Computers & Industrial Engineering 186 (2023) 109718

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

Dynamic scheduling for flexible job shop with insufficient transportation resources via graph neural network and deep reinforcement learning

Min Zhang, Liang Wang *, Fusheng Qiu, Xiaorui Liu

School of Mechanical Engineering, Tongji University, Shanghai, China

ARTICLE INFO

Keywords:
Flexible job shop scheduling
Insufficient transportation resources
Dynamic scheduling
Graph neural network
Deep reinforcement learning

ABSTRACT

The smart workshop is a powerful tool for manufacturing companies to reduce waste and improve production efficiency through real-time data analysis for self-organized production. Automated Guided Vehicles (AGVs) have been widely used for material handling in smart workshop due to their high degree of autonomy, flexibility and powerful end-to-end capability to cope with logistics tasks in production modes such as multiple species and small batch, and mass customization. However, the highly dynamic, complex and uncertain nature of the smart job shop environment makes production scheduling with insufficient transportation resources in mind a challenge. To this end, this paper addresses the dynamic flexible job shop scheduling problem with insufficient transportation resources (DFJSP-ITR), and learn high-quality priority dispatching rule (PDR) end-to-end to minimize makespan by the proposed deep reinforcement learning (DRL) method. To achieve integrated decision making for operation, machine and AGV, an architecture based on heterogeneous graph neural network (GNN) and DRL is proposed. Considering the impact of different AGV distribution methods on the scheduling objective, this paper compares two different AGV distribution methods. Experiments show that the proposed method has superiority and good generalization ability compared with the current PDRs-based methods regardless of the AGV distribution strategy used.

1. Introduction

The production scheduling problem is a complex combinatorial optimization problem, which mainly consists of three parts: constraints condition, scheduling objective and scheduling scheme (Li et al., 2022). Production scheduling is not only an important tool to improve job shop productivity, but also can effectively reduce energy consumption, shorten production cycles and increase customer satisfaction (Li et al., 2020). The flexible job shop scheduling problem (FJSP) is an important branch of production scheduling, which is Nondeterministic Polynomial hard (NP-hard) (Li et al., 2022). FJSP allows operations to be processed on any one machine in a set of compatible machines, so it is critical to select the appropriate processing machine for each operation. AGVs have been widely used for material handling in flexible manufacturing system due to their high degree of autonomy, flexibility and powerful end-to-end capability (Hu et al., 2020). Many scholars have studied the production scheduling problem with AGV transport constraints, but the number of AGVs is usually set to be sufficient (Li et al., 2022). And most studies on production scheduling problems with AGV transportation constraints do not consider the dynamic events that may occur in actual production. Since the distribution process of AGVs is crucial to speed up the material flow to shorten the production cycle and improve the intelligence of shop floor logistics, and dynamic events often occur in actual production. Therefore, the study of DFJSP-ITR is necessary.

For small-scale static job shop scheduling problem (JSP) with known task information, many researchers used mathematical programming (Caumond et al., 2009; Heger & Voss, 2018) or branch and bound method (Brucker et al., 1994) to solve the problem. In complex real-world manufacturing environments, obtaining all task information in advance can be challenging. In solving dynamic scheduling problems, the most common methods are PDRs and *meta*-heuristic algorithms. While the PDRs can react immediately when dynamic events occur, they do not even guarantee local optimum (Luo, 2020). In contrast, meta-heuristics, such as genetic algorithm (Kundakcı & Kulak, 2016), artificial bee colony algorithm (Gao et al., 2015; Shahgholi Zadeh et al., 2019) and particle swarm optimization (Ning et al., 2016; Nouiri et al., 2018), by decomposing the dynamic scheduling problem into a series of static sub-problems to be solved, can yield higher quality scheduling

* Corresponding author.
E-mail addresses: 15070271366@163.com (M. Zhang), liangwang@tongji.edu.cn (L. Wang), 1911401@tongji.edu.cn (F. Qiu), 2232696@tongji.edu.cn (X. Liu).

https://doi.org/10.1016/j.cie.2023.109718
Received 15 June 2023; Received in revised form 26 October 2023; Accepted 29 October 2023
Available online 31 October 2023

solutions yet are very time consuming (Lou et al., 2012). In addition, hybrid algorithms combine the advantages of fast convergence and robustness of each algorithm in solution search (Yildiz, 2013), so they have been used by some researchers. Qin et al. (2019) constructed a multi-objective FJSP model with processing interval and job transportation time constraints and solved it using a hybrid discrete multi-objective gray wolf optimizer. Peng et al. (2022) established a multi-objective FJSP model constrained by job transportation time and learning effect, and developed a hybrid discrete multi-objective empire competition algorithm to solve the model.

Efficient PDR often require a lot of professional knowledge and trial-and-error (Zhang et al., 2020), while reinforcement learning (RL) does not require any pre-collected data or knowledge, but only needs to learn through interaction with the environment to effectively deal with various dynamic events in actual production. However, the traditional RL-based scheduling method cannot deal with the huge state space effectively, and the strategies trained by RL method are often only applicable to the single scale DFJSP-ITR.

To solve the above problems, this paper proposes an end-to-end approach using GNN and DRL to learn a high-quality scheduling policy for DFJSP-ITR (Song et al., 2023), and represent the state of DFJSP-ITR using a heterogeneous graph including operation nodes, machine nodes, and AGV nodes. A three-stage node embedding method is used to extract the node features. The node embeddings processed by average pooling will be used as input to the scheduling policy for integrated decision making for operations, machines and AGVs. Finally, the policy network is trained and optimized using proximal policy optimization (PPO) (Schulman et al., 2017). Considering the impact of different AGV distribution methods on the scheduling performance of the same instance, two different AGV distribution methods are used in this paper. In order to cope with dynamic event (new jobs insertion) during scheduling, we propose a dynamic strategy. During scheduling, the next action decision cannot be made because all the current unfinished jobs' processable machines or transportable AGVs are occupied. To address this situation, this paper proposes machines and AGVs occupancy state update strategy. The main contributions of this paper can be summarized as follows:

- The Markov Decision Process (MDP) model of DFJSP-ITR is established. A new heterogeneous graph structure is adopted to represent the states of DFJSP-ITR, and the process of state updating and the three-stage node embedding method used to extract node features are described in detail;
- Different AGV distribution methods are introduced and the impact of different AGV distribution methods on the scheduling performance of DFJSP-ITR is compared;
- We present a dynamic strategy for coping with new jobs insertion and detail machines and AGVs occupancy state update strategy that ensures smooth scheduling;
- An end-to-end DRL method capable of solving DFJSP-ITR of different sizes is proposed.

The rest of this paper is organized as follows. Section 2 describes the relevant literature. Section 3 first formulates the DFJSP-ITR as an MDP and describes its heterogeneous graph representation, then the dynamic strategy and machines and AGVs occupancy state update strategy are described. Section 4 describes in detail the proposed end-to-end DRL method for solving DFJSP-ITR of different sizes. Section 5 analyses the experimental results. Section 6 presents the conclusions and future work.

2. Literature review

The agent based on RL makes extensive exploration and learning through the interaction with the environment, and finally realizes the adaptive decision at each decision point. Due to its strong self-learning ability, RL is widely used in dynamic job shop scheduling problem (DJSP). Zhang et al. (2012) solve an uncorrelated parallel machine dynamic scheduling problem with random job arrivals using an average-reward RL approach. Xanthopoulos et al. (2013) proposed a RL-based method to minimize the average early arrival time and average late arrival time for the DJSP on a single machine. Shahrabi et al. (2017) adopted RL algorithm with q factor to improve the performance of scheduling methods based on variable neighborhood search, so as to solve DJSP. Shiue et al. (2018) implemented the RL-based real-time scheduling by incorporating off-line learning module and Q-learning-based RL module. Wang (2020) used a weighted Q-learning algorithm based on clustering and dynamic search to solve DJSP.

DeepMind, a pioneering artificial intelligence company, developed the Go artificial intelligence program AlphaGo. AlphaGo's monumental success is rooted in deep learning. Its historic triumph over a world-renowned Go champion underscores the immense potential of Deep Reinforcement Learning (DRL) in addressing complex sequential decision-making challenges (Mnih et al., 2015; Silver et al., 2016).In recent years, more and more researchers have applied DRL to combinational optimization problems (Lei et al., 2022), such as traveling salesman problem and vehicle routing problem (Kool et al., 2018; Lei et al., 2021), graph optimization problem (Khalil et al., 2017; Li et al., 2018).Since DRL can not only effectively handle the huge state space of the FJSP, but also quickly respond to the dynamic events that occur, many scholars have adopted DRL to solve dynamic flexible job shop scheduling problem (DFJSP). Hu et al. (2020) proposed a real-time scheduling method for AGVs based on adaptive DRL with the optimization objectives of makespan and delay rate, and conducted simulation experiments in Plant Simulation (PS) software. Li et al. (2022) developed a multi-objective optimization model for DFJSP-ITR and designed a hybrid deep Q-network for policy training. Gui et al. (2023) used a deep deterministic policy gradient algorithm to train a policy network with state as inputs and weights as output to minimize the average tardiness of DFJSP. However, the trained strategies from these RL methods are often only applicable to a single scale scheduling problem. Therefore, many scholars have recently used DRL and GNN to extract scheduling information from graphs containing information about scheduling problem instances, so as to automatically generate scheduling policy that can be applied to scheduling problems of different sizes. After representing the state of JSP with a disjunctive graph, DRL and GNN are used to learn high-quality scheduling policy that can be used to solve JSP of different sizes (Zhang et al., 2020; Park et al., 2021). Ni et al. (2021) formulated the Gantt chart of the instance into the multi-graph-structured data, and proposed a novel Multi-Graph Attributed RL based Optimization algorithm to solve large-scale hybrid flow shop scheduling problem. Lei et al. (2022) formulated the FJSP as an MDP and proposed a multi-pointer graph network architecture and a multi-Proximal Policy Optimization algorithm to learn scheduling policy that can be used to solve FJSP of different sizes. Song et al. (2023) proposed a heterogeneous GNN architecture to capture complex relationships between operations and machines, and a new DRL approach to learn high-quality PDRs end-to-end.

In previous research, many researchers have utilized RL or DRL methods to solve DJSP or DFJSP well. However, these trained policies are often tailored to single-scale DFJSP scenarios, and the end-to-end DRL methods used primarily address static scheduling problems. The utilization of DRL method for DFJSP-ITR remains rare, with no existing end-to-end DRL solution designed for DFJSP-ITR.

3. Problem formulation

The DFJSP-ITR considered in this paper can be defined as follows. The notations used for problem formulation are shown in Table 1.

An DFJSP-ITR instance have n successively arriving jobs $J = \{J_1, J_2, \ldots, J_n\}$,which are delivered to m machines $M = \{M_1, M_2, \ldots, M_m\}$ for processing via l AGVs $A = \{A_1, A_2, \ldots, A_l\}$. Each job J_i consists of n_i

Table 1
The notations used for problem formulation.

Notations	Meaning
n	Total number of jobs.
m	Total number of machines.
l	Total number of AGVs.
J_i	The ith job.
M_i	The ith machine.
A_i	The ith AGV.
O_{ij}	The jth operation of job J_i.
M_{ij}	The compatible machine set of operation O_{ij}.
A_{ij}	The compatible AGV set of operation O_{ij}.
pt_{ijk}	The processing time of operation O_{ij} on machine M_k.
rt_{ijl}	The total delivery time of operation O_{ij} by AGV A_l.
bt_{ijl}	The back time of AGV A_l returns to carport after delivering operation O_{ij} to the target machine.
AR_i	The arrival time of job J_i.
C_{ij}	The completion time of O_{ij}

operations that must be processed in a specific order. Each operation O_{ij} can be processed on any machine M_k selected from a compatible machine set $M_{ij} \subseteq M$ for a processing time pt_{ijk}. And each operation O_{ij} must be delivered by an AGV A_l selected from a compatible AGV set $A_{ij} \subseteq A$ to the target machine for processing with a delivery time rt_{ijl}. The delivery time is actually composed of two parts: the time when the AGV A_l acquires the operation and the time when the AGV A_l delivers the operation to the target machine. When an AGV A_l completes the task, it will be travel back to the carport with a back time bt_{ijl}. The arrival time of job J_i is AR_i, and the completion time of O_{ij} is C_{ij}. The objective is to minimize the makespan $C_{\max} = max_{ij}\{C_{ij}\}$ of all jobs while determining the machine selection and AGV selection for all operations.

