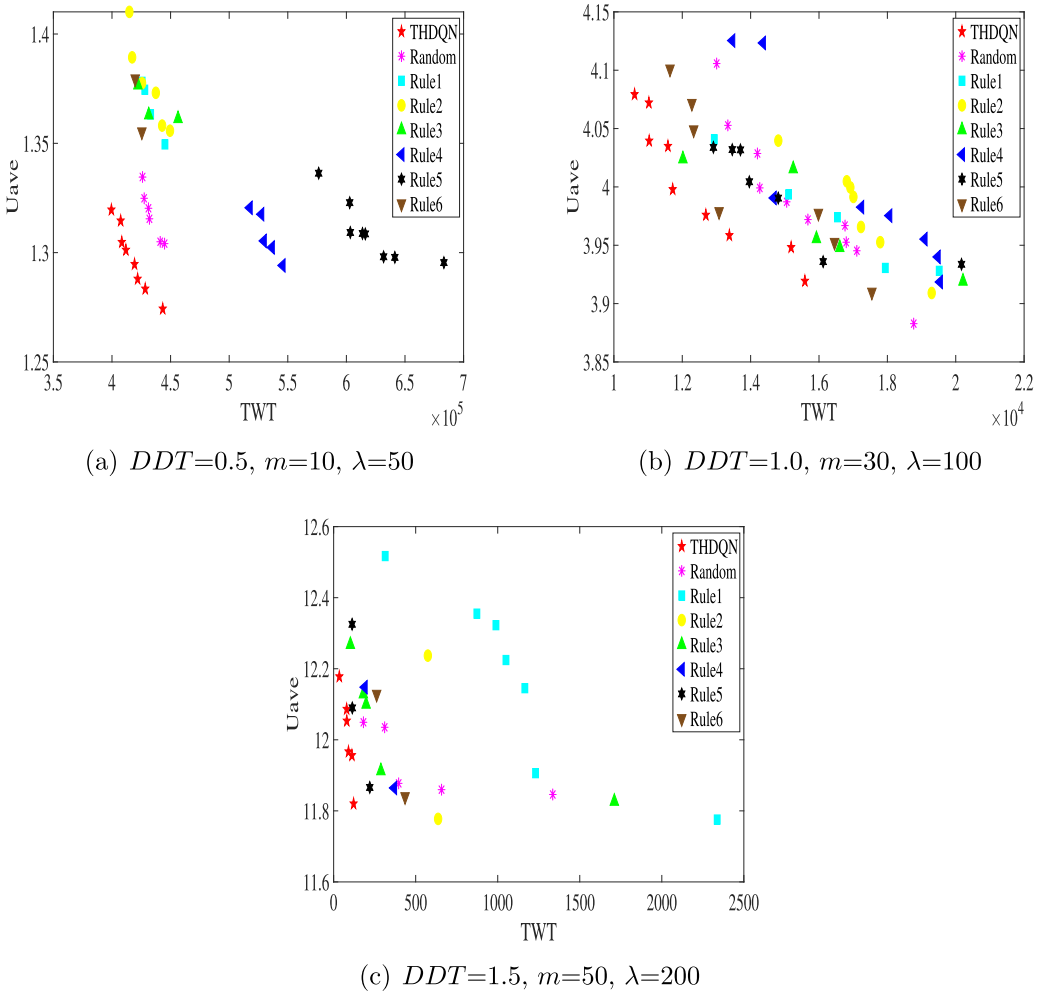
5.1.1. Task-specific features

The task-specific features provide the basic information of a specific task, by utilizing which the THDQN can be generalized to different production environments. Three task-specific features are designed as follows.

(1) total number of machines (*m*) in the shop floor.

(2) due date tightness (*DDT*).

DDT indicates the tightness of a job's slack time from its arrival time to its due date. For a job J_i with n_i operations arriving at time point A_i, its



(a) DDT=0.5, m=10, λ=50 (b) DDT=1.0, m=30, λ=100

(c) DDT=1.5, m=50, λ=200

Fig. 2. Pareto-optimal fronts obtained by the THDQN and each composite dispatching rule for some representative instances.

due date D_i can be calculated as $D_i = A_i + \left(\sum_{j=1}^{n_i} \bar{t}_{i,j}\right) \cdot DDT$, where $\bar{t}_{i,j} = \text{mean}_{k \in M_{i,j}} t_{i,j,k}$ is the average processing time of operation $O_{i,j}$ on all its available machines. It is clear that a smaller DDT means the job is more urgent than others.

(3) Expected value (λ) of the interarrival time (subject to exponential distribution $exp(1/\lambda)$) between two successive new job arrivals.

5.1.2. Environment-specific features

The environment-specific features comprehensively reflect the production status of a rescheduling point. Let $CT_k(t)$ denote the completion time of the last operation assigned on machine M_k at rescheduling point t, and $OP_i(t)$ denote the current number of operations belonging to job J_i that have been assigned. Define the utilization rate of machine M_k at time t as $U_k(t)$, which can be calculated by $U_k(t) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{OP_i(t)} t_{i,j,k} X_{i,j,k}}{CT_k(t)}$. Moreover, define the completion rate of job J_i at t as $CRJ_i(t)$, which can be calculated as $CRJ_i\left(t\right) = \frac{OP_i(t)}{n_i}$. Based on the notations above, the environment-specific state features at each rescheduling point t can be derived as follows.

(1) Average utilization rate of machines $U_{ave}(t)$, as defined in Eq. (13):

$$U_{ave}\left(t\right) = \frac{\sum_{k=1}^{m} U_k\left(t\right)}{m} \tag{13}$$

(2) The standard deviation of machine utilization rate $U_{std}(t)$, as defined in Eq. (14):

$$U_{std}\left(t\right) = \sqrt{\frac{\sum_{k=1}^{m} \left(U_k(t) - U_{ave}(t)\right)^2}{m}} \tag{14}$$

(3) Completion rate of all operations $CRO(t)$, as defined in Eq. (15):

$$CRO\left(t\right) = \frac{\sum_{i=1}^{n} OP_i\left(t\right)}{\sum_{i=1}^{n} n_i} \tag{15}$$

(4) Average completion rate of jobs $CRJ_{ave}(t)$, as defined in Eq. (16):

$$CRJ_{ave}\left(t\right) = \frac{\sum_{i=1}^{n} CRJ_i\left(t\right)}{n} \tag{16}$$

Table 7
GD values for the Pareto-optimal fronts obtained by the THDQN and other well-known dispatching rules.

DDT	*m*	λ	THDQN	FIFO	EDD	MRT	SPT	LPT	CR
0.5	10	50	**0.00e+00**	9.29e−01	1.02e+00	2.67e−01	9.70e−01	1.02e+00	2.60e−01
		100	**0.00e+00**	3.46e−01	9.84e−01	2.64e−01	5.26e−01	1.03e+00	8.24e−02
		200	**8.09e−02**	1.63e−01	1.38e−01	3.38e−01	2.56e−01	1.18e−01	2.53e−01
	30	50	**0.00e+00**	1.42e−01	1.77e−01	2.62e−01	4.94e−01	3.46e−01	2.72e−01
		100	**0.00e+00**	1.01e−01	1.99e−01	4.92e−01	1.66e−01	1.76e−01	1.75e−01
		200	**7.75e−02**	8.58e−02	2.07e−01	1.37e−01	1.42e−01	1.65e−01	9.89e−02
	50	50	**6.71e−02**	1.40e−01	1.16e−01	2.10e−01	4.99e−01	9.47e−01	3.70e−01
		100	**6.49e−02**	1.30e−01	9.11e−02	2.09e−01	2.52e−01	1.94e−01	1.23e−01
		200	1.04e−01	**0.00e+00**	9.46e−02	1.85e−01	1.58e−01	1.61e−01	1.28e−01
1.0	10	50	**0.00e+00**	5.29e−01	1.02e+00	3.49e−01	3.61e−01	5.27e−01	3.38e−01
		100	**0.00e+00**	4.68e−01	4.97e−01	3.22e−01	3.53e−01	1.02e+00	1.31e−01
		200	**0.00e+00**	3.05e−01	1.59e−01	1.29e−01	2.42e−01	2.38e−01	3.69e−01
	30	50	**0.00e+00**	1.14e−01	1.21e−01	2.12e−01	3.40e−01	2.57e−01	1.37e−01
		100	**0.00e+00**	2.56e−01	1.37e−01	3.62e−01	3.42e−01	3.25e−01	2.18e−01
		200	**0.00e+00**	1.26e−01	2.15e−01	1.72e−01	1.74e−01	2.99e−01	1.53e−01
	50	50	1.02e−01	**0.00e+00**	1.33e−01	2.12e−01	9.68e−01	3.47e−01	2.80e−01
		100	**1.11e−01**	1.85e−01	1.25e−01	1.67e−01	3.91e−01	1.58e−01	1.34e−01
		200	1.37e−01	1.46e−01	1.38e−01	3.76e−01	2.35e−01	1.95e−01	**1.14e−01**
1.5	10	50	**0.00e+00**	1.01e+00	5.26e−01	2.68e−01	3.57e−01	5.31e−01	2.51e−01
		100	**0.00e+00**	3.35e−01	4.80e−01	2.72e−01	3.50e−01	5.24e−01	2.19e−01
		200	**0.00e+00**	1.85e−01	2.36e−01	3.43e−01	4.80e−01	3.27e−01	1.50e−01
	30	50	**0.00e+00**	1.46e−01	1.64e−01	3.51e−01	5.09e−01	3.55e−01	3.11e−01
		100	**0.00e+00**	2.81e−01	1.68e−01	2.11e−01	3.15e−01	1.95e−01	2.86e−01
		200	**1.02e−01**	1.57e−01	1.33e−01	2.94e−01	1.79e−01	3.13e−01	1.71e−01
	50	50	**1.20e−01**	1.69e−01	1.94e−01	2.63e−01	3.23e−01	3.36e−01	2.44e−01
		100	2.72e−01	2.15e−01	1.76e−01	**1.35e−01**	2.55e−01	2.65e−01	3.12e−01
		200	2.76e−01	1.56e−01	**0.00e+00**	4.37e−01	3.05e−01	2.07e−01	3.21e−01

Table 8
IGD values for the Pareto-optimal fronts obtained by the THDQN and other well-known dispatching rules.

