Real-Time Scheduling for Dynamic Partial-No-Wait Multiobjective Flexible Job Shop by Deep Reinforcement Learning

Shu Luo, Linxuan Zhang, and Yushun Fan

***Abstract*— In modern discrete flexible manufacturing systems, dynamic disturbances frequently occur in real time and each job may contain several special operations in partial-no-wait constraint due to technological requirements. In this regard, a hierarchical multiagent deep reinforcement learning (DRL)-based real-time scheduling method named hierarchical multi-agent proximal policy optimization (HMAPPO) is developed to address the dynamic partial-no-wait multiobjective flexible job shop scheduling problem (DMOFJSP-PNW) with new job insertions and machine breakdowns. The proposed HMAPPO contains three proximal policy optimization (PPO)-based agents operating in different spatiotemporal scales, namely, objective agent, job agent, and machine agent. The objective agent acts as a higher controller periodically determining the temporary objectives to be optimized. The job agent and machine agent are lower actuators, respectively, choosing a job selection rule and machine assignment rule to achieve the temporary objective at each rescheduling point. Five job selection rules and six machine assignment rules are designed to select an uncompleted job and assign the next operation of which together with its successors in no-wait constraint on the corresponding processing machines. A hierarchical PPO-based training algorithm is developed. Extensive numerical experiments have confirmed the effectiveness and superiority of the proposed HMAPPO compared with other well-known dynamic scheduling methods.**

***Note to Practitioners*—The motivation of this article stems from the need to develop real-time scheduling methods for modern discrete flexible manufacturing factories, such as aerospace product manufacturing and steel manufacturing, where dynamic events frequently occur, and each job may contain several operations subjected to the no-wait constraint. Traditional dynamic scheduling methods, such as metaheuristics or dispatching rules, either suffer from poor time efficiency or fail to ensure good solution quality for multiple objectives in the long-term run. Meanwhile, few of the previous studies have considered the partial-no-wait constraint among several operations from the same job, which widely exists in many industries. In this article, we propose a hierarchical multiagent deep reinforcement learning (DRL)-based real-time scheduling method named HMAPPO to address the dynamic partial-no-wait multiobjective flexible job shop scheduling problem (DMOFJSP-PNW) with new job insertions and machine breakdowns. The proposed HMAPPO uses three DRL-based agents to adaptively select the temporary objectives and choose the most feasible dispatching rules to achieve them at different rescheduling points, through which the rescheduling can be made in real time and a good compromise among different objectives can be obtained in the long-term schedule. Extensive experimental results have demonstrated the effectiveness and superiority of the proposed HMAPPO. For industrial applications, this method can be extended to many other production scheduling problems, such as hybrid flow shops and open shop with different uncertainties and objectives.**

***Index Terms*— Deep reinforcement learning (DRL), flexible job shop, multiagent, multiobjective, partial-no-wait.**

I. Introduction

THE flexible job shop scheduling problem (FJSP), playing an important role in the production management of modern discrete flexible manufacturing systems, such as steel manufacturing and semiconductor manufacturing, is of great research significance for both industry and academia, which has been intensively studied over the past decades. It has been proven to be NP-hard [1] and is more intractable than classical job shop scheduling problem (JSP) since each operation can be assigned on one or more available machines.

In most actual scenarios of FJSP, due to technological requirements, each job may contain several interdependent operations subjected to the no-wait constraint, which should be continuously processed without interruption [2], [3]. For example, in aerospace product manufacturing, the inspection step must be immediately conducted after the heat-treatment or aging treatment to ensure the quality of products. In steel manufacturing, the hot work-in-process must continuously go through the subsequent production steps to avoid unwanted cooling. Based on the spatial dependence of constrained operations, the no-wait constraints can be further divided into homogeneous constraints among operations on the same machine and heterogeneous constraints among operations on different machines. In the classical complete-no-wait JSP (NWJSP), all the operations from the same job must be continuously processed one after another in a strict no-wait mode. Therefore, the original NWJSP can be decomposed into two subproblems, namely, the sequencing and the timetabling subproblem, which corresponds to finding an optimal processing

Manuscript received 6 April 2021; revised 2 July 2021; accepted 6 August 2021. Date of publication 24 August 2021; date of current version 13 October 2022. This article was recommended for publication by Associate Editor F. Chu and Editor J. Li upon evaluation of the reviewers' comments. This work was 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 University–Weichai Power Company Ltd., Intelligent Manufacturing Institute, under Grant JIM02/20182912121, and in part by the Dongguan Innovative Research Team Program under Grant 2018607202007. *(Corresponding author: Linxuan Zhang.)*

The authors are with the Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: luos17@mails.tsinghua.edu.cn; lxzhang@mail.tsinghua.edu.cn; fanyus@tsinghua.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TASE.2021.3104716.

Digital Object Identifier 10.1109/TASE.2021.3104716

sequence of jobs and the optimal start times of the jobs under the given sequence, respectively. Due to the strong no-wait constraint, the solution space is greatly reduced, and numerous efficient methods have been proposed to address this problem [4]. However, few of the previous work has studied the partial-no-wait FJSP (FJSP-PNW) where only several special operations from the same job are subjected to the no-wait constraint due to its higher complexity and variety.

Meanwhile, in the rapidly changing and highly customized manufacturing environments of modern Industry 4.0 smart factories, multiple uncertainties, such as new job insertions, machine breakdowns, and material unavailability, are inevitable to be considered [5], leading to an urgent need for dynamic scheduling methods with the ability to deal with uncertain disturbances in real time. To summarize, the existing methods for dynamic job shop scheduling, such as metaheuristics and dispatching rules, either suffer from poor time efficiency or fail to guarantee good solution quality for multiple objectives in the long-term run. In recent years, with the great breakthroughs of deep reinforcement learning (DRL) [6]–[8], DRL-based methods have been widely applied to different dynamic JSPs [9]. Without loss of generality, the scheduling process of dynamic FJSP (DFJSP) can be modeled as a Markov sequential decision process, which is quite suitable to be solved by DRL. By directly taking the production state features as input and adaptively determining the most feasible actions, i.e., the dispatching rules at different rescheduling points through a DRL agent, not only the rescheduling can be made in real time but also a good overall performance can be achieved in the long-term schedule. Despite a large number of successful applications have been reported, there remain two major challenges for current DRL-based dynamic scheduling methods. First, most of them use deep Q network (DQN) [10]–[13] to approximate the action-value functions, which cannot directly optimize over the policy. Second, the existing single-hierarchy DRL methods could not well address multiobjective optimization problems since different objectives lead to different behavioral policies that are hard to balance for a single agent. Recently, hierarchical reinforcement learning (HRL) has set a good example in dealing with complex tasks with multiple objectives [14], [15]. HRL learns to operate over different levels of spatiotemporal 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 intelligently choosing the appropriate objectives to be optimized during different time periods, a good compromise over all objectives can be made.

With the motivations above, in this article, a hierarchical multiagent proximal policy optimization (PPO)-based real-time scheduling method named hierarchical multi-agent proximal policy optimization (HMAPPO) is developed to address the dynamic partial-no-wait multiobjective FJSP (DMOFJSP-PNW) with new job insertions and machine breakdowns. The proposed HMAPPO conducts rescheduling at each time when a new job arrives, or a machine breakdown occurs, or a machine is freed. Three practical objectives, including total weighted tardiness (TWT), average machine utilization rate, and variance of machine workload, are considered simultaneously, and our aim is to find the Pareto-optimal schedules that can well compromise different objectives. The main contributions of this article are listed as follows.

1) Three PPO-based agents operating in different spatiotemporal levels are proposed, including an objective agent, a job agent, and a machine agent. The objective agent serves as a higher controller determining the temporary optimization objective every C rescheduling points. The job agent and machine agent are lower actuators respectively choosing a job selection rule and machine assignment rule to achieve the higher objective at each rescheduling point.
2) Five job selection rules and six machine assignment rules are designed to select an uncompleted job and assign the next operation of which together with its successors in no-wait constraint on the corresponding processing machines.
3) A hierarchical PPO-based training algorithm is designed to train the three intelligent agents. Particularly, 20 well-designed state features are extracted to comprehensively reflect the production status, and a novel reward function is developed to calculate the rewards corresponding to different higher objectives.
4) Extensive numerical experiments on different production environments have confirmed both the effectiveness and superiority of the proposed HMAPPO.

The remainder of this article is organized as follows. Section II introduces the related works on dynamic job shop scheduling methods. Section III presents the background of reinforcement learning (RL) and PPO. The mathematical model of the DMOFJSP-PNW addressed in this article is established in Section IV. The implementation details of the proposed HMAPPO are successively given in Section V. Section VI provides the results of numerical experiments. Finally, conclusions are drawn in Section VII.

II. Related Works

The dynamic JSP has been widely investigated over the last decades. In this article, we divide the existing methods for solving this problem into three main categories, including rule-based methods, metaheuristics-based methods, and RL-based methods.

A. Rule-Based Methods

Dispatching rules, such as first in first out (FIFO), most remaining time, and earliest due date (EDD), simply choose an unprocessed operation and assign it on a processing machine at each rescheduling point, thus achieving high time efficiency [16]. Various dispatching rules have been proposed in the last decades, which can be mainly classified into basic priority rules, composite rules, and heuristic rules [17]–[20]. Recently, hyperheuristics, such as genetic programming (GP) and gene expression programming (GEP), have arisen as a popular method for discovering effective dispatching rules for JSPs. For example, Zhang *et al.* [21] proposed a GP hyperheuristics (GPHH) to evolve scheduling rules for a DFJSP with new job arrivals. Nie *et al.* [22] developed an approach based on GEP to simultaneously construct reactive scheduling rules

for the routing problem and sequencing problem in a DFJSP with new job insertions. Jun *et al.* [23] suggested a random-forest-based approach called random forest for obtaining rules for scheduling (RANFORS) to extract dispatching rules in a DFJSP with release times. Despite the ease of implementation, most dispatching rules are myopic which fail to guarantee even a local optimum in the long-term run [20]. Moreover, it is difficult to make a compromise among different objectives by a single rule since different rules are suitable for different objectives and production environments [16], [20].

B. Metaheuristics-Based Methods

Metaheuristics always decompose the original dynamic scheduling problem into a series of static subproblems and solve them separately [24]. Typically, an initial schedule is created at first and executed until a rescheduling point occurs. At each rescheduling point (i.e., the occurrence time of a new job insertion or a machine breakdown), the global information of unscheduled operations and available machines is taken into consideration and through which a new schedule is generated and executed until the next rescheduling point. Zhang *et al.* [25] developed a hybrid genetic algorithm (hGA) to address the dynamic job shop with random job arrivals and machine breakdowns considering both the schedule efficiency and stability as objectives. Gao *et al.* [26] proposed an two-stage artificial bee colony (TABC) algorithm with several improvements for the DFJSP with fuzzy processing times and new job insertions. The first stage solves the problem using the artificial bee colony (ABC) algorithm to obtain high-quality solutions, and the second stage reschedules new jobs and the existing jobs' operations that have not started. Nouiri *et al.* [27] suggested a particle swarm optimization (PSO) for the DFJSP under machine breakdowns to optimize the makespan, robustness, and stability of the obtained schedules. Zhang and Wong [28] developed a hybrid multiagent system negotiation and ant colony optimization (ACO) approach for the DFJSP with different types of disruptions, including rush orders, job cancelation, and machine breakdown/repair. Gao *et al.* [29] proposed a discrete Jaya algorithm enhanced with five objective-oriented local search operators for a DFJSP with new job insertions. Shahgholi Zadeh *et al.* [30] developed an ABC algorithm for a DFJSP with processing time deviations. An initial scheduling is created according to the estimated processing times, and then, rescheduling is performed after determining the machine setup time for each operation. Baykasoğlu *et al.* [31] suggested a greedy randomized adaptive search procedure (GRASP) for the DFJSP considering sequence-dependent setup times and dynamic events, such as new order arrivals, changes in due dates, and machine breakdowns. Other metaheuristics, such as teaching–learning-based optimization algorithm (TLBO), estimation of distribution algorithm (EDA), imperialist competitive algorithm (ICA), artificial immune system algorithm (AIS), and iterated greedy algorithm (IGA), can also be found in [32]–[36]. Metaheuristics can obtain near-optimal solutions since the global information is utilized to generate a new schedule at each rescheduling point but suffer from poor time efficiency due to their complicated evolutionary process and searching modes in the huge search space [24]. This limits their practical application in rapidly varying production environments where a new disturbance may suddenly occur even before the new rescheduling scheme has been made.

C. RL-Based Methods

RL-based methods have been intensively investigated by researchers for solving dynamic JSPs. Compared to the dispatching rules and metaheuristics, by adaptively determining the most feasible actions (dispatching rules) at different rescheduling points through an RL agent, both the real-time scheduling and long-term schedule performance can be achieved. For the ease of implementation, earlier RL-based approaches always resort to classical learning methods, such as Q-learning or SARSA [37]–[39]. However, in most real-world production environments, the state space is continuous, and the number of production states is infinite, making it impossible to store all the state-action pairs in a single lookup Q-table.

In recent years, DRL has achieved great success in handling the state-explosion dilemma by using deep neural networks (DNNs) as the function approximator, which directly takes the continuous state features as input. In general, most of the DRL-based scheduling methods adopt DQN [7], [8] as the Q-function approximator substituting for the Q-table used by classical RL methods. Wei *et al.* [40] proposed an intelligent online job scheduling framework for applications in clouds with uncertainties and fluctuations of workloads. A DQN-based job scheduler is designed to dispatch jobs to limited resources under the quality of service (QoS) requirement constraints. Luo [41] developed a DQN agent adaptively selecting the most feasible composite dispatching rules to minimize the total tardiness in a flexible job shop with new job insertions. Zhou *et al.* [10] suggested a DRL-based method to minimize the makespan in a dynamic flexible job shop with new task arrivals. A DQN agent is used to choose a suitable service among all candidate services for each arriving task. Han and Yang [11] developed a dueling double deep Q-network with prioritized replay (DDDQNPR) to select various heuristic rules in a job shop with random processing times, where the manufacturing states are represented by multichannel images and a deep convolution neural network (CNN) is used to approximate the state-action values.

The major limitation of DQN-based methods is that they only estimate the value functions that cannot optimize over the policy. To address this issue, other DRL techniques based on policy search or actor–critic mechanism (i.e., A3C, DDPG, TRPO, and PPO) [6], [42]–[44] are developed where a policy network (actor) is introduced to generate the probability distribution over candidate actions, which is always trained using the feedback from another value network (critic). Zhu *et al.* [45] suggested a DRL-based online learning algorithm for real-time scheduling in cloud manufacturing. A DNN is used as the policy approximator to map the observed production features to the probability distribution over different actions, the policy parameters of which are optimized by policy gradient. Liu *et al.* [46] proposed an actor–critic multiagent DRL method for solving dynamic JSP with machine breakdowns and new job insertions. Each machine is assigned to one agent consisting of an actor network and a critic

network. A parallel training method combining asynchronous update and deep deterministic policy gradient (DDPG) is proposed to train the model. Kuhnle *et al.* [47] developed 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 a policy approximator to dispatch orders to feasible machines. Moreover, a recent review of DRL-based job shop scheduling methods is given in [9].

III. Background of RL and PPO

In general, RL deals with the Markov decision process (MDP) where an intelligent agent interacts with its surrounding environment by trial and error so as to maximize the expected cumulative long-term reward. A classical MDP could be represented by a five-tuple representation ($\boldsymbol{S}$, $\boldsymbol{A}$, $\boldsymbol{P}$, γ, and $\boldsymbol{R}$). 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 that, it enters a new state s_{t+1} with transition probability $p(s_{t+1}|s_t, a_t) \in \boldsymbol{P}(\boldsymbol{S}\times\boldsymbol{A} \rightarrow \boldsymbol{S})$ and receives an immediate reward $r_t \in \boldsymbol{R}(\boldsymbol{S}\times\boldsymbol{A}\times\boldsymbol{S} \rightarrow \mathbb{R})$. Denote $Q_\pi(s, a)$ as the expected cumulative reward upon taking an action a in state s and following a specific policy π thereafter (also known as the action-value function or Q-function), as defined as follows:

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

where $\gamma \in (0, 1]$ is the discount factor differentiating the relative importance of short-term reward and long-term reward. Analogously, denote $V_\pi(s)$ as the expected cumulative reward from state s and following a specific policy π thereafter (also known as the state-value function), as defined in (2), which satisfies $V_\pi(s) = \sum_a \pi(s,a) Q_\pi(s,a)$

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

The objective of an RL agent is to find an optimal policy π^* maximizing the expected cumulative reward among all possible states, as shown in the following:

$$\pi^* = \arg\max_\pi \mathbb{E}_{s|\pi}[V_\pi(s)]. \tag{3}$$

In DRL-based methods, the policy $\pi(\theta)$ is always represented by a DNN with differentiable parameters θ. In order to guarantee monotonic improvement of (3) between the old policy $\pi_{\text{old}}(\theta_{\text{old}})$ and the new policy $\pi(\theta)$ after parameter updating, TRPO [43] is proposed, which maximizes a surrogate objective subject to a constraint on the Kullback–Leibler (KL) divergence between $\pi_{\text{old}}(\theta_{\text{old}})$ and $\pi(\theta)$. PPO [44] is an extension of TRPO, which preserves the stability and reliability of trust-region methods, but is more compute-efficient and simpler to implement. It replaces the KL divergence constraint with a new objective, which only needs the computation of the likelihood. Define $\text{ratio}_t(\theta) = ((\pi(s_t, a_t; \theta))/(\pi_{\text{old}}(s_t, a_t; \theta_{\text{old}})))$; the objective of PPO is shown in the following:

$$\max_\theta \ \mathbb{E}_t \min(\text{ratio}_t(\theta)A_t, \text{clip}(\text{ratio}_t(\theta), 1-\epsilon, 1+\epsilon)A_t) \tag{4}$$

where $A_t = Q_{\pi_{\text{old}}}(s_t, a_t) - V_{\pi_{\text{old}}}(s_t)$ is the advantage function and $\epsilon \in (0, 1)$ is the clip parameter. Due to the simplicity and efficiency of PPO, it is adopted as the basic learning algorithm in this article.