When the first AGV distribution strategy is used, any one AGV can provide distribution service for all operations. When the second AGV distribution strategy is used, one AGV provides distribution service for all operations of only some jobs. For example, when there are 10 jobs and 2 AGVs, all operations of the first 5 jobs are transported by the first AGV and all operations of the last 5 jobs are transported by another AGV. In particular, when the number of jobs is not a multiple of the number of AGVs, for example, when there are 20 jobs and 3 AGVs, all operations for the first 6 jobs are transported by the first AGV, all operations for the middle 7 jobs are transported by the second AGV, and all operations for the last 7 jobs are transported by the third AGV. And the required assumptions for the DFJSP-ITR are shown in Table 2.

3.1. Heterogeneous graph state representation

Graph node embedding is to obtain the appropriate vector representation of each node by cyclic feature aggregation, which can be used to efficiently perform various end-tasks. Since the disjunctive graph can visually represent information such as precedence constraints and compatible machine set for each operation, the disjunctive graph is widely used for the state representation of JSP or FJSP. However, the traditional disjunction graph is difficult to directly represent the state of DFJSP-ITR. Because DFJSP-ITR needs to select not only the machine to process the operation, but also the AGV to transport the operation. Therefore, in order to represent the state of DFJSP-ITR more visually, we defined a novel disjunctive graph $H = (O, M, A, C, D)$ based on (Song et al., 2023), as shown in Fig. 1. Where O is operation node set; M is machine node set; A is AGV node set; C is conjunctive arc set; D is a set including operation-machine (O-M) arc and operation-AGV (O-A) arc. The processing time of each operation on different machines can be represented by different operation-machine (O-M) arcs, while the operation-AGV (O-A) arc represent the average delivery time of the operation. By designing feature vectors for nodes, the heterogeneous graph H can contain dynamically changing state information to make better action decisions (Park et al., 2021). This paper adopts node and edge features similar to (Song et al., 2023), as shown below:

Operation nodes:

- Status: if O_{ij} has been scheduled, the status is 1, otherwise it is 0.
- Number of compatible machines of O_{ij}.
- Number of compatible AGVs of O_{ij}.
- Processing time and delivery time: the sum of the processing time and delivery time of O_{ij}.
- Start time: the estimated or actual start time of O_{ij}.
- Number of unscheduled operations in the job.
- Job completion time: the estimated or actual completion time of O_{ij}.

For calculations related to time values, such as the starting time of

Table 2
The required assumptions for the DFJSP-ITR.

Number	Assumption
1	Jobs are independent, all machines and AGVs are available at time zero.
2	Each machine or AGV can only process or transport one job at a time.
3	All AGVs have the same delivery time between any two task locations.
4	The loading and unloading time of each operation is included in the delivery time of the AGV.
5	All AGVs are waiting at the carport at time zero, and all AGVs travel at constant speed.
6	After each AGV delivers a job to the target machine, it returns to carport to wait for the next task.

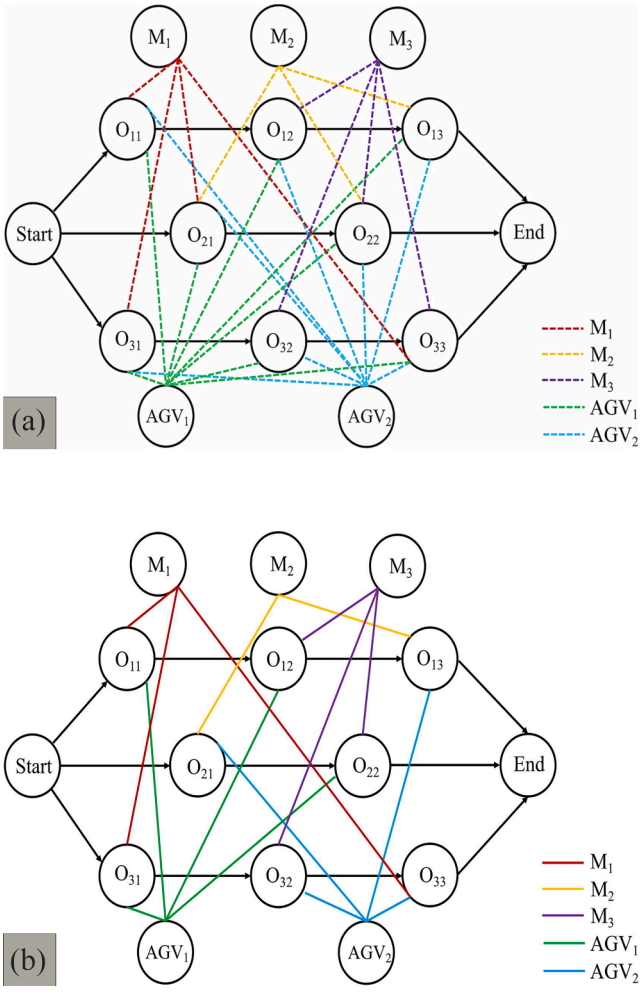

Fig. 1. Heterogeneous graph of DFJSP-ITR: (a) An instance, (b) A solution. The dotted line means processable or transportable, while the solid line means scheduled.

operation nodes, see the state section below.

Machine nodes:

- Available time: the time when machine M_k can process O_{ij}.
- Number of O_{ij} that M_k can process.
- Utilization: the ratio of cumulative processing time to total production time of M_k.

AGV nodes:

- Available time: the time when AGV A_l can delivery O_{ij}.
- Number of O_{ij} that A_l can delivery.
- Utilization: the ratio of cumulative delivery time to total routing time of A_l.

O-M arcs: the processing time of O_{ij} on the M_k.

O-A arcs: the delivery time of O_{ij} by the A_l. Note that when O_{ij} is not scheduled, O-A arc represents an average calculated based on the delivery time of all possible delivery methods.

3.2. MDP formulation of DFJSP-ITR

RL formulates sequential decision-making problems as an MDP, represented by a 6-tuple (S, A, P, γ, R, π). At decision step t, the agent selects an action a_t from the action space A based on the current state s_t and the policy π, and receives an immediate reward r_t. Then enter the new state s_{t+1} according to the transition probability function P. After completing the entire iterative training process, the agent learns a policy that maximizes the expected cumulative sum of future returns. Traditional RL approaches, which rely on policy tables to store all possible states and actions, face challenges when dealing with problems featuring large state spaces. In response, DRL marries deep learning with RL, utilizing neural networks (NN) to establish a functional mapping between state-action pairs and value functions. This not only mitigates the loss of state information but also efficiently addresses large-scale problems in dynamic environments (Li et al., 2022).

In this paper, the problem of DFJSP-ITR is formulated as an MDP, which is a six-element tuple (S, A, P, γ, R, π).

(1) State: In order to represent the machine and AGV information of all operations more intuitively and efficiently, we use the heterogeneous graph H to represent the state of DFJSP-ITR. S is the state set, the state $s_t \in S$ contains all static features and dynamic features of DFJSP-ITR at decision step t (Park et al., 2021). The initial state s_0 is an DFJSP-ITR instance drawn from a distribution (Song et al., 2023). If O_{ij} is the first operation of the job J_i, suppose operation O_{ij} is delivered to machine M_k by AGV A_l for processing and its start time $S_{ij}(t) = max(AR_i, rt_{ijkl})$; otherwise $S_{ij}(t) = max(AR_i, \overline{rt}_{ij})$.Where AR_i is the arrival time of job J_i, $\overline{rt}_{ij} = \sum_{M_k \in M_{ij}} rt_{ijkl} / |M_{ij}|$ is an estimated delivery time of O_{ij}. When the job J_i is not newly inserted, its arrival time is equal to 0. So the task start time $ST_{ij}(t)$ of AGV A_l is equal to $(S_{ij}(t) - rt_{ijkl})$, and the task end time $AT_{ij}(t)$ of AGV A_l is equal to $(S_{ij}(t) + bt_{ijkl})$. If O_{ij} is not the first operation of the job J_i and its immediate predecessor O_{ij-1} is delivered to machine M_k by AGV A_l, then $S_{ij}(t) = S_{ij-1}(t) + pt_{ij-1kl} + rt_{ijkl}$; otherwise $S_{ij}(t) = S_{ij-1}(t) + \overline{pt}_{ij-1} + \overline{rt}_{ij}$, where $\overline{pt}_{ij} = \sum_{M_k \in M_{ij}} pt_{ijkl} / |M_{ij}|$ is an estimated processing time of O_{ij}. Especially, if O_{ij} and its immediate predecessor O_{ij-1} are not scheduled, to better illustrate the estimated delivery time $\overline{rt}_{ij}$ in this case, an example will be used. Assume that the compatible machine set for O_{ij-1} and O_{ij} are $M_{ij-1} = \{M_1, M_2, M_4\}$ and $M_{ij} = \{M_1, M_3\}$, and T_{ij} is the AGV delivery time of O_{ij} when the processing machine of O_{ij-1} is M_i and the processing machine of O_{ij} is M_j. As a result, $\overline{rt}_{ij} = [(T_{11} + T_{21} + T_{41}) + (T_{13} + T_{23} + T_{43})]/3$. When O_{ij-1} and O_{ij} are scheduled on machine M_k, the delivery time rt_{ijkl} of O_{ij} and back time bt_{ijkl} of AGV A_l are equal to 0.That is, $S_{ij}(t)$ equals $ST_{ij}(t)$.

(2) Action: A is the action set. Since DFJSP-ITR needs to select not only the machine to process the operation, but also the AGV to transport the operation. Action $a_t \in A$ as a result of the composite decision at step t is defined as a feasible operation-machine-AGV pair (O_{ij}, M_k, A_l). Specifically, when the delivery time rt_{ijkl} of O_{ij} is equal to 0, O_{ij} actually does not need an AGV for delivery, but an AGV is still selected for O_{ij} in order to keep the consistency of the action a_t. This means that this AGV node does not need to perform node feature updates.

(3) Transition: Based on the current state s_t and the sampled action a_t, the agent interacts with the environment to transit to a new state s_{t+1}. And the transition probability function P represents the probability of going from state s_t to next state s_{t+1} due to taking action a_t.

(4) Reward: R is the reward function that gives an immediate reward r_t after a transition. Since the objective of this paper is to minimize makespan, the immediate reward is defined as: $r_t = C_{\max}(s_t) - C_{\max}(s_{t+1})$. And the discount factor $\gamma = 1$. Since $C_{\max}(s_0)$ for a particular problem instance is a constant, minimizing $C_{\max}$ and maximizing R are equivalent (Song et al., 2023).

(5) Policy: π is the policy that represents the mapping from the state set S to the action set A (Hu et al., 2020). That is, the policy function determines the choice of action and directly affects the final scheduling scheme.

3.3. Dynamic strategy

When one or more new jobs are inserted at some point, in order to generate a new and complete scheduling plan in real time, the operation nodes contained in the newly inserted jobs are added to the original heterogeneous graph to generate a new heterogeneous graph, and then node embedding calculation and action decision are made based on the new heterogeneous graph until the arrival of the next new job or the end of scheduling. The newly inserted operation node contains the same node feature type as in the original heterogeneous graph. The heterogeneous graph before and after the new job is inserted is shown in Fig. 2.

3.4. Machines and AGVs occupancy state update strategy

During scheduling, the next action decision cannot be made because all the current unfinished jobs' processable machines or transportable AGVs are occupied. To address this situation, this paper adopts an update strategy to ensure the smooth progress of scheduling. Assume that the state of the job being processed or transported, the machine in process, and the AGV in transport are all 'True', otherwise 'False'. The latest processing task completion time for all machines is ET={ET_1, ET_2, …,ET_m}, and the time for all AGVs to return to carport is AT={AT_1, AT_2, …, AT_l}.