DDT	*m*	λ	THDQN	FIFO	EDD	MRT	SPT	LPT	CR
0.5	10	50	**0.00e+00**	1.24e+00	1.23e+00	9.68e−01	1.07e+00	1.18e+00	4.60e−02
		100	**1.45e−02**	8.75e−01	8.95e−01	8.16e−01	1.18e+00	1.32e+00	1.53e−02
		200	**3.62e−02**	3.61e−01	2.12e−01	7.27e−01	5.69e−01	5.73e−01	9.46e−02
	30	50	**0.00e+00**	4.66e−02	4.06e−02	6.60e−01	6.51e−01	8.35e−01	8.97e−02
		100	**2.59e−02**	9.10e−02	8.89e−02	5.72e−01	4.07e−01	6.66e−01	1.18e−01
		200	**8.50e−02**	8.73e−02	2.80e−01	2.74e−01	3.26e−01	3.91e−01	2.31e−01
	50	50	**1.33e−02**	3.31e−02	3.74e−02	5.15e−01	8.43e−01	7.98e−01	1.28e−01
		100	**3.49e−02**	6.31e−02	4.02e−02	6.61e−01	3.04e−01	3.65e−01	7.56e−02
		200	1.47e−01	1.09e−01	**6.19e−02**	2.53e−01	2.96e−01	3.07e−01	1.36e−01
1.0	10	50	**0.00e+00**	1.27e+00	1.21e+00	9.18e−01	1.30e+00	1.20e+00	6.65e−01
		100	**0.00e+00**	3.69e−01	6.97e−01	3.91e−01	1.03e+00	1.33e+00	1.58e−01
		200	**8.45e−02**	2.81e−01	2.38e−01	5.76e−01	4.06e−01	4.09e−01	2.44e−01
	30	50	**2.73e−03**	1.88e−02	3.60e−02	5.50e−01	7.89e−01	6.75e−01	6.37e−02
		100	**1.33e−01**	2.33e−01	2.26e−01	3.88e−01	4.05e−01	3.79e−01	2.18e−01
		200	**3.25e−02**	7.53e−02	1.77e−01	3.90e−01	5.87e−01	1.99e−01	1.43e−01
	50	50	**2.66e−02**	3.90e−02	4.08e−02	7.40e−01	9.26e−01	8.98e−01	1.49e−01
		100	5.36e−02	9.82e−02	**4.68e−02**	2.23e−01	2.60e−01	2.34e−01	6.99e−02
		200	**9.69e−02**	1.75e−01	1.11e−01	6.57e−01	3.16e−01	3.63e−01	9.98e−02
1.5	10	50	**0.00e+00**	1.08e+00	1.12e+00	9.81e−01	1.14e+00	1.27e+00	6.08e−01
		100	**1.18e−01**	6.62e−01	4.57e−01	1.35e−01	1.02e+00	1.20e+00	1.29e−01
		200	**0.00e+00**	2.00e−01	1.91e−01	6.59e−01	4.95e−01	4.33e−01	1.34e−01
	30	50	**5.98e−03**	1.17e−02	8.04e−02	9.79e−01	8.55e−01	1.04e+00	2.99e−01
		100	**0.00e+00**	1.25e−01	1.21e−01	5.88e−01	3.15e−01	3.62e−01	1.85e−01
		200	**1.18e−01**	1.36e−01	1.29e−01	3.00e−01	1.62e−01	3.79e−01	2.25e−01
	50	50	**3.05e−02**	3.13e−02	4.32e−02	5.24e−01	6.48e−01	8.49e−01	1.40e−01
		100	2.98e−01	**2.44e−02**	2.09e−01	3.28e−01	6.74e−01	6.95e−01	5.97e−01
		200	**2.94e−02**	3.99e−02	4.01e−02	5.57e−01	4.52e−01	4.15e−01	2.71e−01

Table 9
Δ values for the Pareto-optimal fronts obtained by the THDQN and other well-known dispatching rules.

DDT	*m*	*λ*	THDQN	FIFO	EDD	MRT	SPT	LPT	CR
0.5	10	50	**6.92e−01**	9.76e−01	7.04e−01	9.85e−01	9.86e−01	7.72e−01	8.12e−01
		100	**6.76e−01**	9.93e−01	7.54e−01	9.76e−01	9.68e−01	8.26e−01	8.92e−01
		200	**7.76e−01**	9.04e−01	8.35e−01	8.92e−01	7.94e−01	9.34e−01	8.90e−01
	30	50	**5.25e−01**	9.01e−01	9.15e−01	8.68e−01	8.92e−01	8.95e−01	8.80e−01
		100	**6.45e−01**	8.30e−01	6.52e−01	7.83e−01	7.74e−01	8.97e−01	7.37e−01
		200	7.39e−01	8.53e−01	9.55e−01	7.99e−01	7.93e−01	7.89e−01	**6.95e−01**
	50	50	**6.78e−01**	7.82e−01	7.66e−01	9.28e−01	8.00e−01	8.21e−01	7.89e−01
		100	8.07e−01	**7.30e−01**	8.52e−01	9.50e−01	8.59e−01	9.98e−01	1.02e+00
		200	8.36e−01	8.45e−01	**7.07e−01**	8.17e−01	8.75e−01	8.35e−01	7.75e−01
1.0	10	50	**6.18e−01**	9.80e−01	7.82e−06	9.94e−01	9.81e−01	9.79e−01	9.79e−01
		100	**6.35e−01**	8.25e−01	9.39e−01	8.82e−01	9.51e−01	8.36e−01	9.52e−01
		200	8.57e−01	**7.53e−01**	8.91e−01	9.47e−01	8.72e−01	9.29e−01	8.86e−01
	30	50	**5.71e−01**	7.25e−01	9.24e−01	9.15e−01	9.57e−01	9.33e−01	9.87e−01
		100	**6.94e−01**	9.14e−01	9.67e−01	7.86e−01	8.01e−01	8.47e−01	8.75e−01
		200	**5.72e−01**	6.94e−01	8.78e−01	8.58e−01	8.01e−01	7.76e−01	7.02e−01
	50	50	7.04e−01	6.99e−01	7.07e−01	9.36e−01	**6.61e−01**	9.38e−01	7.17e−01
		100	7.43e−01	7.48e−01	**6.13e−01**	8.66e−01	6.97e−01	7.45e−01	6.24e−01
		200	**7.50e−01**	9.44e−01	9.33e−01	9.16e−01	9.08e−01	9.00e−01	8.76e−01
1.5	10	50	**6.51e−01**	7.33e−01	9.52e−01	9.90e−01	9.86e−01	9.53e−01	8.43e−01
		100	8.71e−01	9.74e−01	8.77e−01	**7.26e−01**	9.72e−01	9.35e−01	8.79e−01
		200	**6.59e−01**	9.22e−01	9.20e−01	7.47e−01	7.84e−01	8.88e−01	7.69e−01
	30	50	**8.79e−01**	9.26e−01	9.71e−01	9.91e−01	9.25e−01	9.62e−01	9.45e−01
		100	**5.27e−01**	6.99e−01	9.25e−01	8.26e−01	8.95e−01	8.51e−01	1.02e+00
		200	7.12e−01	9.14e−01	9.82e−01	7.69e−01	**6.69e−01**	8.02e−01	8.63e−01
	50	50	**7.15e−01**	8.96e−01	8.45e−01	8.84e−01	9.83e−01	9.44e−01	8.24e−07
		100	8.71e−01	**8.20e−01**	9.14e−01	9.25e−01	9.84e−01	9.40e−01	9.09e−01
		200	**6.03e−01**	8.10e−01	8.29e−01	7.84e−01	8.66e−01	8.94e−01	6.36e−01

(5) The standard deviation of job completion rate $CRJ_{std}(t)$, as defined in Eq. (17):

$$CRJ_{std}\left(t\right) = \sqrt{\frac{\sum_{i=1}^{n}\left(CRJ_i(t) - CRJ_{ave}(t)\right)^2}{n}} \tag{17}$$

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

The estimated tardiness rate $Tard_e(t)$ is defined as the number of estimated tardy operations divided by the number of all unassigned operations, 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: $Tard_e(t)$
$T_{cur} \leftarrow \text{mean}_k CT_k(t)$
$N_{tard} \leftarrow 0$
$N_{left} \leftarrow 0$
for $i = 1 : n$ **do**
if $OP_i(t) < n_i$ **then**
  $N_{left} \leftarrow N_{left} + n_i - OP_i(t)$
  $T_{left} \leftarrow 0$
  for $j = OP_i(t) + 1 : n_i$ **do**
    $T_{left} \leftarrow T_{left} + \bar{t}_{i,j}$
    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
$Tard_e\left(t\right) \leftarrow \frac{N_{tard}}{N_{left}}$
Return $Tard_e(t)$

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

The actual tardiness rate $Tard_a(t)$ is defined as the number of actual tardy operations divided by the number of all unassigned operations, 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
$Tard_a\left(t\right) \leftarrow \frac{N_{tard}}{N_{left}}$
Return $Tard_a(t)$

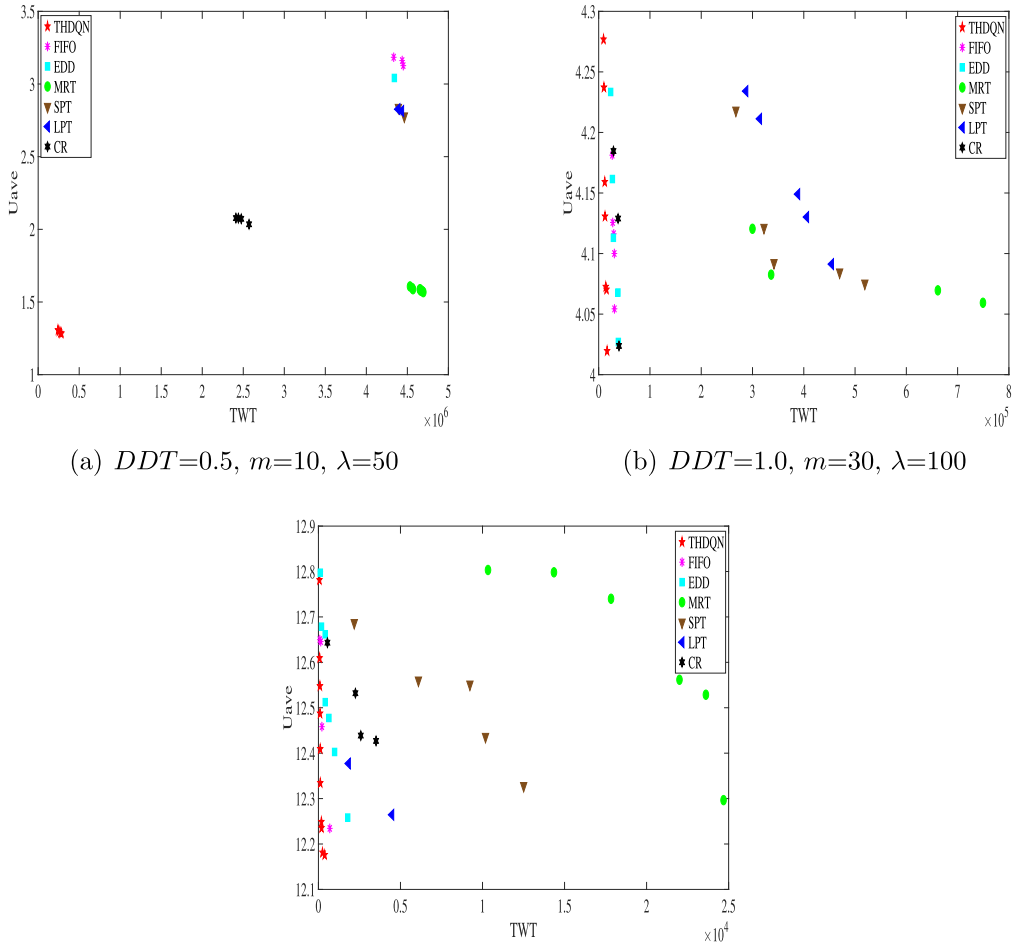
5.2. The proposed composite dispatching rules

Since no single dispatching rule can achieve the best performance over all shop configurations due to its myopia (Nie et al., 2013), six composite dispatching rules are developed to simultaneously select an operation and assign it on a feasible machine at each rescheduling point. All of the proposed rules are designed to reduce the total weighted tardiness or enhance the machine utilization rate.