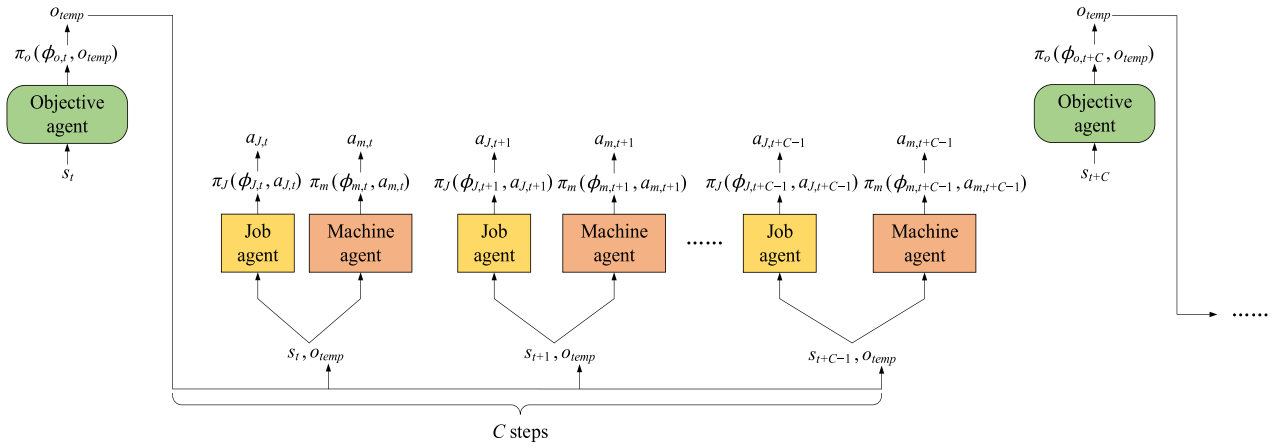
IV. Problem Formulation

The DMOFJSP-PNW addressed in this article 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 $\boldsymbol{M} = \{M_1, M_2, \ldots, M_m\}$. The arrival time and due date of job J_i are, respectively, A_i and D_i. The urgency degree of job J_i is denoted by Pr_i, where a higher urgency degree indicates a higher punishment on tardiness. Each job J_i consists of n_i operations with precedence constraint. $O_{i,j}$ $(j = 1, 2, \ldots, n_i)$ is the jth operation of job J_i. $\boldsymbol{ND_i}$ denotes the set of operations belonging to job J_i subjected to heterogeneous no-wait constraint with its direct successor. That is, if an operation $O_{i,j} \in \boldsymbol{ND_i}$, its successor $O_{i,j+1}$ should be immediately processed on a different machine after $O_{i,j}$ is completed. $\boldsymbol{NS_i}$ denotes the set of operations belonging to job J_i subjected to homogeneous no-wait constraint with its successors. $ns_{i,j}$ is the number of $O_{i,j}$'s successors with homogeneous no-wait constraint. That is, if an operation $O_{i,j} \in \boldsymbol{NS_i}$, its successors $O_{i,j+l}$ $(l = 1, 2, \ldots, ns_{i,j})$ should be immediately processed one after another on the same machine after $O_{i,j}$ is completed. Each operation $O_{i,j}$ can be processed on any machine M_k selected from a set of available machines $\boldsymbol{M_{i,j}}$ $(M_k \in \boldsymbol{M_{i,j}}, \boldsymbol{M_{i,j}} \subseteq \boldsymbol{M})$. The operations subjected to homogeneous no-wait constraint possess the same available machine set. The processing time of operation $O_{i,j}$ on machine M_k is denoted by $t_{i,j,k}$. $S_{i,j}$ and $C_{i,j}$ are, respectively, the start time and completion time of operation $O_{i,j}$. Each machine M_k might break down randomly; the start time of the rth breakdown on machine M_k is denoted by $\text{BS}_{k,r}$. Correspondingly, its repair time is denoted by $bt_{k,r}$. Three objectives, including TWT, average machine utilization rate (U_{ave}), and variance of machine workload (W_{std}), are considered to be optimized 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) Transportation times and setup times are negligible.
4) The maximum number of successive operations subjected to heterogeneous no-wait constraint is two.
5) Once an operation is interrupted by a machine breakdown, it should be immediately resumed on the same machine after the repairment.

Based on the notations above, the mathematical model of the DMOFJSP-PNW can be derived as follows:

$$\text{Min} \begin{cases} \text{TWT} = \displaystyle\sum_{i=1}^{n} \max(C_{i,n_i} - D_i, 0) \cdot Pr_i & (5) \\ \dfrac{1}{U_{\text{ave}}} = \dfrac{1}{\frac{\sum_{k=1}^{m} U_k}{m}} & (6) \\ W_{\text{std}} = \sqrt{\dfrac{\sum_{k=1}^{m} (W_k - \frac{\sum_{l=1}^{m} W_l}{m})^2}{m}} & (7) \end{cases}$$



t: rescheduling point
s_t: production state at rescheduling point t
o_{temp}: temporary higher objective
$\phi_{o,t}$: state feature vector of the objective agent at rescheduling point t, $\phi_{o,t} = \phi(s_t)$
$\phi_{J,t}$: state feature vector of the job agent at rescheduling point t, $\phi_{J,t} = [\phi(s_t), o_{temp}]$
$\phi_{m,t}$: state feature vector of the machine agent at rescheduling point t, $\phi_{m,t} = [\phi(s_t), o_{temp}]$
π_o: objective selection policy of the objective agent
π_J: rule selection policy of the job agent
π_m: rule selection policy of the machine agent
$a_{J,t}$: job selection rule at rescheduling point t
$a_{m,t}$: machine assignment rule at rescheduling point t
C: the interval time length for higher objective adjustment

Fig. 1. Overall operating mechanism of the proposed HMAPPO in different spatiotemporal levels. The objective agent serves as the higher controller periodically adjusting the higher objective. The job agent and machine agent are lower actuators respectively selecting the feasible job selection rules and machine assignment rules to achieve the higher objective at each rescheduling point.

$$
\boldsymbol{s.t.}\ \begin{cases}
\sum\limits_{k\in \boldsymbol{M_{i,j}}} X_{i,j,k} = 1, \quad \forall i, j & (8)\\
(C_{i,1} - t_{i,1,k} - A_i)X_{i,1,k} \geq 0, \quad \forall i, k & (9)\\
(C_{i,j} - t_{i,j,k} - C_{i,j-1})X_{i,j,k} \geq 0, \quad \forall i, j, k & (10)\\
(C_{h,g} - t_{h,g,k} - C_{i,j})X_{i,j,k}X_{h,g,k}(Y_{i,j,h,g} + 1) & \\
+(C_{i,j} - t_{i,j,k} - C_{h,g})X_{i,j,k}X_{h,g,k}(1 - Y_{i,j,h,g}) & \\
\geq 0, \quad \forall i, j, h, g, k & (11)\\
(C_{i,j+1} - t_{i,j+1,k} - C_{i,j})X_{i,j+1,k} = 0, & \\
\quad \forall i, k, j \in \boldsymbol{ND_i} & (12)\\
(C_{i,j+l} - t_{i,j+l,k} - C_{i,j+l-1})X_{i,j+l,k} = 0, & \\
\quad \forall i, k, j \in \boldsymbol{NS_i}, l = 1, 2, \ldots, ns_{i,j} & (13)\\
X_{i,j,k} = X_{i,j+l,k} & \\
\quad \forall i, k, j \in \boldsymbol{NS_i}, l = 1, 2, \ldots, ns_{i,j} & (14)\\
(C_{i,j} - t_{i,j,k} - (\mathrm{BS}_{k,r} + bt_{k,r}))X_{i,j,k} \geq 0 & \\
\vee(\mathrm{BS}_{k,r} - C_{i,j})X_{i,j,k} \geq 0 & \\
\vee(C_{i,j} - S_{i,j} - t_{i,j,k} - bt_{k,r})X_{i,j,k} = 0 & \\
\quad \forall i, j, k, r & (15)
\end{cases}
$$

$$
X_{i,j,k} = \begin{cases} 1, & \text{if } O_{i,j} \text{ is assigned on machine } M_k \\ 0, & \text{otherwise} \end{cases} \quad (16)
$$

$$
Y_{i,j,h,g} = \begin{cases} 1, & \text{if } O_{i,j} \text{ is a predecessor of } O_{h,g} \\ -1, & \text{if } O_{i,j} \text{ is a successor of } O_{h,g} \end{cases} \quad (17)
$$

$$
W_k = \sum_{i=1}^{n}\sum_{j=1}^{n_i} t_{i,j,k}X_{i,j,k} \quad (18)
$$

$$
U_k = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n_i} t_{i,j,k}X_{i,j,k}}{\max_{i,j} C_{i,j}X_{i,j,k}}. \quad (19)
$$

Objective (5) is TWT. Objective (6) is average machine utilization rate U_{ave}. Objective (7) is the variance of machine workload W_{std}. Equation (8) indicates that each operation should be assigned on only one machine from the available machine set. Equation (9) ensures that each operation can only be processed after its arrival time. Equation (10) is the precedence constraint. Equation (11) is the capacity constraint. Equation (12) denotes the heterogeneous no-wait constraint among two adjacent operations. Equation (13) denotes the homogeneous no-wait constraint among a set of successive operations. Equation (14) suggests that the operations subjected to homogeneous no-wait constraints should be assigned on the same machine. Equation (15) indicates that, once an operation is interrupted by a machine breakdown, it should be resumed on the same machine after the repairment is finished.

V. Real-Time Scheduling Method HMAPPO

In this section, the details of the proposed real-time scheduling method HMAPPO are successively provided, including the definition of state features, the proposed job selection rules and machine assignment rules, the definition of the reward function, and the network structures of the proposed agents. Finally, the training and implementation frameworks of the HMAPPO are derived. Fig. 1 gives the overall operating mechanism of the proposed HMAPPO in different spatiotemporal levels.

A. Definition of State Features

The state features $\phi(s_t)$ of production state s_t, which serve as input of the neural networks, are important for the DRL agents to make appropriate decisions at each rescheduling point t. In this article, 20 state features are extracted to comprehensively reflect the production status. Before giving the detailed definitions of state features, we first summarize the important notations used along the rest of this article as follows.

$\mathrm{CT}_k(t)$*:* The completion time of the last operation that has been assigned on machine M_k until rescheduling point t, i.e., the earliest available time of M_k.

$T_{\text{cur}} = ((\sum_{k=1}^{m} \text{CT}_k(t))/m)$*:* The average completion time of all machines.

$\text{OP}_i(t)$*:* Total number of operations belonging to job J_i that have been scheduled until rescheduling point t.

$C_{i,\text{OP}_i(t)}$*:* The completion time of the last scheduled operation of job J_i until rescheduling point t.

$\overline{W}_k(t) = \left(\left(\sum_{i=1}^{n}\sum_{j=1}^{\text{OP}_i(t)} t_{i,j,k}X_{i,j,k}\right)/\left(\max_l \sum_{i=1}^{n}\sum_{j=1}^{\text{OP}_i(t)} t_{i,j,l}X_{i,j,l}\right)\right)$*:* The normalized workload of machine M_k at rescheduling point t.

$U_k(t) = \left(\left(\sum_{i=1}^{n}\sum_{j=1}^{\text{OP}_i(t)} t_{i,j,k}X_{i,j,k}\right)/\left(\text{CT}_k(t)\right)\right)$*:* The utilization rate of machine M_k at rescheduling point t.

$\text{CRJ}_i(t) = ((\text{OP}_i(t))/n_i)$*:* The completion rate of job J_i at rescheduling point t.

$\overline{t}_{i,j} = \left(\left(\sum_{k\in \boldsymbol{M}_{i,j}} t_{i,j,k}\right)/\left(|\boldsymbol{M}_{i,j}|\right)\right)$*:* The average processing time of operation $O_{i,j}$ on all its available machines.

$\boldsymbol{UC}_{\text{job}}(\boldsymbol{t}) = \{J_i|\text{OP}_i(t) < n_i\}$*:* The set of uncompleted jobs at rescheduling point t.

Based on the notations above, the state features at each rescheduling point t can be derived as follows.

1) Total number of machines (m) in the shop floor.
2) Average utilization rate of machines $U_{\text{ave}}(t)$, as defined in the following:

$$U_{\text{ave}}(t) = \frac{\sum_{k=1}^{m} U_k(t)}{m}. \tag{20}$$

3) The standard deviation of machine utilization rate $U_{\text{std}}(t)$, as defined in the following:

$$U_{\text{std}}(t) = \sqrt{\frac{\sum_{k=1}^{m}(U_k(t) - U_{\text{ave}}(t))^2}{m}}. \tag{21}$$

4) Completion rate of operations $\text{CRO}(t)$, as defined in the following:

$$\text{CRO}(t) = \frac{\sum_{i=1}^{n}\text{OP}_i(t)}{\sum_{i=1}^{n} n_i}. \tag{22}$$

5) Average job completion rate $\text{CRJ}_{\text{ave}}(t)$, as defined in the following:

$$\text{CRJ}_{\text{ave}}(t) = \frac{\sum_{i=1}^{n}\text{CRJ}_i(t)}{n}. \tag{23}$$

6) The standard deviation of job completion rate $\text{CRJ}_{\text{std}}(t)$, as defined in the following:

$$\text{CRJ}_{\text{std}}(t) = \sqrt{\frac{\sum_{i=1}^{n}(\text{CRJ}_i(t) - \text{CRJ}_{\text{ave}}(t))^2}{n}}. \tag{24}$$

7) Average normalized machine workload $\overline{W}_{\text{ave}}(t)$, as defined in the following:

$$\overline{W}_{\text{ave}}(t) = \frac{\sum_{k=1}^{m}\overline{W}_k(t)}{m}. \tag{25}$$

8) The standard deviation of normalized machine workload $\overline{W}_{\text{std}}(t)$, as defined in the following:

$$\overline{W}_{\text{std}}(t) = \sqrt{\frac{\sum_{k=1}^{m}\left(\overline{W}_k(t) - \overline{W}_{\text{ave}}(t)\right)^2}{m}}. \tag{26}$$

9) *Estimated Tardiness Rate $\text{Tard}_e(t)$:* An operation $O_{i,j}$ is estimated to be tardy if $\max(T_{\text{cur}}, C_{i,\text{OP}_i(t)}) + \sum_{l=\text{OP}_i(t)+1}^{j}\overline{t}_{i,l} > D_i$. The estimated tardy operations of job J_i at rescheduling point t are denoted by $\boldsymbol{ETO_i(t)} = \{O_{i,j}|\max(T_{\text{cur}}, C_{i,\text{OP}_i(t)}) + \sum_{l=\text{OP}_i(t)+1}^{j}\overline{t}_{i,l} > D_i\}$. The estimated tardiness rate $\text{Tard}_e(t)$ is defined as the number of estimated tardy operations divided by the number of all unscheduled operations, as shown in the following:

$$\text{Tard}_e(t) = \frac{\sum_{i\in \boldsymbol{UC}_{\text{job}}(\boldsymbol{t})}|\boldsymbol{ETO_i(t)}|}{\sum_{i\in \boldsymbol{UC}_{\text{job}}(\boldsymbol{t})}(n_i - \text{OP}_i(t))}. \tag{27}$$

10) *Actual Tardiness Rate $\text{Tard}_a(t)$:* Denote $\boldsymbol{ATJ(t)} = \{J_i|\text{OP}_i(t) < n_i \;\&\&\; D_i < \max(C_{i,\text{OP}_i(t)}, \min_k \text{CT}_k(t))\}$ as the actual set of tardy jobs at t. The actual tardiness rate $\text{Tard}_a(t)$ is defined as the number of actual tardy operations divided by the number of all unscheduled operations, as shown in the following:

$$\text{Tard}_a(t) = \frac{\sum_{i\in \boldsymbol{ATJ(t)}}(n_i - \text{OP}_i(t))}{\sum_{i\in \boldsymbol{UC}_{\text{job}}(\boldsymbol{t})}(n_i - \text{OP}_i(t))}. \tag{28}$$

Based on the features defined above, we can obtain the feature vector $\boldsymbol{V_t} = [m, U_{\text{ave}}(t), U_{\text{std}}(t), \text{CRO}(t), \text{CRJ}_{\text{ave}}(t), \text{CRJ}_{\text{std}}(t), \overline{W}_{\text{ave}}(t), \overline{W}_{\text{std}}(t), \text{Tard}_e(t), \text{ and } \text{Tard}_a(t)]$ at each rescheduling point t. Note that the changing trend of state features also contains important information, which directly reflects if the schedule performance has been improved [47]. In this regard, we calculate the difference vector $\boldsymbol{D_t} = \boldsymbol{V_t} - \boldsymbol{V_{t-1}}$ between $\boldsymbol{V_t}$ and its previous value at $t-1$. Ultimately, the state feature vector $\phi(s_t)$ of state s_t is defined as the concatenation of $\boldsymbol{V_t}$ and $\boldsymbol{D_t}$, i.e., $\phi(s_t) = [\boldsymbol{V_t}, \boldsymbol{D_t}]$.

