Real-Time Scheduling for Flexible Job Shop With AGVs Using Multiagent Reinforcement Learning and Efficient Action Decoding

Yuxin Li, Qingzheng Wang, Xinyu Li, *Member, IEEE*, Liang Gao, *Senior Member, IEEE*, Ling Fu, Yanbin Yu, and Wei Zhou

***Abstract*—The application of automated guided vehicle (AGV) greatly improves the production efficiency of workshop. However, machine flexibility and limited logistics equipment increase the complexity of collaborative scheduling, and frequent dynamic events bring uncertainty. Therefore, this article proposes a real-time scheduling method for dynamic flexible job shop scheduling problem with AGVs using multiagent reinforcement learning (MARL). Specifically, a real-time scheduling framework is proposed in which a multiagent scheduling architecture is designed for achieving task selection, machine allocation and AGV allocation. Then, an action space and an efficient action decoding algorithm are proposed, which enable agents to explore in the high-quality solution space and improve the learning efficiency. In addition, a state space with generalization, a reward function considering machine idle time and a strategy for handling four disturbance events are designed to minimize the total tardiness cost. Comparison experiments show that the proposed method outperforms the priority dispatching rules, genetic programming and four popular reinforcement learning (RL)-based methods, with performance improvements mostly exceeding 10%. Furthermore, experiments considering four disturbance events demonstrate that the proposed method has strong robustness, and it can provide appropriate scheme for uncertain manufacturing system.**

***Index Terms*—Automated guided vehicle (AGV), disturbance events, flexible job shop, multiagent reinforcement learning (MARL).**

Received 11 September 2024; revised 4 November 2024; accepted 8 December 2024. Date of publication 7 January 2025; date of current version 19 February 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 52188102 and Grant U21B2029; in part by the Fundamental Research Funds for the Central Universities under Grant 2024BRA004; and in part by the Key Research and Development Program of Hubei Province under Grant 2021AAB001. This article was recommended by Associate Editor D. O. Olson. *(Corresponding author: Xinyu Li.)*

Yuxin Li, Qingzheng Wang, Xinyu Li, and Liang Gao are with the State Key Laboratory of Intelligent Manufacturing Equipment and Technology, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: liyuxinsc@outlook.com; wqzheng98@hust.edu.cn; lixinyu@mail.hust.edu.cn; gaoliang@mail.hust.edu.cn).

Ling Fu and Yanbin Yu are with the Department of Simulation and Digital Twin, Siemens Technology, Shanghai 200082, China (e-mail: ling.fu@siemens.com; yanbin.yu@siemens.com).

Wei Zhou is with the Department of Simulation and Digital Twin, Siemens Technology, Wuhan 430074, China (e-mail: zhou.charles@siemens.com).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TSMC.2024.3520381.

Digital Object Identifier 10.1109/TSMC.2024.3520381

I. Introduction

With the arrival of mass customization [1], enterprises have widely adopted logistics systems to accelerate material transfer and timely handle diverse customer needs. However, research on production systems shows that transportation costs account for 20%–50% of total costs [2]. Meanwhile, the timeliness of job transmission greatly affects the processing efficiency of machines [3]. Therefore, the efficient collaboration between production resources and transportation resources (such as AGV, i.e., automated guided vehicle) has become a key issue. In addition, due to a series of inevitable and unpredictable dynamic events such as order placement and machine breakdown [4], the manufacturing systems are required to have strong timely response capability and self-organization. Therefore, it is of great significance to design a method to efficiently solve dynamic flexible job shop scheduling problem with AGVs (DFJSP-AGVs).

Flexible job shop scheduling problem (FJSP) is an NP-hard problem, and the addition of AGVs makes it more difficult. DFJSP-AGVs has the following characteristics: 1) machine flexibility, limited AGVs, and highly coupled production-logistics process [5] make the problem very complex; 2) customers have a requirement on delivery time [6]; and 3) disturbance events occur frequently in the workshop. It makes the static scheduling method infeasible, such as mixed integer linear programming (MILP) [7]. Because each event makes the original scheme inconsistent with the actual production.

The uncertainty of manufacturing process [8] has forced scholars to develop many dynamic scheduling methods. The commonly used method in actual production is the priority dispatching rule (PDR), such as shortest processing time (SPT) and first come first service (FCFS). Although PDR can provide a scheme within seconds, its performance cannot be guaranteed. Because no single PDR can provide strong performance along all objectives in various manufacturing scenarios [6]. To this end, scholars have proposed hyper-heuristics, such as genetic programming (GP). It explores a search space of heuristics to discover those that work effectively through a series of operators [9]. However, the composite rules generated by hyper-heuristics still rely on simple computations and logical operations [6]. In addition, in order to pursue high-quality schedules, scholars have designed rescheduling methods using meta-heuristics. However, these methods require the time-consuming process of iterative evaluation to regenerate a new

2168-2216 © 2025 IEEE. All rights reserved, including rights for text and data mining, and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on March 06,2025 at 12:39:15 UTC from IEEE Xplore. Restrictions apply.

scheme, which does not meet the need of rapid response for enterprises [10].

Since Mnih et al. [11] proposed the standard deep Q network (DQN) algorithm in 2015, deep reinforcement learning (DRL) has become a hot research field in artificial intelligence. DRL has the ability to process complex data with the help of deep neural network (DNN), and can provide appropriate decisions based on the environment state. These features make DRL a promising scheduling approach in a data-intensive manufacturing system [6]. However, DRL is mostly used in classic problems such as job shop scheduling problem (JSP) [12] and FJSP [6], but the logistics factor is not taken into account. At the same time, current research often only considers the new job arrival [10] or machine breakdown [4], while ignoring other disturbance events.

To address the above challenges, this article proposes a real-time scheduling method for DFJSP-AGVs to minimize the total tardiness cost using multiagent reinforcement learning (MARL). The main contributions of this article are as follows.

1) A real-time scheduling framework is proposed for DFJSP-AGVs, and a multiagent scheduling architecture is designed to realize the resource allocation. It can achieve the precise update of task pool, and the sequential decision-making and collaborative training of multiple agents.
2) An action space and an efficient action decoding algorithm are proposed using PDR weighting and PDR adjustment. It not only enables agents to obtain all possible solutions, but also guides them to explore high-quality solution spaces and improves the learning efficiency.
3) A generic state space and a reward function considering machine idle time are designed. In addition, the strategy for handling disturbance events is given to enhance robustness.
4) Experimental results show that the proposed method has superiority and generality compared with other methods, and can effectively deal with various disturbance events.

The remainder of this article is organized as follows. Section II introduces the related literature and techniques. Section III defines the research problem. Section IV provides the implementation details of the proposed method. Section V presents the experimental results. Finally, Section VI concludes the work and discusses the future research direction.

II. Related Work

A. Production-Logistics Collaborative Scheduling

The static scheduling methods mainly include the exact algorithm, heuristics and meta-heuristics. Exact algorithms can obtain the optimal solution, such as [7], [13], and [14]. But they are only suitable for solving small-scale problems. Heuristics are constructed from human experience. Zhang et al. [15] proposed a shifting bottleneck heuristic for JSP with transportation constraints. Amirteimoori et al. [16] developed a novel parallel heuristic for hybrid JSP with conflict-free AGV routing. But the performance of heuristics cannot be guaranteed. Meta-heuristics can achieve a good balance between the superiority and efficiency of solutions [17]. Pan et al. [18] proposed a learning-based multipopulation evolutionary optimization for FJSP with finite AGVs. Liu et al. [19] designed an improved genetic algorithm for integrated process planning and scheduling problem considering AGVs.

Research on dynamic scheduling is little, and methods include meta-heuristics [2], multiagent system [20], DRL [21], etc. Luo et al. [22] proposed a real-time edge scheduling framework for order-level requirements of smart factory. Cai et al. [23] proposed the earliest task release moment and a new real-time scheduling model for JSP with AGVs. Zhang et al. [24] proposed a graph neural network (GNN) and DRL based method for FJSP with AGVs. However, these methods do not fully explore the underlying laws behind the production data, and there is still room for further optimization.

B. DRL-Based Dynamic Scheduling Approaches

Due to DRL's outstanding performance in complex problems such as Go and autonomous driving, many dynamic scheduling methods have been developed in recent years. The neural network models used include fully connected network [4], convolution neural network [25], GNN [26], etc.

From the perspective of action space, DRL-based methods mainly include: 1) PDR-based; 2) end-to-end; and 3) special. PDR-based methods utilize directly rules as actions to realize the decision-making, such as [6] and [10]. However, such methods cannot obtain many feasible schedules [27]. At the same time, end-to-end methods can explore all possible solutions of problem through special action design, such as [12] and [26]. Special action space can achieve the full coverage of solution space [27], [28] or the action quality improvement [21] through careful design.

On the other hand, the multiple resources of manufacturing system facilitate the integration of MARL and scheduling methods. The agents in MARL can be physical entities, such as tasks [29], machines [30], equipment [31] and jobs [32]. At the same time, the agents in MARL can be subproblems, such as job selection. Wang et al. [33] proposed a MARL approach for hybrid flow shop with batch machines. Zhu et al. [34] proposed a three-agents-based real-time scheduling method using DRL for dual-resource flexible job shop. Luo et al. [4] used MARL to solve the dynamic partial-no-wait multiobjective FJSP. The above methods rely on MARL and reliable architecture to collaboratively train multiple agents, thereby achieving efficient scheduling.

In summary, although there exist many studies on DRL-based scheduling, the logistics factor is often overlooked. Meanwhile, the existing methods often ignore the design of efficient actions. In addition, disturbance events in the existing studies are insufficient. Therefore, the goal of this article is to design an efficient real-time scheduling method for DFJSP-AGVs considering various disturbance events using MARL.

III. Problem Formulation for DFJSP-AGVs

A. Problem Description

The DFJSP-AGVs is defined as follows. The layout of flexible job shop with AGVs is shown in Fig. 1. There exist

Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on March 06,2025 at 12:39:15 UTC from IEEE Xplore. Restrictions apply.