Denote $T_{cur} = \text{mean}_k CT_k(t)$ as the average completion time of the last operation on all machines at current rescheduling time t. The six composite dispatching rules are described in details as follows.

5.2.1. Composite dispatching rule 1

In dispatching rule 1, we first identify the estimated set of tardy jobs $\boldsymbol{Tard_{job}(t)}$ and the set of all uncompleted jobs $\boldsymbol{UC_{job}(t)}$ at the current decision point t. If $\boldsymbol{Tard_{job}(t)}$ is empty, the next operation of job $J_i \in$



(a) DDT=0.5, m=10, λ=50 (b) DDT=1.0, m=30, λ=100

(c) DDT=1.5, m=50, λ=200

Fig. 3. Pareto-optimal fronts obtained by the THDQN and other well-known dispatching rules for some representative instances.

$\boldsymbol{UC_{job}}(t)$ with the minimum value of $\frac{D_i-\max(T_{cur},C_{i,OP_i(t)})}{n_i-OP_i(t)}\cdot\frac{1}{Pr_i}$ is selected. If $\boldsymbol{Tard_{job}}(t)$ is not empty, the next operation of the job $J_i \in \boldsymbol{Tard_{job}}(t)$ with the maximum estimated weighted tardiness calculated as $(\max(T_{cur}, C_{i,OP_i(t)}) + \sum_{j=OP_i(t)+1}^{n_i} \bar{t}_{i,j} - D_i)\cdot Pr_i$ is chosen. Then the chosen operation $O_{i,j}$ is assigned on the earliest available machine M_k where $k = \text{argmin}_{k\in M_{i,j}}\left(\max(CT_k(t), C_{i,j-1}, A_i)\right)$. The procedure of dispatching rule 1 is shown in Algorithm 4.

Algorithm 4. Procedure of composite dispatching rule 1

1: $T_{cur} \leftarrow \text{mean}_k CT_k(t)$
2: $\boldsymbol{Tard_{job}}(t) \leftarrow \{i | OP_i(t) < n_i \&\& D_i < \max(T_{cur}, C_{i,OP_i(t)})\}$
3: $\boldsymbol{UC_{job}}(t) \leftarrow \{i | OP_i(t) < n_i\}$
4: **if** isempty($\boldsymbol{Tard_{job}}(t)$) **then**
5: $J_i \leftarrow \text{argmin}_{i\in \boldsymbol{UC_{job}}(t)} \frac{D_i - \max(T_{cur}, C_{i,OP_i(t)})}{n_i - OP_i(t)} \cdot \frac{1}{Pr_i}$
6: **else**
7: $J_i \leftarrow \text{argmax}_{i\in \boldsymbol{Tard_{job}}(t)} ((\max(T_{cur}, C_{i,OP_i(t)}) + \sum_{j=OP_i(t)+1}^{n_i} \bar{t}_{i,j} - D_i)\cdot Pr_i)$
8: **end if**
9: $j \leftarrow OP_i(t) + 1$
10: $M_k \leftarrow \text{argmin}_{k\in M_{i,j}} \left(\max(CT_k(t), C_{i,j-1}, A_i)\right)$
11: assign $O_{i,j}$ on M_k

5.2.2. Composite dispatching rule 2

In dispatching rule 2, we first obtain the estimated set of tardy jobs $\boldsymbol{Tard_{job}}(t)$. If no tardy job exists, the next operation of job $J_i \in \boldsymbol{UC_{job}}(t)$ with the minimum value of $\frac{D_i-\max(T_{cur},C_{i,OP_i(t)})}{\sum_{j=OP_i(t)+1}^{n_i}\bar{t}_{i,j}}\cdot\frac{1}{Pr_i}$ is selected. Otherwise, the next operation of the job $J_i \in \boldsymbol{Tard_{job}}(t)$ with the maximum estimated weighted tardiness defined as $(\max(T_{cur}, C_{i,OP_i(t)}) + \sum_{j=OP_i(t)+1}^{n_i} \bar{t}_{i,j} - D_i)\cdot Pr_i$ is selected. Next, the selected operation is assigned on the earliest available machine. The procedure of rule 2 is provided in Algorithm 5.

Algorithm 5. Procedure of composite dispatching rule 2

1: $T_{cur} \leftarrow \text{mean}_k CT_k(t)$
2: $\boldsymbol{Tard_{job}}(t) \leftarrow \{i | OP_i(t) < n_i \&\& D_i < \max(T_{cur}, C_{i,OP_i(t)})\}$
3: $\boldsymbol{UC_{job}}(t) \leftarrow \{i | OP_i(t) < n_i\}$
4: **if** isempty($\boldsymbol{Tard_{job}}(t)$) **then**
5: $J_i \leftarrow \text{argmin}_{i\in \boldsymbol{UC_{job}}(t)} \frac{D_i - \max(T_{cur}, C_{i,OP_i(t)})}{\sum_{j=OP_i(t)+1}^{n_i} \bar{t}_{i,j}} \cdot \frac{1}{Pr_i}$
6: **else**
7: $J_i \leftarrow \text{argmax}_{i\in \boldsymbol{Tard_{job}}(t)} ((\max(T_{cur}, C_{i,OP_i(t)}) + \sum_{j=OP_i(t)+1}^{n_i} \bar{t}_{i,j} - D_i)\cdot Pr_i)$
8: **end if**
9: $j \leftarrow OP_i(t) + 1$
10: $M_k \leftarrow \text{argmin}_{k\in M_{i,j}} \left(\max(CT_k(t), C_{i,j-1}, A_i)\right)$
11: assign $O_{i,j}$ on M_k

Table 10
GD values for the Pareto-optimal fronts obtained by the THDQN and other well-known composite rules.

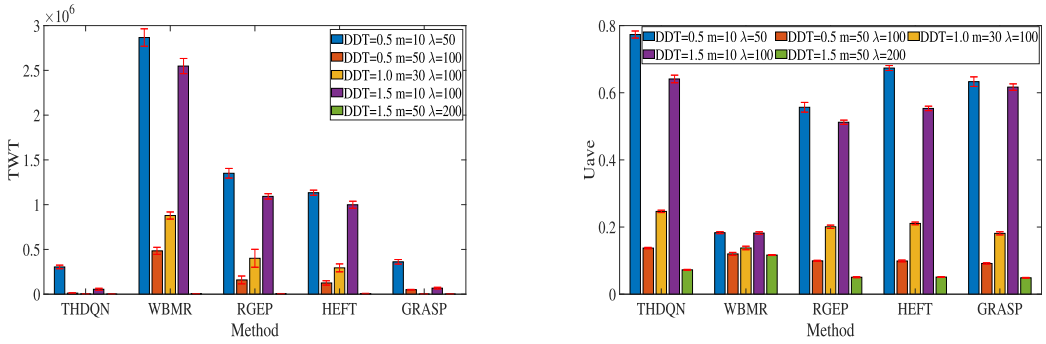
DDT	*m*	λ	THDQN	WBMR	RGEP	HEFT	GRASP
0.5	10	50	**0.00e+00**	3.64e−01	2.67e−01	4.62e−01	2.46e−01
		100	**0.00e+00**	3.55e−01	2.39e−01	2.32e−01	**0.00e+00**
		200	**0.00e+00**	5.28e−01	1.84e−01	2.21e−01	**0.00e+00**
	30	50	**0.00e+00**	1.00e+00	2.37e−01	2.01e−01	**0.00e+00**
		100	**0.00e+00**	5.33e−01	9.04e−01	4.58e−01	**0.00e+00**
		200	1.03e−01	**0.00e+00**	1.75e−01	1.77e−01	**0.00e+00**
	50	50	**0.00e+00**	1.02e+00	4.71e−01	2.00e−01	**0.00e+00**
		100	**0.00e+00**	5.09e−01	1.96e−01	1.87e−01	**0.00e+00**
		200	**0.00e+00**	**0.00e+00**	2.37e−01	1.11e−01	**0.00e+00**
1.0	10	50	3.46e−01	1.03e+00	4.85e−01	2.45e−01	**0.00e+00**
		100	**0.00e+00**	2.77e−01	4.80e−01	4.80e−01	2.02e−01
		200	**0.00e+00**	1.03e+00	3.16e−01	3.01e−01	1.27e−01
	30	50	**0.00e+00**	3.64e−01	3.32e−01	3.29e−01	1.26e−01
		100	**4.29e−02**	3.61e−01	2.48e−01	1.64e−01	1.13e−01
		200	**0.00e+00**	**0.00e+00**	2.05e−01	1.50e−01	3.40e−01
	50	50	**0.00e+00**	1.04e+00	9.65e−01	3.37e−01	1.71e−01
		100	**0.00e+00**	1.00e+00	2.10e−01	1.77e−01	1.78e−01
		200	**0.00e+00**	**0.00e+00**	2.56e−01	3.39e−01	1.73e−01
1.5	10	50	**0.00e+00**	5.34e−01	4.65e−01	4.54e−01	**0.00e+00**
		100	**3.22e−02**	5.34e−01	9.16e−01	4.67e−01	1.26e−01
		200	**0.00e+00**	5.35e−01	4.80e−01	2.46e−01	1.81e−01
	30	50	**0.00e+00**	2.76e−01	9.40e−01	3.30e−01	1.49e−01
		100	**1.15e−01**	1.00e+00	3.47e−01	2.14e−01	2.66e−01
		200	**0.00e+00**	**0.00e+00**	3.31e−01	1.73e−01	3.41e−01
	50	50	1.60e−01	3.63e−01	4.88e−01	2.04e−01	**7.46e−02**
		100	**0.00e+00**	9.77e−01	2.68e−01	2.63e−01	2.64e−01
		200	**0.00e+00**	**0.00e+00**	1.79e−01	2.60e−01	4.83e−01

Table 11
IGD values for the Pareto-optimal fronts obtained by the THDQN and other well-known composite rules.

