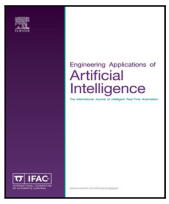
Engineering Applications of Artificial Intelligence 139 (2025) 109557

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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



Research paper

An effective multi-agent-based graph reinforcement learning method for solving flexible job shop scheduling problem

Lanjun Wan [a,*], Long Fu [a], Changyun Li [a], Keqin Li [b]

[a] School of Computer Science, Hunan University of Technology, Zhuzhou 412007, China
[b] Department of Computer Science, State University of New York, New Paltz, NY, 12561, USA

ARTICLE INFO

Keywords:
Flexible job shop scheduling problem
Graph attention network
Graph reinforcement learning
Multi-agent

ABSTRACT

Flexible job shop scheduling problem (FJSP) is a complex optimization problem in intelligent manufacturing and plays a key role in improving productivity, which is characterized by that each operation can be processed by multiple machines. Most current research into FJSP focuses on finding a higher-quality scheduling scheme in a shorter time. However, existing studies are hard to optimize the operation sequencing and machine assignment strategies simultaneously, which is critical for making the optimal scheduling decision. Therefore, a multi-agent-based graph reinforcement learning (MAGRL) method is proposed to effectively solve FJSP. Firstly, the FJSP is modeled into two Markov decision processes (MDPs), where the operation and machine agents are adopted to control the operation sequencing and machine assignment respectively. Secondly, to effectively predict the operation sequencing and machine assignment strategies, an encoder-double-decoder architecture is designed, including an improved graph attention network (IGAT)-based encoder, an operation strategy network-based decoder, and a machine strategy network-based decoder. Thirdly, an automatic entropy adjustment multi-agent proximal policy optimization (AEA-MAPPO) algorithm is proposed for effectively training the operation and machine strategy networks to optimize the operation sequencing and machine assignment strategies simultaneously. Finally, the effectiveness of MAGRL is verified through experimental comparisons with the classical scheduling rules and state-of-the-art methods to solve FJSP. The results achieved on the randomly generated FJSP instances and two common benchmarks indicate that MAGRL can consume less solution time to achieve higher solution quality in solving different-sized FJSP instances, and the overall performance of MAGRL is superior to that of the comparison methods.

1. Introduction

Nowadays, the industrial field is going through a revolutionary changes. The wide applications of the digital technology and automation systems are changing the face of the manufacturing industry. The traditional manufacturing is difficult to adapt to the personalized and customized market demands (Jiang et al., 2023). As the key to achieve efficient and flexible productions under the intelligent manufacturing environment, modern manufacturing is shifting its research focus into flexible job shop scheduling (Zhang et al., 2023). FJSP is a variant of the job shop scheduling problem (JSSP), and both of them belong to the non-deterministic polynomial hard combinatorial optimization problems (Dauzère-Pérès et al., 2024). FJSP considers the diversity and flexibility of the production equipment. Each operation in a job can be assigned to different compatible machines for processing, significantly increasing the complexities of combination and decision-making.

The traditional methods for solving FJSP mainly involve the exact algorithms, the scheduling rules, and the meta-heuristic algorithms (MHAs) (Li et al., 2022). The exact algorithms (Fekih et al., 2023; Müller et al., 2022) are the earliest methods used to solve FJSP, which seeks the optimal solution of FJSP by mathematically modeling it. Fekih et al. (2023) proposed the mixed-integer linear programming model and constrained programming model for FJSP and verified their ability to accurately solve FJSP of different sizes in a reasonable time. Müller et al. (2022) studied five different constraint programming solvers for small and medium-sized FJSP instances, aiming to obtain minimum makespan. Wan et al. (2023) designed a self-triggered finite-time controller to minimize resource consumption. Song et al. (2023) proposed a self-triggered control solution to optimize the performance of resource-constrained systems. Tao et al. (2024) studied the quantized iterative learning control for improving the system performance under limited resources. The scheduling rules (Teymourifar et al., 2020; Jun et al., 2019) can quickly generate the feasible scheduling schemes by designing the heuristic rules according to experience and problem characteristics. In addition, Teymourifar et al. (2020) presented