In the process of training or testing, before each environment state updates, it is pre-judged whether there is a feasible solution (a feasible operation-machine-AGV pair (O_{ij}, M_k, A_l)) in the action space. The update strategy is shown in Strategy 1, if there is no feasible solution in the action space, first determine whether the cause is machine occupancy, AGV occupancy, or both machine and AGV occupancy, and then perform the corresponding update operation. After one or more update operations, if there is at least one feasible solution in the action space at this point, it is executed directly. Otherwise, determine again which occupancy situation it is and perform the corresponding update operation until the next action decision can be made. The update operation when machine occupancy or AGV occupancy is similar to (Song et al., 2023), prioritizing the update of the occupied machine or AGV with the smallest completion time for the latest task.

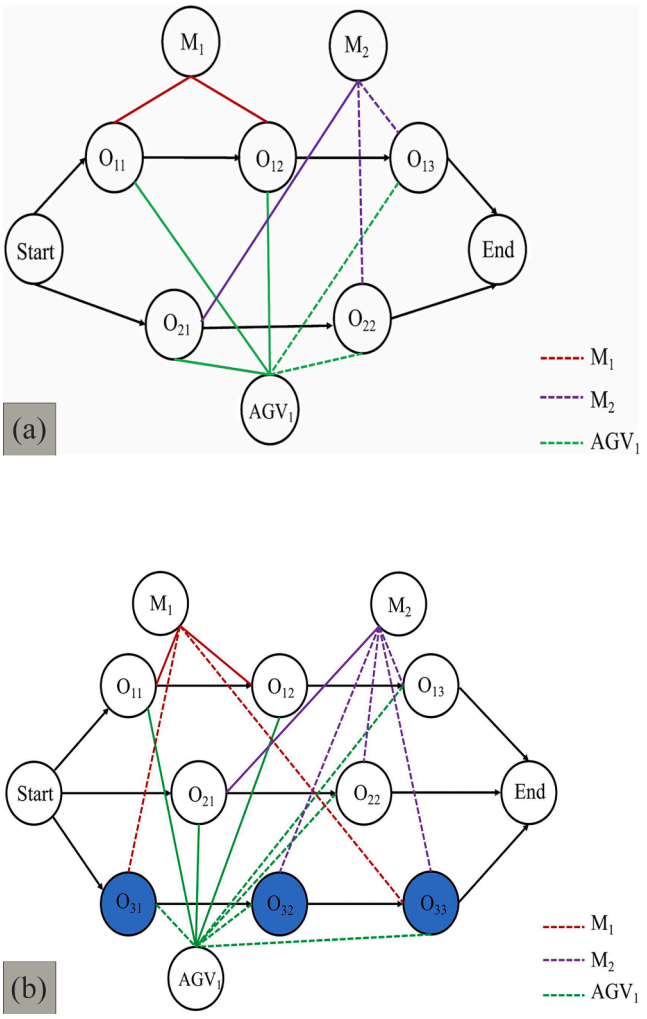

Fig. 2. Heterogeneous graph before and after inserting the new job: (a) before the new job is inserted, (b) after the new job is inserted. The dark blue is the newly inserted operation node.

4. Method

4.1. Node embedding procedure

Since each action is derived using node embedding computed by graph neural networks, the extraction and embedding of all state information in the disjunctive graph directly affects the final scheduling results. Therefore, this paper proposes a three-stage node embedding method and policy network structure similar to that of the literature (Song et al., 2023). It is worth noting that the AGVs distribution strategy can be changed by simply changing the connection relationship between the operation nodes and AGV nodes in the heterogeneous graph before training or testing. The notations used for node embedding procedure is shown in appendix.

4.1.1. Machine node embedding

Since the heterogeneous graph proposed in this paper contains the operation node O_{ij}, machine node M_k, AGV node A_l, O-M arc connecting the operation node and machine node and O-A arc connecting the operation node and AGV node, in order to fully incorporate the structure and feature information of the heterogeneous graph into the final machine node feature vector, machine nodes need to efficiently integrate the feature information of their first-order neighborhood (operation nodes) and second-order neighborhood (AGV nodes). First, the original feature vectors of three types of nodes and two types of arcs are mapped into 8-dimensional feature vectors by linear transformation, where x_{ij} is the feature vector of operation node O_{ij}, y_k is the feature vector of machine node M_k, z_l is the feature vector of AGV node A_l, u_{ijk} is the feature vector of O-M arc. v_{ijl} is the feature vector of O-A arc. Then calculate the attention coefficients between the three types of nodes. The formula for calculating the attention coefficient of AGV node A_l to operation node O_{ij} is as follows:

$$e_{ijl} = \text{LeakyReLU}\left(a[x_{ij}||z_l]\right) \tag{1}$$

where $\bullet \,||\, \bullet$ is the concatenation operation. The formula for calculating the attention coefficient of operation node O_{mn} to operation node O_{ij} is as follows:

$$e_{ij} = \text{LeakyReLU}\left(a[x_{ij}||x_{mn}]\right) \tag{2}$$

The formula for calculating the attention coefficient of operation node O_{ij} to machine node M_k is as follows:

$$e_{ijk} = \text{LeakyReLU}\left(a[y_k||x_{ij}]\right) \tag{3}$$

The formula for calculating the attention coefficient of machine node M_l to machine node M_k is as follows:

$$e_k = \text{LeakyReLU}\left(a[y_k||y_l]\right) \tag{4}$$

Where a is a 16-dimensional vector. By using the softmax function to normalize the attention coefficients of the two groups, the corresponding attention coefficients $(\alpha_{ijl}, \alpha_{ij})$ and (α_{ijk}, α_k) can be obtained. Finally, the machine node embedding can be obtained by integrating the operation node features that have integrated the AGV node features with the machine node features. The operation node embedding calculation integrating AGV nodes features is as follows:

$$x_{ij1} = \sigma\left(\alpha_{ij}x_{ij} + \sum_{A_l \in N_t(A_l)} \alpha_{ijl}\left(z_l + v_{ijl}\right)\right) \tag{5}$$

Where $N_t(A_l)$ is the number of compatible AGVs of O_{ij}, and σ is sigmoid function. The computation of embedding for machine node is as follows:

$$y_k^{'} = \sigma\left(\alpha_k y_k + \sum_{O_{ij} \in N_t\left(O_{ijk}\right)} \alpha_{ijk}\left(x_{ij1} + u_{ijk}\right)\right) \tag{6}$$

Where $N_t\left(O_{ijk}\right)$ is the number of O_{ij} that M_k can process. The node embedding process for a particular machine node is shown in Fig. 3.

4.1.2. AGV node embedding

The calculated f_{ijk}, f_{ij}, f_{ijl} and f_l are normalized in two groups using the softmax function to obtain two sets of attention coefficients $(\beta_{ijk}, \beta_{ij})$ and (β_{ijl}, β_l). Where f_{ijk} is the attention coefficients of the machine nodes to the operation node, f_{ij} is the attention coefficients between operation nodes, f_{ijl} is the attention coefficients of the operation nodes to the AGV node and f_l is the attention coefficients between AGV nodes. Therefore, the operation node embedding calculation integrating machine nodes features is as follows:

$$x_{ij2} = \sigma\left(\beta_{ij}x_{ij} + \sum_{M_k \in N_t(M_k)} \beta_{ijk}\left(y_k^{'} + u_{ijl}\right)\right) \tag{7}$$

Where $N_t(M_k)$ is the number of compatible machines of O_{ij}. The computation of embedding for AGV node is as follows:

$$z_l^{'} = \sigma\left(\beta_l z_l + \sum_{O_{ij} \in N_t\left(O_{ijl}\right)} \beta_{ijl}\left(x_{ij2} + v_{ijl}\right)\right) \tag{8}$$

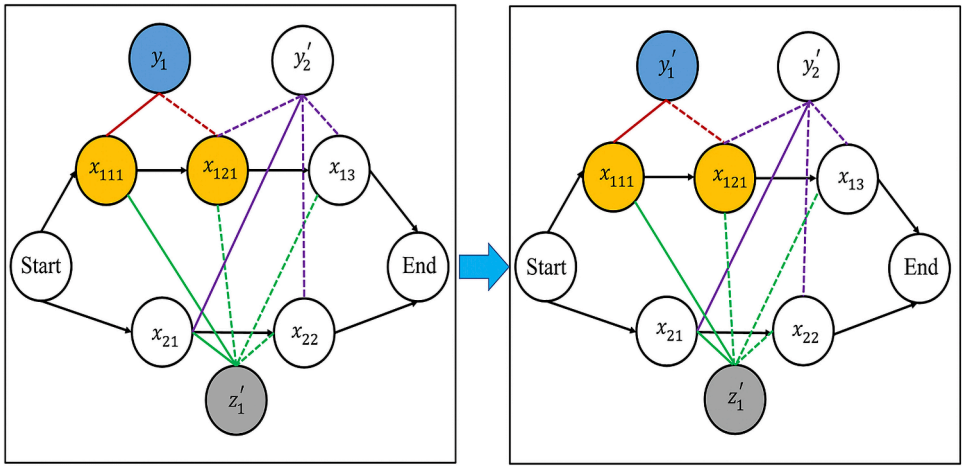

Fig. 3. Update of machine embedding y_1. The node embedding of the first-order neighborhood (operation nodes) of this machine node is updated first, and then the node embedding of this machine node is updated.

Where $N_t(O_{ijl})$ is the number of O_{ij} that A_l can transport.

4.1.3. Operation node embedding

Due to the heterogeneity of the arcs and nodes connected to the operation nodes, the attention-based mechanism for operation node embedding is not very useful (Song et al., 2023). So multilayer perceptron (MLP) will be used for operation node embedding, similar to (Song et al., 2023). Each MLP consists of two 128-dimensional hidden layers and exponential linear unit (ELU) activation. After MLP processing of $x_{i,j-1}$, $x_{i,j}$, $x_{i,j+1}$, $\overline{y}_k'$ and $\overline{z}_l'$ respectively, they are concatenated together to obtain a 40-dimensional vector F_t. The concatenation function for calculating F_t is as follows:

$$F_t = \mathrm{MLP}_2(x_{i,j-1}) \| \mathrm{MLP}_3(x_{i,j+1}) \| \mathrm{MLP}_4(\overline{y}_k') \| \mathrm{MLP}_5(\overline{z}_l') \| \mathrm{MLP}_6(x_{ij}) \tag{9}$$

Where $x_{i,j-1}$, x_{ij} and $x_{i,j+1}$ are the node features of operation nodes $O_{i,j-1}$, $O_{i,j}$ and $O_{i,j+1}$ after linear transformation mapping, respectively. Then the ELU activation and MLP processing of vector F_t are carried out successively, and finally an 8-dimensional embedding vector of operation node $O_{i,j}$ is obtained. The computation of embedding for operation node is as follows:

$$x_{ij}' = \mathrm{MLP}_1(\mathrm{ELU}(F_t)) \tag{10}$$

4.2. Decision making with node embedding

A GNN is constructed by stacking two embedding layers to obtain the final node embedding x_{ij}'',y_k'' and z_l''. The obtained three types of node embeddings are pooled on average, and then three types of node embeddings are stitched with the pooled node embeddings $(\overline{x}_{ij}'', \overline{y}_k'', \overline{z}_l'')$ into a 48-dimensional vector Q_t as the input of the policy network $\pi(a_t|s_t)$. Q_t is shown below:

$$Q_t = (x_{ij}'' \| \overline{x}_{ij}'' \| y_k'' \| \overline{y}_k'' \| z_l'' \| \overline{z}_l'') \tag{11}$$

The probability of choosing each action a_t can be calculated as:

$$\pi(a_t|s_t) = \frac{\exp(P(a_t, s_t))}{\sum_{a_i' \in A_t} \exp(P(a_i', s_t))} \tag{12}$$

where $P(a_t, s_t) = \mathrm{MLP}_7(Q_t)$, similar to (Song et al., 2023). MLP_7 has two 96-dimensional hidden layers and hyperbolic tangent activation. In order to get a better scheduling scheme, the action a_t is obtained by random sampling of the probability distribution.