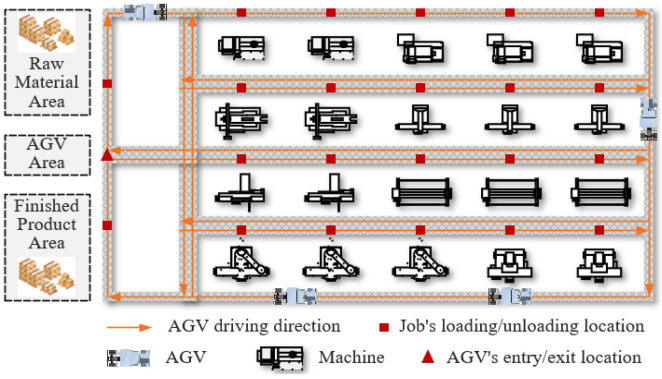

Fig. 1. Layout of flexible job shop with AGVs.

multiple types of machines $\mathbb{M} = \{\mathbb{M}_x, x = 1, \ldots, m\}$ in the workshop, and each type contains multiple machines $\mathbb{M}_x = \{M_{xk}, k = 1, \ldots, N_x\}$. Each machine has an input buffer and an output buffer for temporary job storage. Meanwhile, multiple AGVs $\mathbb{V} = \{V_y, y = 1, \ldots, l\}$ are running in the workshop. All AGVs are at the AGV area in the beginning, and they are responsible for transferring jobs from one location to another location for operation processing.

In the production process, many jobs $\mathbb{J} = \{J_i, i = 1, \ldots, n\}$ are successively released to the workshop, and the arrival time and due date of job J_i are A_i and D_i respectively. Each job consists of a set of operations $J_i = \{O_{ij}, j = 1, \ldots, N_i\}$, which need to be processed in sequence. Each operation O_{ij} can be processed on any one in a prespecified type of machines $\mathbb{M}_x$ with different processing times. Two successive operations correspond to different machine types. When a new job arrival occurs, the job is initially stored in the raw material area. After finishing all processing tasks, the job needs to return to the finished product area for storage. When all jobs are finished, the production process ends. In addition, the assumptions are given in the "Section A of Supplementary Material".

The objective of DFJSP-AGVs is to minimize the total tardiness cost (TTC), and it is calculated by

$$\min \mathrm{TTC} = \min \sum_{i=1}^{n} (w_i \cdot \max(0,\ C_i - D_i)) \tag{1}$$

where w_i is the weight of job J_i and it represents the urgency degree; C_i is the completion time of job J_i.

B. Disturbance Events

1) New Job Arrivals: There exist some jobs in the workshop at the beginning, and new jobs arrive following Poisson distribution. In other words, the interarrival time between successive job arrivals is subjected to exponential distribution exp(1/MTBA), where MTBA represents the mean time between arrivals. Referring to [6], this article designed the formula of MTBA considering logistics factor, and it is shown in

$$\mathrm{MTBA} = \frac{\overline{N_{\mathrm{ope}}} \cdot \overline{T^{\mathrm{PT}}}}{U \cdot \sum_{x=1}^{m} N_x} + \frac{\left(\overline{N_{\mathrm{ope}}} + 1\right) \cdot \overline{T^{\mathrm{LT}}}}{U \cdot l} \tag{2}$$

where U is the utilization rate of workshop; $\overline{N_{\mathrm{ope}}}$ is the mean of operation numbers for jobs; $\overline{T^{\mathrm{PT}}}$ is the mean of times for all processing tasks; $\overline{T^{\mathrm{LT}}}$ is the mean of times for delivery parts of all logistics tasks. It represents that the interarrival time is equal to the ratio of total task quantity to workshop capacity (processing and logistics). It should be noted that the higher the utilization rate U, the more frequent the new job arrival.

The due date of job adopts the Total Work Content method [4]. Considering logistics factor, its formula is shown in

$$D_i = A_i + \mathrm{DDT}_i \cdot \sum_{j=1}^{N_i} \left(\sum_{k=1}^{N_x} T_{ijxk}^{\mathrm{PT}} / N_x \right) + \mathrm{DDT}_i \cdot (N_i + 1) \cdot \overline{T^{\mathrm{LT}}} \tag{3}$$

where DDT_i is the due date tightness of job J_i; T_{ijxk}^{PT} is the time of processing task for operation O_{ij} on machine M_{xk}. The smaller the DDT_i, the more likely the job J_i is to delay.

2) Machine Breakdowns and Repairs: The relevant details are as follows. 1) When a machine fails, it needs to stop processing immediately. When a machine is repaired, its processing capacity is restored. 2) For the job that is being processed when the machine fails, this article adopts the from scratch with the interrupt-repeat mode [21]. 3) The repair time of the machine is unpredictable.

The interval time between two successive machine breakdowns and the repair time of failed machine both follow the exponential distribution, and they are exp(1/MTBF) and exp(1/MTTR) respectively [4]. MTBF is the mean time between failures, and MTTR is the mean time to repair. The breakdown level of workshop A_g is expressed by

$$A_g = \mathrm{MTTR}/(\mathrm{MTBF} + \mathrm{MTTR}). \tag{4}$$

Note that when MTTR is fixed, the larger the A_g, the smaller the MTBF, and the more frequent machine failures occur.

3) Job Reworks: In actual production, the material quality, improper worker operation and other reasons will cause the job quality to be unqualified, which leads to the requirement of job rework.

4) Fuzzy Transportation Time: Due to path congestion, AGV acceleration, AGV deceleration and other reasons, the actual transportation time of AGV will fluctuate based on the predetermined time.

IV. MARL-Based Real-Time Scheduling Method

A. Overall Framework

1) Offline Training and Online Application: The proposed real-time scheduling framework is shown in Fig. 2. It includes the shop floor environment and decision system, which interact with real-time information, including job data, machine data and AGV data. The decision system mainly includes the task pool and the neural network models of three agents. The task pool stores a series of tasks, and each task is an operation or the last logistics task of a job.

The real-time scheduling framework has two modes: offline training and online application.

Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on March 06,2025 at 12:39:15 UTC from IEEE Xplore. Restrictions apply.

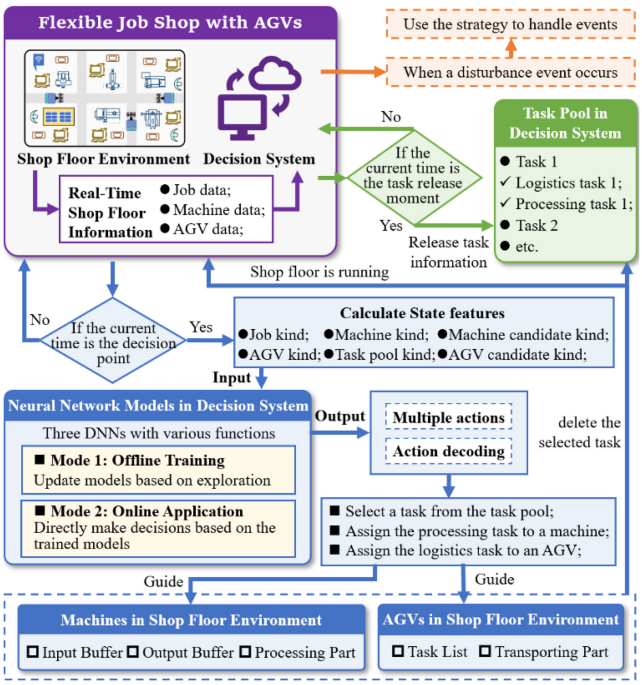

Fig. 2. Proposed real-time scheduling framework.

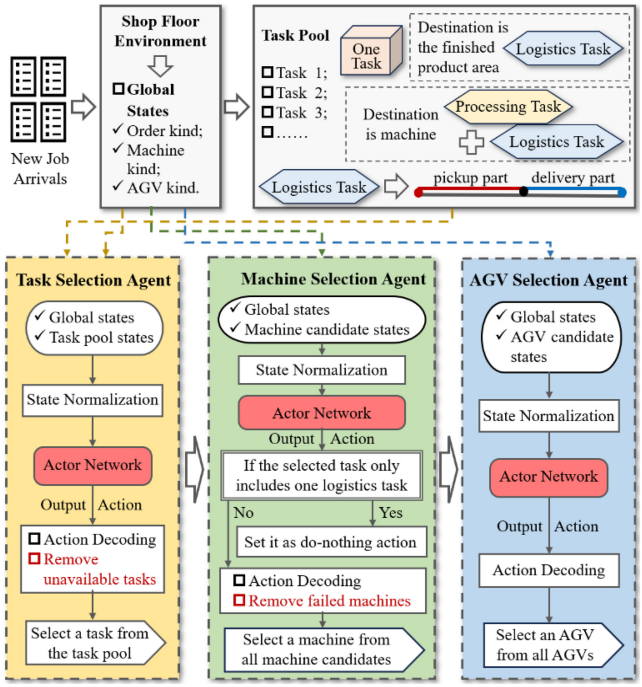

Fig. 3. Resource allocation process of agents.

a) Offline training:

Step 1: When the task pool is not empty and there exists one or more idle AGVs, the current time is the decision point.

Step 2: When the current time is a decision point, the decision system calculates six kinds of state features based on synchronized information. After inputting the state, the neural network models of agents output multiple actions. Through action decoding, the agents complete the machine allocation and AGV allocation of a task. Finally, the selected processing task is added to the input buffer of a machine, and the selected logistics task is added to the task list of an AGV.

Step 3: After receiving the task assigned by agents, the machine/AGV continues to run according to its own logic, and the shop floor environment runs to the next decision point. The running logic of machine/AGV is given below.

Machine: 1) *Input Buffer:* Store jobs that need processing; 2) *Output Buffer:* Store jobs that have finished processing; and 3) *Processing Part:* When the machine is idle, it uses FCFS to select a job from the input buffer for processing.

AGV: 1) *Task List:* Store logistics tasks that need to be executed and 2) *Transporting Part:* When the AGV is idle, it uses first assign first service (FAFS) to select a logistics task from the task list to execute.

Step 4: The neural network models receive a reward from the shop floor environment. Then, repeat steps 2, 3 and 4.

Step 5: Based on a series of exploration trajectories, the neural network models utilize multiagent proximal policy optimization (MAPPO) [35] for self-updating, and eventually can provide appropriate allocation commands according to the real-time workshop state.

b) Online application: After offline training is completed, practitioners can enter the "online application" mode. Based on the trained DNNs, the decision system continuously performs the interactions in steps 2 and 3 to efficiently finish the production of an order.

In addition, it should be emphasized that in the offline training or online application, when one disturbance event occurs, the decision system adopts the strategy to handle it.

2) Update of Task Pool:

a) Task release: When the current time is the task release moment, the shop floor environment adds a task into task pool. Based on the earliest task release moment in [23], this article proposes an improved task release moment. At the beginning, the job releases the task of the first operation. Then, when the job only has one task that has been assigned an AGV but is not finished, it releases its next task. This task release moment can not only reduce the job's waiting time for AGV, but also avoid unnecessary premature decision-making. More details are given in the "Section B of Supplementary Material."

b) Task deletion: After the neural network models make a decision, the task pool deletes the assigned tasks.