B. Job Selection Rules

Five job selection rules are developed in this article, which serve as the candidate action set of the job agent. At each rescheduling point t, the job agent chooses a job selection rule $a_{J,t}$ based on the current state features $\phi(s_t)$ and temporary objective o_{temp}, after which an uncompleted job J_i can be selected by $a_{J,t}$. Denote $\boldsymbol{ETJ(t)} = \{i|\text{OP}_i(t) < n_i \;\&\&\; D_i < \max(T_{\text{cur}}, C_{i,\text{OP}_i(t)})\}$ as the set of estimated tardy jobs at t. The details of the proposed job selection rules are given as follows. To summary, each rule (except for rule 5) first sorts all the jobs according to a specific indicator of tardiness and then chooses the job most likely to be tardy to be processed next.

1) Job Selection Rule 1: If $\boldsymbol{ETJ(t)}$ is empty, $J_i \leftarrow \arg\min_{i\in \boldsymbol{UC}_{\text{job}}(\boldsymbol{t})} ((D_i - \max(T_{\text{cur}}, C_{i,\text{OP}_i(t)}))/(n_i - \text{OP}_i(t))) \cdot (1/Pr_i)$. Else, $J_i \leftarrow \arg\max_{i\in \boldsymbol{ETJ(t)}} \left(\left(\max(T_{\text{cur}}, C_{i,\text{OP}_i(t)}\right) + \sum_{j=\text{OP}_i(t)+1}^{n_i}\overline{t}_{i,j} - D_i\right) \cdot Pr_i\right)$.

2) Job Selection Rule 2: If $\boldsymbol{ETJ(t)}$ is empty, $J_i \leftarrow \arg\min_{i\in \boldsymbol{UC}_{\text{job}}(\boldsymbol{t})} \left(\left(D_i - \max\left(T_{\text{cur}}, C_{i,\text{OP}_i(t)}\right)\right)/\left(\sum_{j=\text{OP}_i(t)+1}^{n_i}\overline{t}_{i,j}\right)\right) \cdot \left(1/Pr_i\right)$. Else, $J_i \leftarrow \arg\max_{i\in \boldsymbol{ETJ(t)}} \left(\left(\max(T_{\text{cur}}, C_{i,\text{OP}_i(t)}\right) + \sum_{j=\text{OP}_i(t)+1}^{n_i}\overline{t}_{i,j} - D_i\right) \cdot Pr_i\right)$.

3) Job Selection Rule 3: $J_i \leftarrow \arg\max_{i\in \boldsymbol{UC}_{\text{job}}(\boldsymbol{t})} (\max(T_{\text{cur}}, C_{i,\text{OP}_i(t)}) + \sum_{j=\text{OP}_i(t)+1}^{n_i}\overline{t}_{i,j} - D_i < 0 \;?\; \max(T_{\text{cur}}, C_{i,\text{OP}_i(t)}) + \sum_{j=\text{OP}_i(t)+1}^{n_i}\overline{t}_{i,j} - D_i : (\max(T_{\text{cur}}, C_{i,\text{OP}_i(t)}) + \sum_{j=\text{OP}_i(t)+1}^{n_i}\overline{t}_{i,j} - D_i) \cdot Pr_i)$.

4) Job Selection Rule 4: If $\boldsymbol{ETJ(t)}$ is empty, $J_i \leftarrow \arg\min_{i\in \boldsymbol{UC}_{\text{job}}(\boldsymbol{t})}((\text{OP}_i(t))/n_i)\cdot(D_i-\max(T_{\text{cur}}, C_{i,\text{OP}_i(t)}))\cdot(1/Pr_i)$. Else, $J_i \leftarrow \arg\max_{i\in \boldsymbol{ETJ(t)}}(n_i/(\text{OP}_i(t)))\cdot(\max(T_{\text{cur}}, C_{i,\text{OP}_i(t)})+\sum_{j=\text{OP}_i(t)+1}^{n_i}\overline{t}_{i,j}-D_i)\cdot Pr_i$.

5) Job Selection Rule 5: Randomly choose an uncompleted job J_i from $\boldsymbol{UC}_{\text{job}}(\boldsymbol{t})$.

Once a job J_i is selected, the next operation $O_{i,j}$ of J_i where $j=\text{OP}_i(t)+1$ will be investigated. If $O_{i,j}\in \boldsymbol{NS_i}$, we choose operation $O_{i,j}$ and all its successors $O_{i,j+l}$ $(l=1,2,\ldots,ns_{i,j})$ in homogeneous no-wait constraint to be processed next. Else if $O_{i,j}\in \boldsymbol{ND_i}$, we choose both operation $O_{i,j}$ and its successor $O_{i,j+1}$ in heterogeneous no-wait constraint. Otherwise, we directly choose $O_{i,j}$ to be processed next.

C. Machine Assignment Rules

Six machine assignment rules are developed in this article, which serves as the candidate action set of the machine agent. At each rescheduling point t, the machine agent chooses a machine assignment rule $a_{m,t}$ based on the current state features $\phi(s_t)$ and temporary objective o_{temp}, through which the processing machines for the chosen operation $O_{i,j}$ and its successors in no-wait constraint (if any) can be determined. The details of the proposed machine assignment rules are given as follows. In order to address the partial-no-wait constraint, each rule successively investigates if $O_{i,j}\in \boldsymbol{NS_i}$ or $\boldsymbol{ND_i}$ and then assigns $O_{i,j}$ together with its successors in no-wait constraint (if any) on the feasible processing machines. To summary, rules 1, 4, and 5 are designed to reduce the TWT. Rules 2 and 3 are designed to, respectively, enhance the average machine utilization rate and reduce the variance of machine workload, while rule 6 is a random rule.

1) Machine Assignment Rule 1: Assign $O_{i,j}$ on $M_{k_1}\leftarrow \arg\min_{k_1\in \boldsymbol{M}_{i,j}}\max(\text{CT}_{k_1}(t), C_{i,j-1})$. If $O_{i,j}\in \boldsymbol{NS_i}$, assign all the successors of $O_{i,j}$ in homogeneous no-wait constraint on M_{k_1}. Else if $O_{i,j}\in \boldsymbol{ND_i}$, let $T_{\text{earliest}}=\max(\text{CT}_{k_1}(t), C_{i,j-1})+t_{i,j,k_1}$, and assign $O_{i,j+1}$ on $M_{k_2}\leftarrow \arg\min_{k_2\in \boldsymbol{M}_{i,j+1}}\max(\text{CT}_{k_2}(t), T_{\text{earliest}})$. This rule intends to assign the selected operation on the machine with the earliest available time.

2) Machine Assignment Rule 2: Assign $O_{i,j}$ on $M_{k_1}\leftarrow \arg\min_{k_1\in \boldsymbol{M}_{i,j}}\left(\left(\sum_{i=1}^{n}\sum_{l=1}^{\text{OP}_i(t)}t_{i,l,k_1}X_{i,l,k_1}\right)/\left(\text{CT}_{k_1}(t)\right)\right)$. If $O_{i,j}\in \boldsymbol{NS_i}$, assign all the successors of $O_{i,j}$ in homogeneous no-wait constraint on M_{k_1}. Else if $O_{i,j}\in \boldsymbol{ND_i}$, assign $O_{i,j+1}$ on $M_{k_2}\leftarrow \arg\min_{k_2\in \boldsymbol{M}_{i,j+1}}\left(\left(\sum_{i=1}^{n}\sum_{l=1}^{\text{OP}_i(t)}t_{i,l,k_2}X_{i,l,k_2}\right)/\left(\text{CT}_{k_2}(t)\right)\right)$. This rule intends to assign the selected operation on the machine with the lowest utilization rate.

3) Machine Assignment Rule 3: Assign $O_{i,j}$ on $M_{k_1}\leftarrow \arg\min_{k_1\in \boldsymbol{M}_{i,j}}\sum_{i=1}^{n}\sum_{l=1}^{\text{OP}_i(t)}t_{i,l,k_1}X_{i,l,k_1}$. If $O_{i,j}\in \boldsymbol{NS_i}$, and assign all the successors of $O_{i,j}$ in homogeneous no-wait constraint on M_{k_1}. Else if $O_{i,j}\in \boldsymbol{ND_i}$, assign $O_{i,j+1}$ on $M_{k_2}\leftarrow \arg\min_{k_2\in \boldsymbol{M}_{i,j+1}}\sum_{i=1}^{n}\sum_{l=1}^{\text{OP}_i(t)}t_{i,l,k_2}X_{i,l,k_2}$. This rule intends to assign the selected operation on the machine with the lowest workload.

4) Machine Assignment Rule 4: Assign $O_{i,j}$ on $M_{k_1}\leftarrow \arg\min_{k_1\in \boldsymbol{M}_{i,j}}t_{i,j,k_1}$. If $O_{i,j}\in \boldsymbol{NS_i}$, assign $O_{i,j}$ and all its successors in homogeneous no-wait constraint on $M_{k_1}\leftarrow \arg\min_{k_1\in \boldsymbol{M}_{i,j}}\sum_{l=0}^{ns_{i,j}}t_{i,j+l,k_1}$. Else if $O_{i,j}\in \boldsymbol{ND_i}$, and assign $O_{i,j+1}$ on $M_{k_2}\leftarrow \arg\min_{k_2\in \boldsymbol{M}_{i,j+1}}t_{i,j+1,k_2}$. This rule intends to assign the selected operation on the machine with the shortest processing time (SPT).

5) Machine Assignment Rule 5: Assign $O_{i,j}$ on $M_{k_1}\leftarrow \arg\min_{k_1\in \boldsymbol{M}_{i,j}}(\max(\text{CT}_{k_1}(t), C_{i,j-1})+t_{i,j,k_1})$. If $O_{i,j}\in \boldsymbol{NS_i}$, assign $O_{i,j}$ and all its successors in homogeneous no-wait constraint on $M_{k_1}\leftarrow \arg\min_{k_1\in \boldsymbol{M}_{i,j}}\left(\max(\text{CT}_{k_1}(t), C_{i,j-1})+\sum_{l=0}^{ns_{i,j}}t_{i,j+l,k_1}\right)$. Else if $O_{i,j}\in \boldsymbol{ND_i}$, let $T_{\text{earliest}}=\max(\text{CT}_{k_1}(t), C_{i,j-1})+t_{i,j,k_1}$, and assign $O_{i,j+1}$ on $M_{k_2}\leftarrow \arg\min_{k_2\in \boldsymbol{M}_{i,j+1}}\left(\max(\text{CT}_{k_2}(t), T_{\text{earliest}})+t_{i,j+1,k_2}\right)$. This rule intends to assign the selected operation on the machine with the earliest finish time (EFT).

6) Machine Assignment Rule 6: Assign $O_{i,j}$ on M_{k_1} randomly selected from $\boldsymbol{M_{i,j}}$. If $O_{i,j}\in \boldsymbol{NS_i}$, assign all the successors of $O_{i,j}$ in homogeneous no-wait constraint on M_{k_1}. Else if $O_{i,j}\in \boldsymbol{ND_i}$, assign $O_{i,j+1}$ on M_{k_2} randomly selected from $\boldsymbol{M_{i,j+1}}$. This rule intends to assign the selected operation on a random machine from its available set.

It should be noted that, if $O_{i,j}\in \boldsymbol{ND_i}$, the earliest start and completion time of $O_{i,j}$ on machine M_{k_1} are, respectively, $\max(\text{CT}_{k_1}(t), C_{i,j-1})$ and $\max(\text{CT}_{k_1}(t), C_{i,j-1})+t_{i,j,k_1}$. Thus, the start time of $O_{i,j+1}$ on machine M_{k_2} is $\max(\text{CT}_{k_2}(t), \max(\text{CT}_{k_1}(t), C_{i,j-1})+t_{i,j,k_1})$. In order to meet the heterogeneous no-wait constraint, if $\text{CT}_{k_2}(t)>\max(\text{CT}_{k_1}(t), C_{i,j-1})+t_{i,j,k_1}$, the start time of $O_{i,j}$ will be delayed to $\text{CT}_{k_2}(t)-t_{i,j,k_1}$ so that its completion time is equal to $\text{CT}_{k_2}(t)$.

D. Definition of Reward Function

At each rescheduling point t, three indicators corresponding to three objectives are taken for calculating the reward r_t upon taking the job selection rule $a_{J,t}$ and machine assignment rule $a_{m,t}$. For objective TWT, we take the estimated total weighted tardiness $\text{ETWT}(t)$ as the indicator, which can be calculated as $\text{ETWT}(t)=\sum_{i\in \boldsymbol{UC}_{\text{job}}(\boldsymbol{t})}\max\left(\max(T_{\text{cur}}, C_{i,\text{OP}_i(t)})+\sum_{j=\text{OP}_i(t)+1}^{n_i}\overline{t}_{i,j}-D_i, 0\right)\cdot Pr_i$. For objective U_{ave}, we directly take the average machine utilization rate $U_{\text{ave}}(t)$ as the indicator, as defined in (20). For objective W_{std}, we take the standard deviation of normalized machine workload $\overline{W}_{\text{std}}(t)$ as the indicator, as defined in (26).

The reward r_t is obtained by considering if the indicator corresponding to the temporary objective o_{temp} at rescheduling point $t+1$, i.e., $\text{ETWT}(t+1)$, $U_{\text{ave}}(t+1)$, or $\overline{W}_{\text{std}}(t+1)$ is improved compared to its original value. The definition of reward function is given in Algorithm 1.

E. Network Structures of the Proposed Agents

Each of the proposed agents contains a policy network and a value network. The policy network generates the probability distribution over different actions. The value network estimates the state-value functions of different states that are used as feedback to train the policy network. The objective policy network π_o is a DNN containing a Relu input layer with 20 nodes, five Relu hidden layers with 200 nodes each layer, and a softmax output layer with three nodes. The objective value network v_o is a DNN containing a Relu input layer

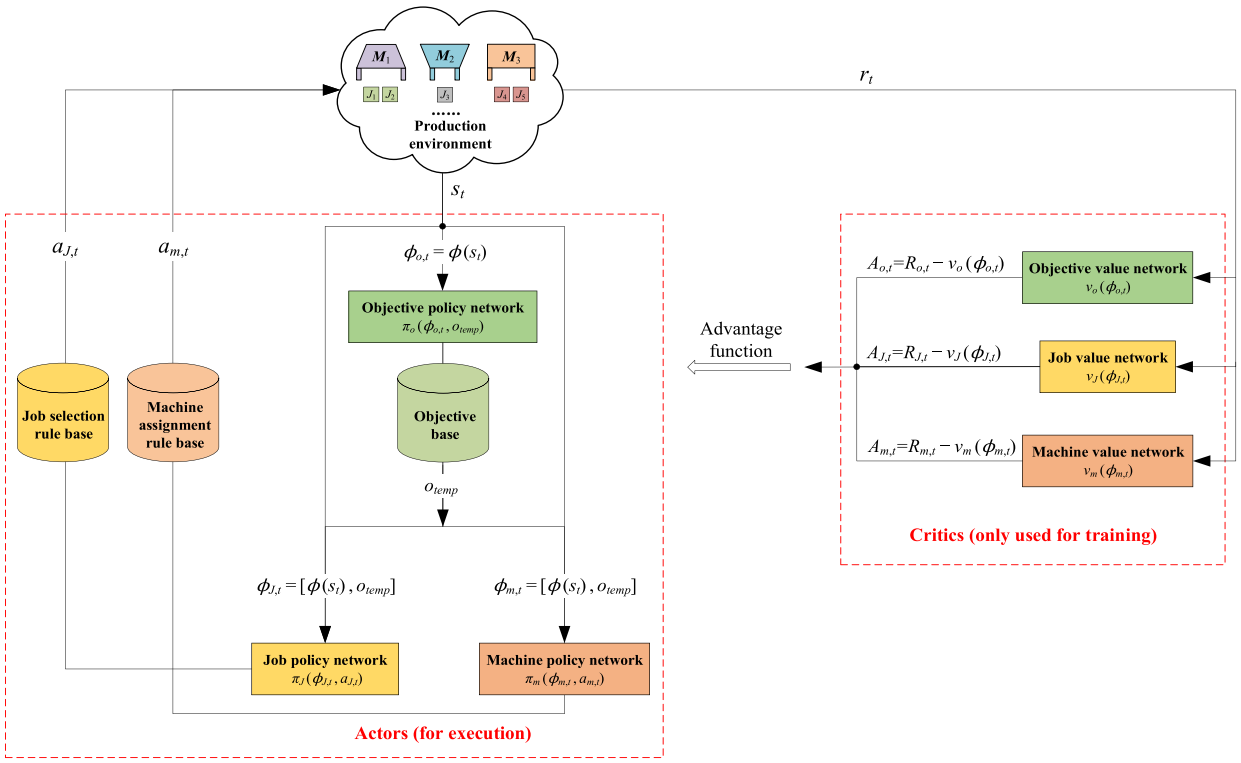


Fig. 2. Cooperation mechanism of different networks used in the three proposed agents.

Algorithm 1 Definition of the Reward r_t at Each Rescheduling Point t

Input: The temporary objective o_{temp}
Output: The reward r_t at rescheduling point t

if $o_{temp} == 1$ **then**
  if $ETWT(t+1) < ETWT(t)$ **then**
    $r_t \leftarrow 1$
  else if $ETWT(t+1) == ETWT(t)$ **then**
    $r_t \leftarrow 0$
  else
    $r_t \leftarrow -1$
else if $o_{temp} == 2$ **then**
  if $U_{ave}(t+1) > U_{ave}(t)$ **then**
    $r_t \leftarrow 1$
  else if $U_{ave}(t+1) \geq U_{ave}(t) \cdot 0.97$ **then**
    $r_t \leftarrow 0$
  else
    $r_t \leftarrow -1$
else if $o_{temp} == 3$ **then**
  if $\overline{W}_{std}(t+1) < \overline{W}_{std}(t) \cdot 0.9$ **then**
    $r_t \leftarrow 1$
  else if $\overline{W}_{std}(t+1) \leq \overline{W}_{std}(t)$ **then**
    $r_t \leftarrow 0$
  else
    $r_t \leftarrow -1$
end if
return r_t

with 20 nodes, three Relu hidden layers with 200 nodes each layer, and a fully connected output layer with one node. The job policy network π_J and the machine policy network π_m are, respectively, a DNN containing a Relu input layer with 21 nodes, five Relu hidden layers with 200 nodes each layer, and a softmax output layer with five nodes for the job policy network and six nodes for the machine policy network. Both the job value network v_J and the machine value network v_m are a DNN containing a Relu input layer with 21 nodes, three Relu hidden layers with 200 nodes each layer, and a fully connected output layer with one node.