In the training process, random sampling can get the best solution among N copies of a given instance, i.e., random sampling helps to explore the environment to get a better training model (Kool et al., 2018; Lei et al., 2022). Therefore, in this paper, the actions are sampled according to the policy π during training, while the greedy strategy is used in testing: the action with the maximum probability is selected according to the policy π in each state.

4.3. Training procedure

In this paper, we use PPO to train the parameters $\theta = \{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5\}$ of the proposed scheduling model, where θ_1, θ_2 and θ_3 are the machine node embedding layer, AGV node embedding layer and operation node embedding layer parameters, respectively, θ_4 and θ_5 are the actor network and critic network parameters. Both the actor network and the critic network adopt MLP with two hidden layer dimensions of 128 and 96. Algorithm 1 gives the pseudo-code of the whole training process. Training is performed for a total of 500 iterations, and the policy is validated on a validation set containing 100 instances every 5 iterations. And every 5 iterations, the 20 instances used for training were replaced. The overall framework of each iteration in the training process is shown in Fig. 4. In this paper, fixed hyperparameters are used to train GNNs of different sizes, as shown in Table 3. In order for the model to learn the assignment strategy of AGVs better, three AGVs are used uniformly in the training. And the model was created by Pytorch. The hardware used is a 13th Gen Intel Core i9-13900HX CPU and an Nvidia GeForce RTX 4060 Laptop GPU.

5. Experimental results

The instances used for training and testing are generated in a similar way to (Brandimarte, 1993; Song et al., 2023), but with additional AGV delivery time, AGV back time, and job arrival time. Nine instances of different sizes as shown in Table 4 were generated and trained on six smaller sizes. In light of practical scenarios where job shop operation processing time often exceed its delivery time. Without considering the job shop layout, it is assumed that the delivery time of the operation and the return time of the AGV obey the uniform distributions U (3,7) and U (2,4), respectively. And the trained policy obtained by using Strategy 1 is named Greedy. We then conduct experiments on a new public benchmarks proposed by (Ham, 2020), and compare the proposed method with the most advanced constraint programming (CP) (Ham, 2020) and tabu search (TS) (Berterottière et al., 2023). Finally, in order to verify the usability of the proposed method in actual production, we

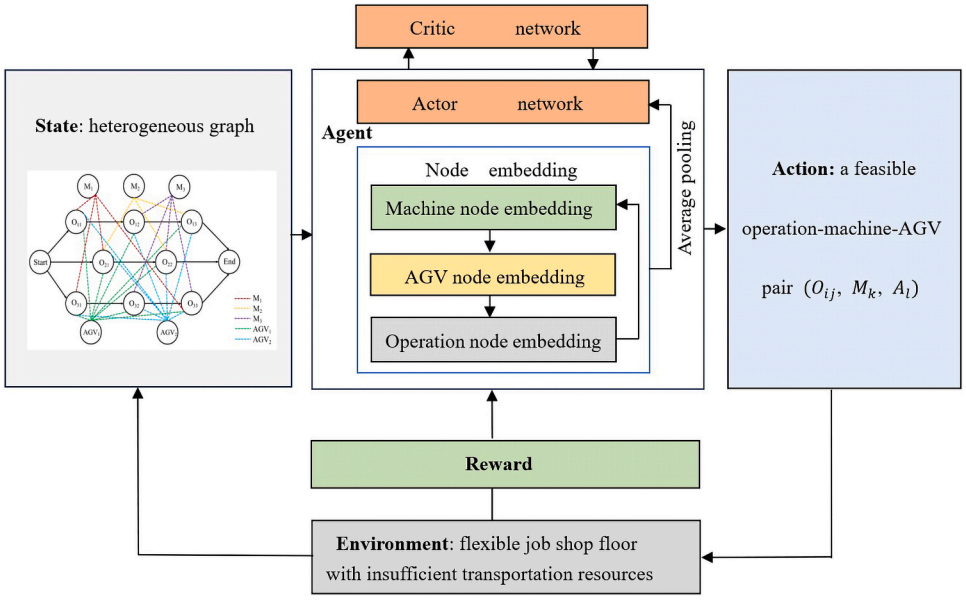

Fig. 4. The overall framework of each iteration in the training process.

Table 3
Training hyperparameters.

Hyperparameters	Value
Optimizer	Adam
Learning rate	2×10^{-4}
Discount factor	1
Clip ratio	0.2
Policy loss coefficient	1
Value loss coefficient	0.55
Entropy bonus coefficient	0.001
Number of episodes per update	3

carry out experimental analysis on 3 order cases based on a real radar microwave module job shop.

5.1. Baseline methods

As of now, there are many PDRs for solving JSP and FJSP with varying performance (Zhang et al., 2020; Lei et al.,2022). Doh et al. (2013) tested 36 combinations of PDRs in solving FJSP. In order to verify the performance of the proposed method on randomly generated instances of different sizes, we selected four PDRs with better performance from the 36 PDRs, similar to (Lei et al., 2022). Below, you'll find the four PDR elements:

- Most Operations Remaining (MOR): select the job with the most remaining operations.
- First In First Out (FIFO): select the first arrived job.
- Most Work Remaining (MWKR): select the job with the most total processing time remaining.
- Shortest Processing Time (SPT): select the job with the shortest processing time.

Four PDR are combined with two AGV distribution strategies proposed in this paper to generate eight baseline PDRs. When multiple machines are available for an operation, select the machine with the least processing time. When the first AGV distribution strategy is used, if multiple AGVs are available at the same time, select the one with the minimum available time.

Table 4
Randomly generated instances.

Size	*n*	*ope*	*mac*	*pt*	*rt*	*bt*	*at*
10 × 5 × 2	1	U(4,6)	U(1,5)	U(8,40)	U(3,7)	U(2,4)	U(30,120)
15 × 5 × 2	2	U(4,6)	U(1,5)	U(8,40)	U(3,7)	U(2,4)	U(30,120)
20 × 5 × 2	2	U(4,6)	U(1,5)	U(8,40)	U(3,7)	U(2,4)	U(30,120)
10 × 10 × 2	1	U(8,12)	U(1,10)	U(8,40)	U(3,7)	U(2,4)	U(30,120)
15 × 10 × 3	2	U(8,12)	U(1,10)	U(8,40)	U(3,7)	U(2,4)	U(30,120)
15 × 15 × 3	3	U(12,18)	U(1,15)	U(8,40)	U(3,7)	U(2,4)	U(30,300)
20 × 10 × 3	3	U(8,12)	U(1,10)	U(8,40)	U(3,7)	U(2,4)	U(30,300)
30 × 10 × 3	4	U(8,12)	U(1,10)	U(8,40)	U(3,7)	U(2,4)	U(30,300)
40 × 10 × 3	5	U(8,12)	U(1,10)	U(8,40)	U(3,7)	U(2,4)	U(30,300)

Size: number of jobs × number of machines × number of AGVs;
n: number of newly arrived jobs;
ope: number of operations in a job;
mac: number of compatible machines for operation ;
pt: processing time of operation;
rt: delivery time of operation by AGV;
bt: back time of AGV;
at: arrival time of newly arrived jobs.

Table 5
Results on randomly generated instances when the first AGV distribution strategy is used.

Size		Greedy	MOR	FIFO	MWKR	SPT
10 × 5 × 2	C_{max}	286.32	396.10	391.56	396.73	718.75
	Gap	0.00 %	38.34 %	36.76 %	38.56 %	151.03 %
	Time (s)	0.44	0.06	0.05	0.05	0.05
15 × 5 × 2	C_{max}	378.20	589.27	608.37	581.24	973.49
	Gap	0.00 %	55.81 %	60.86 %	53.69 %	157.40 %
	Time (s)	0.71	0.11	0.11	0.09	0.07
20 × 5 × 2	C_{max}	553.76	649.57	653.97	653.73	1397.24
	Gap	0.00 %	17.30 %	18.10 %	18.05 %	152.32 %
	Time (s)	1.12	0.16	0.16	0.11	0.10
10 × 10 × 2	C_{max}	406.69	668.22	680.41	634.91	1528.4
	Gap	0.00 %	64.31 %	67.30 %	56.12 %	275.81 %
	Time (s)	1.17	0.14	0.16	0.14	0.11
15 × 10 × 3	C_{max}	497.52	752.17	729.00	736.21	2244.84
	Gap	0.00 %	51.18 %	46.53 %	47.98 %	351.21 %
	Time (s)	2.24	0.26	0.22	0.20	0.15
15 ×15 ×3	C_{max}	607.35	1078.06	1036.79	1039.48	3266.89
	Gap	0.00 %	77.50 %	70.71 %	71.15 %	437.89 %
	Time (s)	5.12	0.45	0.49	0.48	0.38
20 × 10 × 3	C_{max}	624.89	965.59	994.19	957.29	3035.57
	Gap	0.00 %	54.52 %	59.10 %	53.19 %	385.78 %
	Time (s)	4.20	0.36	0.37	0.29	0.22
30 × 10 × 3	C_{max}	842.38	1228.42	1215.17	1196.90	4325.88
	Gap	0.00 %	45.83 %	44.25 %	42.09 %	413.53 %
	Time (s)	8.80	0.83	0.82	0.48	0.35
40 × 10 × 3	C_{max}	1081.03	1473.10	1437.96	1437.84	5708.46
	Gap	0.00 %	36.26 %	33.02 %	33.01 %	428.06 %
	Time (s)	16.56	1.52	1.56	0.69	0.51

Table 6
Results on randomly generated instances when the second AGV distribution strategy is used.

Size		Greedy	MOR	FIFO	MWKR	SPT
10 × 5 × 2	C_{max}	310.10	417.07	400.43	411.84	763.07
	Gap	0.00 %	34.50 %	29.13 %	32.81 %	146.07 %
	Time (s)	0.50	0.06	0.05	0.05	0.05
15 × 5 × 2	C_{max}	385.37	663.83	660.88	643.01	1012.18
	Gap	0.00%	72.26%	71.49%	66.86%	162.65%
	Time (s)	0.73	0.09	0.13	0.08	0.08
20 × 5 × 2	C_{max}	570.47	690.38	667.93	666.56	1462.11
	Gap	0.00 %	21.02 %	17.08 %	16.84 %	156.30 %
	Time (s)	1.07	0.18	0.16	0.10	0.10
10 × 10 × 2	C_{max}	450.13	738.39	742.46	705.12	1473.88
	Gap	0.00%	64.04%	64.94%	56.65%	227.43%
	Time (s)	1.11	0.13	0.15	0.13	0.10
15 × 10 × 3	C_{max}	555.56	823.08	774.93	790.09	2243.53
	Gap	0.00 %	48.15 %	39.49 %	42.22 %	303.83 %
	Time (s)	2.48	0.25	0.22	0.20	0.18
15 × 15 × 3	C_{max}	783.00	1253.53	1217.19	1218.42	3162.50
	Gap	0.00%	60.09%	55.45%	55.61%	303.90%
	Time (s)	5.15	0.54	0.50	0.46	0.30
20 × 10 × 3	C_{max}	684.40	1062.42	1078.40	1015.14	3050.74
	Gap	0.00 %	55.23 %	57.57 %	48.33 %	345.75 %
	Time (s)	4.35	0.37	0.41	0.29	0.27
30 × 10 × 3	C_{max}	912.05	1347.51	1308.98	1293.75	4411.06
	Gap	0.00 %	47.75 %	43.52 %	41.85 %	383.64 %
	Time (s)	9.11	0.78	0.80	0.47	0.36
40 × 10 × 3	C_{max}	1150.46	1622.93	1575.20	1566.32	5854.41
	Gap	0.00 %	41.07 %	36.92 %	36.15 %	408.88 %
	Time (s)	16.60	1.57	1.55	0.69	0.52

5.2. Performance on randomly generated instances

Three indicators (average makespan, relative gap of average makespan, running time) were used to evaluate nine instances of different sizes, and each size includes 100 test instances. The experimental results are shown in Tables 5 and 6. The results on the six different sizes (10 × 5 × 2, 15 × 5 × 2, 20 × 5 × 2, 10 × 10 × 2, 15 × 10 × 3 and 15 × 15 × 3) test instances show that the proposed approach significantly outperforms all baseline PDRs regardless of which AGV distribution strategy is used. Moreover, for the six different sizes test instances, the first AGV distribution strategy performed better. This may be because the second AGV distribution strategy has more constraints on DFJSP-ITR, resulting in poor scheduling performance. It can be seen that on the 20 × 5 × 2 test instances, the proposed method shows relatively little improvement over the baseline PDRs, which is similar to the results of (Song et al., 2023). This may be due to the fact that there are fewer machine nodes and AGV nodes connected to each operation node in 20 × 5 × 2 instances, making it difficult to effectively capture the relationship between these two types of nodes and operation nodes during training. For a more intuitive comparison, we give the Gantt charts obtained by solving the 15 × 10 × 3 instances with the proposed method and the best baseline PDR, both of which use the first AGV distribution strategy. It is obvious from Fig. 5 that the proposed method outperforms all the baseline PDRs.