3) Multiagent Scheduling Architecture: The core of offline training lies in the resource allocation of step 2. Therefore, this article establishes a multiagent scheduling architecture, which is shown in Fig. 3.

The shop floor environment forms the global states and task pool. The task pool contains two types of tasks. 1) T_1^{Pool}: Its destination is the machine, and it corresponds to an operation of a job. It consists of a processing task and a logistics task. 2) T_2^{Pool}: Its destination is the finished product area, and it corresponds to the last logistics task of a job. It only contains one logistics task. Note that the logistics task in each task consists of two parts: pickup part and delivery part.

At each decision point, the details of resource allocation process are given below.

Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on March 06,2025 at 12:39:15 UTC from IEEE Xplore. Restrictions apply.

TABLE I
REAL-TIME ATTRIBUTES OF WORKSHOP

Notations	Descriptions
Variables for manufacturing system	
t	Current time
Variables for task pool	
T_g^P	The g-th task in the task pool, $g = 1, \ldots, NTP$, where NTP is the number of tasks in the task pool
J_g^P	The job corresponding to task T_g^P
Jth	The sequence number of operation for task T_g^P in job J_g^P
T_{gj}^{PT}	Time of the j-th processing task for job J_g^P
FT_{gj}^{PT}	Time that the j-th processing task for job J_g^P has been finished
T_{ga}^{DLT}	Time of the delivery part of the a-th logistics task for job J_g^P
FT_{ga}^{DLT}	Time that the delivery part of the a-th logistics task for job J_g^P has been finished
RT_g^P	Remaining time of job J_g^P (consider processing and logistics)
EAT_g^P	Earliest available time of job J_g^P
EWT_g^P	Estimated weighted tardiness of job J_g^P
Variables for all machines	
$RTING_{xk}^M$	Remaining time of the in-process task for machine M_{xk}
T_{xkw}^{PT}	Time of the w-th processing task in the buffer for machine M_{xk}, $w = 1, \ldots, N_{xk}$, where N_{xk} is the number of processing tasks in the machine's buffer
CT_{xk}^M	Shortest time for completing the in-process task and all tasks in the buffer for machine M_{xk}
DPT_{gxk}	The sum of delivery time and processing time of task T_g^P for machine M_{xk}
Variables for all AGVs	
$RTING_y^A$	Remaining time of the in-transport task for AGV V_y
T_{ye}^{LT}	Time of the e-th logistics task in the list for AGV V_y, $e = 1, \ldots, N_y$, where N_y is the number of logistics tasks in the AGV's list
CT_y^A	Shortest time for completing the in-transport task and all tasks in the list for AGV V_y
PUT_{yg}	Pickup time of AGV V_y for task T_g^P

TABLE II
SIX KINDS OF STATE FEATURES

State	Content
State features for all jobs	
SF_1	the completion rate of the order; the mean/std of completion rates for all jobs.
State features for all machines	
SF_2	the mean/std of utilization rates for all machines; the mean/std of CT_{xk}^M for all machines; the proportion of faulty machines among all machines.
State features for all AGVs	
SF_3	the mean/std of utilization rates for all AGVs; the mean/std of CT_y^A for all AGVs.
State features for task pool	
SF_4	the number of tasks; the mean/std of processing times for all tasks; the mean/std of completion rates for all jobs; the mean/std of RT_g^P for all jobs; the mean/std/min/max of EAT_g^P for all jobs; the mean/std/min/max of EWT_g^P for all jobs.
State features for all machine candidates of the selected job	
SF_5	the mean/std of utilization rates; the mean/std of CT_{xk}^M; the mean of DPT_{gxk}; the proportion of faulty machines among all machine candidates; the EAT_g^P of the selected job; the EWT_g^P of the selected job.
State features for all AGV candidates of the selected job	
SF_6	the $CT_y^A + PUT_{yg}$ of each AGV for the selected job; the EAT_g^P of the selected job; the EWT_g^P of the selected job; the CT_{xk}^M of the assigned machine.

Step 1: The task selection agent receives the normalized global states and task pool states and outputs an action. Through action decoding and removing unavailable tasks (all machine candidates of the task are faulty), a task is selected from the task pool.

Step 2: Determine the machine candidates of the selected task. The machine selection agent receives the normalized global states and machine candidate states and outputs an action. If the selected task belongs to T_1^{Pool}, one machine is selected from all candidates for processing by action decoding and removing failed machines. If the selected task belongs to T_2^{Pool}, set the agent's action as the do-nothing action δ_0.

Step 3: The AGV selection agent receives the normalized global states and AGV candidate states (all AGVs) and outputs an action. Through action decoding, one AGV is selected from all AGVs for transportation.

B. Real-Time Workshop Attribute Model

To facilitate the description of state, action and reward in the multiagent scheduling architecture, a real-time workshop attribute model is established. It is shown in Table I.

Some attributes in Table I need further elaboration. RT_g^P, EAT_g^P and EWT_g^P are given in

$$\text{RT}_g^P = \sum_{j=1}^{N_g}\left(T_{gj}^{\text{PT}} - \text{FT}_{gj}^{\text{PT}}\right) + \sum_{a=1}^{N_g+1}\left(T_{ga}^{\text{DLT}} - \text{FT}_{ga}^{\text{DLT}}\right) \quad (5)$$

$$\text{EAT}_g^P = \sum_{a=1}^{\text{Jth}}\left(T_{ga}^{\text{DLT}} - \text{FT}_{ga}^{\text{DLT}}\right) + \sum_{j=1}^{\text{Jth}}\left(T_{gj}^{\text{PT}} - \text{FT}_{gj}^{\text{PT}}\right) \quad (6)$$

$$\text{EWT}_g^P = \max\left(0,\ t + \text{RT}_g^P - D_g\right) \cdot w_g. \quad (7)$$

Define the location of previous task of T_g^P as L_1, and the location of machine M_{xk} as L_2. The processing time of task T_g^P on machine M_{xk} is set as T_{gxk}^{PT}. Hence, CT_{xk}^M and DPT_{gxk} are calculated by

$$\text{CT}_{xk}^M = \text{RTING}_{xk}^M + \sum_{w=1}^{N_{xk}} T_{xkw}^{\text{PT}} \quad (8)$$

$$\text{DPT}_{gxk} = \Delta t(L_1,\ L_2) + T_{gxk}^{\text{PT}}. \quad (9)$$

Define the destination of task $T_{yN_y}^{\text{LT}}$ as L_1, and the pickup location of the selected task T_g^P as L_2. Therefore, CT_y^A and PUT_{yg} are expressed by

$$\text{CT}_y^A = \text{RTING}_y^A + \sum_{e=1}^{N_y} T_{ye}^{\text{LT}} \quad (10)$$

$$\text{PUT}_{yg} = \Delta t(L_1,\ L_2). \quad (11)$$

C. State Space

The real-time workshop state is divided into six parts, where $\{\text{SF}_1,\ \text{SF}_2,\ \text{SF}_3\}$ represents the global states, and $\{\text{SF}_4,\ \text{SF}_5,\ \text{SF}_6\}$ describes different decision objects. They are given in Table II. The state spaces of three agents are $S_{\text{TS}} = \{\text{SF}_1,\ \text{SF}_2,\ \text{SF}_3,\ \text{SF}_4\}$, $S_{\text{MS}} = \{\text{SF}_1,\ \text{SF}_2,\ \text{SF}_3,\ \text{SF}_5\}$, and $S_{\text{AS}} = \{\text{SF}_1,\ \text{SF}_2,\ \text{SF}_3,\ \text{SF}_6\}$ respectively. Most state features are statistical, and each agent can understand real-time production information through the input state. It should be noted that this article adopts Z-Score normalization to alleviate the negative effects caused by scale differences of different state features.

Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on March 06,2025 at 12:39:15 UTC from IEEE Xplore. Restrictions apply.

The input-output of three agents adopts a serial relationship. In this way, each agent can clarify the decision object and use its detailed data as input, so as to make reasonable decisions.

D. Action Space

Based on PDR weighting and PDR adjustment, this article designs an action space and an efficient action decoding algorithm to ensure efficient training and learning of agents.

1) Task Selection Agent: The involved PDRs are given below. 1) BEWT / 2) LWKR / 3) CR [6] / 4) EDD / 5) S/RT: the job with biggest EWT_g^P / least RT_g^P / smallest critical ratio/earliest D_g/smallest slack per remaining time is selected first. Therefore, the agent outputs a rule weight vector, which is shown in

$$a^{\mathrm{TS}}(t) = \left[w_{\mathrm{BEWT}},\ w_{\mathrm{LWKR}},\ w_{\mathrm{CR}},\ w_{\mathrm{EDD}},\ w_{S/RT}\right]. \quad (12)$$

2) Machine Selection Agent: The involved PDRs are given below. 1) SURM / 2) SDPT / 3) SCTM: the machine with smallest utilization rate / smallest DPT_{gxk} / smallest CT_{xk}^M is selected first. Therefore, the agent outputs a rule weight vector, which is shown in

$$a^{\mathrm{MS}}(t) = [w_{\mathrm{SURM}},\ w_{\mathrm{SDPT}},\ w_{\mathrm{SCTM}}]. \quad (13)$$

3) AGV Selection Agent: The involved PDRs are given below. (1) SPUT / (2) SCTA / (3) SCPT: the AGV with smallest PUT_{yg} / smallest CT_y^A / smallest $\mathrm{CT}_y^A + \mathrm{PUT}_{yg}$ is selected first. Therefore, the agent outputs a rule weight vector, and it is shown in

$$a^{\mathrm{AS}}(t) = [w_{\mathrm{SPUT}},\ w_{\mathrm{SCTA}},\ w_{\mathrm{SCPT}}]. \quad (14)$$

4) Action Decoding Algorithm: Based on the above actions, this article proposes an action decoding algorithm, and it is shown in Algorithm 1. First, the PDR priority set ($\mathrm{PDR}_{\mathrm{TS}}$, $\mathrm{PDR}_{\mathrm{MS}}$, $\mathrm{PDR}_{\mathrm{AS}}$) is calculated. Then, the priority vectors about all tasks, all machine candidates and all AGV candidates ($\mathrm{PV}_{\mathrm{TS}}$, $\mathrm{PV}_{\mathrm{MS}}$, $\mathrm{PV}_{\mathrm{AS}}$) can be obtained based on PDR weighting and PDR adjustment. Finally, three agents can give the appropriate task selection and resource allocation. Next, take the task selection agent (line 2 and 3) as an example for explanation.