Fig. 2 illustrates the cooperation mechanism of different networks used in the three proposed agents. To sum up, the policy networks π_o, π_J, and π_m are actors that are responsible for choosing the objectives and dispatching rules, while the value networks v_o, v_J, and v_m are critics responsible for estimating the state-value functions and then calculating the advantage functions that are used as feedback for training the policy networks.

F. Training Algorithm for the HMAPPO

A hierarchical PPO-based training algorithm is developed to train the three agents. The rescheduling point t is defined as the time when a new job arrives, or a machine breakdown occurs, or a machine is freed (i.e, the last operation assigned on which is completed). At each rescheduling point, the job policy network π_J and the machine policy network π_m, respectively, choose a job selection rule $a_{J,t}$ and a machine assignment rule $a_{m,t}$ according to the current state features $\phi(s_t)$ and temporary objective o_{temp}. Consequently, an uncompleted job J_i is selected by $a_{J,t}$, and the next operation of job J_i together with its successors in no-wait constraint is assigned on the processing machines chosen by $a_{m,t}$. In every C rescheduling steps, the temporary objective o_{temp} is adjusted by the objective

policy network π_o. The parameters of the job agent and machine agent are updated every M rescheduling steps according to Algorithm 3, while the parameters of the objective agent are updated slower every $C \cdot M$ rescheduling steps according to Algorithm 4. The training procedure is repeated for L epochs where each epoch is conducted in a randomly generated production environment so as to enhance the generality of the proposed HMAPPO. The overall framework of the hierarchical PPO-based training algorithm is provided in Algorithm 2.

Algorithm 2 Hierarchical PPO-Based Training Algorithm

Randomly initialize the objective policy network $\pi_o(\theta_o)$, the job policy network $\pi_J(\theta_J)$ and the machine policy network $\pi_m(\theta_m)$
Initialize the old objective policy network $\pi_{o,old}(\theta_{o,old}) = \pi_o(\theta_o)$, the old job policy network $\pi_{J,old}(\theta_{J,old}) = \pi_J(\theta_J)$ and the old machine policy network $\pi_{m,old}(\theta_{m,old}) = \pi_m(\theta_m)$
Randomly initialize the objective value network $v_o(w_o)$, the job value network $v_J(w_J)$ and the machine value network $v_m(w_m)$
for $epoch = 1 : L$ **do**
  for $t = 0 : T$ (t is the rescheduling point, T is the terminal time when all the operations have been scheduled) **do**
    if a new job J_i arrives **then**
      Add the new job to the set of uncompleted jobs $\boldsymbol{UC_{job}(t)}$
    end if
    if the rth breakdown of machine M_k occurs **then**
      if there exists an operation $O_{i,j}$ being processed on M_k **then**
        $C_{i,j} \leftarrow C_{i,j} + bt_{k,r}$, $CT_k(t) \leftarrow C_{i,j}$
      else
        $CT_k(t) \leftarrow BS_{k,r} + bt_{k,r}$
      end if
    end if
    Observe the current state s_t, extract the current state feature vector $\phi(s_t)$
    if $t \,\%\, C == 0$ **then**
      $\phi_{o,t} \leftarrow \phi(s_t)$, choose the temporary objective $o_{temp} \leftarrow \pi_{o,old}(\phi_{o,t}, o_{temp}; \theta_{o,old})$
      $t' \leftarrow t$, $G_{t'} \leftarrow 0$, $o_{t'} \leftarrow o_{temp}$
    end if
    $\phi_{J,t} \leftarrow [\phi(s_t), o_{temp}]$, choose a job selection rule $a_{J,t} \leftarrow \pi_{J,old}(\phi_{J,t}, a_{J,t}; \theta_{J,old})$
    $\phi_{m,t} \leftarrow [\phi(s_t), o_{temp}]$, choose a machine assignment rule $a_{m,t} \leftarrow \pi_{m,old}(\phi_{m,t}, a_{m,t}; \theta_{m,old})$
    Select an uncompleted job J_i by $a_{J,t}$, assign the next operation of job J_i and its successors in no-wait constraint (if any) on the corresponding machines chosen by $a_{m,t}$
    Enter the new state s_{t+1}, calculate the intermediate reward r_t by Algorithm 1
    $G_{t'} \leftarrow G_{t'} + \gamma^{t-t'} \cdot r_t$
    if $t \,\%\, M == 0$ **then**
      Update $\pi_J(\theta_J)$, $v_J(w_J)$, $\pi_m(\theta_m)$ and $v_m(w_m)$ by Algorithm 3
      $\pi_{J,old}(\theta_{J,old}) \leftarrow \pi_J(\theta_J)$, $\pi_{m,old}(\theta_{m,old}) \leftarrow \pi_m(\theta_m)$
    end if
    if $t \,\%\, (C \cdot M) == 0$ **then**
      Update $\pi_o(\theta_o)$ and $v_o(w_o)$ by Algorithm 4
      $\pi_{o,old}(\theta_{o,old}) \leftarrow \pi_o(\theta_o)$
    end if
  end for
end for

Algorithm 3 Parameter Updating Procedure for $\pi_J(\theta_J)$, $v_J(w_J)$, $\pi_m(\theta_m)$, and $v_m(w_m)$

$R_{J,t} = v_J(\phi_{J,t}; w_J)$
$R_{m,t} = v_m(\phi_{m,t}; w_m)$
for $i = t-1 : -1 : t-M$ **do**
  $R_{J,i} = r_i + \gamma R_{J,i+1}$
  $A_{J,i} = R_{J,i} - v_J(\phi_{J,i}; w_J)$
  $ratio_{J,i}(\theta_J) = \frac{\pi_J(\phi_{J,i}, a_{J,i}; \theta_J)}{\pi_{J,old}(\phi_{J,i}, a_{J,i}; \theta_{J,old})}$
  $R_{m,i} = r_i + \gamma R_{m,i+1}$
  $A_{m,i} = R_{m,i} - v_m(\phi_{m,i}; w_m)$
  $ratio_{m,i}(\theta_m) = \frac{\pi_m(\phi_{m,i}, a_{m,i}; \theta_m)}{\pi_{m,old}(\phi_{m,i}, a_{m,i}; \theta_{m,old})}$
end for
$JPPO(\theta_J) = -\sum_{i=t-M}^{t-1}$
$\min(ratio_{J,i}(\theta_J)A_{J,i}, \text{clip}(ratio_{J,i}(\theta_J), 1-\epsilon, 1+\epsilon)A_{J,i})$
$JPPO(\theta_m) = -\sum_{i=t-M}^{t-1}$
$\min(ratio_{m,i}(\theta_m)A_{m,i}, \text{clip}(ratio_{m,i}(\theta_m), 1-\epsilon, 1+\epsilon)A_{m,i})$
$MSE(w_J) = \sum_{i=t-M}^{t-1}(R_{J,i} - v_J(\phi_{J,i}; w_J))^2$
$MSE(w_m) = \sum_{i=t-M}^{t-1}(R_{m,i} - v_m(\phi_{m,i}; w_m))^2$
Update θ_J and θ_m by gradient descent w.r.t. $JPPO(\theta_J)$ and $JPPO(\theta_m)$ for N_p times
Update w_J and w_m by gradient descent w.r.t. $MSE(w_J)$ and $MSE(w_m)$ for N_v times
return θ_J, w_J, θ_m and w_m

Algorithm 4 Parameter Updating Procedure for $\pi_o(\theta_o)$ and $v_o(w_o)$

$R_{o,t} = v_o(\phi_{o,t}; w_o)$
for $i = t-C : -C : t-C \cdot M$ **do**
  $R_{o,i} = G_i + \gamma R_{o,i+C}$
  $A_{o,i} = R_{o,i} - v_o(\phi_{o,i}; w_o)$
  $ratio_{o,i}(\theta_o) = \frac{\pi_o(\phi_{o,i}, o_i; \theta_o)}{\pi_{o,old}(\phi_{o,i}, o_i; \theta_{o,old})}$
end for
$JPPO(\theta_o) = -\sum_{i=t-C\cdot M:C:t-C}$
$\min(ratio_{o,i}(\theta_o)A_{o,i}, \text{clip}(ratio_{o,i}(\theta_o), 1-\epsilon, 1+\epsilon)A_{o,i})$
$MSE(w_o) = \sum_{i=t-C\cdot M:C:t-C}(R_{o,i} - v_o(\phi_{o,i}; w_o))^2$
Update θ_o by gradient descent w.r.t. $JPPO(\theta_o)$ for N_p times
Update w_o by gradient descent w.r.t. $MSE(w_o)$ for N_v times
return θ_o, w_o

G. Overall Scheduling Framework of the HMAPPO

Note that the value networks of the proposed agents are only used in the training process for calculating the advantage functions. When applied to the practical scheduling process,

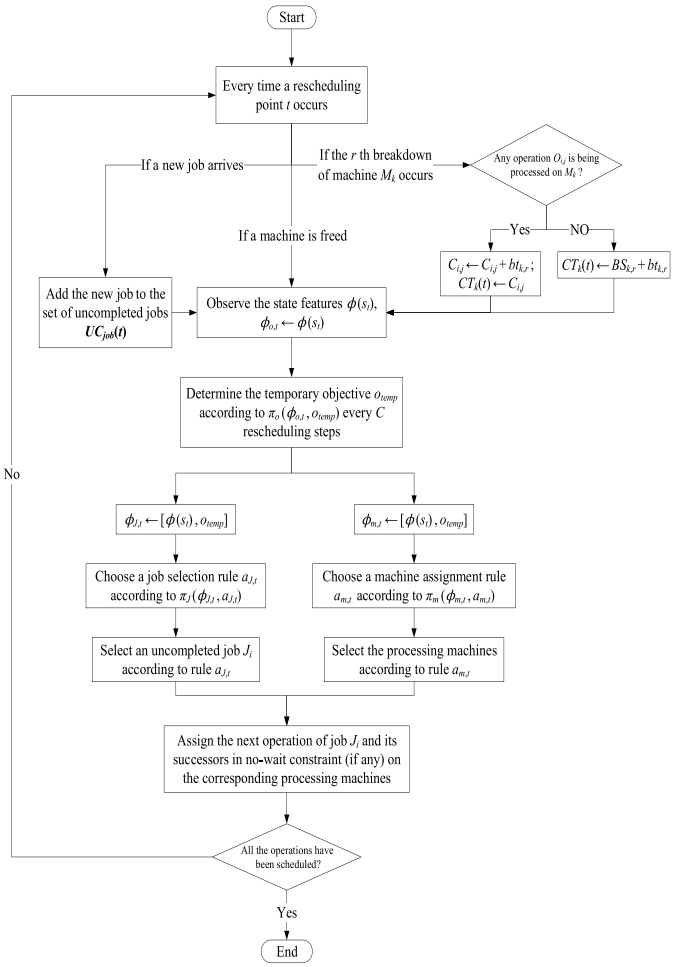


Fig. 3. Overall scheduling framework of the HMAPPO.

we directly use the trained policy networks π_o, π_J, and π_m to select the objectives and dispatching rules at each rescheduling point. The overall scheduling framework of HMAPPO in the implementation process is shown in Fig. 3.

VI. Numerical Experiments

A. Parameter Settings

In training and testing environments, the new jobs are assumed to arrive following Poisson distribution, that is, the interarrival time between two successive new job insertions is subjected to exponential distribution $\exp(1/\lambda_{\text{new}})$ with the mean value of λ_{new}. The due date D_i of job J_i is calculated by $D_i = A_i + (\sum_{j=1}^{n_i} \overline{t}_{i,j}) \cdot \text{DDT}$, where DDT is the due date tightness. A smaller DDT indicates that the job is more prone to be tardy. For each job, $\delta\%$ of its operations is randomly chosen and half of which are subjected to homogeneous no-wait constraint, while the other half of which are subjected to heterogeneous no-wait constraint. The interval time between two successive machine breakdowns follows an exponential distribution $\exp(1/\lambda_{\text{MTBF}})$ with the mean value of λ_{MTBF}. Furthermore, the repair time for a machine breakdown follows an exponential distribution $\exp(1/\lambda_{\text{MTTR}})$ with the mean value of λ_{MTTR}, which satisfies $\lambda_{\text{MTTR}} = \overline{p}$ and $((\lambda_{\text{MTTR}})/(\lambda_{\text{MTTR}} + \lambda_{\text{MTBF}})) = 0.05$ [48], where $\overline{p}$ is the mean processing time of an operation. The training and testing benchmarks used in this article are randomly generated according to the parameters given in Table I, where the symbols "randi" and "randf" denote uniform distribution of integers and real numbers, respectively.

TABLE I
Parameter Settings for Training and Testing Benchmarks

Parameter	Value
total number of machines (m)	randi$[10, 30]$
number of available machines per operation	randi$[1, m]$
number of initial jobs at beginning	randi$[1, 20]$
total number new inserted jobs	randi$[100, 150]$
number of operations per job	randi$[1, 40]$
number of successive operations subjected to homogeneous no-wait constraint	randi$[1, 5]$
urgency degree (Pr_i) of each job	randi$[1, 5]$
processing time of an operation on each available machine	randi$[1, 50]$
mean time between failure (λ_{MTBF})	475
mean time to repair (λ_{MTTR})	25
mean interarrival time of new jobs (λ_{new})	randi$[25, 100]$
due date tightness (DDT) of each job	randf$[0.5, 1.5]$
the percentage (δ) of operations per job subjected to no-wait constraint	randf$[0, 0.3]$

TABLE II
Parameter Settings of the Training Algorithm

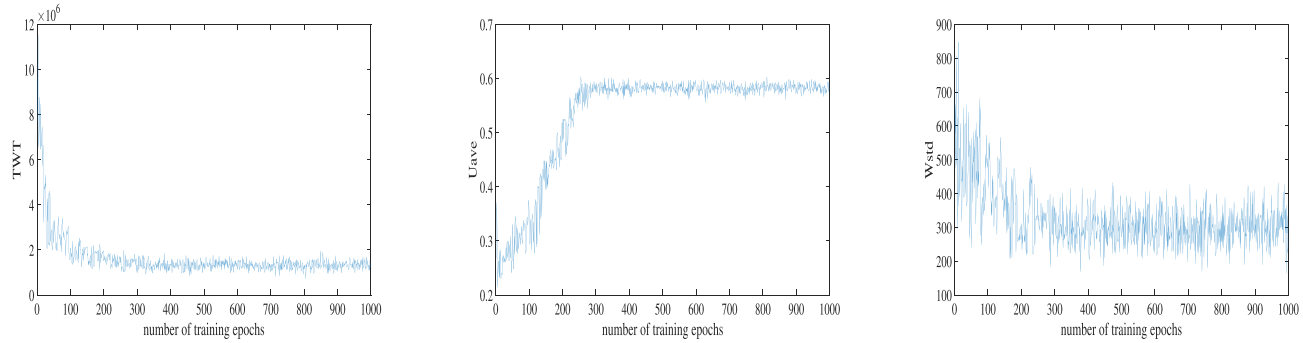
Parameter	Value
number of training epochs L	1000
minibatch size M	32
number of gradient descents for policy networks per update N_p	10
number of gradient descents for value networks per update N_v	10
step length C for higher objective adjustment	10
discount factor γ	0.9
clip parameter ϵ	0.2
optimizer	Adam

B. Details of Training Process

The proposed HMAPPO is implemented by python on a PC with Intel Core i7-6700 @ 4.0-GHz CPU and 8-GB RAM. The parameter settings for the training process are listed in Table II. The HMAPPO is trained for 1000 epochs where each epoch is conducted in a randomly generated environment to enhance its robustness and generality. After each training epoch, we test the trained HMAPPO on a validation instance where δ, m, and λ_{new} are, respectively, set as 0.2, 20, and 25. Fig. 4 illustrates the values of three investigated objectives per training epoch on the validation instance. It can be observed that all the objectives are continuously improved during the training process.

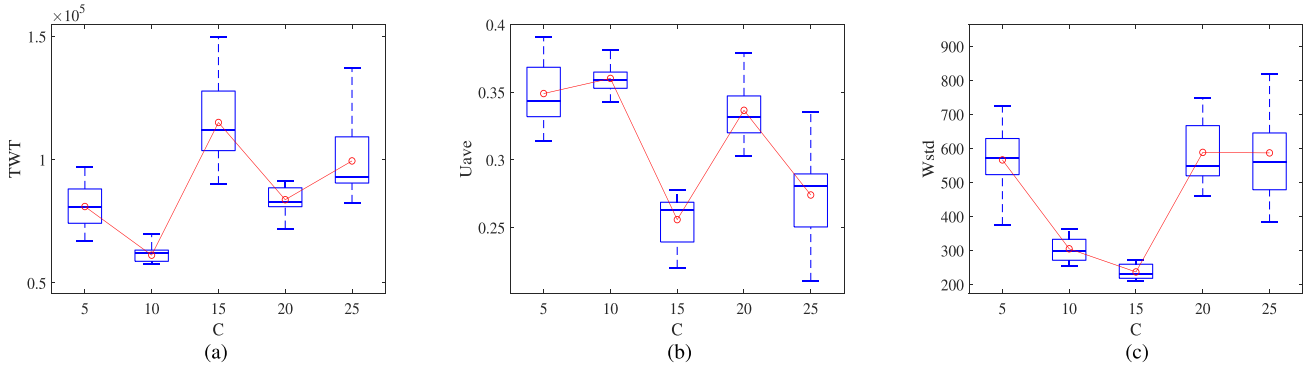
C. Performance Metrics

For the multiobjective optimization problem considered in this article, three performance metrics are used to evaluate the quality of obtained solutions in Pareto optimality including generational distance (GD) [49], inverse GD (IGD) [50], and spread (Δ) [51]. The GD and Δ are, respectively, the convergence and diversity metric. The IGD is a comprehensive metric evaluating both the convergence and diversity of the



(a) (b) (c)

Fig. 4. Values of three objectives per training epoch on the validation instance where δ, m, and λ_{new} are, respectively, set as 0.2, 20, and 25. (a) Value of TWT per training epoch. (b) Value of U_{ave} per training epoch. (c) Value of W_{std} per training epoch.