For further analysis, based on makespan of 100 test instances, we calculated the relative gap of each baseline PDRs to the proposed method. Due to the poor performance of SPT, this paper omits them from the baseline method. As shown in Figs. 6 and 7, the proposed method outperforms all the baseline PDRs in more than 75 % of the instances in six different sizes, regardless of which AGV distribution strategy is used. To verify the generalization ability of the trained policy, the policy trained at 15 × 15 × 3 scale is run directly on three medium-scale (20 × 10 × 3, 30 × 10 × 3 and 40 × 10 × 3) instances, and the results show that the proposed method still outperforms all baseline PDRs, so the trained scheduling policy based on the proposed method not only have high quality, but also have good generalization performance. The running time of the proposed method is greater than all baseline PDRs on instances of different sizes, but they are all acceptable. The training time of the proposed method is also acceptable at all six scales of 10 × 5 × 2, 15 × 5 × 2, 20 × 5 × 2, 10 × 10 × 2, 15 × 10 × 3 and 15 × 15 × 3, which require 0.40 h, 1.41 h, 2.08 h, 2.16 h, 4.76 h and 9.84 h, respectively.

5.3. Performance on public benchmarks

Ham (2020) combined the benchmark instances proposed by Hurink et al. (1994) with three machine layouts to propose new benchmark instances. Considering that the operation processing time of the actual job shop is often greater than its delivery time, the first machine layout (the operation delivery time is 0.6 times its processing time) proposed by Ham (2020) is adopted. Since these instances require that even if two consecutive operations of the same job are processed on the same machine, an AGV is required for transportation, and considering that our

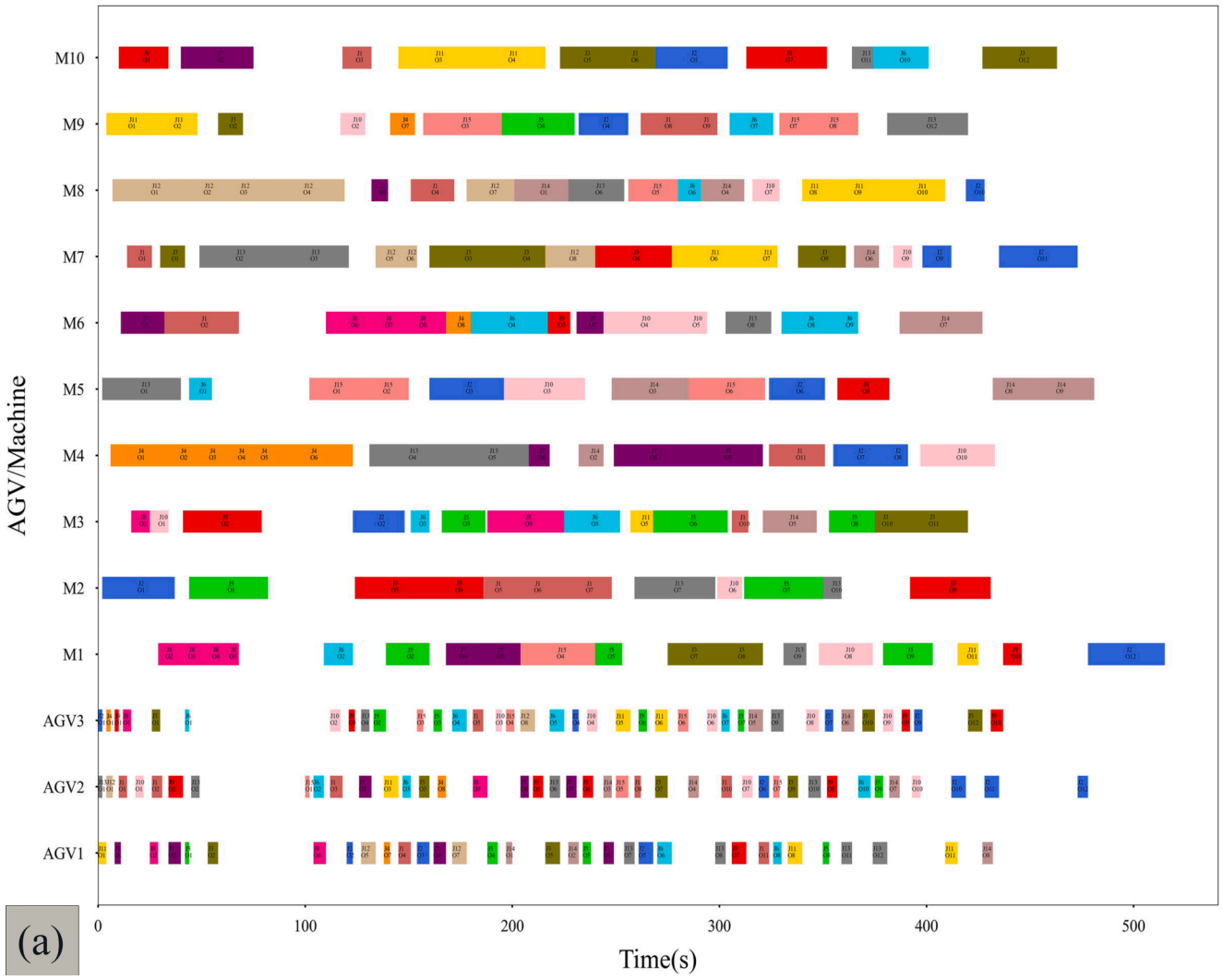

Fig. 5. The Gantt charts of our method and the best PDR for 15 × 10 × 3 DFJSP-ITR instance: (a) The Gantt chart of our method, (b) The Gantt chart of the best PDR.

original operation delivery time is given by uniform distribution, the delivery time of the operation is directly set to 0.6 times of the operation processing time during training and testing. AGV back time and job arrival time are set to 0.

C_{max} and Time in the Table 7 denote the average makespan and average CPU run time of the 40 instances. Since we only trained on six scale instances, the policy trained at 15 × 15 × 3 size is run directly on two medium-scale (20 × 10 × 2 and 30 × 10 × 2) instances. We name the method of testing the corresponding six sizes benchmark instances (10 × 5 × 2, 15 × 5 × 2, 20 × 5 × 2, 10 × 10 × 2, 15 × 10 × 2 and 15 × 15 × 2) with six different training policies respectively Greedy1. Greedy2 means that all benchmark instances are tested with policy trained on the 15 × 15 × 3 scale. It can be seen that the performance of our proposed method on sdata and edata is significantly better than CP and TS, while the performance on rdata is not much different from CP and TS. In relative gap of average makespan, the results of CP and TS only surpass Greedy1 4.79 % and 4.07 % on rdata. The poor performance of the proposed method on vdata may be due to the small number of AGVs, so it is difficult to effectively capture the complex relationship between AGV nodes and operation nodes during training. Although the results of TS surpass ours in average makespan on rdata and vdata, the average running time of the proposed method is much less than TS.

5.4. Case study

The radar microwave module job shop belongs to batch mixed line production and has a wide variety of products. In this paper, only three products (products 1, 2 and 3) produced in one of the production lines are selected for experimental analysis. Each product can be processed in batches, with batches of 20,5 and 5 for products 1, 2 and 3, respectively. The batch size of product 1 is 20 indicating that each processing unit that processes product 1 can process 20 products 1 at the same time, and the specific processing information of the three products is shown in Table 8. And the transportation time of AGV between carport, buffer (BF) and processing unit is shown in appendix. In Table 11, we use P and M_i to represent the carport and the ith processing unit, respectively. BF is the storage location of raw materials of the three products, and carport is the AGV's initial departure location and waiting location after completing the transportation task. The transportation time after all operations have been completed for each product is not considered. Note that the unit of machine processing time and AGV transport time are minutes.

Suppose that an order needs to process 20 pieces of each of the three products, then the batches of the three products are 1, 4 and 4, respectively, consisting of 9 workpieces (1 job1, 4 job2 and 4 job3) and 229 operations. In order to balance the production time of the order and the efficiency of the AGVs, we conducted an experimental analysis of

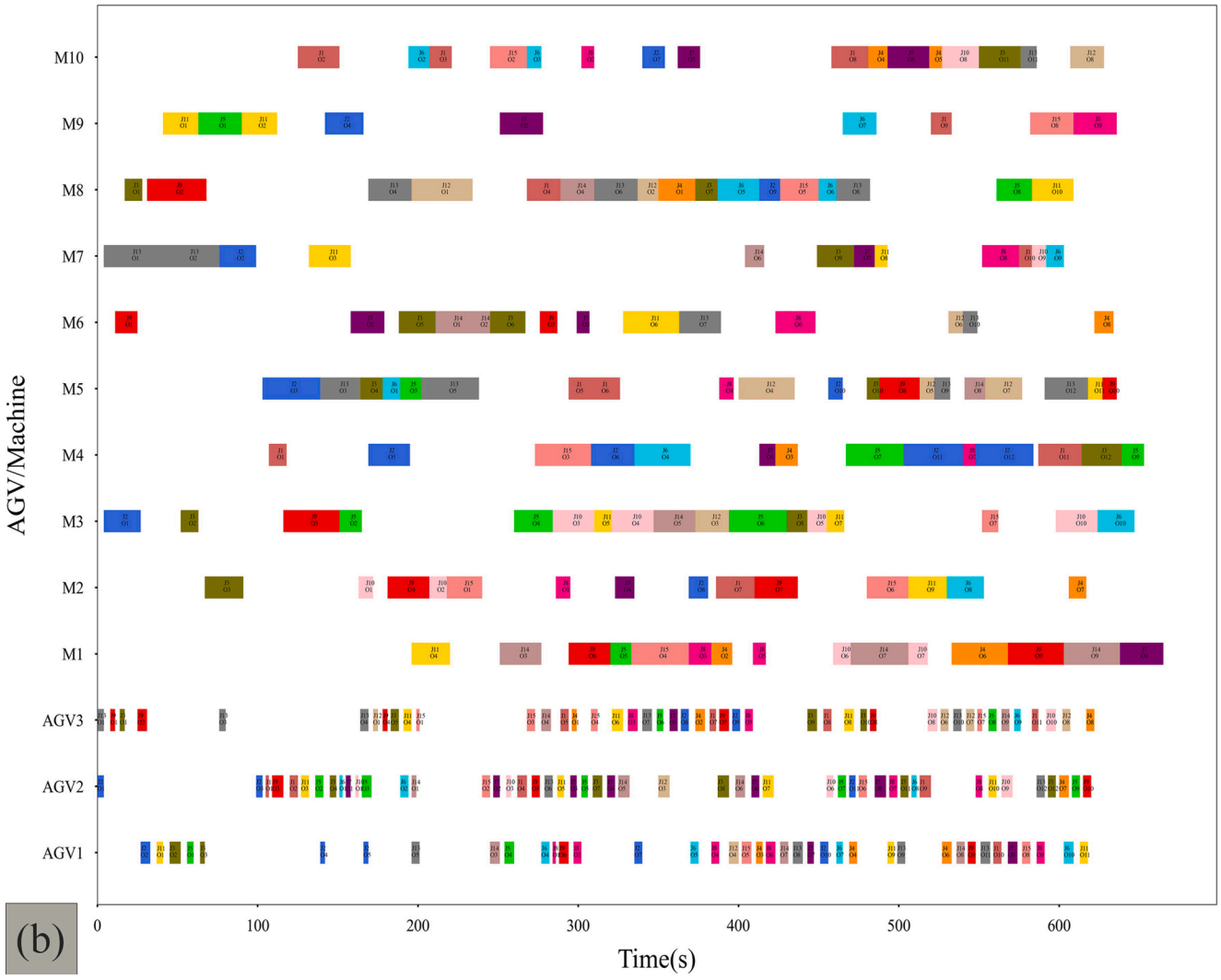

Fig. 5. (*continued*).

orders of different sizes by varying the number of AGVs. As can be seen from Table 9, with the increase of the number of AGVs, the makespan becomes larger instead after getting a small amount of improvement. This is due to the fact that the processing time of each operation is much greater than its transport time, resulting in the AGV being in a waiting state for a long time. Therefore, when the order quantity is large, two AGVs can be considered, otherwise one can meet the production.