* Corresponding author.
E-mail address: wanlanjun@hut.edu.cn (L. Wan).

https://doi.org/10.1016/j.engappai.2024.109557
Received 20 June 2024; Received in revised form 20 September 2024; Accepted 23 October 2024

a new heuristic scheduling rule extraction method, and the extracted scheduling rules can effectively cope with dynamic flexible job shop scheduling problem (DFJSP) with random job arrivals and machine breakdowns. Jun et al. (2019) proposed a new scheduling rule generation method that uses a random forest algorithm to generate the best scheduling rule to solve FJSP and can quickly obtain better makespan. The MHAs Li et al. (2023a), Liu et al. (2024) and Fan et al. (2024) can generate a variety of solutions by combining multiple heuristic rules to improve the ability of FJSP in finding the optimal solution space. Li et al. (2023a) used an artificial bee colony algorithm combined with reinforcement learning (RL) to train the optimal scheduling strategy and the optimal scheduling scheme of sublots through different stages, which improves the solution quality of FJSP with lot streaming. Liu et al. (2024) proposed a multi-objective adaptive large neighborhood search approach to solve the multi-objective DFJSP with transportation resources, significantly enhancing the exploration ability of multi-objective solutions of DFJSP. Fan et al. (2024) introduced an improved tuna swarm optimization algorithm for DFJSP with machine breakdowns, which effectively reduces the makespan of DFJSP. The existing studies indicated that the exact algorithms and MHAs can achieve high solution quality while the scheduling rules can consume less computing time when solving FJSP instances. However, when the size of FJSP instances increases, the computational complexities of these algorithms also increase significantly, posing a huge challenge to the solution efficiency and the requirement for computing resources. In addition, the solution quality is not high and it is difficult to effectively deal with FJSP in different scenarios when using the scheduling rules.

Due to the traditional methods to solve FJSP have some limitations, recently the deep reinforcement learning (DRL) has brought a new paradigm for solving FJSP (Li et al., 2023b). DRL utilizes the perceptual ability of deep learning (Gheisari et al., 2023) to extract multiple scheduling states related to FJSP and treat them as the actions of the agent, in which the agent continuously learns and improves scheduling strategies to find the optimal scheduling scheme by exploiting the decision-making ability of RL (Wang et al., 2021). Luo (2020) proposed a DRL method based on deep Q-networks for DFJSP with new job inserts, in which the extracted scheduling state information is designed as a universal scheduling rule for the agent to train, minimizing the total delay of DFJSP. Yuan et al. (2024) developed a new DRL framework based on the multi-layer perceptron (MLP), effectively extracting the scheduling state information of FJSP for the agent to train the scheduling strategy and thereby improving the solution quality of FJSP. Zhao et al. (2024) designed a new proximal policy optimization (PPO) algorithm to train the real-time scheduling strategy, and a strategy network based on the attention mechanism is used to improve the training efficiency of the scheduling strategy, effectively solving DFJSP with random job arrivals. The aforementioned DRL methods to solve FJSP typically model the FJSP as a Markov decision process (MDP), the corresponding states, actions, and rewards are designed to train the agent by extracting the scheduling state information of FJSP, and the agent continuously learns and optimizes the scheduling strategy, enabling the trained scheduling strategy can effectively solve FJSP instances of different sizes. However, the quality of the trained scheduling strategy largely depends on whether the scheduling state information of FJSP can be effectively extracted. Therefore, it is a key problem how to effectively extract the scheduling state information of different FJSP instances.