(a) (b) (c)

Fig. 5. Boxplots of three objectives, together with their average values marked as red dots, under different levels of C on the validation instance where δ, m, and λ_{new} are, respectively, set as 0.2, 20, and 50. (a) Boxplots of TWT. (b) Boxplots of U_{ave}. (c) Boxplots of W_{std}.

obtained solutions. The definitions of the three metrics are listed as follows.

1) Convergence Metric: GD:

$$\text{GD}(\boldsymbol{A}, \boldsymbol{P}) = \frac{\sqrt{\sum_{i=1}^{|\boldsymbol{A}|} d_{i,A,P}^2}}{|\boldsymbol{A}|} \tag{29}$$

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 minimum Euclidean distance of the ith solution in $\boldsymbol{A}$ to the solutions in $\boldsymbol{P}$.

2) Comprehensive Metric: IGD:

$$\text{IGD}(\boldsymbol{P}, \boldsymbol{A}) = \frac{1}{|\boldsymbol{P}|} \sum_{i=1}^{|\boldsymbol{P}|} d_{i,P,A} \tag{30}$$

where $d_{i,P,A}$ is the minimum Euclidean distance of the ith solution in $\boldsymbol{P}$ to the solutions in $\boldsymbol{A}$.

3) Diversity Metric: Spread (Δ):

$$\Delta = \frac{\sum_{j=1}^{n_o} d_{j,A,P}^e + \sum_{i=1}^{|\boldsymbol{A}|} |d_{i,A,A} - \overline{d}_{A,A}|}{\sum_{j=1}^{n_o} d_{j,A,P}^e + |\boldsymbol{A}| \cdot \overline{d}_{A,A}} \tag{31}$$

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 solutions of $\boldsymbol{A}$ and $\boldsymbol{P}$ in the sense of 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. Since the true Pareto-optimal front $\boldsymbol{P}$ is not known in advance, in our numerical experiments, all the Pareto-optimal solutions obtained by the HMAPPO and other comparative methods from multiple runs are merged at first and then the nondominated ones among which are taken as $\boldsymbol{P}$. This ensures that $\boldsymbol{P}$ can be regarded as a close approximation of the true Pareto-optimal front and is good enough for performance comparisons.

D. Sensitivity Study on the Control Parameter C

Note that the control parameter C, which determines the frequency of higher objective adjustment, plays an important role in affecting the overall performance of the proposed HMAPPO. A smaller value of C can quickly react to uncertain events and adjust the higher objective but may cause the schedule to be unstable in turn, and vice versa for a higher value of C. In order to choose a feasible value of C, we increase it from 5 to 25 with a step size of five. Under each parameter level, we test the trained HMAPPO independently for 20 times on a validation instance where δ, m, and λ_{new} are, respectively, set as 0.2, 20, and 50. The boxplots of three investigated objectives under different levels of C are shown in Fig. 5. The effects of C on three objectives are, respectively, analyzed using the analysis of variance (ANOVA) technique. The analysis results are given in Table III, which demonstrates

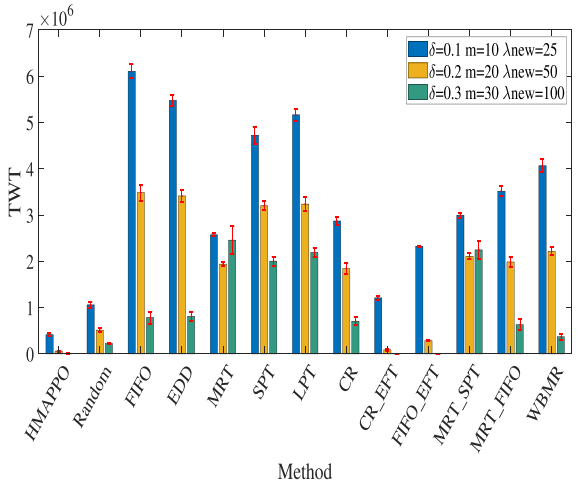


Fig. 6. Average values of TWT obtained by the HMAPPO and other dispatching rules on three representative instances.

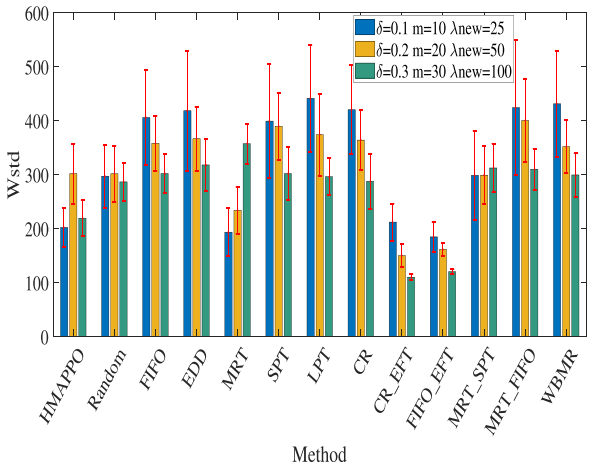


Fig. 8. Average values of W_{std} obtained by the HMAPPO and other dispatching rules on three representative instances.

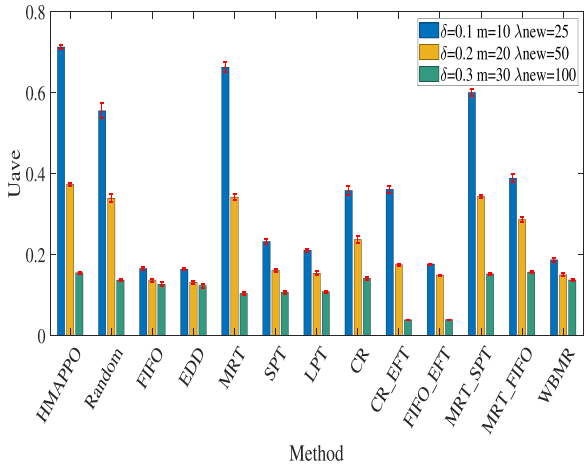


Fig. 7. Average values of U_{ave} obtained by the HMAPPO and other dispatching rules on three representative instances.

TABLE III
ANOVA RESULTS ON THREE INVESTIGATED OBJECTIVES FOR THE CONTROL PARAMETER C

Objective	Source	SS	df	MS	F	Prob>F
TWT	C	3.30e+10	4	8.24e+09	67.31	6.80e-27
	Error	1.16e+10	95	1.22e+08		
	Total	4.46e+10	99			
U_{ave}	C	1.77e-01	4	4.42e-02	93.89	3.82e-32
	Error	4.47e-02	95	4.75e-04		
	Total	2.21e-01	99			
W_{std}	C	2.35e+06	4	5.88e+05	74.89	1.57e-28
	Error	7.46e+05	95	7.85e+03		
	Total	3.10e+06	99			

that the parameter C has significant effects on all the objectives (p-value < 0.05). Meanwhile, it can be seen from Fig. 5 that, when $C = 10$, the trained HMAPPO can achieve the best compromise among three objectives compared with other parameter levels, which obtains the lowest TWT, the highest U_{ave}, and relatively lower W_{std}. Thus, it can be concluded that $C = 10$ is recommended in this article.

E. Comparisons With Well-Known Dispatching Rules

In order to confirm the effectiveness and superiority of the proposed HMAPPO, we take 11 well-known dispatching rules from previous studies [19], [20], [37], [52], [53] as comparisons, including FIFO, EDD, most remaining processing time (MRT), SPT, longest processing time (LPT), critical ratio (CR), combination of CR and EFT (CR_EFT), combination of FIFO and EFT (FIFO_EFT), combination of MRT and SPT (MRT_SPT), combination of MRT and FIFO (MRT_FIFO), and weight biased modified RRrule (WBMR). Meanwhile, in order to confirm if the proposed HMAPPO has learned a feasible rule selection strategy, we also take a random strategy into comparison, i.e., randomly choose a job selection rule and machine assignment rule at each rescheduling point. A wide range of test instances with different parameter settings of production environments, as shown in Table I, is used to verify the generality of the proposed HMAPPO. Each method is repeated 20 times on each test instance, the performance metrics of the final Pareto-optimal fronts obtained by the comparative methods are provided in Tables IV–VI with the best results highlighted in bold font.

First, compared with the random action selection strategy, the proposed HMAPPO can achieve better results for all metrics on almost all instances. This confirms that the HMAPPO has learned an efficient strategy to select the feasible objectives and dispatching rules at each rescheduling point. Next, compared with other well-known dispatching rules, the HMAPPO can also attain the best results for the GD and IGD metrics on 24 and 20 instances out of the 27 test instances. This indicates that the solutions obtained by the HMAPPO demonstrate the best convergence performance to the true Pareto-optimal front. For the diversity metric Δ, since there are three objectives to be investigated and different rules are suitable for different objectives, which naturally results in various distributions of solutions obtained by the comparative methods in the objective space, the HMAPPO cannot always achieve the best diversity performance for different test instances. However, the HMAPPO can still attain the most number of best results for Δ compared to other methods. In conclusion, the

TABLE IV

GD VALUES FOR THE PARETO-OPTIMAL FRONTS OBTAINED BY THE HMAPPO AND OTHER WELL-KNOWN DISPATCHING RULES

δ	m	λ_{new}	HMAPPO	Random	FIFO	EDD	MRT	SPT	LPT	CR	CR_EFT	FIFO_EFT	MRT_SPT	MRT_FIFO	WBMR
		25	**0.00e+00**	1.05e-01	3.08e-01	3.87e-01	**0.00e+00**	2.13e-01	3.33e-01	1.25e-01	1.35e-01	2.59e-01	5.47e-02	1.62e-01	3.25e-01
	10	50	**0.00e+00**	1.68e-01	1.83e-01	2.18e-01	1.10e-01	1.54e-01	2.62e-01	1.96e-01	1.65e-01	1.80e-01	1.15e-01	1.89e-01	1.80e-01
		100	**0.00e+00**	1.07e-01	5.05e-01	3.50e-01	1.11e-01	1.81e-01	2.12e-01	1.71e-01	**0.00e+00**	2.25e-01	1.55e-01	1.99e-01	2.01e-01
		25	**0.00e+00**	2.79e-01	1.40e-01	3.50e-01	**0.00e+00**	1.35e-01	5.02e-01	8.84e-01	1.15e-01	1.28e-01	2.11e-01	2.46e-01	1.53e-01
0.1	20	50	**0.00e+00**	2.01e-01	1.71e-01	2.03e-01	1.79e-01	1.04e-01	3.18e-01	2.45e-01	**0.00e+00**	9.96e-02	1.10e-01	1.64e-01	4.70e-01
		100	**0.00e+00**	9.57e-02	2.39e-01	1.43e-01	1.40e-01	1.57e-01	1.40e-01	1.78e-01	**0.00e+00**	**0.00e+00**	9.91e-02	8.28e-02	4.30e-01
		25	**0.00e+00**	1.74e-01	1.83e-01	1.82e-01	1.15e-01	3.50e-01	1.82e-01	2.56e-01	**0.00e+00**	1.72e-01	1.29e-01	2.04e-01	2.59e-01
	30	50	**0.00e+00**	1.24e-01	1.99e-01	1.98e-01	3.48e-01	2.63e-01	1.10e-01	1.64e-01	**0.00e+00**	1.32e-01	1.57e-01	2.37e-01	2.27e-01
		100	8.76e-02	9.67e-02	9.92e-02	1.45e-01	1.25e-01	2.47e-01	1.96e-01	1.71e-01	1.05e-01	2.59e-01	1.10e-01	1.78e-01	**0.00e+00**
		25	**0.00e+00**	**0.00e+00**	2.70e-01	2.71e-01	**0.00e+00**	1.35e-01	3.43e-01	3.15e-01	3.06e-01	9.80e-01	2.17e-01	1.40e-01	2.13e-01
	10	50	**0.00e+00**	1.53e-01	1.39e-01	3.50e-01	**0.00e+00**	2.12e-01	2.59e-01	1.65e-01	1.02e-01	3.32e-01	1.43e-01	1.86e-01	1.21e-01
		100	**0.00e+00**	1.10e-01	2.65e-01	1.24e-01	8.67e-02	1.52e-01	1.11e-01	1.68e-01	**0.00e+00**	1.15e-01	1.08e-01	3.13e-01	2.11e-01
		25	**0.00e+00**	3.97e-01	3.54e-01	3.52e-01	**0.00e+00**	5.01e-01	1.78e-01	1.44e-01	9.05e-02	2.38e-01	2.19e-01	1.60e-01	1.53e-01
0.2	20	50	**0.00e+00**	1.95e-02	4.63e-01	5.43e-01	**0.00e+00**	2.71e-01	3.36e-01	1.64e-01	**0.00e+00**	6.03e-02	3.74e-02	1.54e-01	2.31e-01
		100	5.44e-02	9.36e-02	3.83e-01	1.82e-01	1.58e-01	2.43e-01	1.29e-01	1.39e-01	1.27e-01	**0.00e+00**	1.18e-01	1.01e-01	**0.00e+00**
		25	**0.00e+00**	1.83e-01	5.17e-01	2.17e-01	1.24e-01	1.80e-01	1.57e-01	3.30e-01	**0.00e+00**	**0.00e+00**	2.31e-01	1.33e-01	1.76e-01
	30	50	**0.00e+00**	1.43e-01	1.99e-01	4.65e-01	2.67e-01	1.55e-01	2.15e-01	1.92e-01	**0.00e+00**	1.94e-01	2.65e-01	1.69e-01	2.27e-01
		100	**0.00e+00**	1.19e-01	9.17e-02	1.05e-01	1.47e-01	1.47e-01	1.45e-01	7.50e-02	1.17e-01	1.80e-01	2.36e-01	9.29e-02	7.58e-02
		25	**0.00e+00**	1.55e-01	3.54e-01	2.69e-01	**0.00e+00**	3.35e-01	2.10e-01	1.42e-01	3.11e-01	2.14e-01	1.27e-01	2.40e-01	2.13e-01
	10	50	**0.00e+00**	1.44e-01	2.14e-01	1.38e-01	1.20e-01	1.51e-01	3.40e-01	2.48e-01	1.66e-01	2.09e-01	1.56e-01	3.05e-01	1.19e-01
		100	**0.00e+00**	8.38e-02	3.30e-01	1.50e-01	1.01e-01	1.70e-01	1.73e-01	1.40e-01	**0.00e+00**	6.52e-02	1.59e-01	1.42e-01	1.01e-01
		25	**0.00e+00**	1.62e-01	3.51e-01	1.83e-01	1.42e-01	3.40e-01	1.21e-01	4.73e-01	9.36e-02	2.45e-01	1.52e-01	1.64e-01	1.54e-01
0.3	20	50	**0.00e+00**	9.29e-02	2.12e-01	1.77e-01	1.01e-01	3.51e-01	1.23e-01	1.28e-01	**0.00e+00**	7.10e-02	1.36e-01	1.96e-01	1.30e-01
		100	**0.00e+00**	8.18e-02	1.47e-01	2.87e-01	9.81e-02	3.24e-01	2.53e-01	2.90e-01	1.08e-01	1.19e-01	2.50e-01	8.27e-02	9.44e-02
		25	**0.00e+00**	**0.00e+00**	1.52e-01	4.82e-01	1.44e-01	2.09e-01	9.33e-01	1.60e-01	7.05e-02	**0.00e+00**	1.43e-01	1.84e-01	3.12e-01
	30	50	**0.00e+00**	1.01e-01	2.47e-01	1.46e-01	1.87e-01	2.47e-01	2.44e-01	4.71e-01	**0.00e+00**	8.98e-02	1.08e-01	4.64e-01	1.64e-01
		100	5.84e-02	7.65e-02	1.25e-01	1.50e-01	9.32e-02	1.95e-01	1.89e-01	1.44e-01	**0.00e+00**	1.30e-01	1.87e-01	8.60e-02	6.56e-02