DDT	*m*	λ	THDQN	WBMR	RGEP	HEFT	GRASP
0.5	10	50	**0.00e+00**	1.38e+00	3.25e−01	4.05e−01	6.52e−02
		100	**2.32e−01**	1.23e+00	4.02e−01	2.97e−01	2.96e−01
		200	1.19e−01	1.28e+00	1.76e−01	1.51e−01	**7.91e−02**
	30	50	**1.64e−01**	1.05e+00	6.90e−01	5.28e−01	8.29e−01
		100	7.35e−02	1.34e+00	3.92e−01	3.31e−01	**1.61e−02**
		200	3.76e−01	1.12e+00	2.80e−01	2.91e−01	**2.74e−01**
	50	50	**1.13e−01**	1.25e+00	4.14e−01	4.54e−01	1.61e−01
		100	**2.36e−01**	1.03e+00	6.02e−01	3.88e−01	3.16e−01
		200	**4.05e−01**	5.51e−01	6.75e−01	5.20e−01	5.84e−01
1.0	10	50	1.87e−02	1.41e+00	5.10e−01	4.05e−01	**0.00e+00**
		100	**2.32e−02**	1.38e+00	4.65e−01	4.67e−01	5.23e−02
		200	**0.00e+00**	1.41e+00	2.99e−01	2.03e−01	2.67e−02
	30	50	**0.00e+00**	1.38e+00	5.62e−01	5.09e−01	9.84e−02
		100	**1.72e−01**	1.29e+00	4.05e−01	3.53e−01	3.70e−01
		200	**3.36e−01**	6.65e−01	7.35e−01	7.68e−01	9.12e−01
	50	50	**0.00e+00**	1.41e+00	7.00e−01	6.19e−01	4.67e−01
		100	**0.00e+00**	1.00e+00	7.50e−01	6.97e−01	7.51e−01
		200	**4.34e−01**	4.90e−01	6.96e−01	6.99e−01	6.57e−01
1.5	10	50	2.87e−02	1.38e+00	3.66e−01	2.97e−01	**1.57e−02**
		100	**3.70e−03**	1.37e+00	4.18e−01	3.55e−01	2.66e−02
		200	**0.00e+00**	1.39e+00	4.01e−01	3.65e−01	1.42e−01
	30	50	**5.46e−02**	1.35e+00	5.39e−01	5.15e−01	1.03e−01
		100	**2.07e−01**	1.02e+00	8.55e−01	8.39e−01	8.44e−01
		200	**3.98e−01**	5.93e−01	7.84e−01	7.57e−01	9.34e−01
	50	50	2.35e−01	1.36e+00	5.29e−01	5.14e−01	**1.74e−01**
		100	**1.79e−01**	7.97e−01	9.78e−01	8.50e−01	8.25e−01
		200	**2.28e−01**	5.50e−01	7.89e−01	7.78e−01	9.27e−01

Table 12
Δ values for the Pareto-optimal fronts obtained by the THDQN and other well-known composite rules.

DDT	*m*	λ	THDQN	WBMR	RGEP	HEFT	GRASP
0.5	10	50	**9.77e−01**	9.98e−01	9.84e−01	9.90e−01	9.85e−01
		100	**9.72e−01**	9.90e−01	9.97e−01	9.79e−01	9.95e−01
		200	9.28e−01	9.67e−01	8.91e−01	**8.20e−01**	1.01e+00
	30	50	9.68e−01	**8.57e−01**	9.95e−01	1.01e+00	9.54e−01
		100	8.67e−00	9.72e−01	9.23e−01	9.62e−01	**7.87e−01**
		200	9.52e−01	9.83e−01	9.58e−01	9.68e−01	**9.00e−01**
	50	50	**7.79e−01**	9.85e−01	9.12e−01	9.68e−01	7.85e−01
		100	9.20e−01	8.77e−01	9.67e−01	**8.25e−01**	9.65e−01
		200	**8.97e−01**	9.73e−01	9.80e−01	9.70e−01	9.53e−01
1.0	10	50	**7.29e−01**	9.37e−01	9.26e−01	9.87e−01	8.51e−01
		100	**8.55e−01**	9.86e−01	9.54e−01	9.73e−01	8.65e−01
		200	**9.25e−01**	9.74e−01	9.91e−01	9.33e−01	9.29e−01
	30	50	9.20e−00	9.86e−01	9.79e−01	9.72e−01	**8.29e−01**
		100	**7.62e−01**	9.24e−01	8.33e−01	9.65e−01	9.90e−01
		200	9.56e−01	8.99e−01	**8.95e−01**	9.78e−01	9.66e−01
	50	50	**8.41e−01**	9.36e−01	9.72e−01	8.85e−01	9.47e−01
		100	**8.38e−01**	1.03e+00	8.81e−01	8.89e−01	9.66e−01
		200	9.81e−01	9.09e−01	9.36e−01	**8.96e−01**	1.02e+00
1.5	10	50	8.05e−01	9.89e−01	9.70e−01	9.71e−01	**7.07e−01**
		100	**8.55e−01**	9.76e−01	8.72e−01	9.66e−01	9.01e−01
		200	1.26e+00	9.81e−01	**7.80e−01**	9.66e−01	9.02e−01
	30	50	**7.49e−01**	8.76e−01	9.40e−01	9.30e−01	7.73e−01
		100	8.22e−01	1.10e+00	8.95e−01	**7.90e−01**	9.84e−01
		200	**9.09e−01**	9.36e−01	9.18e−01	9.16e−01	9.39e−01
	50	50	**7.95e−01**	9.66e−01	8.96e−01	9.35e−01	8.56e−01
		100	**8.61e−01**	1.05e+00	9.22e−01	9.16e−01	9.82e−01
		200	9.68e−01	**8.47e−01**	9.38e−01	9.63e−01	9.55e−01



(a) TWT (b) U_{ave}

Fig. 4. Average values of two objectives obtained by the THDQN and other existing composite rules on some representative instances.

5.2.3. *Composite dispatching rule 3*

In dispatching rule 3, The next operation of job $J_i \in \boldsymbol{UC_{job}}(t)$ with the maximum estimated weighted tardiness calculated as $\max(T_{cur}, C_i, OP_i(t)) + \sum_{j=OP_i(t)+1}^{n_i} \bar{t}_{i,j}$

$$-D_i < 0$$

$$? \ \max(T_{cur}, C_{i,OP_i(t)}) + \sum_{j=OP_i(t)+1}^{n_i} \bar{t}_{i,j}$$

$$-D_i : \left(\max\left(T_{cur}, C_{i,OP_i(t)}\right) + \sum_{j=OP_i(t)+1}^{n_i} \bar{t}_{i,j} - D_i\right) \cdot Pr_i$$

is selected. 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 composite dispatching rule 3

1: $T_{cur} \leftarrow \text{mean}_k CT_k(t)$
2: $\boldsymbol{UC_{job}}(t) \leftarrow \{i | OP_i(t) < n_i\}$
3: $J_i \leftarrow \text{argmax}_{i \in \boldsymbol{UC_{job}}(t)} (\max(T_{cur}, C_{i,OP_i(t)}) + \sum_{j=OP_i(t)+1}^{n_i} \bar{t}_{i,j} - D_i < 0 \ ? \ \max(T_{cur}, C_{i,OP_i(t)}) + \sum_{j=OP_i(t)+1}^{n_i} \bar{t}_{i,j} - D_i : (\max(T_{cur}, C_{i,OP_i(t)}) + \sum_{j=OP_i(t)+1}^{n_i} \bar{t}_{i,j} - D_i) \cdot Pr_i)$
4: $j \leftarrow OP_i(t) + 1$
5: Generate a random number r in $[0, 1]$
6: **if** $r < 0.5$ **then**
7: $\quad M_k \leftarrow \text{argmin}_{k \in M_{ij}} \frac{\sum_{i=1}^{n} \sum_{j=1}^{OP_i(t)} t_{i,j,k} X_{i,j,k}}{CT_k(t)}$
8: **else**
9: $\quad M_k \leftarrow \text{argmin}_{k \in M_{ij}} \sum_{i=1}^{n} \sum_{j=1}^{OP_i(t)} t_{i,j,k} X_{i,j,k}$
10: **end if**
11: assign $O_{i,j}$ on M_k

Table 13
GD values for the Pareto-optimal fronts obtained by the THDQN and other RL based scheduling methods.

DDT	*m*	λ	THDQN	Q −learning	SHDQN	DDQN
0.5	10	50	**0.00e+00**	5.33e−01	5.22e−01	4.83e−01
		100	**0.00e+00**	1.83e−01	4.92e−01	8.44e−01
		200	1.72e−01	2.35e−01	2.09e−01	**0.00e+00**
	30	50	**0.00e+00**	3.39e−01	4.69e−01	**0.00e+00**
		100	**0.00e+00**	9.53e−01	2.59e−01	2.11e−01
		200	**0.00e+00**	**0.00e+00**	1.46e−01	1.83e−01
	50	50	**0.00e+00**	9.76e−01	2.63e−01	5.07e−01
		100	**0.00e+00**	**0.00e+00**	**0.00e+00**	1.30e−01
		200	**0.00e+00**	**0.00e+00**	2.49e−01	1.43e−01
1.0	10	50	**0.00e+00**	2.74e−01	5.15e−01	2.53e−01
		100	**0.00e+00**	2.22e−01	5.13e−01	2.53e−01
		200	**0.00e+00**	5.16e−01	3.29e−01	1.62e−01
	30	50	**0.00e+00**	5.24e−01	3.45e−01	7.88e−01
		100	**0.00e+00**	9.98e−01	3.29e−01	**0.00e+00**
		200	**0.00e+00**	**0.00e+00**	2.18e−01	3.46e−01
	50	50	**0.00e+00**	2.72e−01	3.40e−01	4.16e−01
		100	**0.00e+00**	5.01e−01	1.97e−01	2.27e−01
		200	2.48e−01	**0.00e+00**	2.31e−01	**0.00e+00**
1.5	10	50	**0.00e+00**	5.33e−01	5.17e−01	9.34e−01
		100	**0.00e+00**	1.03e+00	1.00e+00	1.73e−01
		200	**0.00e+00**	9.89e−01	4.66e−01	1.85e−01
	30	50	**0.00e+00**	1.02e+00	9.91e−01	8.99e−01
		100	**0.00e+00**	3.49e−01	4.60e−01	1.37e−01
		200	1.17e−01	**0.00e+00**	**0.00e+00**	**0.00e+00**
	50	50	**0.00e+00**	1.00e+00	4.76e−01	2.62e−01
		100	**0.00e+00**	**0.00e+00**	**0.00e+00**	1.65e−01
		200	**0.00e+00**	1.30e−01	**0.00e+00**	**0.00e+00**

Table 14
IGD values for the Pareto-optimal fronts obtained by the THDQN and other RL based scheduling methods.