6. Conclusions and future work

In this paper, we use an innovative heterogeneous graph to represent the state of DFJSP-ITR. We present an end-to-end approach for acquiring a high-quality scheduling policy tailored specifically for DFJSP-ITR. After constructing the MDP model of DFJSP-ITR, a three-stage node embedding method is used to extract the node features essential for the integrated decision-making processes. Ultimately, we employ PPO algorithm in training the policy network. The results demonstrate that when two different AGV distribution strategies are used separately, our proposed method yields scheduling schemes superior to the baseline PDRs. Notably, when the first AGV distribution strategy is used, the results are better. A significant finding is that the policies trained on randomly generated instances can be directly used for testing on public benchmarks with good results. However, as the scale of DFJSP-ITR increases, the required computing time also escalate rapidly. This escalation primarily arises from the growth in the number of nodes, which consequently prolongs the computation time required for node embedding. Therefore, there exists a critical need to explore more effective graph state representation and node embedding method to mitigate computation time.

From the case study of the radar microwave module job shop, it can be found that there is a marginal effect of single-load AGVs (a maximum of one job can be transported at a time), that is, as the number of AGVs increases the makespan of the order is increased after a small decrease. Therefore, it is necessary to integrate multi-load AGVs into DRL environment to reduce the number of transportation tasks and the start time of job processing. In addition, the job shop simulation model established in PS software is used as the DRL environment. Simulation model interacts with algorithm written in Python software to optimize scheduling schemes, which makes sense for actual production.

This study exclusively focuses on production scheduling problem where the transportation time of operations is less than their processing time. However, different industries exhibit variations in the ratio of operations transportation time to their processing time, the number of operations per job, and the number of processable machines for

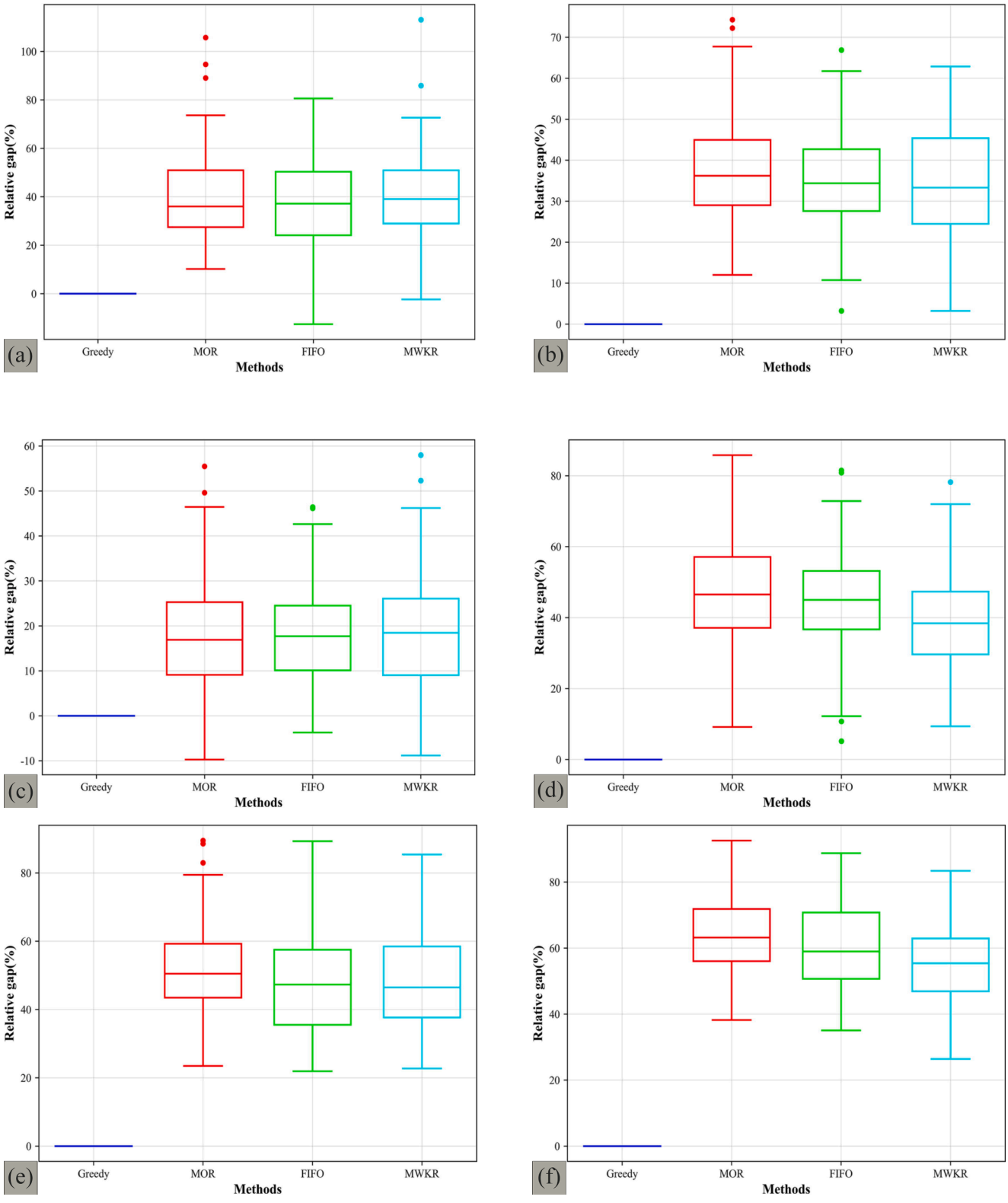

Fig. 6. Relative gaps of the PDRs to the Greedy policy when the first AGV distribution strategy is used:(a) 10 × 5 × 2, (b) 15 × 5 × 2, (c) 20 × 5 × 2, (d) 10 × 10 × 2, (e) 15 × 10 × 3, (f) 15 × 15 × 3.

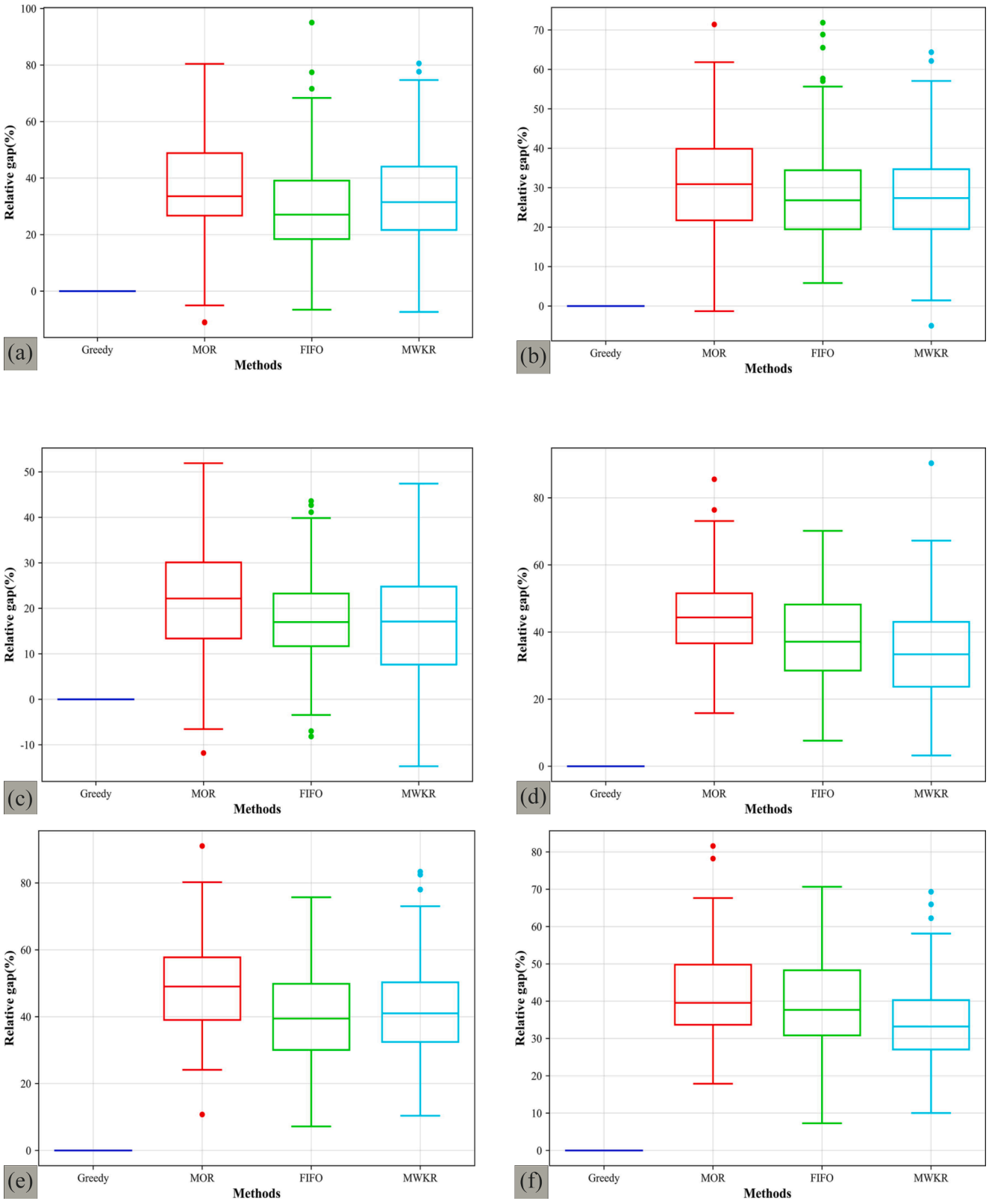

Fig. 7. Relative gaps of the PDRs to the Greedy policy when the second AGV distribution strategy is used: (a) 10 × 5 × 2, (b) 15 × 5 × 2, (c) 20 × 5 × 2, (d) 10 × 10 × 2, (e) 15 × 10 × 3, (f) 15 × 15 × 3.

Table 7
Comparison on benchmark instances from Ham (2020).

HUdata		CP	TS	Greedy1	Greedy2
sdata	C_{max}	2975.1	2984.2	2597.2	2568.3
	Time(s)	–	1542	2.9	2.8
edata	C_{max}	2920.5	2873.6	2561.9	2564.1
	Time(s)	–	1592	2.9	2.9
rdata	C_{max}	2398.9	2415.6	2513.8	2551.6
	Time(s)	–	1661	2.9	2.9
vdata	C_{max}	2538.5	1787.0	2524.4	2582.1
	Time(s)	–	1673	2.9	2.9

Table 8
Processing information for the three products.