TABLE V

IGD VALUES FOR THE PARETO-OPTIMAL FRONTS OBTAINED BY THE HMAPPO AND OTHER WELL-KNOWN DISPATCHING RULES

δ	m	λ_{new}	HMAPPO	Random	FIFO	EDD	MRT	SPT	LPT	CR	CR_EFT	FIFO_EFT	MRT_SPT	MRT_FIFO	WBMR
		25	**1.59e-01**	1.94e-01	1.15e+00	1.19e+00	1.76e-01	8.44e-01	9.20e-01	3.76e-01	3.07e-01	8.47e-01	2.62e-01	4.74e-01	9.14e-01
	10	50	**1.38e-01**	1.97e-01	1.26e+00	1.25e+00	4.19e-01	9.45e-01	1.00e+00	5.14e-01	4.32e-01	9.03e-01	4.88e-01	4.57e-01	8.88e-01
		100	3.55e-01	3.56e-01	9.30e-01	1.05e+00	9.89e-01	1.01e+00	9.38e-01	5.53e-01	**3.44e-01**	5.23e-01	8.71e-01	5.44e-01	6.37e-01
		25	**2.62e-01**	3.04e-01	1.11e+00	9.85e-01	2.95e-01	8.13e-01	8.10e-01	3.94e-01	3.36e-01	7.48e-01	3.46e-01	4.28e-01	7.72e-01
0.1	20	50	**4.75e-01**	6.38e-01	7.19e-01	6.63e-01	9.13e-01	8.42e-01	9.73e-01	6.60e-01	4.98e-01	5.63e-01	8.77e-01	6.97e-01	6.42e-01
		100	**4.90e-01**	5.16e-01	6.15e-01	6.60e-01	7.29e-01	7.86e-01	8.54e-01	5.52e-01	5.54e-01	5.33e-01	7.12e-01	5.11e-01	5.37e-01
		25	4.96e-01	4.75e-01	1.06e+00	1.02e+00	8.15e-01	9.45e-01	1.02e+00	7.68e-01	**3.39e-01**	4.27e-01	7.67e-01	8.28e-01	7.57e-01
	30	50	**2.52e-01**	2.77e-01	5.43e-01	5.53e-01	9.75e-01	8.96e-01	9.19e-01	4.80e-01	7.84e-01	8.12e-01	1.01e+00	4.77e-01	3.46e-01
		100	**1.52e-01**	1.72e-01	1.89e-01	1.99e-01	4.45e-01	6.34e-01	7.46e-01	2.15e-01	1.03e+00	1.01e+00	3.33e-01	1.73e-01	1.76e-01
		25	2.33e-01	2.44e-01	1.15e+00	1.10e+00	**1.51e-01**	8.85e-01	8.59e-01	4.41e-01	3.36e-01	8.63e-01	3.04e-01	4.40e-01	8.55e-01
	10	50	**2.44e-01**	3.62e-01	1.13e+00	1.06e+00	4.09e-01	9.25e-01	9.31e-01	5.23e-01	3.96e-01	8.78e-01	4.40e-01	3.66e-01	9.12e-01
		100	**1.99e-01**	3.20e-01	1.00e+00	1.14e+00	4.02e-01	8.87e-01	1.03e+00	4.97e-01	4.23e-01	8.18e-01	5.08e-01	3.45e-01	8.04e-01
		25	**2.15e-01**	2.52e-01	1.06e+00	1.11e+00	3.36e-01	8.38e-01	9.13e-01	5.47e-01	4.06e-01	6.66e-01	4.50e-01	5.72e-01	7.95e-01
0.2	20	50	**3.02e-01**	3.50e-01	1.01e+00	1.04e+00	5.02e-01	8.83e-01	8.96e-01	5.00e-01	3.57e-01	5.30e-01	5.49e-01	5.20e-01	7.46e-01
		100	**4.08e-01**	4.11e-01	4.74e-01	4.50e-01	6.36e-01	8.24e-01	8.56e-01	4.38e-01	6.06e-01	6.21e-01	5.65e-01	4.17e-01	4.15e-01
		25	**3.55e-01**	4.98e-01	1.04e+00	1.03e+00	6.77e-01	9.38e-01	1.01e+00	7.37e-01	4.19e-01	4.82e-01	7.34e-01	8.81e-01	7.99e-01
	30	50	5.41e-01	5.70e-01	7.30e-01	6.68e-01	1.23e+00	9.93e-01	1.07e+00	6.99e-01	**4.63e-01**	5.19e-01	1.15e+00	8.31e-01	5.60e-01
		100	**4.49e-01**	4.86e-01	4.89e-01	4.97e-01	6.59e-01	7.29e-01	6.92e-01	5.13e-01	5.72e-01	6.28e-01	6.56e-01	4.88e-01	4.84e-01
		25	2.51e-01	3.26e-01	1.13e+00	1.15e+00	**2.01e-01**	7.37e-01	8.18e-01	4.32e-01	3.99e-01	8.90e-01	3.54e-01	4.08e-01	8.62e-01
	10	50	**2.93e-01**	3.94e-01	1.08e+00	1.11e+00	5.43e-01	8.03e-01	8.94e-01	5.59e-01	3.78e-01	7.22e-01	5.53e-01	4.77e-01	7.67e-01
		100	5.25e-01	5.83e-01	8.61e-01	9.06e-01	7.81e-01	7.64e-01	9.59e-01	5.77e-01	**5.08e-01**	5.15e-01	7.11e-01	5.73e-01	5.59e-01
		25	**3.81e-01**	4.11e-01	1.11e+00	1.10e+00	3.90e-01	8.83e-01	9.64e-01	5.12e-01	3.91e-01	6.94e-01	4.93e-01	5.28e-01	8.45e-01
0.3	20	50	**4.13e-01**	4.98e-01	8.33e-01	8.01e-01	1.03e+00	9.53e-01	9.12e-01	6.31e-01	4.99e-01	4.56e-01	1.03e+00	6.74e-01	5.88e-01
		100	**4.60e-01**	5.41e-01	6.02e-01	6.20e-01	7.63e-01	7.59e-01	8.32e-01	6.08e-01	4.80e-01	5.16e-01	7.83e-01	5.75e-01	5.72e-01
		25	6.44e-01	7.04e-01	9.73e-01	8.52e-01	7.80e-01	9.94e-01	8.79e-01	6.97e-01	4.52e-01	**4.40e-01**	8.00e-01	7.17e-01	5.65e-01
	30	50	**4.52e-01**	4.59e-01	6.61e-01	7.05e-01	8.18e-01	9.62e-01	9.12e-01	6.61e-01	6.97e-01	7.07e-01	8.38e-01	6.50e-01	5.69e-01
		100	**3.00e-01**	4.31e-01	3.11e-01	3.34e-01	6.70e-01	9.18e-01	8.14e-01	3.52e-01	8.09e-01	8.47e-01	6.80e-01	3.49e-01	3.03e-01

proposed HMAPPO can intelligently determine the feasible objectives and select the most appropriate rules to achieve them at each rescheduling point; hence, a good compromise among different objectives can be made in the long-term schedule and due to which the HMAPPO is superior to other methods.

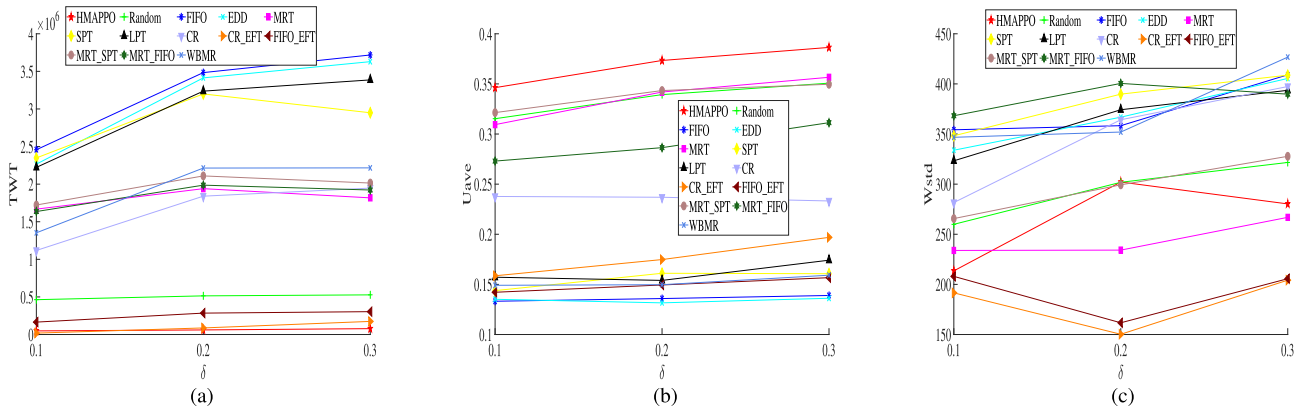


Fig. 9. Average values of three objectives obtained by the HMAPPO and other dispatching rules under different levels of δ when $m = 20$ and $\lambda_{\text{new}} = 50$. (a) TWT under different levels of δ. (b) U_{ave} under different levels of δ. (c) W_{std} under different levels of δ.

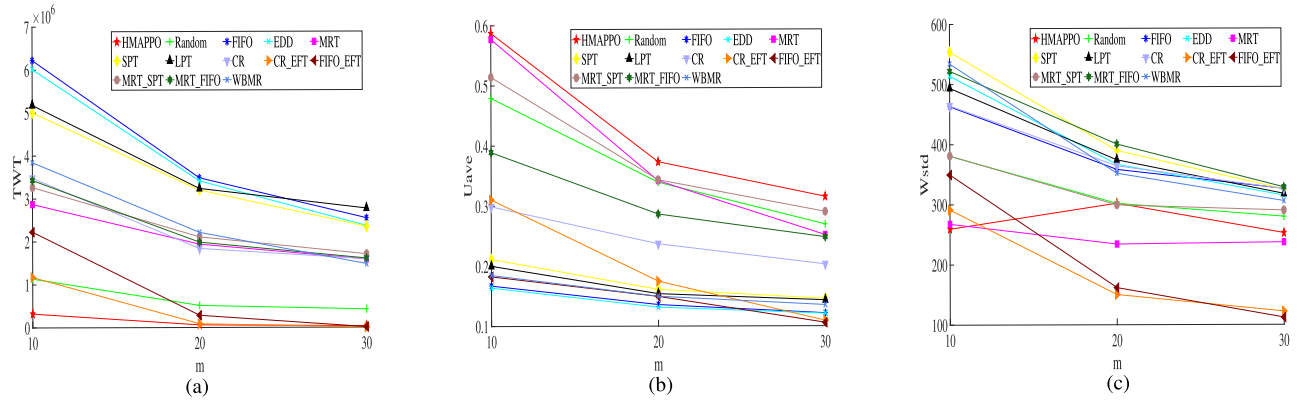


Fig. 10. Average values of three objectives obtained by the HMAPPO and other dispatching rules under different levels of m when $\delta = 0.2$ and $\lambda_{\text{new}} = 50$. (a) TWT under different levels of m. (b) U_{ave} under different levels of m. (c) W_{std} under different levels of m.

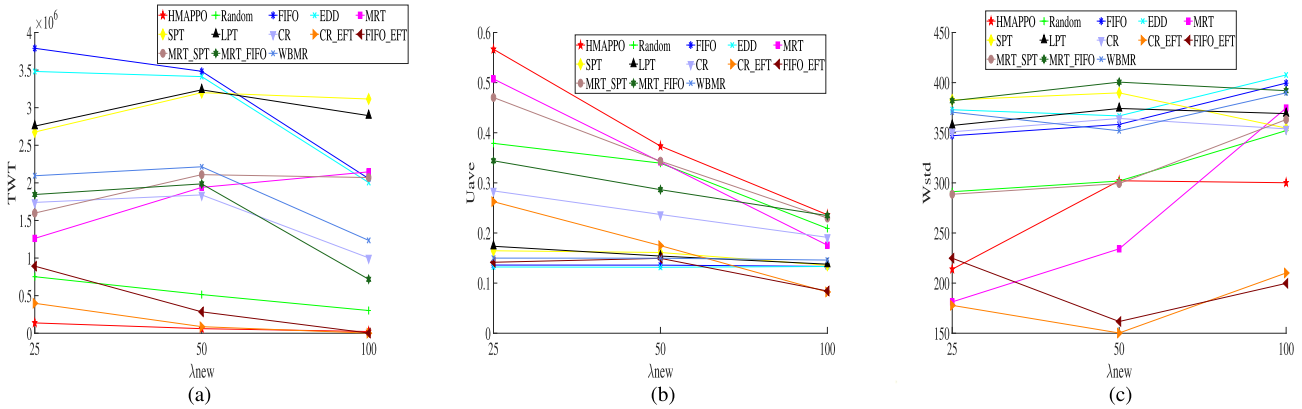


Fig. 11. Average values of three objectives obtained by the HMAPPO and other dispatching rules under different levels of λ_{new} when $\delta = 0.2$ and $m = 20$. (a) TWT under different levels of λ_{new}. (b) U_{ave} under different levels of λ_{new}. (c) W_{std} under different levels of λ_{new}.

Figs. 6–8 show the average values of three objectives from 20 runs obtained by the HMAPPO and other dispatching rules on three representative instances. It can be seen that the proposed HMAPPO can obtain the best compromise among three investigated objectives, which achieves the smallest TWT and highest U_{ave} compared with other dispatching rules. Although the CR_EFT and FIFO_EFT can obtain relatively smaller values in W_{std}, their performances in U_{ave} are much worse than that of the HMAPPO. This further confirms that a single rule can never guarantee the best performance for all objectives in all situations. Meanwhile, the necessity and effectiveness of using the HMAPPO to choose the feasible objectives and dispatching rules at different rescheduling points can also be validated.

Figs. 9–11 show the changing trends of three objectives under different parameter levels of δ, m, and λ_{new}. It can

TABLE VI

Δ VALUES FOR THE PARETO-OPTIMAL FRONTS OBTAINED BY THE HMAPPO AND OTHER WELL-KNOWN DISPATCHING RULES

δ	m	λ_{new}	HMAPPO	Random	FIFO	EDD	MRT	SPT	LPT	CR	CR_EFT	FIFO_EFT	MRT_SPT	MRT_FIFO	WBMR
0.1	10	25	**8.66e-01**	8.75e-01	9.93e-01	9.86e-01	9.23e-01	9.82e-01	9.75e-01	9.67e-01	9.79e-01	9.99e-01	9.92e-01	9.82e-01	9.87e-01
		50	**8.73e-01**	9.77e-01	9.70e-01	9.83e-01	9.75e-01	9.72e-01	9.83e-01	9.67e-01	9.17e-01	9.88e-01	9.75e-01	9.77e-01	9.56e-01
		100	9.43e-01	9.40e-01	9.14e-01	8.98e-01	9.72e-01	9.60e-01	9.31e-01	9.57e-01	**8.61e-01**	9.65e-01	9.78e-01	9.74e-01	9.33e-01
	20	25	**8.34e-01**	9.95e-01	9.47e-01	9.22e-01	9.74e-01	9.63e-01	9.11e-01	9.40e-01	9.41e-01	9.79e-01	9.77e-01	9.02e-01	9.86e-01
		50	9.81e-01	9.96e-01	9.67e-01	9.81e-01	9.78e-01	9.37e-01	9.61e-01	9.96e-01	**9.33e-01**	9.83e-01	9.62e-01	9.54e-01	9.75e-01
		100	9.56e-01	9.82e-01	1.00e+00	9.19e-01	9.31e-01	9.61e-01	9.59e-01	9.43e-01	9.71e-01	9.91e-01	9.75e-01	9.60e-01	**9.02e-01**
	30	25	9.50e-01	9.68e-01	9.33e-01	9.28e-01	9.87e-01	9.38e-01	9.40e-01	**9.07e-01**	9.26e-01	9.86e-01	9.81e-01	9.76e-01	9.61e-01
		50	9.97e-01	9.70e-01	9.54e-01	9.82e-01	9.84e-01	9.88e-01	9.66e-01	9.70e-01	9.95e-01	9.92e-01	9.85e-01	**9.44e-01**	9.52e-01
		100	**9.04e-01**	9.91e-01	9.92e-01	9.90e-01	9.96e-01	9.46e-01	9.52e-01	9.72e-01	9.67e-01	9.96e-01	9.12e-01	9.73e-01	9.75e-01
0.2	10	25	9.61e-01	9.76e-01	9.82e-01	9.86e-01	9.18e-01	9.74e-01	9.18e-01	**8.99e-01**	9.54e-01	9.15e-01	9.25e-01	9.56e-01	1.01e+00
		50	**8.96e-01**	9.75e-01	9.58e-01	9.56e-01	9.26e-01	9.46e-01	9.50e-01	9.58e-01	9.49e-01	9.95e-01	9.99e-01	9.63e-01	9.55e-01
		100	**8.86e-01**	9.46e-01	9.63e-01	9.38e-01	9.43e-01	9.38e-01	9.50e-01	9.00e-01	8.87e-01	9.87e-01	9.24e-01	9.33e-01	9.53e-01
	20	25	9.28e-01	**8.04e-01**	9.48e-01	9.82e-01	9.41e-01	9.51e-01	9.12e-01	9.71e-01	9.38e-01	9.89e-01	9.44e-01	9.45e-01	9.84e-01
		50	**9.03e-01**	9.53e-01	9.92e-01	9.46e-01	9.22e-01	9.67e-01	9.75e-01	9.84e-01	9.83e-01	9.79e-01	9.75e-01	9.12e-01	9.45e-01
		100	9.59e-01	9.77e-01	9.75e-01	**9.33e-01**	9.85e-01	9.70e-01	9.64e-01	9.59e-01	9.58e-01	9.87e-01	9.78e-01	9.72e-01	9.74e-01
	30	25	**9.03e-01**	9.50e-01	9.31e-01	9.68e-01	9.71e-01	9.56e-01	9.12e-01	9.48e-01	9.08e-01	9.22e-01	9.75e-01	9.58e-01	9.35e-01
		50	9.63e-01	9.88e-01	9.83e-01	**9.02e-01**	9.81e-01	9.65e-01	9.75e-01	9.46e-01	9.77e-01	9.72e-01	9.79e-01	9.57e-01	9.59e-01
		100	9.74e-01	9.83e-01	9.83e-01	9.90e-01	**9.64e-01**	9.65e-01	9.69e-01	9.76e-01	9.94e-01	9.96e-01	9.79e-01	9.87e-01	9.96e-01
0.3	10	25	9.44e-01	9.93e-01	9.61e-01	9.67e-01	**8.22e-01**	9.73e-01	9.80e-01	8.87e-01	9.43e-01	9.87e-01	9.76e-01	9.54e-01	9.84e-01
		50	**9.00e-01**	9.78e-01	9.79e-01	9.50e-01	9.24e-01	9.77e-01	9.59e-01	9.64e-01	9.63e-01	9.77e-01	9.28e-01	9.29e-01	9.37e-01
		100	9.38e-01	9.16e-01	9.63e-01	9.53e-01	9.05e-01	9.27e-01	9.41e-01	8.92e-01	9.64e-01	9.83e-01	9.22e-01	9.46e-01	**8.70e-01**
	20	25	**8.27e-01**	9.36e-01	9.89e-01	9.54e-01	9.45e-01	9.77e-01	9.52e-01	9.07e-01	9.43e-01	9.69e-01	9.85e-01	9.10e-01	9.53e-01
		50	9.39e-01	9.47e-01	9.67e-01	1.00e+00	9.69e-01	9.70e-01	9.46e-01	9.38e-01	9.83e-01	9.90e-01	9.71e-01	9.04e-01	**9.02e-01**
		100	**8.75e-01**	9.79e-01	9.77e-01	9.59e-01	9.40e-01	9.30e-01	9.68e-01	9.64e-01	9.77e-01	9.99e-01	9.49e-01	9.65e-01	9.80e-01
	30	25	9.64e-01	9.71e-01	**9.16e-01**	9.31e-01	9.58e-01	9.65e-01	9.68e-01	9.52e-01	9.85e-01	9.57e-01	9.92e-01	9.42e-01	9.61e-01
		50	**9.41e-01**	9.78e-01	9.46e-01	9.88e-01	9.70e-01	9.45e-01	9.75e-01	9.82e-01	9.84e-01	9.82e-01	9.83e-01	9.84e-01	9.86e-01
		100	9.75e-01	9.87e-01	9.97e-01	9.87e-01	9.84e-01	9.82e-01	9.99e-01	9.79e-01	9.94e-01	9.89e-01	9.74e-01	**9.63e-01**	9.85e-01

be observed that, for most methods, all the three objectives increase with the increment of δ and decrease with the increment of m. The TWT and U_{ave} decrease, while the W_{std} increases with the increment of λ_{new}. Meanwhile, the proposed HMAPPO achieves the best performance in TWT and U_{ave} under all parameter levels.