DDT	*m*	λ	THDQN	Q −learning	SHDQN	DDQN
0.5	10	50	**0.00e+00**	1.32e+00	1.06e+00	4.54e−01
		100	**0.00e+00**	1.04e+00	4.75e−01	2.21e−01
		200	3.22e−01	4.55e−01	3.14e−01	**2.47e−01**
	30	50	**1.18e−01**	7.81e−01	4.99e−01	2.10e−01
		100	**0.00e+00**	7.77e−01	7.28e−01	6.90e−01
		200	4.90e−01	2.97e−01	**1.84e−01**	4.02e−01
	50	50	**0.00e+00**	9.04e−01	8.28e−01	6.51e−01
		100	**2.28e−01**	3.77e−01	3.13e−01	3.81e−01
		200	**2.90e−01**	4.07e−01	3.29e−01	3.47e−01
1.0	10	50	**0.00e+00**	1.16e+00	8.90e−01	5.04e−01
		100	**0.00e+00**	1.23e+00	8.53e−01	5.46e−01
		200	**0.00e+00**	9.96e−01	3.89e−01	3.40e−01
	30	50	**0.00e+00**	1.19e+00	8.13e−01	1.79e−01
		100	2.16e−01	1.04e+00	5.53e−01	**1.66e−01**
		200	**2.50e−01**	5.64e−01	3.40e−01	3.95e−01
	50	50	**0.00e+00**	1.15e+00	7.22e−01	1.30e−01
		100	**6.90e−02**	8.49e−01	1.04e−01	3.63e−01
		200	3.85e−01	6.26e−01	2.56e−01	**2.51e−01**
1.5	10	50	**0.00e+00**	1.32e+00	9.86e−01	5.14e−01
		100	**0.00e+00**	1.38e+00	1.04e+00	5.59e−01
		200	**7.37e−02**	1.04e+00	5.52e−01	2.30e−01
	30	50	**0.00e+00**	1.36e+00	1.01e+00	4.62e−01
		100	**5.39e−02**	1.03e+00	3.17e−01	7.98e−02
		200	5.46e−01	4.57e−01	**2.75e−01**	4.63e−01
	50	50	**7.34e−02**	1.14e+00	6.05e−01	1.41e−01
		100	5.74e−01	6.06e−01	2.87e−01	**2.49e−01**
		200	**3.16e−01**	8.16e−01	3.28e−01	3.47e−01

Table 15
Δ values for the Pareto-optimal fronts obtained by the THDQN and other RL based scheduling methods.

DDT	*m*	λ	THDQN	Q −learning	SHDQN	DDQN
0.5	10	50	**7.52e−01**	8.94e−01	9.20e−01	7.89e−01
		100	**6.47e−01**	9.54e−01	6.90e−01	9.91e−01
		200	7.17e−01	7.73e−01	**6.88e−01**	7.27e−01
	30	50	8.55e−01	8.55e−01	8.31e−01	**7.19e−01**
		100	**4.32e−01**	1.00e+00	8.92e−01	9.98e−01
		200	**7.65e−01**	8.09e−01	8.12e−01	8.57e−01
	50	50	8.41e−01	1.00e+00	8.86e−01	**7.03e−01**
		100	**8.54e−01**	9.92e−01	8.65e−01	9.16e−01
		200	**8.47e−01**	9.09e−01	8.56e−01	8.97e−01
1.0	10	50	**8.29e−01**	8.40e−01	8.34e−01	9.71e−01
		100	**7.66e−01**	9.80e−01	8.36e−01	9.60e−01
		200	9.23e−01	**7.49e−01**	9.03e−01	8.83e−01
	30	50	**7.69e−01**	9.47e−01	8.52e−01	1.00e+00
		100	8.95e−01	1.00e+00	7.86e−01	**7.31e−01**
		200	**7.10e−01**	9.26e−01	7.75e−01	9.41e−01
	50	50	**5.89e−01**	9.03e−01	9.51e−01	5.99e−01
		100	9.05e−01	6.72e−01	**6.65e−01**	7.58e−01
		200	8.60e−01	7.49e−01	7.09e−01	**6.09e−01**
1.5	10	50	**8.07e−01**	9.46e−01	9.11e−01	9.97e−01
		100	**7.67e−01**	9.75e−01	1.00e+00	9.26e−01
		200	8.41e−01	9.24e−01	7.73e−01	**7.08e−01**
	30	50	**7.55e−01**	9.33e−01	9.47e−01	9.41e−01
		100	**7.22e−01**	8.51e−01	8.16e−01	8.25e−01
		200	8.53e−01	8.95e−01	**8.11e−01**	1.00e+00
	50	50	**6.68e−01**	1.00e+00	7.87e−01	8.24e−01
		100	9.40e−01	8.89e−01	8.41e−01	**7.91e−01**
		200	**7.54e−01**	8.53e−01	7.72e−01	8.86e−01

5.2.4. Composite dispatching rule 4

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

Algorithm 7. Procedure of composite dispatching rule 4

1: $\boldsymbol{UC_{job}(t)} \leftarrow \{i | OP_i(t) < n_i\}$
2: Randomly choose an uncompleted job J_i from $\boldsymbol{UC_{job}(t)}$
3: $j \leftarrow OP_i(t) + 1$
4: $M_k \leftarrow \text{argmin}_{k \in M_{i,j}} \left(\max(CT_k(t), C_{i,j-1}, A_i)\right)$
5: assign $O_{i,j}$ on M_k

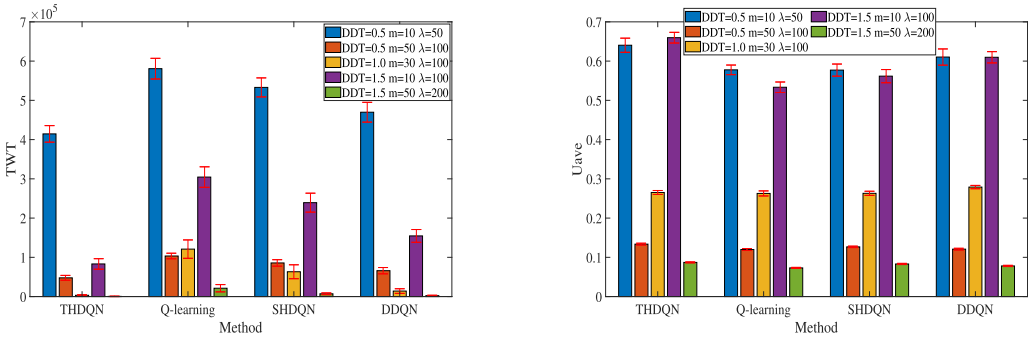
5.2.5. Composite dispatching rule 5

In dispatching rule 5, we first find the estimated tardy set $\boldsymbol{Tard_{job}(t)}$. If no tardy job exists, the next operation of job $J_i \in \boldsymbol{UC_{job}(t)}$ with the minimum value of $\frac{OP_i(t)}{n_i} \cdot \left(D_i - \max\left(T_{cur}, C_{i,OP_i(t)}\right)\right) \cdot \frac{1}{Pr_i}$ is selected. Otherwise, the next operation of job $J_i \in \boldsymbol{Tard_{job}(t)}$ with the maximum value of $\frac{n_i}{OP_i(t)} \cdot \left(\max\left(T_{cur}, C_{i,OP_i(t)}\right) + \sum_{j=OP_i(t)+1}^{n_i} \bar{t}_{ij} - D_i\right) \cdot Pr_i$ is selected. Finally, the selected operation is assigned on the earliest available machine. The procedure of rule 5 is given in Algorithm 8.

Algorithm 8. Procedure of composite dispatching rule 5

1: $T_{cur} \leftarrow \text{mean}_k CT_k(t)$
2: $\boldsymbol{Tard_{job}(t)} \leftarrow \{i | OP_i(t) < n_i \&\& D_i < \max(T_{cur}, C_{i,OP_i(t)})\}$
3: $\boldsymbol{UC_{job}(t)} \leftarrow \{i | OP_i(t) < n_i\}$
4: **if** isempty($\boldsymbol{Tard_{job}(t)}$) **then**
5: $J_i \leftarrow \text{argmin}_{i \in \boldsymbol{UC_{job}(t)}} \frac{OP_i(t)}{n_i} \cdot \left(D_i - \max\left(T_{cur}, C_{i,OP_i(t)}\right)\right) \cdot \frac{1}{Pr_i}$

(*continued on next page*)



(a) TWT (b) U_{ave}

Fig. 5. Average values of two objectives obtained by the THDQN and other RL based scheduling methods on some representative instances.

(*continued*)

6: **else**
7: $J_i \leftarrow \text{argmax}_{i \in \boldsymbol{Tard}_{job}(t)} \frac{n_i}{OP_i(t)} \cdot \left(\max\left(T_{cur}, C_{i,OP_i(t)}\right) + \sum_{j=OP_i(t)+1}^{n_i} \bar{t}_{i,j} - D_i\right) \cdot Pr_i$
8: **end if**
9: $j \leftarrow OP_i(t) + 1$
10: $M_k \leftarrow \text{argmin}_{k \in M_{i,j}} \left(\max(CT_k(t), C_{i,j-1}, A_i)\right)$
11: assign $O_{i,j}$ on M_k

5.2.6. Composite dispatching rule 6

In dispatching rule 6, we select the next operation of job $J_i \in \boldsymbol{UC}_{job}(t)$ with the maximum estimated weighted tardiness and assign it on the machine with the earliest available time. The procedure of rule 6 is given in Algorithm 9.

Algorithm 9. Procedure of composite dispatching rule 6

1: $T_{cur} \leftarrow \text{mean}_k CT_k(t)$
2: $\boldsymbol{UC}_{job}(t) \leftarrow \{i | OP_i(t) < n_i\}$
3: $J_i \leftarrow \text{argmax}_{i \in \boldsymbol{UC}_{job}(t)} (\max(T_{cur}, C_{i,OP_i(t)}) + \sum_{j=OP_i(t)+1}^{n_i} \bar{t}_{i,j} - D_i < 0 \ ? \ \max(T_{cur}, C_{i,OP_i(t)}) + \sum_{j=OP_i(t)+1}^{n_i} \bar{t}_{i,j} - D_i : (\max(T_{cur}, C_{i,OP_i(t)}) + \sum_{j=OP_i(t)+1}^{n_i} \bar{t}_{i,j} - D_i) \cdot Pr_i)$
4: $j \leftarrow OP_i(t) + 1$
5: $M_k \leftarrow \text{argmin}_{k \in M_{i,j}} \left(\max(CT_k(t), C_{i,j-1}, A_i)\right)$
6: assign $O_{i,j}$ on M_k

5.3. Definition of higher goals and reward functions

Four different higher goals $g \in \{1, 2, 3, 4\}$ are used in this paper, corresponding to four different kinds of reward functions in the training process. The proposed goals respectively optimize four different indicators of tardiness or machine utilization rate including estimated total weighted tardiness $ETWT$, actual tardiness rate $Tard_a$, estimated tardiness rate $Tard_e$ and average machine utilization rate U_{ave}, the definitions of which are given in Algorithm 10, 3, 2 and Eq. (13). By adaptively selecting different higher optimization goals at different rescheduling points, a good compromise between the two objectives can be made in the long-term schedule.