Job (Batch size)	(Machine, Processing time) pair of all operations
Job1(20)	O_1(1,40) O_2(2,50) O_3(3,85) O_4(2,20) O_5(4,36) O_6(5,300) O_7(6,145) O_8(7,65) O_9(8,50) O_{10}((9,6), (10,6)) O_{11}((11,150), (12,150)) O_{12}(13,50) O_{13}((14,3),(15,3)) O_{14}(16,135) O_{15}(17,120) O_{16}(18,270) O_{17}(19,100) O_{18}(20,20) O_{19}(21,135) O_{20}(22,80) O_{21}(23,83) O_{22}((24,240), (25,240), (26,240)) O_{23}(20,15) O_{24}(27,100) O_{25}(20,15) O_{26}(20,43) O_{27}(28,33) O_{28}(20,20) O_{29}(28,107)
Job2(5)	O_1(3,106) O_2(2,20) O_3(17,30) O_4(18,270) O_5(8,13) O_6((11,180), (12,180)) O_7(16,34) O_8(22,20) O_9(19,25) O_{10}(20,5) O_{11}(21,34) O_{12}(22,20) O_{13}(23,21) O_{14}((24,240), (25,240), (26,240)) O_{15}(28,8) O_{16}(20,4) O_{17}(27,25) O_{18}(20,4) O_{19}(20,11) O_{20}(20,5) O_{21}(28,27)
Job3(5)	O_1(1,10) O_2(2,50) O_3(3,85) O_4(2,20) O_5(4,9) O_6(5,75) O_7(6,145) O_8(7,16) O_9(8,13) O_{10}((9,6), (10,6)) O_{11}((11,150), (12,150)) O_{12}(13,13) O_{13}((14,3),(15,3)) O_{14}(16,34) O_{15}(17,30) O_{16}(18,270) O_{17}(19,25) O_{18}(20,5) O_{19}(21,34) O_{20}(22,20) O_{21}(23,21) O_{22}((24,240), (25,240), (26,240)) O_{23}(28,8) O_{24}(20,4) O_{25}(27,25) O_{26}(20,4) O_{27}(20,11) O_{28}(20,5) O_{29}(28,27)

Table 9
Orders makeapan with different number of AGVs.

Size	1	2	3	4	5	6
1 × 4 × 4	3329	3238.5	3222.5	3222.5	3282	3282
3 × 12 × 12	9310	8905.5	8875	8872	9049.5	9049.5
5 × 20 × 20	14,933	14247.5	14205.5	14199.5	14229.5	14229.5

Size: number of job1 × number of job2 × number of job3.

operations under distinct production models. As is widely recognized, Design of Experiments allows for the exploration and validation of the impact of different factors on outcomes with a smaller experimental scale and a shorter experimental duration. Ozturk et al. (2019) better validated the superiority of the proposed method over PDRs and composite dispatching rules by utilizing the analysis of variance and Tukey test. Consequently, statistical methods such as analysis of variance and Tukey test can be used in the future to more efficiently corroborate the effectiveness of the proposed approach under the above three different conditions.

CRediT authorship contribution statement

Min Zhang: Investigation, Conceptualization, Methodology, Formal analysis, Writing – original draft, Writing – review & editing. **Liang Wang:** Supervision, Methodology, Data curation, Formal analysis. **Fusheng Qiu:** Investigation, Data curation, Formal analysis. **Xiaorui Liu:** Investigation, Writing – review & editing.

Declaration of Competing Interest

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

Data availability

The authors do not have permission to share data.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.cie.2023.109718. Supplementary data includes pseudocode for Algorithm 1 and Strategy 1.

Appendix B. Supplementary table

Supplementary table consists of three tables. Table 10: the notations used for node embedding procedure in Section 4, Table 11: the travel time of AGV between two destinations in the case study in Section 5, Table 12: all abbreviations in the paper.

Table 10
The notations used for node embedding procedure.

Notations	Meaning
O_{ij}	The operation node in heterogeneous graph
M_k	The machine node in heterogeneous graph
A_l	The AGV node in heterogeneous graph
O-M	The arc connecting node O_{ij} and node M_k
O-A	The arc connecting node O_{ij} and node A_l
x_{ij}	The feature vector of node O_{ij} after linear transformation mapping
y_k	The feature vector of node M_k after linear transformation mapping
z_l	The feature vector of node A_l after linear transformation mapping
u_{ijk}	The feature vector of O-M arc after linear transformation mapping
v_{ijl}	The feature vector of O-A arc after linear transformation mapping
e_{ijl}	The attention coefficient of node A_l to node O_{ij}
e_{ij}	The attention coefficient of node O_{mn} to node O_{ij}
e_{ijk}	The attention coefficient of node O_{ij} to node M_k
e_k	The attention coefficient of node M_l to node M_k
α_{ijl}	The attention coefficient of node A_l to node O_{ij} after normalization
α_{ij}	The attention coefficient of node O_{mn} to node O_{ij} after normalization
α_{ijk}	The attention coefficient of node O_{ij} to node M_k after normalization
α_k	The attention coefficient of node M_l to node M_k after normalization
f_{ijk}	The attention coefficient of node M_k to node O_{ij}
f_{ij}	The attention coefficient of node O_{mn} to node O_{ij}
f_{ijl}	The attention coefficient of node O_{ij} to node A_l
f_l	The attention coefficient of node A_k to node A_l
β_{ijk}	The attention coefficient of node M_k to node O_{ij} after normalization
β_{ij}	The attention coefficient of node O_{mn} to node O_{ij} after normalization
β_{ijl}	The attention coefficient of node O_{ij} to node A_l after normalization
β_l	The attention coefficient of node A_k to node A_l after normalization
$N_t(A_l)$	Number of compatible AGVs of node O_{ij}
$N_t(M_k)$	Number of compatible machines of node O_{ij}
$N_t(O_{ijk})$	Number of O_{ij} that M_k can process
$N_t(O_{ijl})$	Number of O_{ij} that A_l can transport
x_{ij1}	The operation node embedding integrating AGV nodes features
x_{ij2}	The operation node embedding integrating machine nodes features
$y_k^{'}/y_k^{''}/\overline{y}_k^{''}$	The machine node embedding integrating all types nodes features, final machine node embedding and pooled machine node embedding
$z_l^{'}/z_l^{''}/\overline{z}_l^{''}$	The AGV node embedding integrating all types nodes features, final AGV node embedding and pooled AGV node embedding
$x_{ij}^{'}/x_{ij}^{''}/\overline{x}_{ij}^{''}$	The operation node embedding integrating all types nodes features, final operation node embedding and pooled operation node embedding

Table 11
Travel time between machines.

	P	BF	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}	M_{16}	M_{17}	M_{18}	M_{19}	M_{20}	M_{21}	M_{22}	M_{23}	M_{24}	M_{25}	M_{26}	M_{27}	M_{28}
P	0	2	3.5	4.5	5	4.5	5	5.5	5	4	3	3.5	2.5	3	3	2	2.5	4	3	4	1.5	0.5	2	1.5	1	3	2.5	2	3	2.5
BF	2	0	1.5	2.5	4	4.5	5	5.5	5	4	3	3.5	2.5	3	3	2	2.5	4	3	4	1.5	2.5	4	3.5	3	5	4.5	4	3	2.5
M_1	3.5	1.5	0	1	2.5	3	3.5	4	3.5	2.5	2.5	2	2	1.5	3.5	3.5	3	2.5	1.5	2.5	3	4	5.5	5	4.5	6.5	6	5.5	4.5	4
M_2	4.5	2.5	1	0	1.5	2	2.5	3	2.5	3.5	3.5	3	3	2.5	4.5	4.5	4	1.5	1.5	1.5	4	5	6.5	6	5.5	7.5	7	6.5	5.5	5
M_3	5	4	2.5	1.5	0	0.5	1	1.5	3	4	4	3.5	3.5	3	5	5	4.5	2	2	1	4.5	5.5	7	6.5	6	8	7.5	7	7	6.5
M_4	4.5	4.5	3	2	0.5	0	0.5	1	2.5	3.5	3.5	3	3	2.5	4.5	4.5	4	1.5	1.5	0.5	4	5	6.5	6	5.5	7.5	7	6.5	7.5	7
M_5	5	5	3.5	2.5	1	0.5	0	0.5	2	3	3	2.5	2.5	2	4	4	3.5	1	2	1	3.5	4.5	6	5.5	5	7	6.5	7	8	7.5
M_6	5.5	5.5	4	3	1.5	1	0.5	0	1.5	2.5	2.5	2	3	2.5	3.5	3.5	3	1.5	2.5	1.5	4	5	5.5	6	5.5	6.5	7	7.5	8.5	8
M_7	5	5	3.5	2.5	3	2.5	2	1.5	0	1	2	1.5	2.5	2	2	3	2.5	1	2.0	2.0	3.5	4.5	5	5.5	5	6	6.5	7	8	7.5
M_8	4	4	2.5	3.5	4	3.5	3	2.5	1	0	1	0.5	1.5	1	1	2	1.5	2	2	3	2.5	3.5	4	4.5	4	5	5.5	6	7	6.5
M_9	3	3	2.5	3.5	4	3.5	3	2.5	2	1	0	0.5	0.5	1	1	1	0.5	2	2	3	1.5	2.5	3	3.5	3	4	4.5	5	6	5.5
M_{10}	3.5	3.5	2	3	3.5	3	2.5	2	1.5	0.5	0.5	0	1	0.5	1.5	1.5	1	1.5	1.5	2.5	2	3	3.5	4	3.5	4.5	5	5.5	6.5	6
M_{11}	2.5	2.5	2	3	3.5	3	2.5	3	2.5	1.5	0.5	1	0	0.5	1.5	1.5	1	1.5	1.5	2.5	1	2	3.5	3	2.5	4.5	4	4.5	5.5	5
M_{12}	3	3	1.5	2.5	3	2.5	2	2.5	2	1	1	0.5	0.5	0	2	2	1.5	1	1	2	1.5	2.5	4	3.5	3	5	4.5	5	6	5.5
M_{13}	3	3	3.5	4.5	5	4.5	4	3.5	2	1	1	1.5	1.5	2	0	1	0.5	3	3	4	1.5	2.5	3	3.5	3	4	4.5	5	6	5.5
M_{14}	2	2	3.5	4.5	5	4.5	4	3.5	3	2	1	1.5	1.5	2	1	0	0.5	3	3	4	0.5	1.5	2	2.5	2	3	3.5	4	5	4.5
M_{15}	2.5	2.5	3	4	4.5	4	3.5	3	2.5	1.5	0.5	1	1	1.5	0.5	0.5	0	2.5	2.5	3.5	1	2	2.5	3	2.5	3.5	4	4.5	5.5	5
M_{16}	4	4	2.5	1.5	2	1.5	1	1.5	1	2	2	1.5	1.5	1	3	3	2.5	0	1	1	2.5	3.5	5	4.5	4	6	5.5	6	7	6.5
M_{17}	3	3	1.5	1.5	2	1.5	2	2.5	2	2	2	1.5	1.5	1	3	3	2.5	1	0	1	2.5	3.5	5	4.5	4	6	5.5	5	6	5.5
M_{18}	4	4	2.5	1.5	1	0.5	1	1.5	2	3	3	2.5	2.5	2	4	4	3.5	1	1	0	3.5	4.5	6	5.5	5	7	6.5	6	7	6.5
M_{19}	1.5	1.5	3	4	4.5	4	3.5	4	3.5	2.5	1.5	2	1	1.5	1.5	0.5	1	2.5	2.5	3.5	0	1	2.5	2	1.5	3.5	3	3.5	4.5	4
M_{20}	0.5	2.5	4	5	5.5	5	4.5	5	4.5	3.5	2.5	3	2	2.5	2.5	1.5	2	3.5	3.5	4.5	1	0	1.5	1	0.5	2.5	2	2.5	3.5	3
M_{21}	2	4	5.5	6.5	7	6.5	6	5.5	5	4	3	3.5	3.5	4	3.0	2	2.5	5	5	6	2.5	1.5	0	0.5	1	1	1.5	2	3	2.5
M_{22}	1.5	3.5	5	6	6.5	6	5.5	6	5.5	4.5	3.5	4	3	3.5	3.5	2.5	3	4.5	4.5	5.5	2	1	0.5	0	0.5	1.5	1	1.5	2.5	2
M_{23}	1	3	4.5	5.5	6	5.5	5	5.5	5	4	3	3.5	2.5	3	3	2	2.5	4	4	5	1.5	0.5	1	0.5	0	2	1.5	2	3	2.5
M_{24}	3	5	6.5	7.5	8	7.5	7	6.5	6	5	4	4.5	4.5	5	4	3	3.5	6	6	7	3.5	2.5	1	1.5	2	0	0.5	1	2	2.5
M_{25}	2.5	4.5	6	7	7.5	7	6.5	7	6.5	5.5	4.5	5	4	4.5	4.5	3.5	4	5.5	5.5	6.5	3	2	1.5	1	1.5	0.5	0	0.5	1.5	2
M_{26}	2	4	5.5	6.5	7	6.5	7	7.5	7	6	5	5.5	4.5	5	5	4	4.5	6	5	6	3.5	3.5	2	1.5	2	1	0.5	0	1	1.5
M_{27}	3	3	4.5	5.5	7	7.5	8	8.5	8	7	6	6.5	5.5	6	6	5	5.5	7	6	7	4.5	3.5	3	2.5	3	2	1.5	1	0	0.5
M_{28}	2.5	2.5	4	5	6.5	7	7.5	8	7.5	6.5	5.5	6	5	5.5	5.5	4.5	5	6.5	5.5	6.5	4	3	2.5	2	2.5	2.5	2	1.5	0.5	0

Table 12
The abbreviations in the paper.