1) Define PDR $\mathcal{p} \in \mathcal{P}$, where the PDR set $\mathcal{P} = \{\mathrm{BEWT},\ \mathrm{LWKR},\ CR,\ \mathrm{EDD},\ S/RT\}$.
2) Calculate the priority vector of PDR $\mathcal{p}$: $\mathrm{PV}_{\mathcal{p}} \in \mathbb{R}^{1\times \mathrm{NTP}}$. First, define the production attribute vector of PDR $\mathcal{p}$ as $[\mathrm{SV}_{\mathcal{p},g},\ g = 1,\ \ldots,\ \mathrm{NTP}]$. For example, the $\mathrm{SV}_{\mathcal{p},g}$ of BEWT or LWKR is EWT_g^P or $-\mathrm{RT}_g^P$. Then, $\mathrm{PV}_{\mathcal{p}}$ using Min-Max normalization is as follows:

$$\mathrm{PV}_{\mathcal{p}} = \left[\frac{\mathrm{SV}_{\mathcal{p},g} - \mathrm{SV}_{\mathcal{p}}^{\min}}{\mathrm{SV}_{\mathcal{p}}^{\max} - \mathrm{SV}_{\mathcal{p}}^{\min}},\ g = 1,\ \ldots,\ \mathrm{NTP}\right]. \quad (15)$$

3) Calculate $\mathrm{Sigmoid}(a^{\mathrm{TS}}(t)) \in \mathbb{R}^{1\times 5}$ based on $a^{\mathrm{TS}}(t)$

$$\mathrm{Sigmoid}\left(a^{\mathrm{TS}}(t)\right) = \left[\frac{1}{1+e^{-w_{\mathcal{p}}}},\ w_{\mathcal{p}} \in a^{\mathrm{TS}}(t)\right]. \quad (16)$$

Algorithm 1 Action Decoding

1: Task selection agent receives S_{TS}, and output $a^{JS}(t)$
2: Calculate $PDR_{TS} = [PV_{BEWT},\ PV_{LWKR}, PV_{CR},\ PV_{EDD}, PV_{S/RT}]$
3: Calculate $PV_{TS} = Norm\left(Sigmoid\left(a^{TS}(t)\right) \cdot PDR_{TS}\right) + w^{TS} \cdot PV_{SEAT}$
4: Consider machine breakdown and remove unavailable tasks
5: Select the task with highest priority from the task pool
6: Machine selection agent receives S_{MS}, and output $a^{MS}(t)$
7: **if** the selected task $T_g^P \in T_1^{Pool}$ **then**
8: Calculate $PDR_{MS} = [PV_{SURM},\ PV_{SDPT},\ PV_{SCTM}]$
9: Calculate $PV_{MS} = Norm\left(a^{MS}(t) \cdot PDR_{MS}\right) + w^{MS} \cdot PV_{SCTM}$
10: Remove failed machines
11: select the machine with highest priority from all machine candidates
12: **else**
13: Set $a^{MS}(t) = \delta_0$ ($\delta_0 = [0,\ 0,\ 0]$), and represents the do-nothing action
14: AGV selection agent receives S_{AS} and output $a^{AS}(t)$
15: Calculate $PDR_{AS} = [PV_{SPUT},\ PV_{SCTA},\ PV_{SCPT}]$
16: Calculate $PV_{AS} = Norm\left(a^{AS}(t) \cdot PDR_{AS}\right) + w^{AS} \cdot PV_{SCTA}$
17: Select the AGV with highest priority from all AGV candidates

4) Define $X = [x_g,\ g = 1,\ \ldots,\ NTP] = \mathrm{Sigmoid}(a^{\mathrm{TS}}(t)) \cdot \mathrm{PDR}_{\mathrm{TS}}$. Its matrix shape is $\mathbb{R}^{1\times 5} \cdot \mathbb{R}^{5\times NTP} = \mathbb{R}^{1\times NTP}$

$$x_g = \sum_{\mathcal{p}\in\mathcal{P}} \left(\frac{1}{1+e^{-w_{\mathcal{p}}}} \cdot \frac{\mathrm{SV}_{\mathcal{p},g} - \mathrm{SV}_{\mathcal{p}}^{\min}}{\mathrm{SV}_{\mathcal{p}}^{\max} - \mathrm{SV}_{\mathcal{p}}^{\min}}\right). \quad (17)$$

5) Based on Min-Max normalization, the half of $\mathrm{PV}_{\mathrm{TS}}$, i.e., $Y = \mathrm{Norm}(\mathrm{Sigmoid}(a^{\mathrm{TS}}(t)) \cdot \mathrm{PDR}_{\mathrm{TS}}) = \mathrm{Norm}(X) \in \mathbb{R}^{1\times NTP}$, is as follows:

$$Y = \left[\frac{x_g - x_g^{\min}}{x_g^{\max} - x_g^{\min}},\ g = 1,\ \ldots,\ NTP\right]. \quad (18)$$

6) Another half of $\mathrm{PV}_{\mathrm{TS}}$ is $Z = w^{\mathrm{TS}} \cdot \mathrm{PV}_{\mathrm{SEAT}} \in \mathbb{R}^{1\times NTP}$. w^{TS} is the adjustment weight. Rule SEAT represents that the task with smallest EAT_g^P is selected first, and the $\mathrm{SV}_{\mathcal{p},g}$ of SEAT is EAT_g^P. So Z is as follows:

$$Z = \left[w^{\mathrm{TS}} \cdot \frac{\mathrm{SV}_{\mathcal{p},g} - \mathrm{SV}_{\mathcal{p}}^{\min}}{\mathrm{SV}_{\mathcal{p}}^{\max} - \mathrm{SV}_{\mathcal{p}}^{\min}},\ g = 1, \ldots, NTP, \mathcal{p} = \mathrm{SEAT}\right]. \quad (19)$$

7) Finally, the decision basis $\mathrm{PV}_{\mathrm{TS}}$ of task selection agent is equal to $Y+Z$, and its shape is $\mathbb{R}^{1\times NTP}$. Each element P_g in $\mathrm{PV}_{\mathrm{TS}}$ is expressed by (20), and it represents the priority of task T_g^P in task pool

$$P_g = \frac{x_g - x_g^{\min}}{x_g^{\max} - x_g^{\min}} + w^{\mathrm{TS}} \cdot \frac{\mathrm{SV}_{\mathcal{p},g} - \mathrm{SV}_{\mathcal{p}}^{\min}}{\mathrm{SV}_{\mathcal{p}}^{\max} - \mathrm{SV}_{\mathcal{p}}^{\min}}. \quad (20)$$

In the action decoding of task selection agent, Y represents PDR weighting, which ensures that the agent can select different tasks based on different states (covering all possible

Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on March 06,2025 at 12:39:15 UTC from IEEE Xplore. Restrictions apply.

Algorithm 2 Reward Calculation

1: **for** job $i = 1$ **to** n **do**
2: **if** job J_i has been finished **then**
3: Calculate $EWT_i = \max(0,\ C_i - D_i) \cdot w_i$
4: **else**
5: Calculate $EWT_i = \max(0,\ t + RT_i - D_i) \cdot w_i$
6: **end if**
7: **next for**
8: Calculate $r_t = \sum_{i=1}^{n} EWT_i(t) - \sum_{i=1}^{n} EWT_i(t+1)$
9: Calculate $r_t = r_t - \alpha \cdot \left(\sum_{x=1}^{m} \sum_{k=1}^{N_x} ITM_{xk}\right) / \sum_{x=1}^{m} N_x$

solutions). Z represents PDR adjustment, which ensures that the selection logic of task selection agent is close to the rule SEAT, that is, the task selection agent tends to choose the task with smaller EAT_g^P. It can reduce the waiting time of AGV for the job. Furthermore, since all PDRs in $\mathcal{P}$ have a positive effect on reducing TTC, Sigmoid operation is used in (16) to transform $a^{\mathrm{TS}}(t)$ into a vector containing positive values. Therefore, the above action decoding can make task selection agent explore in a good trajectory direction, and finally improve the production efficiency.

Similar to line 2 and 3 in Algorithm 1 for task selection agent, line 8 and 9 represent the action decoding of machine selection agent, and line 15 and 16 represent the action decoding of AGV selection agent. w^{MS} and w^{AS} are the corresponding adjustment weights. Rule SCTM and SCTA enable the job to start processing and transporting as quickly as possible. Moreover, in the machine selection part, we set a do-nothing action δ_0 to deal with the situation that the selected task is only the last logistics task of a job, which does not need machine assignment.

E. Reward Function

The goal of DRL is to maximize cumulative rewards. The purpose of DFJSP-AGVs is to minimize TTC. To connect the two objectives, this article designs a reward function based on the estimated weighted tardiness (EWT) of jobs, while considering the machine idle time. It is shown in Algorithm 2. It should be noted that three agents act together at each decision point to achieve the resource allocation of a job and optimize TTC. Therefore, three agents share a reward.

In Algorithm 2, RT_i is the remaining time of job J_i, and its formula is shown in (5); $\mathrm{EWT}_i(t)$ is the EWT of job J_i at time t; ITM_{xk} is the idle time of machine M_{xk} between t and $t+1$; α represents the influence degree of machine idle time to reward. Line 7 represents that the agents are encouraged to make decisions that reduce tardiness of each transition. Line 8 indicates that the agents will try to keep all machines busy during training.

F. MAPPO-Based Training Algorithm

MAPPO is a classic MARL algorithm, and its details are provided in the "Section C of Supplementary Material." Therefore, based on the above designs, this article proposes a MAPPO-based training algorithm, as shown in Algorithm 3.