Algorithm 10. Procedure to calculate the estimated total weighted tardiness $ETWT(t)$

Input: $CT_k(t), OP_i(t), A_i, D_i, Pr_i$
Output: $ETWT(t)$
1: $ETWT(t) \leftarrow 0$
2: $T_{cur} \leftarrow \text{mean}_k CT_k(t)$
3: **for** $i = 1 : n$ **do**
4: **if** $OP_i(t) < n_i$ **then**
5: **if** $\max(T_{cur}, C_{i,OP_i(t)}) + \sum_{j=OP_i(t)+1}^{n_i} \bar{t}_{i,j} - D_i > 0$ **then**

(*continued on next column*)

(*continued*)

6: $ETWT(t) \leftarrow ETWT(t) + (\max(T_{cur}, C_{i,OP_i(t)}) + \sum_{j=OP_i(t)+1}^{n_i} \bar{t}_{i,j} - D_i) \cdot Pr_i$
7: **end if**
8: **end if**
9: **end for**
10: Return $ETWT(t)$

After the higher optimization goal has been chosen, the specific form of reward function can be determined accordingly. The reward r_t at each rescheduling point t is assigned by investigating if the chosen goal at time $t+1$ after taking the dispatching rule a_t is better than its original value at time t, the definition of which is given in Algorithm 11.

Algorithm 11. Definition of the reward r_t for the state-action pair (s_t, a_t) at each rescheduling point t

Input: The current higher goal g_t, the values of goal indicators before and after taking action a_t
Output: The reward r_t for rescheduling point t
1: **if** $g_t == 1$ **then**
2: **if** $ETWT(t+1) < ETWT(t)$ **then**
3: $r_t \leftarrow 1$
4: **else if** $ETWT(t+1) > ETWT(t)$ **then**
5: $r_t \leftarrow -1$
6: **else**
7: $r_t \leftarrow 0$
8: **else if** $g_t == 2$ **then**
9: **if** $Tard_a(t+1) < Tard_a(t)$ **then**
10: $r_t \leftarrow 1$
11: **else if** $Tard_a(t+1) > Tard_a(t)$ **then**
12: $r_t \leftarrow -1$
13: **else**
14: $r_t \leftarrow 0$
15: **else if** $g_t == 3$ **then**
16: **if** $Tard_e(t+1) < Tard_e(t)$ **then**
17: $r_t \leftarrow 1$
18: **else if** $Tard_e(t+1) > Tard_e(t)$ **then**
19: $r_t \leftarrow -1$
20: **else**
21: $r_t \leftarrow 0$
22: **else if** $g_t == 4$ **then**
23: **if** $U_{ave}(t+1) > U_{ave}(t)$ **then**
24: $r_t \leftarrow 1$
25: **else if** $U_{ave}(t+1) > U_{ave}(t) \cdot 0.95$ **then**
26: $r_t \leftarrow 0$
27: **else**
28: $r_t \leftarrow -1$
29: **End if**
30: Return r_t

5.4. Network structure of the proposed DQN agents

Both of the proposed DQN agents use deep neural network (DNN) as

the Q-function approximator. The higher DQN is a DNN consisting of six fully connected layers, including one input layer with 10 nodes (corresponding to 10 state features), four hidden layers with 200 nodes each layer, and one output layer with 4 nodes (corresponding to 4 higher goals). At each rescheduling point t, it takes the state features as input and produces the Q-value of each goal, based on which the temporary optimization goal for the lower DQN can be determined. The lower DQN is also a DNN consisting of six fully connected layers, including one input layer with 11 nodes (corresponding to ten state features and one higher goal), four hidden layers with 200 nodes each layer, and one output layer with 6 nodes (corresponding to 6 candidate dispatching rules). It takes the state features together with the higher goal as input and generates the Q-value of each dispatching rule. Finally, the most feasible rule at each rescheduling point can be chosen. Both of the two DQN agents use Rectified Linear Unit (ReLU) as the activation function. Fig. 1 gives an illustration of the proposed THDQN.

5.5. Overall training framework of the THDQN

In the training process, the rescheduling point t is defined as every time a new job arrives or an operation is completed. The overall training framework of the proposed THDQN is provided in Algorithm 12, which is mainly based on the procedure of double DQN (DDQN) (Van Hasselt et al., 2016). At each rescheduling point t, the higher DQN takes the current state feature $\phi(s_t)$ as input and determines a temporary optimization goal g_t. The lower DQN takes a concatenation of $\phi(s_t)$ and g_t as input and select a dispatching rule a_t which chooses an unscheduled operation and assigns it on a feasible machine. After executing a_t, the system enters a new rescheduling point $t+1$ and the immediate reward r_t can be calculated based on the chosen goal g_t. Two experience replay buffers are used to respectively store the transitions of the higher and lower DQN agents. In order to balance the exploration and exploitation, we adopt an ε-*greedy* action selection policy in the training process, as shown in Algorithm 13.

Algorithm 12. The DDQN based training algorithm for the THDQN

Initialize the higher replay memory D_{high} with capacity N_{high} and the lower replay memory D_{low} with capacity N_{low}
Initialize higher online DQN Q_{high} with random weights θ_{high}
Initialize higher target DQN $\widehat{Q}_{high}$ with weights $\theta^{-}_{high} = \theta_{high}$
Initialize lower online DQN Q_{low} with random weights θ_{low}
Initialize lower target DQN $\widehat{Q}_{low}$ with weights $\theta^{-}_{low} = \theta_{low}$
for epoch $= 1 : L$ **do**
  Initialize a new production environment with random settings of m, DDT and λ
  Observe the initial state s_0 with state feature vector $\phi(s_0)$
  Set the initial higher feature vector $\phi_{0,high} = \phi(s_0)$
  Select higher goal $g_0 = \varepsilon$-*greedy*$(Q_{high}, \phi_{0,high})$
  for $t = 0 : T$ (t is the rescheduling 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**
    Observe the current state s_t with state feature vector $\phi(s_t)$
    Set $\phi_{t,low} = [\phi(s_t), g_t]$
    Select dispatching rule $a_t = \varepsilon$-*greedy*$(Q_{low}, \phi_{t,low})$
    Execute rule a_t, observe the new state s_{t+1} with state feature vector $\phi(s_{t+1})$, obtain the immediate reward r_t by Algorithm 11
    Set $\phi_{t+1,high} = \phi(s_{t+1})$
    Select higher goal $g_{t+1} = \varepsilon$-*greedy*$(Q_{high}, \phi_{t+1,high})$
    Set $\phi_{t+1,low} = [\phi(s_{t+1}), g_{t+1}]$
    Store transition $(\phi_{t,high}, g_t, r_t, \phi_{t+1,high})$ in D_{high}
    Store transition $(\phi_{t,low}, a_t, r_t, \phi_{t+1,low})$ in D_{low}
    Sample random minibatch of transitions $(\phi_{j,high}, g_j, r_j, \phi_{j+1,high})$ from D_{high}
    Set $y_{high} = r_j + \gamma \widehat{Q}_{high}(\phi_{j+1,high}, \mathrm{argmax}_{g'} Q_{high}(\phi_{j+1,high}, g'; \theta_{high}); \theta^{-}_{high})$
    Perform a gradient descent step on $(y_j - Q_{high}(\phi_{j,high}, g_j; \theta_{high}))^2$ with respect to the parameters θ_{high} of higher online network Q_{high}
    Sample random minibatch of transitions $(\phi_{j,low}, a_j, r_j, \phi_{j+1,low})$ from D_{low}
    Set $y_j = r_j + \gamma \widehat{Q}_{low}(\phi_{j+1,low}, \mathrm{argmax}_{a'} Q_{low}(\phi_{j+1,low}, a'; \theta_{low}); \theta^{-}_{low})$

(*continued on next column*)

(*continued*)

    Perform a gradient descent step on $(y_j - Q_{low}(\phi_{j,low}, a_j; \theta_{low}))^2$ with respect to the parameters θ_{low} of lower online network Q_{low}
    Every C steps reset $\widehat{Q}_{high} = Q_{high}, \widehat{Q}_{low} = Q_{low}$
  end for
end for

Algorithm 13. The procedure of the ε-*greedy* action selection policy

Input: The feature vector $\phi(s_t)$ at time point t, the greedy parameter ε, the candidate action set $\boldsymbol{A}$, the DQN agent Q
Output: The selected action a_t
Generate the Q-value $Q(\phi(s_t), a)$ of each candidate action a
With probability ε randomly select an action $a_t \in \boldsymbol{A}$
otherwise select the action $a_t = \mathrm{argmax}_{a \in A} Q(\phi(s_t), a)$
Return a_t

5.6. Implementation framework of the trained THDQN

In the training process, the action is randomly chosen with probability ε so as to enhance exploration. However, when applied to real-world production process, the action with the highest Q value should be selected at each rescheduling point to obtain a near optimal schedule. The implementation framework of the trained THDQN in actual production process is provided in Algorithm 14.

Algorithm 14. The implementation framework of the trained THDQN

for $t = 0 : T$ (t is the rescheduling 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**
  Observe the current state s_t with feature vector $\phi(s_t)$
  Set $\phi_{t,high} = \phi(s_t)$
  Generate the current goal $g_t = \mathrm{argmax}_g Q_{high}(\phi_{t,high}, g)$
  Set $\phi_{t,low} = [\phi(s_t), g_t]$
  Execute the action $a_t = \mathrm{argmax}_a Q_{low}(\phi_{t,low}, a)$
end for

6. Numerical experiments

In this section, we successively provide the parameter settings in training and testing process, the performance metrics in sense of multi-objective optimization, and the results of performance comparisons between the proposed THDQN and each composite dispatching rule as well as existing well-known dispatching rules. To further validate the superiority of the proposed two-hierarchy reinforcement learning framework, we also compare the THDQN with other three RL based training algorithms with the same composite rules used in this paper as actions. The results of numerical experiments have confirmed both the effectiveness and generality of the proposed THDQN.

6.1. Parameter settings

The training and testing 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, that is, the interarrival time between two successive new job insertions is subjected to exponential distribution $exp(1/\lambda)$. The proposed THDQN is implemented on a PC with Intel Core i7-6700 @ 4.0 GHz CPU and 8 GB RAM. Each instance used in this study is randomly generated according to the parameters listed in Table 2, where the symbols "randf" and "randi" denote uniform distribution of real numbers and integers, respectively. The parameters of training algorithm are elaborately set by preliminary experiments as shown in

Table 3.