Abbreviations	Full title
AGV	Automated Guided Vehicle
BF	Buffer
CP	Constraint programming
DFJSP	Dynamic flexible job shop scheduling problem
DFJSP-ITR	Dynamic flexible job shop scheduling problem with insufficient transportation resources
DJSP	Dynamic job shop scheduling problem
DRL	Deep reinforcement learning
ELU	Exponential linear unit
FIFO	First in first out
FJSP	Flexible job shop scheduling problem
GNN	Graph neural network
JSP	Job shop scheduling problem
MDP	Markov Decision Process
MLP	Multilayer perceptron
MOR	Most Operations Remaining
MWKR	Most Work Remaining
NP-hard	Nondeterministic Polynomial hard
PDR	Priority dispatching rule
PPO	Proximal policy optimization
PS	Plant Simulation
RL	Reinforcement learning
SPT	Shortest Processing Time
TS	Tabu search

References

Brandimarte, P. (1993). Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations Research, 41*(3), 157–183. https://doi.org/10.1007/BF02023073

Brucker, P., Jurisch, B., & Sievers, B. (1994). A branch and bound algorithm for the job-shop scheduling problem. *Discrete Applied Mathematics, 49*(1), 107–127. https://doi.org/10.1016/0166-218X(94)90204-6

Berterottière, L., Dauzère-Pérès, S., & Yugma, C. (2023). Flexible job-shop scheduling with transportation resources. *European Journal of Operational Research*. https://doi.org/10.1016/j.ejor.2023.07.036

Caumond, A., Lacomme, P., Moukrim, A., & Tchernev, N. (2009). An MILP for scheduling problems in an FMS with one vehicle. *European Journal of Operational Research, 199*(3), 706–722. https://doi.org/10.1016/j.ejor.2008.03.051

Doh, H.-H., Yu, J.-M., Kim, J.-S., Lee, D.-H., & Nam, S.-H. (2013). A priority scheduling approach for flexible job shops with multiple process plans. *International Journal of Production Research, 51*(12), 3748–3764. https://doi.org/10.1080/00207543.2013.765074

Gao, K. Z., Suganthan, P. N., Chua, T. J., Chong, C. S., Cai, T. X., & Pan, Q. K. (2015). A two-stage artificial bee colony algorithm scheduling flexible job-shop scheduling problem with new job insertion. *Expert Systems with Applications, 42*(21), 7652–7663. https://doi.org/10.1016/j.eswa.2015.06.004

Gui, Y., Tang, D., Zhu, H., Zhang, Y., & Zhang, Z. (2023). Dynamic scheduling for flexible job shop using a deep reinforcement learning approach. *Computers & Industrial Engineering, 180*, Article 109255. https://doi.org/10.1016/j.cie.2023.109255

Hurink, J., Jurisch, B., & Thole, M. (1994). Tabu search for the job-shop scheduling problem with multi-purpose machines. *Operations-Research-Spektrum, 15*(4), 205–215. https://doi.org/10.1007/BF01719451

Heger, J., & Voss, T. (2018). Optimal Scheduling of AGVs in a Reentrant Blocking Job-shop. *Procedia CIRP, 67*, 41–45. https://doi.org/10.1016/j.procir.2017.12.173

Hu, H., Jia, X., He, Q., Fu, S., & Liu, K. (2020). Deep reinforcement learning based AGVs real-time scheduling with mixed rule for flexible shop floor in industry 4.0. *Computers & Industrial Engineering, 149*, Article 106749. https://doi.org/10.1016/j.cie.2020.106749

Ham, A. (2020). Transfer-robot task scheduling in flexible job shop. *Journal of Intelligent Manufacturing, 31*(7), 1783–1793. https://doi.org/10.1007/s10845-020-01537-6

Khalil, E. B., Dai, H., Zhang, Y., Dilkina, B. N., & Song, L. (2017). *Learning Combinatorial Optimization Algorithms over Graphs*. https://doi.org/10.48550/arXiv.1704.01665.

Kool, W., Hoof, H. V., & Welling, M. (2018). *Attention, Learn to Solve Routing Problems! International Conference on Learning Representations*. https://doi.org/10.48550/arXiv.1803.08475.

Kundakcı, N., & Kulak, O. (2016). Hybrid genetic algorithms for minimizing makespan in dynamic job shop scheduling problem. *Computers & Industrial Engineering, 96*, 31–51. https://doi.org/10.1016/j.cie.2016.03.011

Lou, P., Liu, Q., Zhou, Z., Wang, H., & Sun, S. X. (2012). Multi-agent-based proactive–reactive scheduling for a job shop. *The International Journal of Advanced Manufacturing Technology, 59*(1), 311–324. https://doi.org/10.1007/s00170-011-3482-4

Li, Z., Chen, Q., & Koltun, V. (2018). Combinatorial Optimization with Graph Convolutional Networks and Guided Tree Search. *Neural Information Processing Systems*. https://doi.org/10.48550/arXiv.1810.10659

Luo, S. (2020). Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning. *Applied Soft Computing, 91*, Article 106208. https://doi.org/10.1016/j.asoc.2020.106208

Li, Y., Huang, W., Wu, R., & Guo, K. (2020). An improved artificial bee colony algorithm for solving multi-objective low-carbon flexible job shop scheduling problem. *Applied Soft Computing, 95*, Article 106544. https://doi.org/10.1016/j.asoc.2020.106544

Lei, K., Guo, P., Wang, Y., Wu, X., & Zhao, W. (2021). Solve routing problems with a residual edge-graph attention neural network. *Neurocomputing, 508*, 79–98. https://doi.org/10.48550/arXiv.2105.02730

Lei, K., Guo, P., Zhao, W., Wang, Y., Qian, L., Meng, X., & Tang, L. (2022). A multi-action deep reinforcement learning framework for flexible Job-shop scheduling problem. *Expert Systems with Applications, 205*, Article 117796. https://doi.org/10.1016/j.eswa.2022.117796

Li, X., Guo, X., Tang, H., Wu, R., Wang, L., Pang, S., … Li, X. (2022). Survey of integrated flexible job shop scheduling problems. *Computers & Industrial Engineering, 174*, Article 108786. https://doi.org/10.1016/j.cie.2022.108786

Li, Y., Gu, W., Yuan, M., & Tang, Y. (2022). Real-time data-driven dynamic scheduling for flexible job shop with insufficient transportation resources using hybrid deep Q network. *Robotics and Computer-Integrated Manufacturing, 74*, Article 102283. https://doi.org/10.1016/j.rcim.2021.102283

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., … Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature, 518*(7540), 529–533. https://doi.org/10.1038/nature14236

Ning, T., Huang, M., Liang, X., & Jin, H. (2016). A novel dynamic scheduling strategy for solving flexible job-shop problems. *Journal of Ambient Intelligence and Humanized Computing, 7*(5), 721–729. https://doi.org/10.1007/s12652-016-0370-7

Nouiri, M., Bekrar, A., & Trentesaux, D. (2018). Towards Energy Efficient Scheduling and Rescheduling for Dynamic Flexible Job Shop Problem. *IFAC-PapersOnLine, 51*(11), 1275–1280. https://doi.org/10.1016/j.ifacol.2018.08.357

Ni, F., Hao, J., Lu, J., Tong, X., Yuan, M., Duan, J., Ma, Y., & He, K. (2021). A Multi-Graph Attributed Reinforcement Learning based Optimization Algorithm for Large-scale Hybrid Flow Shop Scheduling Problem. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, Virtual Event, Singapore*. https://doi.org/10.1145/3447548.3467135

Ozturk, G., Bahadir, O., & Teymourifar, A. (2019). Extracting priority rules for dynamic multi-objective flexible job shop scheduling problems using gene expression programming. *International Journal of Production Research, 57*(10), 3121–3137. https://doi.org/10.1080/00207543.2018.1543964

Park, J., Chun, J., Kim, S.-H., Kim, Y., & Park, J. (2021). Learning to schedule job-shop problems: Representation and policy learning using graph neural network and reinforcement learning. *International Journal of Production Research, 59*, 3360–3377. https://doi.org/10.1080/00207543.2020.1870013

Peng, Z., Zhang, H., Tang, H., Feng, Y., & Yin, W. (2022). Research on flexible job-shop scheduling problem in green sustainable manufacturing based on learning effect. *Journal of Intelligent Manufacturing, 33*(6), 1725–1746. https://doi.org/10.1007/s10845-020-01713-8

Qin, H., Fan, P., Tang, H., Huang, P., Fang, B., & Pan, S. (2019). An effective hybrid discrete grey wolf optimizer for the casting production scheduling problem with multi-objective and multi-constraint. *Computers & Industrial Engineering, 128*, 458–476. https://doi.org/10.1016/j.cie.2018.12.061

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., … Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature, 529*(7587), 484–489. https://doi.org/10.1038/nature16961

Shahrabi, J., Adibi, M. A., & Mahootchi, M. (2017). A reinforcement learning approach to parameter estimation in dynamic job shop scheduling. *Computers & Industrial Engineering, 110*, 75–82. https://doi.org/10.1016/j.cie.2017.05.026

Shiue, Y.-R., Lee, K.-C., & Su, C.-T. (2018). Real-time scheduling for a smart factory using a reinforcement learning approach. *Computers & Industrial Engineering, 125*, 604–614. https://doi.org/10.1016/j.cie.2018.03.039

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. ArXiv, abs/1707.06347. https://doi.org/10.48550/arXiv.1707.06347.

Shahgholi Zadeh, M., Katebi, Y., & Doniavi, A. (2019). A heuristic model for dynamic flexible job shop scheduling problem considering variable processing times. *International Journal of Production Research, 57*(10), 3020–3035. https://doi.org/10.1080/00207543.2018.1524165

Song, W., Chen, X., Li, Q., & Cao, Z. (2023). Flexible Job-Shop Scheduling via Graph Neural Network and Deep Reinforcement Learning. *IEEE Transactions on Industrial Informatics, 19*(2), 1600–1610. https://doi.org/10.1109/TII.2022.3189725

Wang, Y.-F. (2020). Adaptive job shop scheduling strategy based on weighted Q-learning algorithm. *Journal of Intelligent Manufacturing, 31*(2), 417–432. https://doi.org/10.1007/s10845-018-1454-3

Xanthopoulos, A. S., Koulouriotis, D. E., Tourassis, V. D., & Emiris, D. M. (2013). Intelligent controllers for bi-objective dynamic scheduling on a single machine with sequence-dependent setups. *Applied Soft Computing, 13*(12), 4704–4717. https://doi.org/10.1016/j.asoc.2013.07.015

Yildiz, A. R. (2013). Comparison of evolutionary-based optimization algorithms for structural design optimization. *Engineering Applications of Artificial Intelligence, 26*(1), 327–333. https://doi.org/10.1016/j.engappai.2012.05.014

Zhang, Z., Zheng, L.i., Li, N.a., Wang, W., Zhong, S., & Kaishun, H.u. (2012). Minimizing mean weighted tardiness in unrelated parallel machine scheduling with reinforcement learning. *Computers & Operations Research, 39*(7), 1315–1324. https://doi.org/10.1016/j.cor.2011.07.019

Zhang, C., Song, W., Cao, Z., Zhang, J., Tan, P. S., & Xu, C. (2020). Learning to Dispatch for Job Shop Scheduling via Deep Reinforcement Learning. https://doi.org/10.48550/arXiv.2010.12367.