Recently, the graph reinforcement learning (GRL) has attracted the attention of researchers in production scheduling, and it solves the difficult of extracting the key scheduling state information of different FJSP instances by combining DRL with graph neural network (GNN) (Munikoti et al., 2024). For instance, Lei et al. (2022) proposed a new GRL method, which uses the graph to represent the local scheduling states of FJSP, the graph isomorphic network to extract the node features of the graph, and the DRL algorithm to train the scheduling strategy to effectively solve FJSP. In addition, Song et al. (2022) developed a heterogeneous GNN framework. In this framework, heterogeneous graph is used for the state representation of MDP, the graph attention network (GAT) is used to process the structure information of heterogeneous graph, and the PPO algorithm is employed for training the strategy network, which improves the solution quality of FJSP. Generally, for the existing research solving FJSP using GRL, first, the graph-based model is utilized to represent FJSP. Second, GNN is used to encode the scheduling state information contained in the operation and machine nodes of the graph. Third, a corresponding strategy network is designed to decode the encoded information from GNN. Finally, a DRL method is adopted to train the scheduling strategy generated after decoding the strategy network. The scheduling models trained by these GRL methods significantly improve the quality and speed of solving different FJSP instances.

Overall, compared with the other methods to solve FJSP, the GRL methods have exhibited better performance. However, to further increase the solution quality and solution efficiency of FJSP, the following issues need to be addressed. (1) Operation sequencing and machine assignment are two core problems that need to be addressed for solving FJSP. Most existing GRL methods use a single agent to control the action training of the operation-machine pairs, which is not conducive to finding the optimal operation sequencing and machine assignment strategies. (2) The design of GNN should consider the feature extractions of operation and machine nodes, which will affect whether more important information related to the scheduling strategy can be extracted. (3) The design of the DRL algorithm is also crucial because it directly affects the quality of the trained scheduling strategy.

By fully considering the above issues, a novel multi-agent-based graph reinforcement learning method is proposed to effectively solve FJSP. First, a heterogeneous graph is adopted to represent the global scheduling states of FJSP, and the FJSP is modeled into two MDPs, in which the operation and machine agents are used to effectively control the operation sequencing and machine assignment, respectively. Next, the encoder-double-decoder architecture is constructed to handle the complex scheduling information of FJSP. The IGAT is designed as an encoder to effectively extract the features of operation and machine nodes, and the operation strategy and machine strategy networks are designed as double decoders for to effectively predict the operation sequencing and machine assignment strategies. In addition, the AEA-MAPPO algorithm is proposed for effectively training the operation and machine strategy networks, aiming to learn the optimal operation sequencing and machine assignment strategies for the scheduling decision in a shorter time. The effectiveness of the proposed MAGRL method is verified through experimental comparisons with the classical scheduling rules and state-of-the-art methods to solve FJSP. The results achieved on the dataset consisting of randomly generated FJSP instances and two common benchmarks show that MAGRL outperforms the comparison methods in solving FJSP instances of different sizes.

The main contributions of this paper are as follows.

- Cooperation of dual-agent. The FJSP is modeled as two Markov decision processes, in which the operation agent and the machine agent are responsible for controlling the selection of the operations and the assignment of the machines, respectively. The operation sequencing and machine assignment can be done simultaneously through the cooperation of the operation and machine agents, which is conducive to making the optimal scheduling decision.
- Construction of encoder-double-decoder architecture. To effectively predicting the operation sequencing and machine assignment strategies, the global scheduling states of FJSP are represented through a heterogeneous graph, and the encoder-double-decoder architecture is constructed to handle the complex scheduling information contained in the operation and machine nodes of the heterogeneous graph. The improved graph attention network is designed as an encoder to encode the node features

Nomenclature

FJSP	Flexible job shop scheduling problem
MAGRL	Multi-agent-based graph reinforcement learning
MDPs	Markov decision processes
IGAT	Improved graph attention network
AEA-MAPPO	Automatic entropy adjustment multi-agent proximal policy optimization
JSSP	Job shop scheduling problem
DFJSP	Dynamic flexible job shop scheduling problem
DRL	Deep reinforcement learning
RL	Reinforcement learning
PPO	Proximal policy optimization
MLP	Multi-layer perceptron
MDP	Markov decision process
GRL	Graph reinforcement learning
GNN	Graph neural network
GAT	Graph attention network
MHAs	Meta-heuristic algorithms
FIFO	First in first out
SPT	Shortest processing time
MOPNR	Most operations remaining
MWKR	Most work remaining
MPPO	Multi-proximal policy optimization
IPSO	Improved particle swarm optimization
SLGA	Self-learning genetic algorithm
SLABC	Self-learning artificial bee colony
GNN-DRL	Graph neural network and deep reinforcement learning