In the training process, a new instance is randomly generated at each epoch to improve the generality of THDQN in different production environments. At each training epoch, the ε-*greedy* parameter ε is linearly decreased from 0.9 to 0.1 per training step so that each rule can be adequately investigated during the training process. Therefore, the algorithm can avoid from falling into local optimum and the diversity of Pareto-optimal schedules is enhanced. Meanwhile, note that in the classical DDQN, the DQN agent is trained in an off-policy mode where a large experience replay memory is used to enhance data efficiency. However, the changing behavior of the lower-level policy creates a non-stationary problem for the higher-level policy. That is, when the lower-level policy is updated, the same higher goals may cause the lower agent to exhibit different transitions (Nachum, Gu, Lee, & Levine, 2018b), thus the original experiences far from now may be infeasible for training the current higher agent. To address this issue, we set the higher replay memory size N_{high} the same as the training minibatch size. This enables the higher DQN to be trained approximately in an on-line mode.

6.2. Performance metrics

For the bi-objective optimization problem considered in this paper, the aim is to find a set of Pareto-optimal solutions uniformly distributed in the Pareto-optimal front. In order to comprehensively measure the quality of the obtained Pareto-optimal solutions, we utilize three different performance metrics including generational distance (GD) (Zitzler, Deb, & Thiele, 2000), inverse generational distance (IGD) (Zitzler & Thiele, 1999) and spread (Δ) (Deb, Pratap, Agarwal, & Meyarivan, 2002). The GD and Δ metrics evaluate the convergence and diversity of the obtained Pareto-optimal solutions, respectively. While the IGD metric is more comprehensive reflecting both the convergence and diversity of the obtained solutions. The definitions of these three metrics are listed as follows.

(1) Convergence metric: Generational Distance (GD)

$$GD\left(\boldsymbol{A},\boldsymbol{P}\right)=\frac{\sqrt{\sum_{i=1}^{|\boldsymbol{A}|}d_{i,A,P}^{2}}}{|\boldsymbol{A}|} \tag{18}$$

where $\boldsymbol{P}$ is the true Pareto-optimal front of an optimization problem and $\boldsymbol{A}$ is the approximate Pareto-optimal front obtained by the algorithm to be evaluated. $d_{i,A,P}$ is the Euclidean distance between the ith solution in $\boldsymbol{A}$ and its closest neighbor in $\boldsymbol{P}$.

(2) Comprehensive metric: Inverse Generational Distance (IGD)

$$IGD\left(\boldsymbol{P},\boldsymbol{A}\right)=\frac{1}{|\boldsymbol{P}|}\sum_{i=1}^{|\boldsymbol{P}|}d_{i,P,A} \tag{19}$$

where $d_{i,P,A}$ is the Euclidean distance between the ith solution in $\boldsymbol{P}$ and its closest neighbor in $\boldsymbol{A}$.

(3) Diversity metric: Spread (Δ)

$$\Delta=\frac{\sum_{j=1}^{n_o}d_{j,A,P}^{e}+\sum_{i=1}^{|\boldsymbol{A}|}\left|d_{i,A,A}-\overline{d}_{A,A}\right|}{\sum_{j=1}^{n_o}d_{j,A,P}^{e}+\left|\boldsymbol{A}\right|\cdot\overline{d}_{A,A}} \tag{20}$$

where $d_{i,A,A}$ is the Euclidean distance of the ith solution in $\boldsymbol{A}$ to its closest neighbor in $\boldsymbol{A}$, and $\overline{d}_{A,A}$ is the average value of all $d_{i,A,A}$. $d_{j,A,P}^{e}$ is the Euclidean distance between the extreme solution of $\boldsymbol{A}$ and the boundary solution of $\boldsymbol{P}$ with regard to the jth objective. n_o is the number of objectives.

In general, a set of solutions with smaller values in GD, IGD and Δ are preferred. It should be noted that the true Pareto-optimal front $\boldsymbol{P}$ for the test benchmarks can not be known in advance, for the ease of implementation, we merge the solutions obtained by all comparative algorithms and take the nondominated ones among which as $\boldsymbol{P}$.

6.3. Comparisons with the proposed composite dispatching rules

To validate the effectiveness and generality of the proposed THDQN, we compare the performance of THDQN with each composite rule on a wide range of instances with different parameter settings of DDT, m and λ, which can be regarded as abstractions of different real-world production environments. The total number of new inserted jobs is set as 200. Meanwhile, to further confirm if the THDQN has learned a feasible behavioral strategy at different rescheduling points, a random strategy, i.e., randomly select a higher goal and lower dispatching rule rule at each rescheduling point, is also taken as comparison. Each method is repeated independently for 20 times on each instance. The GD, IGD and Δ values of the Pareto-optimal front obtained by each method from the 20 runs on different instances are provided in Table 4–6 with the best results highlighted in bold font.

It can be seen from the results that the THDQN outperforms other comparative methods for most production environments. First, compared with the random strategy, the THDQN achieves better performance for almost all instances, indicating that the THDQN has learned an efficient policy to choose the most feasible rule at each decision point. Next, compared with each dispatching rule, the THDQN can also obtain the best results for most instances. This reflects that there does not exist a single rule to perform well for all production environments and further confirms the effectiveness and generality of the proposed THDQN. In conclusion, the higher DQN can determine a feasible goal by comprehensively investigating the current production status at each rescheduling point, and based on which the lower DQN can choose the optimal rule to achieve this goal. Since the goals are designed to optimize the tardiness and machine utilization rate, a good compromise between the two given objectives can be made during the long-term schedule, due to which the proposed THDQN is more effective and generic than each single dispatching rule. Fig. 2 illustrates the Pareto-optimal fronts obtained by the THDQN and each composite dispatching rule for some representative instances.

6.4. Comparisons with other well-known dispatching rules

To further confirm the advantage of the proposed THDQN over other widely-used dispatching rules, we take six well-known dispatching rules from previous work into comparisons, the details of which are listed as follows.

(1) First in first out (FIFO): The highest priority is given to the job J_i with the earliest arrival time A_i.
(2) Earliest due date (EDD): The highest priority is given to the job J_i with the earliest due date D_i.
(3) Most remaining processing time (MRT): The highest priority is given to the job J_i with the most remaining processing time $\sum_{j=OP_i(t)+1}^{n_i}\overline{t}_{ij}$.
(4) Shortest processing time (SPT): The highest priority is given to the job J_i with the shortest average processing time $\overline{t}_{i,OP_i(t)+1}$ of its next operation $O_{i,OP_i(t)+1}$.
(5) Longest processing time (LPT): The highest priority is given to the job J_i with the longest average processing time $\overline{t}_{i,OP_i(t)+1}$ of its next operation $O_{i,OP_i(t)+1}$.
(6) Critical ratio (CR): The highest priority is given to the job J_i with the minimum critical ratio defined as $\frac{D_i-t}{\sum_{j=OP_i(t)+1}^{n_i}\overline{t}_{ij}}$.

It should be noted that these rules do not explicitly determine the processing machines for the chosen jobs. In order to adjust them to be

feasible for flexible job shop scheduling problems, for each comparative dispatching rule, we assign the next operation of the selected job to the machine with the earliest available time among its compatible machine set. Each method is conducted for 20 runs on each instance. The performances comparisons on different instances are provided in Table 7–9 with the best results highlighted in bold font.

As shown in the comparison results, the THDQN demonstrates the best performance for all metrics on most instances. This has verified the superiority of THDQN over existing well-known dispatching rules and further validated the effectiveness of the proposed composite dispatching rules in reducing total weighted tardiness and enhancing machine utilization rate. Fig. 3 illustrates the Pareto-optimal fronts obtained by the THDQN and other well-known dispatching rules for some representative instances.

6.5. Comparisons with existing composite dispatching rules

In order to verify the superiority of the proposed THDQN over existing composite dispatching rules, we take four recently proposed composite rules for the dynamic flexible job shop scheduling problem as comparisons, including the weight biased modified RRrule (WBMR) (Chen & Matis, 2013), the best rule found by gene expression programming (denoted by RGEP) (Ozturk, Bahadir, & Teymourifar, 2019), the heterogeneous earliest finish time (HEFT) algorithm (Cao, Zhou, Hu, & Lin, 2019), and the greedy randomized adaptive search procedure (GRASP) (Baykasoğlu, Madenoğlu, & Hamzadayı, 2020). Each method is repeated for 20 times on each instance, the experimental results are given in Table 10–12 with the best results highlighted in bold font.

It can be observed that compared with the existing efficient composite rules, the THDQN can achieve the best results for most instances. Note that the GRASP and WBMR can also obtain the best GD metric for some instances, this may due to that they can obtain the lowest *TWT* or highest U_{ave} on these instances thus demonstrate the best convergence in the corresponding objective. For example, the WBMR rule can attain better U_{ave} especially for instances where $\lambda = 200$. However, the THDQN outperforms all the other methods in terms of IGD and Δ metrics, which indicates that the THDQN achieves the best compromise among two investigated objectives. Fig. 4 illustrates the average values of two objectives obtained by the THDQN and other composite rules on some representative instances. It can be seen that the THDQN can attain significantly lower *TWT* and relatively higher U_{ave} compared with other methods.

6.6. Comparisons with other RL based scheduling methods

Moreover, to verify the superiority of the proposed two-hierarchy deep reinforcement learning structure in multi-objective optimization, we have also compared the THDQN to other three RL based training algorithms including classical Q-learning, single-hierarchy DQN (SHDQN) where only a single DQN agent is applied to select the dispatching rules without a higher controller adjusting the optimization goals, and the double DQN (DDQN) based training approach proposed by Luo (2020). Since the other comparative algorithms only use a single RL agent which is merely responsible for rule selection, the reward function of the other three methods is set as the weighted sum of total weighted tardiness and average machine utilization rate with the equal weight of 0.5 for each objective. To make fair comparison, each comparative method uses the same set of composite dispatching rules (i.e., the set of available actions) as the THDQN. The comparison results on different production environments are given in Table 13–15.

It can be observed from the comparison results that the THDQN is superior to other RL based training methods for most instances. This confirms the necessity and effectiveness to use the proposed two-hierarchy structure where a higher DQN adaptively chooses the most feasible optimization goals at different rescheduling points so that a good compromise between the total weighted tardiness and average machine utilization rate can be achieved. Meanwhile, note that the classical Q-learning algorithm is outperformed by all the other DRL based approaches on the comprehensive metric IGD, which indicates that the compulsive discretization of continuous state space in classical RL methods is adverse to the model accuracy and further verifies the superiority of DRL based methods. Fig. 5 illustrates the average values of two objectives obtained by the THDQN and other RL based methods on some representative instances. It can be seen that the THDQN can attain the best performance on the two investigated objectives.