Algorithm 3 MAPPO-Based Training Algorithm

1: Initialize the actor networks of three agents $\pi_{TS}(\theta_{TS})$, $\pi_{MS}(\theta_{MS})$, $\pi_{AS}(\theta_{AS})$; the critic networks of three agents $v_{TS}(\phi_{TS})$, $v_{MS}(\phi_{MS})$, $v_{AS}(\phi_{AS})$; three memory buffers MB_{TS}, MB_{MS}, MB_{AS}
2: Initialize all hyperparameters of MAPPO, memory number $MN = 0$
3: **for** *episode* $= 1$: L **do**
4: Initialize the shop floor environment using problem *PROB*
5: **while** all jobs have not returned to the finished product area **do**
6: Implement Algorithm 1, and assign resources to the selected task
7: Execute three actions, and machines and AGVs keep running
8: Update task pool in action execution process
9: Observe the new state $\{SF_1,\ SF_2,\ SF_3,\ SF_4\}_{t+1}$
10: Calculate the reward r_t based on Algorithm 2
11: **if** the current state is the terminal state **then**
12: Capture the observations $\{SF_5,\ SF_6\}_{t+1}$
13: **end if**
14: Put $\left\{S_{TS}(t),\ a^{TS}(t),\ r_t,\ S_{TS}(t+1)\right\}$ into MB_{TS}
15: Put $\left\{S_{MS}(t),\ a^{MS}(t),\ r_t,\ S_{MS}(t+1)\right\}$ into MB_{MS}
16: Put $\left\{S_{AS}(t),\ a^{AS}(t),\ r_t,\ S_{AS}(t+1)\right\}$ into MB_{AS}
17: Update the memory number $MN = MN + 1$
18: **if** one disturbance event occurs **then**
19: Execute the corresponding strategy to handle it
20: **end if**
21: **end while**
22: Form trajectory τ using the transitions of current episode
23: Compute advantage estimate $\widehat{A}$ via GAE using PopArt
24: Compute reward-to-go $\widehat{R}$ on τ and normalize with PopArt
25: **if** $MN \geq$ memory update capacity MUC **then**
26: Adam update $\{\pi_{TS}(\theta_{TS}),\ \pi_{MS}(\theta_{MS}),\ \pi_{AS}(\theta_{AS})\}$ with (21)
27: Adam update $\{v_{TS}(\phi_{TS}),\ v_{MS}(\phi_{MS}),\ v_{AS}(\phi_{AS})\}$ with (23)
28: Regenerate a new training problem *PROB*
29: Clear memory buffers $\{MB_{TS}, MB_{MS}, MB_{AS}\}$ and set $MN = 0$
30: **end if**
31: **next for**

Lines 1 and 2 represent the initialization step. Lines 6–17 represent the agents' exploration and memory storage. Lines 18–20 represent the timely response to disturbance events. Lines 22–30 represent the update of all DNN models. Line 28 means that the agents use a randomly generated training problem in each epoch, which enhances the generalization of the trained DNNs.

In line 26, each actor network is trained to maximize the objective

Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on March 06,2025 at 12:39:15 UTC from IEEE Xplore. Restrictions apply.

$$L(\theta) = 1/B \cdot \sum_{i=1}^{B} (\min(r_i(\theta) \cdot A_i, \text{clip}(r_i(\theta),\ 1-\epsilon,\ 1+\epsilon) \cdot A_i) + \sigma \cdot S[\pi(o_i;\theta)]) \quad (21)$$

where B is the minibatch size; θ is one of $\{\theta_{\text{TS}},\ \theta_{\text{MS}},\ \theta_{\text{AS}}\}$; A_i is the advantage estimation in $\widehat{A}$; ϵ is the clipping parameter; σ is the entropy coefficient; S is the policy entropy; o_i is one of three observations $\{S_{\text{TS}}(t), S_{\text{MS}}(t), S_{\text{AS}}(t)\}$; $r_i(\theta)$ is the probability ratio, and it is calculated by

$$r_i(\theta) = \pi(a_i|o_i;\theta)/\pi(a_i|o_i;\theta_{\text{old}}). \quad (22)$$

Each critic network is trained to minimize the loss

$$L(\phi) = 1/B \cdot \sum_{i=1}^{B} (\max(H(v(s_i;\phi) - \widehat{R}_i) H(\text{clip}(v(s_i;\phi),\ v(s_i;\phi_{\text{old}}) - \epsilon,\ v(s_i;\phi_{\text{old}}) + \epsilon) - \widehat{R}_i))) \quad (23)$$

where ϕ is one of $\{\phi_{\text{TS}},\ \phi_{\text{MS}},\ \phi_{\text{AS}}\}$; $\widehat{R}_i$ is the discounted reward-to-go in $\widehat{R}$; s_i is the concatenated state, and it is calculated by

$$s_i = \text{concat}\Big(\{\text{SF}_1,\ \text{SF}_2,\ \text{SF}_3,\ \text{SF}_4\}, \mathcal{N}(0,\ \varepsilon^2)\Big) \quad (24)$$

where $\mathcal{N}(0,\ \varepsilon^2)$ is a gaussian noise vector. This article randomly generates a noise vector every 100 episodes. It can mitigate the multiagent policies overfitting [36], and more details are given in the "Section C of Supplementary Material." $H(x_1,\ x_2)$ is the Huber loss, and it can bring the improvement of training stability [27], which is given below

$$L_H(x_1,\ x_2) = \begin{cases} 0.5 \cdot (x_1 - x_2)^2, \text{ if } |x_1 - x_2| < 1 \\ |x_1 - x_2| - 0.5, \text{ otherwise.} \end{cases} \quad (25)$$

During the training process, each weight provided by each actor network is sampled from a Gaussian distribution where the mean is the output value and the standard deviation is 1. In this way, the agents can obtain different trajectories and learn from them. In the application process, each output value of each actor network is directly used as a PDR weight.

G. Strategy for Handling Disturbance Events

In essence, the DRL-based scheduling method is to form an effective mapping relationship between workshop state and action. Therefore, the strategy is to transform the occurrence of disturbance events into changes in the workshop state, which is beneficial for agents' perception. On this basis, the agents can learn to deal with disturbance events effectively through the training on relevant instances.

1) New Job Arrivals: When a new job arrives at the workshop, the decision system puts its first task into the task pool, so that the agents can assign machine and AGV to it.

2) Machine Breakdowns: When a machine fails, determine the corresponding disrupted tasks and put them into the task pool. These tasks include: 1) tasks being processed by the faulty machine; 2) tasks in the input buffer of the faulty machine; and 3) tasks of the above jobs that are assigned in advance. The agents will reallocate the disrupted tasks based on the workshop state and job urgency degree, which prevents these jobs from having a large tardiness due to machine failure.

When the machine has finished repairing, determine the corresponding disrupted tasks and put them into the task pool. These tasks include: 1) tasks in the input buffers of same-kind machines; 2) tasks of the above jobs that are assigned in advance; (note that exclude tasks that are being delivered). The agents will reallocate the disrupted tasks based on the workshop state and job urgency degree, which can prevent the repaired machine from being idle for a long time.

3) Job Rework: When a job rework event occurs, put the task corresponding to the rework operation into the task pool, and the agents can reassign machine and AGV to it.

4) Fuzzy Transportation Time: The DRL-based agents can learn to allocate manufacturing resources reasonably under fuzzy transportation time through the relevant training.

V. Numerical Experiments

In this section, the experimental settings are given first. Second, the training and evaluation are executed. Third, the ablation experiments on action decoding and reward function are implemented. Fourth, the comparison with composite PDRs is conducted. Fifthly, the proposed method is compared with GP and four popular DRL methods under various disturbance events. Finally, the behavior analysis of the trained DNN models and some discussions are provided.

We define a performance comparison index to conduct the ablation experiment and comparison experiments, i.e., superior proportion SP_y^x of method X to method Y, and it is as follows:

$$\text{SP}_Y^X = (\text{TTC}_Y - \text{TTC}_X)/\text{TTC}_X \cdot 100\% \quad (26)$$

where TTC_X and TTC_Y are the TTC of method X and method Y respectively. The larger SP_y^x, the better method X is compared to method Y.

Moreover, due to the space limitation, the transportation time matrix, hyperparameter tuning, the details of comparison methods (composite PDRs, GP, four DRL methods), some detailed ablation/comparison results, real-time performance test, and plant simulation model are given in the "Section D–I of Supplementary Material."

A. Experimental Settings

This article used Tecnomatix Plant Simulation software to establish a model based on the defined workshop layout, which is shown in Fig. 4. It is used to test the transportation times of AGV between different locations. Meanwhile, it is used to measure the probability and amplitude of time fluctuations considering path congestion, AGV acceleration, AGV deceleration and other factors.

The production configurations are given below. 1) Number of machine types m: 8, N_x: [2, 3, 2, 3, 2, 3, 3, 2]; 2) Number of AGVs l: 10; 3) Number of initial jobs: 20; 4) Number of operations for a job: randi [4, 10]; 5) Processing time of each operation: randi [10, 30] (unit: min); 6) The range of transportation time for each delivery task is in [17, 204] (unit: second); 7) The job weight: $w_i \in \{1,\ 2,\ 4\}$, and the

Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on March 06,2025 at 12:39:15 UTC from IEEE Xplore. Restrictions apply.

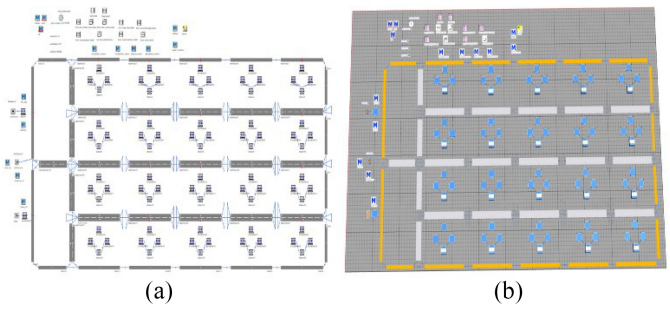

Fig. 4. Plant simulation model. (a) 2-D model. (b) 3-D model.

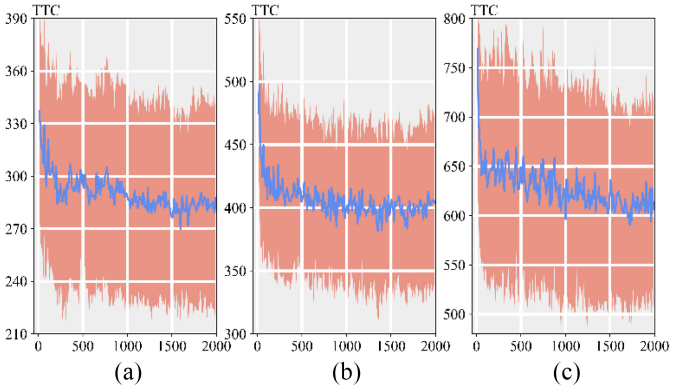

Fig. 5. Evaluation results under different problem settings. (a) Problem setting 1. (b) Problem setting 2. (c) Problem setting 3.

corresponding proportions are 20%, 60%, and 20%; and 8) Due date tightness DDT_i: randf[1, DDT_H], $DDT_H \in \{1.5,\ 2\}$.

The settings of disturbance events are as follows: 1) Utilization rate of workshop: $U \in \{0.9,\ 0.95\}$; 2) Number of new job arrivals J_{new}: randi[100, 200]; 3) Mean time to repair: $\text{MTTR} = 3 \cdot T^{\text{PT}}$; 4) Breakdown level of workshop: $A_g \in \{0.025,\ 0.05\}$; 5) Probability of job rework JRP: $\{0.03,\ 0.06\}$; 6) Through the established Plant Simulation model considering path occupancy, AGV acceleration and AGV deceleration, we found that when the AGV performs a task, the probability of transportation time fluctuation is about 10%, and the amplitude is [0, 10%]. It should be noted that the symbols "randi" and "randf" denote uniform distribution of integers and real numbers, respectively.