Fig. 12 illustrates the Pareto-optimal fronts obtained by different methods for two representative instances, from which we can see that the HMAPPO demonstrates the best convergence to the true Pareto-optimal front.

F. Comparisons With Existing Real-Time Scheduling Methods

In order to further verify the superiority of the HMAPPO over existing dynamic scheduling methods, three recently proposed real-time scheduling methods for solving the DFJSP are reimplemented and taken as comparisons, including the heterogeneous EFT (HEFT) algorithm [54], the GRASP [31], and the double DQN (DDQN)-based scheduling approach [41]. Specifically, to validate that the trained HMAPPO can be well generalized to larger-size problems, we set the number of new inserted jobs as 1000 compared to these real-time scheduling methods, while the other parameter settings are set the same as those in Table I.

Each method is repeated for 20 runs on each test instance; the performance metrics obtained by the comparative methods are provided in Tables VII–IX with the best results highlighted in bold font. It can be seen that the proposed HMAPPO can achieve the best performance on most instances. Note that all the four comparative methods can obtain good results in GD metric; this may be due to that different methods are suitable for different objectives; thus, each of them demonstrates the best convergence performance in the corresponding objective. For example, the HMAPPO and GRASP can attain

TABLE VII

GD VALUES FOR THE PARETO-OPTIMAL FRONTS OBTAINED BY THE HMAPPO AND OTHER REAL-TIME SCHEDULING METHODS

δ	m	λ_{new}	HMAPPO	HEFT	GRASP	DDQN
0.1	10	25	**0.00e+00**	2.15e-01	**0.00e+00**	1.20e-01
		50	2.02e-01	1.73e-01	**0.00e+00**	1.44e-01
		100	**0.00e+00**	7.84e-02	**0.00e+00**	**0.00e+00**
	20	25	**0.00e+00**	1.11e-01	**0.00e+00**	**0.00e+00**
		50	**0.00e+00**	1.45e-01	**0.00e+00**	**0.00e+00**
		100	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
	30	25	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
		50	**0.00e+00**	6.34e-02	**0.00e+00**	**0.00e+00**
		100	**0.00e+00**	**0.00e+00**	1.09e-01	**0.00e+00**
0.2	10	25	**0.00e+00**	1.47e-01	**0.00e+00**	**0.00e+00**
		50	**0.00e+00**	1.02e-01	7.17e-02	**0.00e+00**
		100	**0.00e+00**	1.11e-01	9.21e-02	**0.00e+00**
	20	25	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
		50	**0.00e+00**	9.93e-02	**0.00e+00**	**0.00e+00**
		100	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
	30	25	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
		50	**0.00e+00**	**0.00e+00**	1.22e-01	1.09e-01
		100	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
0.3	10	25	**0.00e+00**	1.41e-01	1.30e-01	1.33e-01
		50	**0.00e+00**	3.07e-01	**0.00e+00**	1.39e-01
		100	**0.00e+00**	8.04e-02	6.59e-02	8.25e-02
	20	25	**0.00e+00**	1.00e-01	**0.00e+00**	**0.00e+00**
		50	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
		100	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
	30	25	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
		50	**0.00e+00**	1.32e-01	9.05e-02	1.16e-01
		100	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**

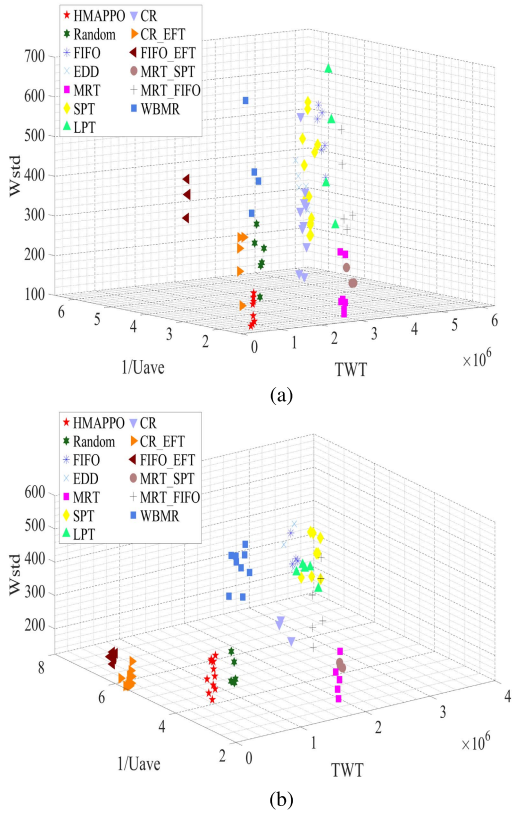


Fig. 12. Pareto-optimal fronts obtained by the HMAPPO and other dispatching rules for two representative instances. (a) $\delta = 0.1$, $m = 10$, and $\lambda_{\text{new}} = 25$. (b) $\delta = 0.2$, $m = 20$, and $\lambda_{\text{new}} = 50$.

TABLE VIII

IGD VALUES FOR THE PARETO-OPTIMAL FRONTS OBTAINED BY THE HMAPPO AND OTHER REAL-TIME SCHEDULING METHODS

δ	m	λ_{new}	HMAPPO	HEFT	GRASP	DDQN
0.1	10	25	**3.08e-01**	1.11e+00	3.11e-01	3.63e-01
		50	6.24e-01	7.20e-01	**4.61e-01**	5.97e-01
		100	**4.69e-01**	8.88e-01	7.50e-01	5.12e-01
	20	25	**4.57e-01**	1.05e+00	6.25e-01	5.34e-01
		50	5.82e-01	8.95e-01	**5.68e-01**	6.91e-01
		100	**4.16e-01**	6.46e-01	8.06e-01	6.80e-01
	30	25	**6.54e-01**	7.68e-01	7.63e-01	6.90e-01
		50	6.76e-01	8.97e-01	7.08e-01	**5.86e-01**
		100	**5.31e-01**	6.45e-01	7.96e-01	7.85e-01
0.2	10	25	4.01e-01	8.37e-01	3.49e-01	**2.89e-01**
		50	**2.90e-01**	1.08e+00	6.70e-01	3.42e-01
		100	**3.87e-01**	8.76e-01	5.89e-01	4.75e-01
	20	25	**5.52e-01**	8.16e-01	8.36e-01	5.63e-01
		50	**5.45e-01**	7.28e-01	7.69e-01	7.22e-01
		100	**5.21e-01**	6.97e-01	8.52e-01	6.28e-01
	30	25	**6.36e-01**	8.76e-01	6.79e-01	6.72e-01
		50	**5.97e-01**	6.40e-01	8.79e-01	7.04e-01
		100	5.75e-01	**4.64e-01**	5.80e-01	8.92e-01
0.3	10	25	**5.11e-01**	9.65e-01	5.27e-01	6.01e-01
		50	4.66e-01	1.08e+00	**2.98e-01**	5.93e-01
		100	**3.28e-01**	8.41e-01	5.88e-01	3.45e-01
	20	25	4.17e-01	9.27e-01	8.37e-01	**4.14e-01**
		50	**4.26e-01**	4.84e-01	6.15e-01	5.75e-01
		100	6.43e-01	**5.96e-01**	6.59e-01	7.02e-01
	30	25	**6.93e-01**	7.20e-01	7.02e-01	8.07e-01
		50	**3.79e-01**	5.65e-01	8.75e-01	5.83e-01
		100	6.06e-01	**5.22e-01**	6.51e-01	7.32e-01

lower TWT, and the DDQN can attain higher U_{ave}, while the HEFT can attain lower W_{std}. However, the proposed

TABLE IX

Δ VALUES FOR THE PARETO-OPTIMAL FRONTS OBTAINED BY THE HMAPPO AND OTHER REAL-TIME SCHEDULING METHODS

δ	m	λ_{new}	HMAPPO	HEFT	GRASP	DDQN
0.1	10	25	**6.82e-01**	9.84e-01	7.99e-01	6.85e-01
		50	9.12e-01	9.03e-01	**6.43e-01**	8.57e-01
		100	8.83e-01	9.65e-01	9.74e-01	**8.47e-01**
	20	25	8.72e-01	9.53e-01	9.56e-01	**8.03e-01**
		50	**8.95e-01**	9.90e-01	9.23e-01	9.49e-01
		100	**9.12e-01**	9.87e-01	9.87e-01	9.17e-01
	30	25	9.54e-01	9.80e-01	9.57e-01	**8.91e-01**
		50	**8.77e-01**	9.67e-01	9.76e-01	8.87e-01
		100	**9.10e-01**	9.66e-01	9.72e-01	9.39e-01
0.2	10	25	1.03e+00	9.50e-01	7.41e-01	**6.87e-01**
		50	9.42e-01	9.75e-01	9.01e-01	**6.93e-01**
		100	**8.41e-01**	9.22e-01	9.18e-01	8.54e-01
	20	25	**8.81e-01**	9.82e-01	9.37e-01	8.97e-01
		50	**8.41e-01**	1.00e+00	9.09e-01	8.46e-01
		100	**9.29e-01**	9.82e-01	1.00e+00	9.38e-01
	30	25	**9.07e-01**	9.64e-01	9.22e-01	9.11e-01
		50	9.20e-01	9.45e-01	9.88e-01	**9.06e-01**
		100	**9.07e-01**	9.15e-01	9.60e-01	9.89e-01
0.3	10	25	**5.52e-01**	8.19e-01	6.43e-01	8.93e-01
		50	**6.20e-01**	9.54e-01	7.40e-01	8.73e-01
		100	9.34e-01	9.26e-01	**8.33e-01**	9.18e-01
	20	25	9.40e-01	9.82e-01	9.15e-01	**8.26e-01**
		50	**8.62e-01**	9.94e-01	9.77e-01	9.02e-01
		100	**7.47e-01**	7.85e-01	9.70e-01	9.44e-01
	30	25	9.50e-01	9.72e-01	**9.24e-01**	9.26e-01
		50	9.33e-01	9.84e-01	9.92e-01	**8.99e-01**
		100	**8.41e-01**	8.88e-01	9.77e-01	9.69e-01

TABLE X

AVERAGE COMPUTING TIME (S) FOR THE HMAPPO AND OTHER REAL-TIME SCHEDULING METHODS TO MAKE A DECISION AT EACH RESCHEDULING POINT

δ	m	λ_{new}	HMAPPO	HEFT	GRASP	DDQN
0.1	10	25	3.83e-01	2.51e-01	**6.37e-02**	2.28e-01
0.1	20	100	3.08e-02	3.92e-03	**2.31e-03**	7.83e-03
0.1	30	50	7.41e-02	7.04e-03	**2.85e-03**	1.61e-02
0.2	10	100	4.44e-02	1.69e-02	**3.32e-03**	4.09e-02
0.2	20	50	9.49e-02	2.01e-02	**3.88e-03**	3.85e-02
0.2	30	25	2.64e-01	1.26e-01	**7.82e-03**	1.33e-01
0.3	10	50	2.08e-01	1.42e-01	**1.13e-02**	1.49e-01
0.3	20	25	2.81e-01	1.62e-01	**1.26e-02**	1.55e-01
0.3	30	100	3.21e-02	3.15e-03	**2.14e-03**	1.27e-02

HMAPPO outperforms all the other methods in the IGD and Δ metrics, which indicates that the HMAPPO achieves the best compromise among the three investigated objectives.

Fig. 13 shows the average values of three objectives from 20 runs obtained by the HMAPPO and other real-time scheduling methods on three representative instances, from which we can see that the proposed HMAPPO achieves the lowest TWT and relatively higher U_{ave} compared with other methods. Despite that the HEFT can achieve the lowest W_{std}, its performance in TWT is much worse than other methods.

Table X gives the average computing times of the comparative methods for making a decision (i.e., select a job and assign the next operation of which together with its successors in no-wait constraint on the corresponding processing machines) at each rescheduling point on some representative instances. It can be observed that each method can make a decision within 1 s; thus, the requirement of real-time scheduling can be satisfied. The proposed HMAPPO consumes a little

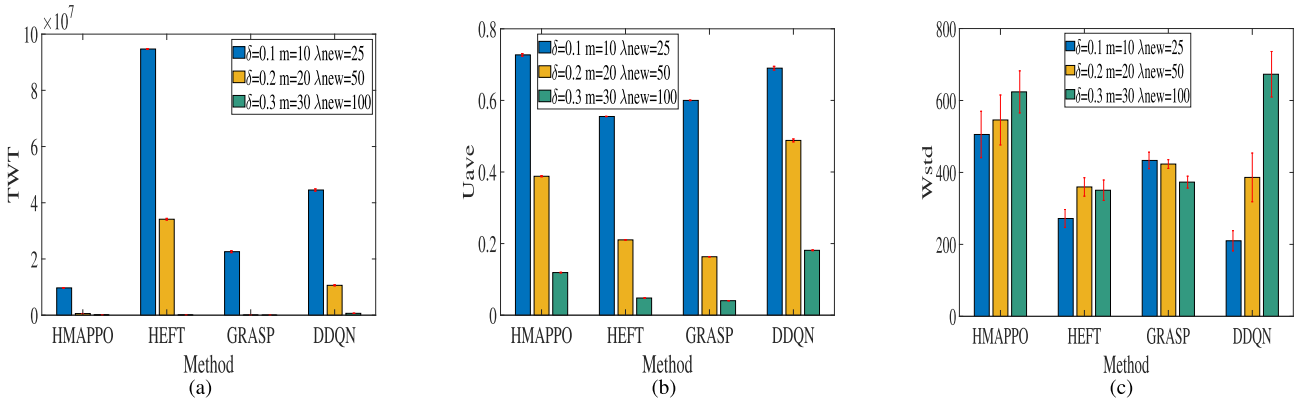


Fig. 13. Average values of three objectives obtained by the HMAPPO and other real-time scheduling methods on three representative instances. (a) TWT. (b) U_{ave}. (c) W_{std}.

TABLE XI
PERFORMANCE COMPARISONS WITH MOPSO AND NSGA-II ON SOME REPRESENTATIVE INSTANCES

δ	m	λ_{new}	HMAPPO				MOPSO				NSGA-II			
			TWT	U_{ave}	W_{std}	time(s)	TWT	U_{ave}	W_{std}	time(s)	TWT	U_{ave}	W_{std}	time(s)
0.1	10	25	**3.51e+05**$^+$	6.15e-01	2.05e+02	**1.51e-02**$^+$	1.16e+06	**6.75e-01**$^+$	2.31e+02	1.97e+02	8.72e+05	6.50e-01	**2.00e+02**	1.21e+02
0.1	20	100	**1.96e+04**	1.91e-01	2.32e+02	**8.02e-03**$^+$	3.93e+04	**2.03e-01**	2.29e+02	2.56e+01	2.12e+04	1.99e-01	**2.01e+02**$^+$	1.59e+01
0.1	30	50	**9.73e+03**$^+$	1.75e-01	**1.59e+02**	**1.28e-02**$^+$	2.40e+04	1.86e-01	1.82e+02	3.04e+01	2.06e+04	**2.05e-01**$^+$	1.65e+02	1.85e+01
0.2	10	100	5.85e+04	**3.71e-01**$^+$	**2.92e+02**$^+$	**8.23e-03**$^+$	4.51e+04	3.10e-01	3.32e+02	2.27e+01	**3.29e+04**$^+$	3.49e-01	3.50e+02	1.43e+01
0.2	20	50	**5.67e+04**$^+$	3.89e-01	2.87e+02	**1.34e-02**$^+$	1.68e+05	**4.55e-01**$^+$	3.01e+02	7.29e+01	1.31e+05	4.33e-01	**2.75e+02**	4.60e+01
0.2	30	25	**6.07e+04**$^+$	3.60e-01	**1.98e+02**$^+$	**3.34e-02**$^+$	1.92e+05	3.95e-01	2.65e+02	1.51e+02	1.57e+05	**4.04e-01**$^+$	2.81e+02	8.22e+01
0.3	10	50	**2.87e+05**$^+$	5.29e-01	**2.76e+02**$^+$	**1.47e-02**$^+$	6.96e+05	**6.15e-01**$^+$	3.04e+02	1.22e+02	7.41e+05	6.03e-01	3.29e+02	7.81e+01
0.3	20	25	**1.32e+05**$^+$	**5.61e-01**$^+$	**1.97e+02**$^+$	**1.72e-02**$^+$	3.22e+05	5.23e-01	3.54e+02	3.26e+02	4.01e+05	5.17e-01	3.91e+02	1.92e+02
0.3	30	100	1.42e+04	1.32e-01	**2.52e+02**$^+$	**8.31e-03**$^+$	2.56e+04	1.50e-01	3.10e+02	2.77e+01	**9.51e+03**$^+$	**1.66e-01**$^+$	2.99e+02	1.96e+01

more computing time compared to other methods due to the additional time for constructing state features and calculating the output of three policy networks. However, this slight increase in computing time will not obstruct its practical application in real-world scheduling problems.