of the heterogeneous graph. The operation strategy network-based decoder and machine strategy network-based decoder are designed to transform the output of the encoder into the probabilities of the operation sequencing and machine assignment actions, respectively.

- Integration of multi-agent and GRL. Different from the traditional RL methods using a single agent to control the action training of the operation-machine pairs, multi-agent and GRL are integrated in MAGRL, and the automatic entropy adjustment multi-agent proximal policy optimization algorithm is proposed. In this algorithm, the operation strategy and machine strategy networks are constructed to learn the operation sequencing and machine assignment actions, respectively. The entropy objective function is introduced to balance the randomness of strategies and actions, improving the ability to find the optimal operation sequencing and machine assignment strategies.

The remainder of the paper is organized as follows. The preliminaries are introduced in Section 2. The proposed method is described in Section 3. The experimental results and analysis are provided in Section 4. Conclusions are given in Section 5.

2. Preliminaries

2.1. Definition of FJSP

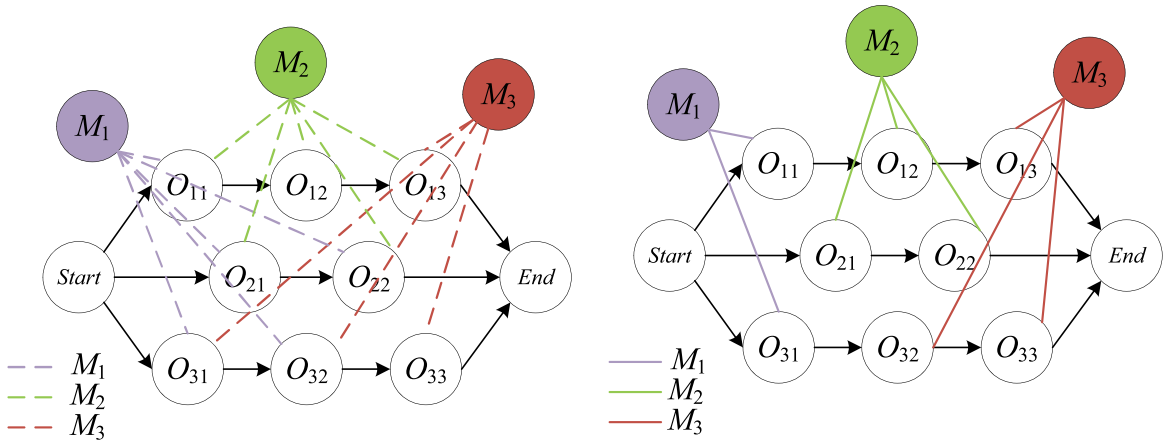
The description of a FJSP instance is as follows. A $n\times m$ FJSP instance contains n jobs and m machines. $J=\{J_1,\ J_2,\ \ldots,\ J_n\}$ is the set of all n jobs, and $M=\{M_1,\ M_2,\ \ldots,\ M_m\}$ is the set of all m machines. The job J_i contains n_i operations O_i with the sequence constraints, where $O_i=\{O_{i1},\ O_{i2},\ \ldots,\ O_{in_i}\}$. For the job J_i, the next operation O_{ij+1} can only be started after the current operation O_{ij} has been completed. For the operation O_{ij}, there is a set of available machines M_{ij}. The operation O_{ij} can be processed on any available machine in M_{ij}, and let P_{ijk} be the processing time of the operation O_{ij} on the available machine M_k. Note that one machine can only be used for processing one operation at the same timestep. In addition, once the operation has been assigned to a machine for processing, it cannot be interrupted until this operation is completed. For an instance of FJSP, the ultimate goal is to consume the minimum makespan to process all operations. Therefore, the objective function to solve FJSP is set to $C_{\max}=\max\{C_{in_i}\}$, where C_{in_i} is the completion time of the job J_i. For a better understanding, a 3×3 FJSP instance is taken as an example, as shown in Table 1.