B. Training and Evaluation

The proposed method is implemented by PyTorch and runs on a server with Intel Xeon W-3365 CPU @ 2.70 GHz and 125-GB RAM. This article adopts the random search [37] to achieve the best-performing hyperparameters.

To evaluate the convergence and generalization of the proposed method, we set three problem settings with different parameters. Each problem setting contains five randomly generated instances. The DNNs saved during the training process were evaluated on these instances, and the results are shown in Fig. 5. Each blue curve represents the mean of total tardiness costs on 5 instances, and the 90% confidence interval is shown shaded in red. The proposed method can decline and converge on instances with different scales, which indicates that it has generalization. It should be noted that due to the addition of rule adjustment, the agents only explore within a solution space filled with good schemes, so the decrease in the curve is relatively small. This article selects the best-performing DNNs from the convergence stage for the next comparison experiments.

TABLE III
ABLATION OF FOUR STRATEGIES IN ACTION DECODING

MAPPO Model	Sigmoid	TS adjustment	MS adjustment	AS adjustment
1	-0.4/7.5	1125.4/593.1	63.6/35.5	8468.6/5485.6
2	3.2/5.2	1164.3/589.8	51.2/36.8	7076/4353.8
3	4.5/9.9	1173.2/619.8	39.1/23.8	6067.6/3392.7
4	2.7/8.5	1245.9/686.6	30.1/15.6	6617.2/6130.3
5	3.8/9.4	1236.9/675.9	32.7/24.4	7362.8/5591.9
6	0.2/6.5	1256.4/660.9	31.4/22.4	9361.1/4663.3
7	4.5/10.6	1262.2/728.2	27.6/25.4	8379.5/5697.5
8	0.9/6.2	1338.1/728.9	23.7/14.8	7215.2/5819.3
9	1.6/6.8	1382.4/776.6	23.6/15.7	8949.3/6736.3
10	2.5/7.7	1498.7/714	14.9/12.3	7503.7/5408.3

C. Ablation Experiment

1) Ablation on Four Strategies in Action Decoding: To demonstrate the effectiveness of four strategies in action decoding, 10 MAPPO models and 20 instances were randomly generated for ablation experiments. Here X is the proposed method, and Y is method X without one strategy. In terms of four strategies, the SP_Y^X of each MAPPO model on 20 instances is shown in Table III, which is represented by mean/standard deviation. TS adjustment, MS adjustment and AS adjustment refer to the rule adjustments among three agents. It can be observed that each strategy has brought performance improvements to the proposed method. The reasons are as follows: 1) Sigmoid operation can effectively integrate various tardiness-oriented PDRs; 2) some tasks in the task pool are put in advance, and their jobs may be being processed on the machine. If these tasks are selected, a long-time wait of AGV will occur. Rule SEAT can avoid this situation; 3) rule SCTM can balance the load of each machine and try to keep each machine running for a long time; 4) rule SCTA can enable each logistics task to be executed as soon as possible.

2) Ablation on Reward Function: At the same time, this article conducted the ablation experiment on reward function. Here X is the proposed reward, and Y is a commonly used EWTs difference reward [28], [31], i.e., line 7 in Algorithm 2. X adds the part of machine idle time on the basis of Y, i.e., line 8 in Algorithm 2. This article carried out two trainings based on X and Y separately. The trained models were tested on 30 randomly generated instances, and the results are shown in Fig. 6. The proposed method achieves better results in 83.3% of instances. The number on each instance represents the corresponding SP_Y^X. Fig. 6 also shows the box plot of 30 SP_Y^X, with an average of 5.66%. It can be found that by adding machine idle time, the proposed reward can better guide the training and learning of agents.

D. Comparison With Composite PDRs

PDR is a popular real-time scheduling method. Based on previous research, this article selected some common job

Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on March 06,2025 at 12:39:15 UTC from IEEE Xplore. Restrictions apply.

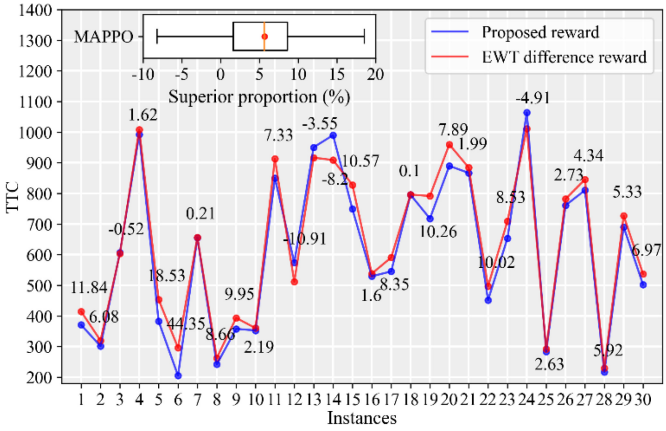

Fig. 6. Ablation of machine idle time in reward function.

TABLE IV
COMPARISON WITH COMPOSITE PDRS CONSIDERING NEW JOB ARRIVALS

J_{new}	DDT	U	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	R_{10}
100	1.5	0.9	37	59	40	33	55	33	58	39	49	79
		0.95	15	28	25	16	29	17	27	26	30	55
	2	0.9	38	57	44	25	41	23	41	34	46	71
		0.95	31	58	47	19	45	25	48	34	41	61
150	1.5	0.9	36	63	46	29	51	27	66	40	52	83
		0.95	23	43	26	21	33	19	32	24	35	56
	2	0.9	50	96	78	43	74	41	76	59	68	118
		0.95	46	77	70	32	66	30	62	72	55	113
200	1.5	0.9	45	71	43	30	64	34	70	52	57	103
		0.95	33	50	39	29	48	29	50	39	46	69
	2	0.9	19	46	38	20	46	17	46	36	34	72
		0.95	32	63	40	28	55	25	49	38	49	79

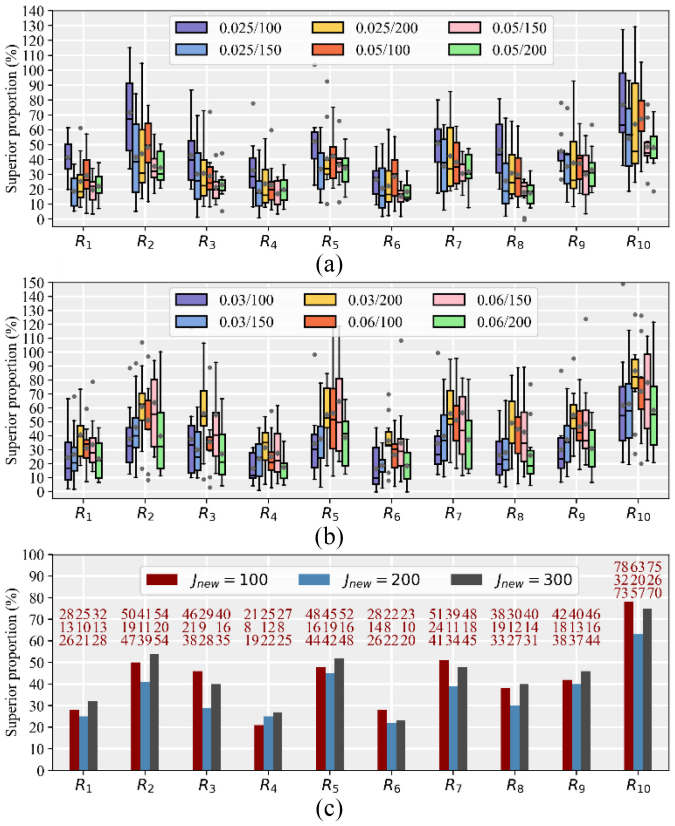

Fig. 7. Comparison with composite PDRs under various disturbance. (a) Robustness comparison under new job arrivals and machine breakdowns. (b) Robustness comparison under new job arrivals and job reworks. (c) Robustness comparison under new job arrivals and fuzzy transportation time.

sorting PDRs, machine selection PDRs and AGV selection PDRs, and execute combination to form 10 best-performing composite PDRs (R_i, $i = 1, \ldots, 10$) for experiments.

Here, X is the proposed method, and Y is one composite PDR. Based on the three parameters (J_{new}, DDT_H, U), 12 sets of instances were generated that consider the new job arrivals, and each set contains five instances. The results are shown in Table IV where each number represents the mean of the corresponding 5 SP_Y^X. The proposed method can obtain better solutions than composite PDR on different instances. The performance improvement is at least 14.58%, with most exceeding 20%. In summary, the agents can make appropriate decisions based on real-time workshop information through training, thereby achieving efficient production.

In order to verify the robustness of the proposed method, this article conducted comparison experiments under four disturbance events. Fig. 7(a) is presented in six sets of parameters (A_g/J_{new}), Fig. 7(b) is presented in six sets of parameters (JRP/J_{new}), and Fig. 7(c) is presented in three sets of parameters (J_{new}). Each set contains 10 randomly generated instances. The comparison results between the proposed method and composite PDRs are shown in Fig. 7. In box plots of Fig. 7(a), the proposed method is lower than R_8 by -0.54% on only one instance. In box plots of Fig. 7(b), the proposed method can achieve better results on all instances. In Fig. 7(c), fuzzy transportation time leads to the fact that a method can obtain different solutions on one instance. Therefore, a method runs 5 times on an instance, and the mean is taken as the final result. Each bar represents the mean of the corresponding 10 SP_Y^X, and the three numbers above it represent the mean, standard deviation, and median respectively. The mean of all SP_Y^X is much greater than 0. In summary, the proposed method can obtain better schemes on instances under various disturbance events, and the performance improvement is mostly more than 20%. This indicates that the proposed method has robustness and can ensure stable production through the designed strategy.

E. Comparison With GP and DRL-Based Methods

In order to further demonstrate the superiority, this article selects the well-known GP [21] and four DRL-based methods (Luo et al. [4], Park and Park [27], Li et al. [21], Lei et al. [26]) for comparison.

Here, X is defined as the proposed method, and Y is one of comparison methods mentioned above. The comparison results on 60 instances considering new job arrivals are shown in Table V, which is presented in a similar manner to Table IV. It can be found that compared with other real-time scheduling methods, the proposed method has superiority and can obtain better schemes on most instances. The superior proportion mostly exceeds 5%. This indicates that the proposed method can provide appropriate PDR weight vectors according to the real-time workshop state, so as to make reasonable allocation of manufacturing resources.

Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on March 06,2025 at 12:39:15 UTC from IEEE Xplore. Restrictions apply.

TABLE V
COMPARISON WITH GP AND DRL CONSIDERING NEW JOB ARRIVALS

J_{new}	DDT_H	U	GR_1	GR_2	GR_3	GR_4	Luo	Park	Li	Lei
100	1.5	0.9	21	28	21	24	41	6	18	-4
		0.95	12	9	10	15	19	7	14	4
	2	0.9	22	27	25	23	42	18	19	9
		0.95	20	26	20	19	27	14	14	3
150	1.5	0.9	26	27	21	21	39	14	19	4
		0.95	18	16	13	11	26	6	12	8
	2	0.9	36	48	36	39	66	20	34	6
		0.95	28	28	21	34	50	11	29	8
200	1.5	0.9	28	29	29	27	47	11	23	3
		0.95	23	21	18	20	33	14	25	4
	2	0.9	17	20	14	17	31	8	15	0
		0.95	22	33	20	16	28	13	23	0

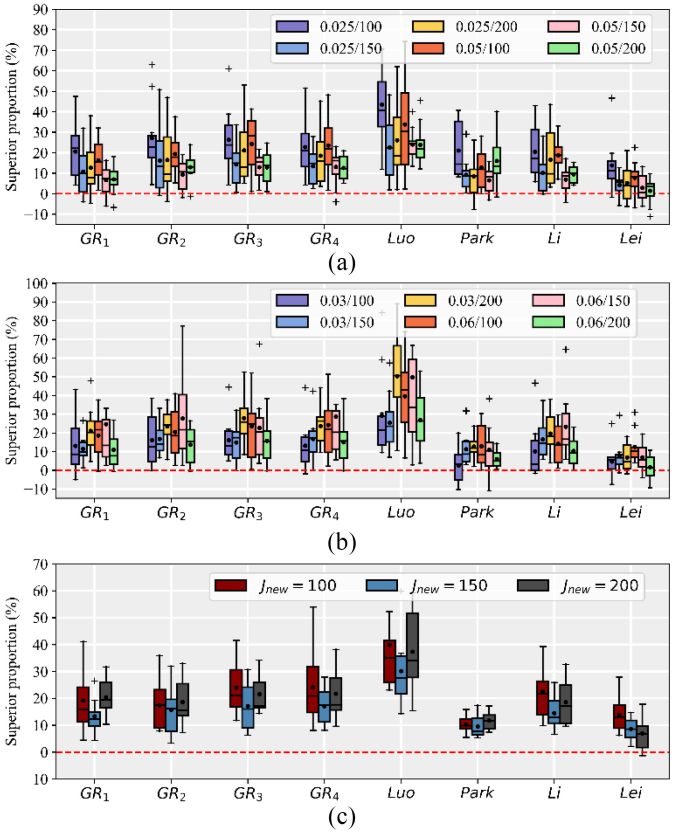

Fig. 8. Comparison with GP and DRL under various disturbance events. (a) Robustness comparison under new job arrivals and machine breakdowns. (b) Robustness comparison under new job arrivals and job reworks. (c) Robustness comparison under new job arrivals and fuzzy transportation time.

Similar to Fig. 7, Fig. 8 shows the comparison results of 150 instances considering various disturbance events, which are presented in the form of SP_Y^X box plots. All methods are retrained because four disturbance events are considered. In three subgraphs of Fig. 8, the proposed method only achieves poor solutions on a few instances compared to other methods, with the worst amplitude being near –10%. However, the proposed methods can obtain better solutions on most instances. The performance improvement for GP rules, Luo, Park, and Li is mostly more than 10%, while the performance improvement for Lei is mostly more than 5%. This proves that the proposed method can implement the reasonable reallocation based on the job urgency degree and resource information when disturbance events occur, thereby maintaining efficient production.

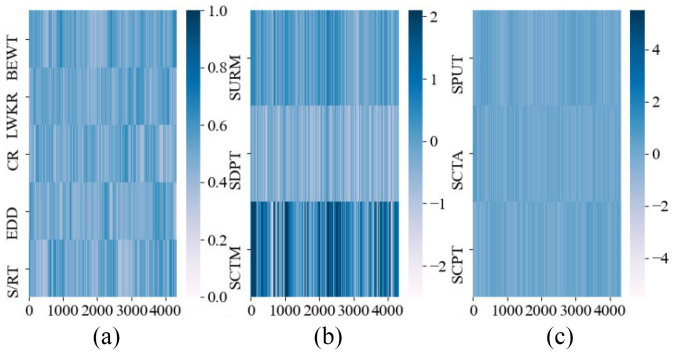

Fig. 9. Heatmap of actions on three instances. (a) Task selection agent. (b) Machine selection agent. (c) AGV selection agent.

F. Behavior Analysis and Discussions

We tested the trained DNN models on three instances (100 jobs, 150 jobs, 200 jobs), and collected the PDR weight vectors output by agents. Its heatmap is shown in Fig. 9, where the output of task selection agent has been reprocessed by Sigmoid operation. It can be found that the machine selection agent is more biased to make decisions based on SCTM, because rule SCTM can effectively balance the loads of all machines and enable each job to start processing as soon as possible. The job selection agent and AGV selection agent can favor different rules at different moments under PDR adjustment, thus achieving reasonable decision-making.

The reasons for the poor performance of the above real-time scheduling methods are as follows.

1) One PDR is only suitable for solving partial scheduling instances, and the quality of solution is not satisfactory.
2) Although GP overcomes the shortcoming of PDR through evolution, it still remains a fixed decision logic, which does not map with the constantly changing workshop state during the scheduling process.
3) Due to the rule-based action space, method Luo or method Li can only select the job with maximum/minimum attribute at each decision point. As a result, these methods miss many good solutions and the training effect is poor.
4) Although end-to-end or special action design can explore all possible solutions, method Park or method Lei is easy to get lost in the huge solution space of DFJSP-AGVs.

In summary, PDR weighting ensures that the proposed method does not miss good solutions, and PDR adjustment gives the agents the ability to obtain good solutions through directional exploration. Ultimately, the agents can make appropriate decisions based on real-time workshop state through training, and effectively deal with various disturbance events.

VI. CONCLUSION AND FUTURE WORK

Dynamic scheduling method with timely response capability is an important part of future factory. This article proposes a MARL-based real-time scheduling method for DFJSP-AGVs to minimize the total tardiness cost. First, a

Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on March 06,2025 at 12:39:15 UTC from IEEE Xplore. Restrictions apply.

real-time scheduling framework is proposed, and a multiagent scheduling architecture is designed to realize the task selection, machine allocation and AGV allocation. Second, based on the PDR weighting and PDR adjustment, the action space and action decoding algorithm are proposed for the full exploration and improvement of learning efficiency. Third, a generic state space, a reward function considering machine idle time, and a strategy for handling four disturbance events are designed to improve performance and enhance robustness.

Extensive experiments demonstrate that the action decoding and reward function are effective, and the proposed method is superior to composite PDRs, GP and four DRL methods. When various disturbance events occur, the proposed method can provide the appropriate resource allocation in time through the trained state-action mapping relationship, and ensures the stable and efficient operation of workshop.

However, compared to other methods, the proposed method cannot achieve the best solution on all instances. The superiority and stability of algorithm deserve further exploration. Therefore, future research directions are as follows: 1) integrate more domain knowledge into the action space decoding; 2) use state-of-the-art deep learning models; and 3) design better reward function based on domain knowledge or inverse reinforcement learning.

REFERENCES

[1] S. Guo, T.-M. Choi, B. Shen, and S. Jung, "Coordination and enhancement schemes for quick response mass customization supply chains with consumer returns and salvage value considerations," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 2, pp. 673–685, Feb. 2020, doi: 10.1109/TSMC.2017.2766207.

[2] A. Goli, E. B. Tirkolaee, and N. S. Aydin, "Fuzzy integrated cell formation and production scheduling considering automated guided vehicles and human factors," *IEEE Trans. Fuzzy Syst.*, vol. 29, no. 12, pp. 3686–3695, Dec. 2021, doi: 10.1109/TFUZZ.2021.3053838.

[3] Z. Li, H. Sang, Q. Pan, K. Gao, Y. Han, and J. Li, "Dynamic AGV scheduling model with special cases in matrix production workshop," *IEEE Trans. Ind. Informat.*, vol. 19, no. 6, pp. 7762–7770, Jun. 2023, doi: 10.1109/TII.2022.3211507.

[4] S. Luo, L. Zhang, and Y. Fan, "Real-time scheduling for dynamic partial-no-wait multiobjective flexible job shop by deep reinforcement learning," *IEEE Trans. Autom. Sci. Eng.*, vol. 19, no. 4, pp. 3020–3038, Oct. 2022, doi: 10.1109/TASE.2021.3104716.

[5] M. Li, M. Li, D. Guo, T. Qu, and G. Q. Huang, "Real-time data-driven out-of-order synchronization for production and intralogistics in multiresource-constrained assembly systems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 53, no. 12, pp. 7513–7525, Dec. 2023, doi: 10.1109/TSMC.2023.3298927.

[6] R. Liu, R. Piplani, and C. Toro, "Deep reinforcement learning for dynamic scheduling of a flexible job shop," *Int. J. Prod. Res.*, vol. 60, no. 13, pp. 4049–4069, Jul. 2022, doi: 10.1080/00207543.2022.2058432.

[7] Y. Yao et al., "A novel mathematical model for the flexible job-shop scheduling problem with limited automated guided vehicles," *IEEE Trans. Autom. Sci. Eng.*, early access, Jan. 30, 2024, doi: 10.1109/TASE.2024.3356255.

[8] Z. Pan, D. Lei, and L. Wang, "A bi-population evolutionary algorithm with feedback for energy-efficient fuzzy flexible job shop scheduling," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 8, pp. 5295–5307, Aug. 2022, doi: 10.1109/TSMC.2021.3120702.

[9] J. Branke, S. Nguyen, C. W. Pickardt, and M. Zhang, "Automated design of production scheduling heuristics: A review," *IEEE Trans. Evol. Comput.*, vol. 20, no. 1, pp. 110–124, Feb. 2016, doi: 10.1109/TEVC.2015.2429314.

[10] 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, doi: 10.1016/j.asoc.2020.106208.

[11] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015, doi: 10.1038/nature14236.

[12] C.-L. Liu and T.-H. Huang, "Dynamic job-shop scheduling problems using graph neural network and deep reinforcement learning," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 53, no. 11, pp. 6836–6848, Nov. 2023, doi: 10.1109/TSMC.2023.3287655.