7. Conclusion

In this paper, a two-hierarchy deep Q network (THDQN) is developed for the dynamic multi-objective flexible job shop scheduling problem (DMOFJSP) with new job insertions aiming at optimizing the total weighted tardiness and average machine utilization rate. The proposed THDQN contains two DQN based agents including a higher controller and a lower actuator. At each rescheduling point, the higher DQN determines the temporary optimization goal for the lower DQN based on the current state features. While the lower DQN takes both the higher optimization goal and current state features as input and chooses a feasible dispatching rule to achieve the given goal. Four different goals corresponding to four different forms of reward functions in the training process are designed, each of which optimizes an indicator of tardiness or machine utilization rate. Six composite dispatching rules are developed to simultaneously select an unprocessed operation and assign it on an available machine. A double DQN based training framework is proposed to train the two DQN agents. By adaptively choosing the feasible goals and appropriate dispatching rules at different rescheduling points, not only the scheduling can be conducted in real time but also a good compromise among different objectives can be made in the long-term run.

Numerical experiments are conducted on a large set of instances which can be regarded as high abstractions of actual manufacturing process to verify the effectiveness and superiority of the proposed THDQN in practical applications. The results demonstrate that the THDQN performs significantly better than each composite dispatching rule, existing well-known dispatching rules as well as other RL based scheduling methods, which can also be well generalized to different production environments.

For future work, more uncertain events such as machine breakdowns and varying processing times will be investigated. Other objectives like mean flow time, energy consumption and production costs are also worthy to be considered to validate the generality of the THDQN over different objectives. Meanwhile, it should be noted that many features are considered as input for the THDQN, the coupling of which may cause the networks to be misleading. Thus introducing a feature selection algorithm can be useful to improve the performance of the deep Q networks. In addition, since the proposed THDQN is essentially a value based method which can not directly optimize over the policy, we will apply other state-of-art policy based methods such as actor-critic algorithm and proximal policy optimization for solving the DMOFJSP.

CRediT authorship contribution statement

Shu Luo: Conceptualization, Methodology, Software, Data curation, Writing - original draft, Visualization, Validation. **Linxuan Zhang:** Writing - review & editing, Project administration, Funding acquisition. **Yushun Fan:** Resources, Supervision.

Acknowledgements

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 in part by the National Key Research and Development Program of China under Grant 2018YFB1703103, in part by the

Research and Development Program of Tsinghua Univ.-Weichai Power Co. Ltd. Intelligent Manufacturing Institute under Grant JIM02/20182912121, in part by the Dongguan Innovative Research Team Program under Grant 2018607202007.

References

Altenmüller, T., Stüker, T., Waschneck, B., Kuhnle, A., & Lanza, G. (2020). Reinforcement learning for an intelligent and autonomous production control of complex job-shops under time constraints. *Production Engineering, 14*, 319–328.

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

Baykasoğlu, A., Madenoğlu, F. S., & Hamzadayı, A. (2020). Greedy randomized adaptive search for dynamic flexible job-shop scheduling. *Journal of Manufacturing Systems, 56*, 425–451.

Bellman, R. (1957). A markovian decision process. *Journal of Mathematics and Mechanics*, 679–684.

Bouazza, W., Sallez, Y., & Beldjilali, B. (2017). A distributed approach solving partially flexible job-shop scheduling problem with a q-learning effect. *IFAC-PapersOnLine, 50*(1), 15890–15895.

Cao, Z., Zhou, L., Hu, B., & Lin, C. (2019). An adaptive scheduling algorithm for dynamic jobs for dealing with the flexible job shop scheduling problem. *Business & Information Systems Engineering, 61*(3), 299–309.

Chen, B., & Matis, T. I. (2013). A flexible dispatching rule for minimizing tardiness in job shop scheduling. *International Journal of Production Economics, 141*(1), 360–365.

Chen, R., Yang, B., Li, S., & Wang, S. (2020). A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem. *Computers & Industrial Engineering, 149*, 106778.

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

Cunha, B., Madureira, A. M., Fonseca, B., & Coelho, D. (2018). Deep reinforcement learning as a job shop scheduling solver: A literature review. In *International Conference on Hybrid Intelligent Systems. Springer* (pp. 350–359).

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation, 6*(2), 182–197.

Gabel, T., & Riedmiller, M. (2012). Distributed policy search reinforcement learning for job-shop scheduling tasks. *International Journal of production research, 50*(1), 41–61.

Garey, M. R., Johnson, D. S., & Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of operations research, 1*(2), 117–129.

Hasselt, H.V. (2010). Double q-learning. In: Advances in Neural Information Processing Systems. pp. 2613–2621.

Holthaus, O., & Rajendran, C. (2000). Efficient jobshop dispatching rules: Further developments. *Production Planning & Control, 11*(2), 171–178.

Howard, R. A. (1960). Dynamic programming and markov processes.

Hu, L., Liu, Z., Hu, W., Wang, Y., Tan, J., & Wu, F. (2020). Petri-net-based dynamic scheduling of flexible manufacturing system via deep reinforcement learning with graph convolutional network. *Journal of Manufacturing Systems, 55*, 1–14.

Kuhnle, A., Schäfer, L., Stricker, N., & Lanza, G. (2019). Design, implementation and evaluation of reinforcement learning for an adaptive order dispatching in job shop manufacturing systems. *Procedia CIRP, 81*, 234–239.

Kundakcı, N., & Kulak, O. (2016). Hybrid genetic algorithms for minimizing makespan in dynamic job shop scheduling problem. *Computers & Industrial Engineering, 96*, 31–51.

Li, A. C., Florensa, C., Clavera, I., & Abbeel, P. (2019). Sub-policy adaptation for hierarchical reinforcement learning. arXiv preprint arXiv:1906.05862.

Li, X., Wang, J., & Sawhney, R. (2012). Reinforcement learning for joint pricing, lead-time and scheduling decisions in make-to-order systems. *European Journal of Operational Research, 221*(1), 99–109.

Li, Y. (2017). Deep reinforcement learning: An overview. arXiv preprint arXiv: 1701.07274.

Liu, C.-L., Chang, C.-C., & Tseng, C.-J. (2020). Actor-critic deep reinforcement learning for solving job shop scheduling problems. *IEEE Access, 8*, 71752–71762.

Lu, C., Li, X., Gao, L., Liao, W., & Yi, J. (2017). An effective multi-objective discrete virus optimization algorithm for flexible job-shop scheduling problem with controllable processing times. *Computers & Industrial Engineering, 104*, 156–174.

Luo, S. (2020). Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning. *Applied Soft Computing, 91*, 106208.

Méndez-Hernández, B. M., Rodríguez-Bazan, E. D., Martinez-Jimenez, Y., Libin, P., & Nowé, A. (2019). A multi-objective reinforcement learning algorithm for jssp. In *International Conference on Artificial Neural Networks* (pp. 567–584).

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature, 518*(7540), 529.

Mohan, J., Lanka, K., & Rao, A.N., 2019. A review of dynamic job shop scheduling techniques. *Procedia Manufacturing* 30, 34–39, digital Manufacturing Transforming Industry Towards Sustainable Growth.

Nachum, O., Gu, S. S., Lee, H., & Levine, S. (2018a). Data-efficient hierarchical reinforcement learning. In: Advances in Neural Information Processing Systems. pp. 3303–3313.

Nachum, O., Gu, S. S., Lee, H., & Levine, S. (2018b). Data-efficient hierarchical reinforcement learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 31* (pp. 3303–3313). Curran Associates, Inc.

Nie, L., Gao, L., Li, P., & Li, X. (2013). A gep-based reactive scheduling policies constructing approach for dynamic flexible job shop scheduling problem with job release dates. *Journal of Intelligent Manufacturing, 24*(4), 763–774.

Ozturk, G., Bahadir, O., & Teymourifar, A. (2019). Extracting priority rules for dynamic multi-objective flexible job shop scheduling problems using gene expression programming. *International Journal of Production Research, 57*(10), 3121–3137.

Rafati, J., & Noelle, D. C. (2019). Learning representations in model-free hierarchical reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence., 33*, 10009–10010.

Rajendran, C., & Holthaus, O. (1999). A comparative study of dispatching rules in dynamic flowshops and jobshops. *European Journal of Operational Research, 116*(1), 156–170.

Riedmiller, S., & Riedmiller, M. (1999). A neural reinforcement learning approach to learn local dispatching policies in production scheduling. In: IJCAI. Vol. 2. Citeseer, pp. 764–771.

Shahrabi, J., Adibi, M. A., & Mahootchi, M. (2017). A reinforcement learning approach to parameter estimation in dynamic job shop scheduling. *Computers & Industrial Engineering, 110*, 75–82.

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

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT Press.

Tang, D., Dai, M., Salido, M. A., & Giret, A. (2016). Energy-efficient dynamic scheduling for a flexible flow shop using an improved particle swarm optimization. *Computers in Industry, 81*, 82–95.

Van Hasselt, H., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double q-learning. In: AAAI. Vol. 2. Phoenix, AZ, p. 5.

Wang, Y.-C., & Usher, J. M. (2004). Learning policies for single machine job dispatching. *Robotics and Computer-Integrated Manufacturing, 20*(6), 553–562.

Wang, Y.-F. (2018). Adaptive job shop scheduling strategy based on weighted q-learning algorithm. *Journal of Intelligent Manufacturing*, 1–16.

Wang, Y.-F. (2020). Adaptive job shop scheduling strategy based on weighted q-learning algorithm. *Journal of Intelligent Manufacturing, 31*(2), 417–432.

Waschneck, B., Reichstaller, A., Belzner, L., Altenmüller, T., Bauernhansl, T., Knapp, A., & Kyek, A. (2018). Optimization of global production scheduling with deep reinforcement learning. *Procedia CIRP, 72*(1), 1264–1269.

Wei, Y., & Mingyang, Z. (2005). A reinforcement learning-based approach to dynamic job-shop scheduling. *Acta Automatica Sinica, 31*(5), 765–771.

Yingzi, W, & Mingyang Z. (2004). Composite rules selection using reinforcement learning for dynamic job-shop scheduling. In: IEEE Conference on Robotics, Automation and Mechatronics, 2004. Vol. 2. pp. 1083–1088.

Zhang, S., Li, X., Zhang, B., & Wang, S. (2020). Multi-objective optimisation in flexible assembly job shop scheduling using a distributed ant colony system. *European Journal of Operational Research, 283*(2), 441–460.

Zhang, W., & Dietterich, T.G. (1995). A reinforcement learning approach to job-shop scheduling. In: IJCAI. Vol. 95. Citeseer, pp. 1114–1120.

Zhu, H., Li, M., Tang, Y., & Sun, Y. (2020). A deep-reinforcement-learning-based optimization approach for real-time scheduling in cloud manufacturing. *IEEE Access, 8*, 9987–9997.

Zitzler, E., Deb, K., & Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation, 8*(2), 173–195.

Zitzler, E., & Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation, 3*(4), 257–271.