G. Comparisons With Metaheuristics-Based Rolling-Horizon Rescheduling Methods

To further validate the effectiveness and efficiency of the proposed HMAPPO compared to metaheuristics-based rescheduling methods, we take two rolling-horizon rescheduling methods (i.e., generate a new schedule at each rescheduling point) based on multi-objective PSO (MOPSO) [55] and nondominated sorting genetic algorithm-II (NSGA-II) [51] as comparisons, where the rescheduling point is defined as the time a new job arrives or a machine breakdown occurs. At each rescheduling point, the population size and iteration number for these two methods are, respectively, set as 200 and 100. The parameter settings for the test instances are set the same as those in Table I. The average values of three objectives and computing times obtained by the comparative methods from 20 independent runs on some representative instances are given in Table XI, where the symbol "time" denotes the average computing time to make a decision at each rescheduling point. The superscript "+" denotes that the best result is significantly better than both of the other two counterparts with regard to two pairwise t tests with 5% significance level between the best result and each counterpart. It can be observed that the proposed HMAPPO can obtain significantly lower TWT and W_{std} for most instances. The NSGA-II can only obtain lower TWT for two instances where new jobs arrive infrequently (i.e., $\lambda_{\text{new}} = 100$). This further confirms the superiority of HMAPPO in the rapidly changing production environments. Despite that the MOPSO and NSGA-II can obtain relatively higher values in U_{ave}, their computing times to make a decision (generate a new schedule) at each rescheduling point are much more (more than 1000 times) than the HMAPPO. Thus, it can be concluded that the HMAPPO demonstrates the best performance in terms of both time efficiency and solution quality.

VII. CONCLUSION

In this article, a hierarchical multiagent PPO-based real-time scheduling method named HMAPPO is developed to address the DMOFJSP-PNW with new job insertions and machine breakdowns aiming at optimizing the TWT, average machine utilization rate, and variance of machine workload simultaneously. The proposed HMAPPO contains three PPO-based agents operating in different spatiotemporal levels, including an objective agent, a job agent, and a machine agent. The objective agent serves as a higher controller periodically determining the temporary objective to be optimized every a fixed length of rescheduling points. The job agent and machine agent are lower actuators respectively choosing a job selection rule and machine assignment rule to achieve the higher objective at each rescheduling point. By adaptively selecting the temporary objectives and the most feasible dispatching rules to achieve these objectives, both the timeliness of scheduling

and a good compromise among different objectives can be achieved. Numerical experiments on a wide range of test instances under different production environments have verified the effectiveness and superiority of the HMAPPO.

The proposed HMAPPO is particularly suitable for real-time scheduling in discrete flexible manufacturing factories, such as aerospace product manufacturing and steel manufacturing, where each job may contain several operations subjected to the no-wait constraint and dynamic disturbances, such as new job insertions and machine breakdowns that frequently occur in real time. It can also be well applied to various modern Industry 4.0 smart factories with high complexity and stochasticity, which requires reactive real-time scheduling methods to achieve product individualization and variety. Moreover, it can be extended to deal with many other practical production scheduling problems, such as hybrid flow shops and open shops with different uncertainties and objectives.

For future work, we will investigate more uncertainties, such as material shortages and variations in processing times. Other objectives, such as energy consumption and production costs, are also worthy to be considered. Meanwhile, a practical heuristic should be designed to estimate the possible machine breakdown times so as to determine if an operation should be assigned on the corresponding machine or not. Moreover, for more complicated and larger manufacturing systems, we will develop a decentralized real-time scheduling framework where each machine is associated with an intelligent DRL agent and study the cooperation mechanism among different agents.

ACKNOWLEDGMENT

The authors would like to thank the editors and the anonymous reviewers for their fruitful comments and suggestions in improving the quality of this article.

REFERENCES

[1] M. R. Garey, D. S. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," *Math. Oper. Res.*, vol. 1, no. 2, pp. 117–129, May 1976.

[2] N. G. Hall and C. Sriskandarajah, "A survey of machine scheduling problems with blocking and no-wait in process," *Oper. Res.*, vol. 44, no. 3, pp. 510–525, Jun. 1996.

[3] A. Aschauer, F. Roetzer, A. Steinboeck, and A. Kugi, "Efficient scheduling of a stochastic no-wait job shop with controllable processing times," *Expert Syst. Appl.*, vol. 162, Dec. 2020, Art. no. 113879.

[4] A. Allahverdi, "A survey of scheduling problems with no-wait in process," *Eur. J. Oper. Res.*, vol. 255, no. 3, pp. 665–686, 2016.

[5] D. Ouelhadj and S. Petrovic, "A survey of dynamic scheduling in manufacturing systems," *J. Scheduling*, vol. 12, no. 4, pp. 417–431, 2009.

[6] Y. Li, "Deep reinforcement learning: An overview," 2017, *arXiv:1701.07274*. [Online]. Available: https://arxiv.org/abs/1701.07274

[7] V. Mnih *et al.*, "Playing atari with deep reinforcement learning," 2013, *arXiv:1312.5602*. [Online]. Available: https://arxiv.org/abs/1312.5602

[8] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.

[9] B. Cunha, A. M. Madureira, B. Fonseca, and D. Coelho, "Deep reinforcement learning as a job shop scheduling solver: A literature review," in *Hybrid Intelligent Systems*, A. M. Madureira, A. Abraham, N. Gandhi, and M. L. Varela, Eds. Cham, Switzerland: Springer, 2020, pp. 350–359.

[10] L. Zhou, L. Zhang, and B. K. P. Horn, "Deep reinforcement learning-based dynamic scheduling in smart manufacturing," *Procedia CIRP*, vol. 93, pp. 383–388, Jan. 2020.

[11] B.-A. Han and J.-J. Yang, "Research on adaptive job shop scheduling problems based on dueling double DQN," *IEEE Access*, vol. 8, pp. 186474–186495, 2020.

[12] L. Hu, Z. Liu, W. Hu, Y. Wang, J. Tan, and F. Wu, "Petri-net-based dynamic scheduling of flexible manufacturing system via deep reinforcement learning with graph convolutional network," *J. Manuf. Syst.*, vol. 55, pp. 1–14, Apr. 2020.

[13] I.-B. Park, J. Huh, J. Kim, and J. Park, "A reinforcement learning approach to robust scheduling of semiconductor manufacturing facilities," *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 3, pp. 1420–1431, Jul. 2020.

[14] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3675–3683.

[15] J. Rafati and D. C. Noelle, "Learning representations in model-free hierarchical reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, Jul. 2019, pp. 10009–10010.

[16] J. H. Blackstone, D. T. Phillips, and G. L. Hogg, "A state-of-the-art survey of dispatching rules for manufacturing job shop operations," *Int. J. Prod. Res.*, vol. 20, no. 1, pp. 27–45, 1982.

[17] S. S. Panwalkar and W. Iskander, "A survey of scheduling rules," *Oper. Res.*, vol. 25, no. 1, pp. 45–61, Jan./Feb. 1977.

[18] V. Sels, N. Gheysen, and M. Vanhoucke, "A comparison of priority rules for the job shop scheduling problem under different flow time- and tardiness-related objective functions," *Int. J. Prod. Res.*, vol. 50, no. 15, pp. 4255–4270, Aug. 2012.

[19] C. Rajendran and O. Holthaus, "A comparative study of dispatching rules in dynamic flowshops and jobshops," *Eur. J. Oper. Res.*, vol. 116, no. 1, pp. 156–170, Jul. 1999.

[20] O. Holthaus and C. Rajendran, "Efficient jobshop dispatching rules: Further developments," *Prod. Planning Control*, vol. 11, no. 2, pp. 171–178, Jan. 2000.

[21] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Evolving scheduling heuristics via genetic programming with feature selection in dynamic flexible job-shop scheduling," *IEEE Trans. Cybern.*, vol. 51, no. 4, pp. 1797–1811, Apr. 2021.

[22] L. Nie, L. Gao, P. Li, and X. Li, "A GEP-based reactive scheduling policies constructing approach for dynamic flexible job shop scheduling problem with job release dates," *J. Intell. Manuf.*, vol. 24, no. 4, pp. 763–774, Aug. 2013.

[23] S. Jun, S. Lee, and H. Chun, "Learning dispatching rules using random forest in flexible job shop scheduling problems," *Int. J. Prod. Res.*, vol. 57, no. 10, pp. 3290–3310, May 2019.

[24] J. Mohan, K. Lanka, and A. N. Rao, "A Review of dynamic job shop scheduling techniques," *Procedia Manuf.*, vol. 30, pp. 34–39, Jan. 2019.

[25] L. Zhang, L. Gao, and X. Li, "A hybrid genetic algorithm and Tabu search for a multi-objective dynamic job shop scheduling problem," *Int. J. Prod. Res.*, vol. 51, no. 12, pp. 3516–3531, Jun. 2013.

[26] K. Z. Gao, P. N. Suganthan, Q. K. Pan, M. F. Tasgetiren, and A. Sadollah, "Artificial bee colony algorithm for scheduling and rescheduling fuzzy flexible job shop problem with new job insertion," *Knowl.-Based Syst.*, vol. 109, pp. 1–16, Oct. 2016.

[27] M. Nouiri, A. Bekrar, A. Jemai, D. Trentesaux, A. C. Ammari, and S. Niar, "Two stage particle swarm optimization to solve the flexible job shop predictive scheduling problem considering possible machine breakdowns," *Comput. Ind. Eng.*, vol. 112, pp. 595–606, Oct. 2017.

[28] S. Zhang and T. N. Wong, "Flexible job-shop scheduling/rescheduling in dynamic environment: A hybrid MAS/ACO approach," *Int. J. Prod. Res.*, vol. 55, no. 11, pp. 3173–3196, Jun. 2017.

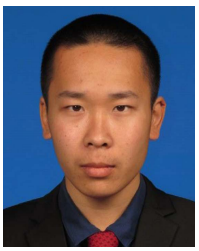
[29] K. Gao, F. Yang, M. Zhou, Q. Pan, and P. Suganthan, "Flexible job-shop rescheduling for new job insertion by using discrete Jaya algorithm," *IEEE Trans. Cybern.*, vol. 49, no. 5, pp. 1944–1955, May 2019.

[30] M. Shahgholi Zadeh, Y. Katebi, and A. Doniavi, "A heuristic model for dynamic flexible job shop scheduling problem considering variable processing times," *Int. J. Prod. Res.*, vol. 57, no. 10, pp. 3020–3035, May 2019.

[31] A. Baykasoğlu, F. S. Madenoğlu, and A. Hamzadayı, "Greedy randomized adaptive search for dynamic flexible job-shop scheduling," *J. Manuf. Syst.*, vol. 56, pp. 425–451, Jul. 2020.

[32] Y. Xu, L. Wang, S.-Y. Wang, and M. Liu, "An effective teaching–learning-based optimization algorithm for the flexible job-shop scheduling problem with fuzzy processing time," *Neurocomputing*, vol. 148, pp. 260–268, Jan. 2015.

[33] B. Liu, Y. Fan, and Y. Liu, "A fast estimation of distribution algorithm for dynamic fuzzy flexible job-shop scheduling problem," *Comput. Ind. Eng.*, vol. 87, pp. 193–201, Sep. 2015.
[34] M. Zandieh, A. R. Khatami, and S. H. A. Rahmati, "Flexible job shop scheduling under condition-based maintenance: Improved version of imperialist competitive algorithm," *Appl. Soft Comput.*, vol. 58, pp. 449–464, Sep. 2017.
[35] J. Li, Z.-M. Liu, C. Li, and Z. Zheng, "Improved artificial immune system algorithm for type-2 fuzzy flexible job shop scheduling problem," *IEEE Trans. Fuzzy Syst.*, early access, Aug. 12, 2020, doi: 10.1109/TFUZZ.2020.3016225.
[36] J.-Q. Li *et al.*, "A hybrid iterated greedy algorithm for a crane transportation flexible job shop problem," *IEEE Trans. Autom. Sci. Eng.*, early access, Mar. 17, 2021, doi: 10.1109/TASE.2021.3062979.
[37] Y. Wei and M. Zhao, "Composite rules selection using reinforcement learning for dynamic job-shop scheduling," in *Proc. IEEE Conf. Robot., Automat. Mechatronics*, vol. 2, Dec. 2004, pp. 1083–1088.
[38] H. Wang, B. R. Sarker, J. Li, and J. Li, "Adaptive scheduling for assembly job shop with uncertain assembly times based on dual Q-learning," *Int. J. Prod. Res.*, pp. 1–17, 2020, doi: 10.1080/00207543.2020.1794075.
[39] Z. Cao, C. Lin, and M. Zhou, "A knowledge-based cuckoo search algorithm to schedule a flexible job shop with sequencing flexibility," *IEEE Trans. Autom. Sci. Eng.*, vol. 18, no. 1, pp. 56–69, Jan. 2021.
[40] Y. Wei, L. Pan, S. Liu, L. Wu, and X. Meng, "DRL-scheduling: An intelligent QoS-aware job scheduling framework for applications in clouds," *IEEE Access*, vol. 6, pp. 55112–55125, 2018.
[41] S. Luo, "Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning," *Appl. Soft Comput.*, vol. 91, Jun. 2020, Art. no. 106208.
[42] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 387–395.
[43] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1889–1897.
[44] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*. [Online]. Available: https://arxiv.org/abs/1707.06347
[45] H. Zhu, M. Li, Y. Tang, and Y. Sun, "A deep-reinforcement-learning-based optimization approach for real-time scheduling in cloud manufacturing," *IEEE Access*, vol. 8, pp. 9987–9997, 2020.
[46] C.-L. Liu, C.-C. Chang, and C.-J. Tseng, "Actor-critic deep reinforcement learning for solving job shop scheduling problems," *IEEE Access*, vol. 8, pp. 71752–71762, 2020.
[47] A. Kuhnle, L. Schäfer, N. Stricker, and G. Lanza, "Design, implementation and evaluation of reinforcement learning for an adaptive order dispatching in job shop manufacturing systems," *Procedia CIRP*, vol. 81, pp. 234–239, Jan. 2019.
[48] O. Holthaus, "Scheduling in job shops with machine breakdowns: An experimental study," *Comput. Ind. Eng.*, vol. 36, no. 1, pp. 137–162, Jan. 1999.
[49] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evol. Comput.*, vol. 8, no. 2, pp. 173–195, 2000.
[50] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 257–271, Nov. 1999.
[51] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.
[52] P. D. D. Dominic, S. Kaliyamoorthy, and M. S. Kumar, "Efficient dispatching rules for dynamic job shop scheduling," *Int. J. Adv. Manuf. Technol.*, vol. 24, no. 2, pp. 70–75, Jul. 2004.
[53] B. Chen and T. I. Matis, "A flexible dispatching rule for minimizing tardiness in job shop scheduling," *Int. J. Prod. Econ.*, vol. 141, no. 1, pp. 360–365, Jan. 2013.
[54] Z. Cao, L. Zhou, B. Hu, and C. Lin, "An adaptive scheduling algorithm for dynamic jobs for dealing with the flexible job shop scheduling problem," *Bus. Inf. Syst. Eng.*, vol. 61, no. 3, pp. 299–309, 2019.
[55] C. A. C. Coello and M. S. Lechuga, "MOPSO: A proposal for multiple objective particle swarm optimization," in *Proc. Congr. Evol. Comput.*, vol. 2, May 2002, pp. 1051–1056.



Shu Luo received the B.S. degree in control science and engineering from Tsinghua University, Beijing, China, in 2017, where he is currently pursuing the Ph.D. degree with the Department of Automation.

His current research interests are focused on advanced planning and scheduling, metaheuristics, and machine learning.

Linxuan Zhang received the Ph.D. degree from Tsinghua University, Beijing, China, in 1998.

He is currently an Associate Professor with the National Engineering Research Centre of Computer Integrated Manufacturing System (CIMS-ERC), Department of Automation, Tsinghua University. His current research interests are focused on manufacturing system modeling and simulation, advanced planning and scheduling, and prognostics and health management.

Yushun Fan received the Ph.D. degree from Tsinghua University, Beijing, China, in 1990.

He is currently a Professor with the National Engineering Research Centre of Computer Integrated Manufacturing System (CIMS-ERC), Department of Automation, Tsinghua University. His current research interests are focused on enterprise modeling methods and optimization analysis, business process reengineering and workflow management, object-oriented technologies and flexible software systems, and workshop management and control.