[13] S. M. Homayouni and D. B. M. M. Fontes, "Production and transport scheduling in flexible job shop manufacturing systems," *J. Glob. Optim.*, vol. 79, no. 2, pp. 463–502, Feb. 2021, doi: 10.1007/s10898-021-00992-6.

[14] A. Ham, "Transfer-robot task scheduling in flexible job shop," *J. Intell. Manuf.*, vol. 31, no. 7, pp. 1783–1793, Oct. 2020, doi: 10.1007/s10845-020-01537-6.

[15] Q. Zhang, H. Manier, and M.-A. Manier, "A modified shifting bottleneck heuristic and disjunctive graph for job shop scheduling problems with transportation constraints," *Int. J. Prod. Res.*, vol. 52, no. 4, pp. 985–1002, Feb. 2014, doi: 10.1080/00207543.2013.828164.

[16] A. Amirteimoori, E. B. Tirkolaee, V. Simic, and G.-W. Weber, "A parallel heuristic for hybrid job shop scheduling problem considering conflict-free AGV routing," *Swarm Evol. Comput.*, vol. 79, Jun. 2023, Art. no. 101312, doi: 10.1016/j.swevo.2023.101312.

[17] Y. Yao, C. Wang, X. Li, and L. Gao, "A knowledge-driven hybrid algorithm for solving the integrated production and transportation scheduling problem in job shop," *IEEE Trans. Intell. Transp. Syst.*, early access, Dec. 17, 2024, doi: 10.1109/TITS.2024.3511998.

[18] Z. Pan, L. Wang, J. Zheng, J.-F. Chen, and X. Wang, "A learning-based multipopulation evolutionary optimization for flexible job shop scheduling problem with finite transportation resources," *IEEE Trans. Evol. Comput.*, vol. 27, no. 6, pp. 1590–1603, Dec. 2023, doi: 10.1109/TEVC.2022.3219238.

[19] Q. Liu, C. Wang, X. Li, and L. Gao, "An improved genetic algorithm with modified critical path-based searching for integrated process planning and scheduling problem considering automated guided vehicle transportation task," *J. Manuf. Syst.*, vol. 70, pp. 127–136, Oct. 2023, doi: 10.1016/j.jmsy.2023.07.004.

[20] C. Sahin, M. Demirtas, R. Erol, A. Baykasoğlu, and V. Kaplanoğlu, "A multiagent based approach to dynamic scheduling with flexible processing capabilities," *J. Intell. Manuf.*, vol. 28, no. 8, pp. 1827–1845, Dec. 2017, doi: 10.1007/s10845-015-1069-x.

[21] Y. Li, W. Gu, M. Yuan, and Y. Tang, "Real-time data-driven dynamic scheduling for flexible job shop with insufficient transportation resources using hybrid deep Q network," *Robot. Comput.-Integr. Manuf.*, vol. 74, Apr. 2022, Art. no. 102283, doi: 10.1016/j.rcim.2021.102283.

[22] Y. Luo, W. Li, W. Yang, and G. Fortino, "A real-time edge scheduling and adjustment framework for highly customizable factories," *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5625–5634, Aug. 2021, doi: 10.1109/TII.2020.3044698.

[23] L. Cai, W. Li, Y. Luo, and L. He, "Real-time scheduling simulation optimisation of job shop in a production-logistics collaborative environment," *Int. J. Prod. Res.*, vol. 61, no. 5, pp. 1373–1393, Mar. 2023, doi: 10.1080/00207543.2021.2023777.

[24] M. Zhang, L. Wang, F. Qiu, and X. Liu, "Dynamic scheduling for flexible job shop with insufficient transportation resources via graph neural network and deep reinforcement learning," *Comput. Ind. Eng.*, vol. 186, Dec. 2023, Art. no. 109718, doi: 10.1016/j.cie.2023.109718.

[25] C.-L. Liu, C.-J. Tseng, T.-H. Huang, and J.-W. Wang, "Dynamic Parallel machine scheduling with deep Q-network," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 53, no. 11, pp. 6792–6804, Nov. 2023, doi: 10.1109/TSMC.2023.3289322.

[26] K. Lei, P. Guo, Y. Wang, J. Zhang, X. Meng, and L. Qian, "Large-scale dynamic scheduling for flexible job-shop with random arrivals of new jobs by hierarchical reinforcement learning," *IEEE Trans. Ind. Informat.*, vol. 20, no. 1, pp. 1007–1018, Jan. 2024, doi: 10.1109/TII.2023.3272661.

[27] I.-B. Park and J. Park, "Scalable scheduling of semiconductor packaging facilities using deep reinforcement learning," *IEEE Trans. Cybern.*, vol. 53, no. 6, pp. 3518–3531, Jun. 2023, doi: 10.1109/TCYB.2021.3128075.

[28] Y. Gui, D. Tang, H. Zhu, Y. Zhang, and Z. Zhang, "Dynamic scheduling for flexible job shop using a deep reinforcement learning approach," *Comput. Ind. Eng.*, vol. 180, Jun. 2023, Art. no. 109255, doi: 10.1016/j.cie.2023.109255.

[29] X. Wang et al., "Dynamic scheduling of tasks in cloud manufacturing with multi-agent reinforcement learning," *J. Manuf. Syst.*, vol. 65, pp. 130–145, Oct. 2022, doi: 10.1016/j.jmsy.2022.08.004.

Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on March 06,2025 at 12:39:15 UTC from IEEE Xplore. Restrictions apply.

[30] 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, doi: 10.1109/TASE.2019.2956762.
[31] Y. Gui, Z. Zhang, D. Tang, H. Zhu, and Y. Zhang, "Collaborative dynamic scheduling in a self-organizing manufacturing system using multi-agent reinforcement learning," *Adv. Eng. Informat.*, vol. 62, Oct. 2024, Art. no. 102646, doi: 10.1016/j.aei.2024.102646.
[32] X. Wang, L. Zhang, T. Lin, C. Zhao, K. Wang, and Z. Chen, "Solving job scheduling problems in a resource preemption environment with multi-agent reinforcement learning," *Robot. Comput.-Integr. Manuf.*, vol. 77, Oct. 2022, Art. no. 102324, doi: 10.1016/j.rcim.2022.102324.
[33] M. Wang, J. Zhang, P. Zhang, L. Cui, and G. Zhang, "Independent double DQN-based multi-agent reinforcement learning approach for online two-stage hybrid flow shop scheduling with batch machines," *J. Manuf. Syst.*, vol. 65, pp. 694–708, Oct. 2022, doi: 10.1016/j.jmsy.2022.11.001.
[34] X. Zhu, J. Xu, J. Ge, Y. Wang, and Z. Xie, "Multi-task multi-agent reinforcement learning for real-time scheduling of a dual-resource flexible job shop with robots," *Processes*, vol. 11, no. 1, p. 267, Jan. 2023, doi: 10.3390/pr11010267.
[35] C. Yu et al., "The surprising effectiveness of PPO in cooperative multi-agent games," in *Proc. 36th Int. Conf. Neural Inf. Process. Syst.*, 2022, pp. 1–14.
[36] S. Wang, W. Chen, J. Hu, S. Hu, and L. Huang, "Noise-regularized advantage value for multi-agent reinforcement learning," *Mathematics*, vol. 10, no. 15, p. 2728, Aug. 2022, doi: 10.3390/math10152728.
[37] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, Feb. 2012.

Yuxin Li received the B.S. and M.S. degrees in mechanical engineering from the College of Mechanical and Electrical Engineering, Hohai University, Changzhou, China, in 2019 and 2022, respectively. He is currently pursuing the Ph.D. degree in mechanical engineering with the School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Wuhan, China.

His current research interests are focused on production scheduling, deep reinforcement learning, and large language model.

Qingzheng Wang received the B.S. degree in mechanical engineering from Qingdao University, Qingdao, China, in 2021. He is currently pursuing the M.S. degree in industrial engineering and management with the School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Wuhan, China.

His current research interests primarily revolve around production and logistics integration scheduling, metaheuristic, and multiobjective algorithms.

Xinyu Li (Member, IEEE) received the Ph.D. degree in industrial engineering from the Huazhong University of Science and Technology (HUST), Wuhan, China, 2009.

He is a Professor with the Department of Industrial and Manufacturing Systems Engineering, State Key Laboratory of Intelligent Manufacturing Equipment and Technology, School of Mechanical Science and Engineering, HUST. He had published more than 100 refereed papers. His research interests include intelligent algorithm, big data, and machine learning etc.

Prof. Li currently serves as an Associate Editor for IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING and the *Expert Systems With Applications*.

Liang Gao (Senior Member, IEEE) received the Ph.D. degree in mechatronic engineering from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2002.

He is a Professor with the Department of Industrial and Manufacturing System Engineering, State Key Laboratory of Intelligent Manufacturing Equipment and Technology, School of Mechanical Science and Engineering, HUST. He had published more than 400 refereed papers. His research interests include operations research and optimization, big data, and machine learning.

Prof. Gao currently serves as the Co-Editor-in-Chief for *IET Collaborative Intelligent Manufacturing*, and an Associate Editor for *Swarm and Evolutionary Computation* and the *Journal of Industrial and Production Engineering*.

Ling Fu received the Ph.D. degree in applied physics from Stanford University, Stanford, CA, USA, in 2010.

She is the Principal Key Expert Research Scientist of Siemens Technology, Alpharetta, GA, USA, and she drives the digital twin-based solutions in the respective field with Siemens Technology and across businesses. Since, she joined Siemens in 2010, she has taken various technology innovation responsibilities at the corporate level, including in-house research and development activities, new product/solution creation and incubation, open innovation ecosystem engagement, and strategic investment.

Yanbin Yu received the M.S. degree in computer science from Shanghai Jiao Tong University, Shanghai, China, in 2008.

He is the Head of Research Group, Siemens Technology, Beijing, China, where he leads the development of digital twin and simulation technology. Prior to this role, he served in various positions with Siemens, including the Product Manager, the Project Manager, and the Venture Technology Manager. He has over ten years of experience in open innovation, as well as five years of experience in digital twin technology research. His expertise includes digital twin and simulation technology, and enterprize digital transformation and upgrading.

Wei Zhou received the M.S. degree in industrial engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2019.

He is a Research Scientist with the Production Digitalization and Optimization Department, Siemens Technology, Beijing, China. His areas of research include production system modeling and simulation, logistic optimization, digital twin in production, and advanced planning and scheduling.

Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on March 06,2025 at 12:39:15 UTC from IEEE Xplore. Restrictions apply.