Table 1
A 3×3 FJSP instance.

Job	Operation	M_1	M_2	M_3
J_1	O_{11}	13	15	–
	O_{12}	–	24	–
	O_{13}	–	–	16
J_2	O_{21}	21	18	–
	O_{22}	15	15	–
J_3	O_{31}	8	–	8
	O_{32}	18	–	18
	O_{33}	–	–	25

2.2. Heterogeneous graph representation of FJSP

The JSSP is usually represented using the disjunctive graph (Su et al., 2023), however it lacks representation for machine nodes, there are some limitations in the representation of FJSP. In order to better represent the operations and machines in FJSP and the relationship between operations and machines in the form of graph, a heterogeneous graph (Lei et al., 2022) is used to represent FJSP. The heterogeneous graph used to represent FJSP is defined as a four tuple $HG=(\mathcal{O},\ \mathcal{M},\ \mathcal{C},\ E_t)$. $\mathcal{O}=\{\mathcal{O}_{ij}\mid \forall i,j\}\cup\{Start, End\}$ denotes all operation nodes, where $\{\mathcal{O}_{ij}\mid \forall i,j\}$ represents all real operation nodes and *Start*



(a) FJSP instance (b) Feasible solution

Fig. 1. Heterogeneous graph representation of a 3 × 3 FJSP instance.

and *End* indicate that the start and end of each job respectively. $\mathscr{M}$ denotes all machine nodes. $\mathscr{C}$ is the set of directed connecting arcs, where the path that connects from *Start* to *End* with a solid line represents the execution sequence constraints between all operations of a job. E_t is the set of the undirected O-M arcs connecting operation and machine nodes. Fig. 1 presents the heterogeneous graph representation of a FJSP instance containing three machines and three jobs.

3. Proposed method

Fig. 2 presents the overall process of solving FJSP with MAGRL. Firstly, the global scheduling states of a FJSP instance are represented through a heterogeneous graph, and the FJSP is modeled as two MDPs, in which the operation agent is responsible for the selection of the operations and the machine agent is responsible for controlling the assignment of the machines. Secondly, an encoder-double-decoder architecture is used to encode the structure information of the heterogeneous graph, an IGAT-based encoder is used for encoding the features of the machine and operation nodes of the heterogeneous graph. A decoder based on the operation strategy network and a decoder based on the machine strategy network transform the features of encoded operation and machine nodes into the probabilities of operation sequencing and machine assignment actions, respectively. Finally, the operation strategy and machine strategy networks are trained using the AEA-MAPPO algorithm to obtain the optimal operation sequencing and machine assignment strategies.

3.1. MDP formulation

The task to solve FJSP can be seen as the following two steps. The first step is to choose an eligible operation that has not been processed in a job. The second step is to assign an available machine to process the selected operation in the first step. Repeating the above two steps until the processing of all operations of all jobs is completed. To better solve the FJSP task, two agents are used to control the process of solving the FJSP task. The operation agent is responsible for selecting an operation, and the machine agent is responsible for assigning an available machine for the selected operation. The MDPs of the operation and machine agents are as follows.

3.1.1. MDP of the operation agent

State: The local state s_t^o indicates the scheduling state of the operation O_{ij} at timestep t. The operation node $\mathscr{O}_{ij}$ contains four features, which are $SSF(O_{ij})$, $PT(O_{ij})$, $\left|NM_t(\mathscr{O}_{ij})\right|$, and $\left|RON_t(J_n)\right|$. $SSF(O_{ij})$ indicates the state flag of the operation O_{ij}. If O_{ij} is not scheduled, the value of $SSF(O_{ij})$ is 0; otherwise, the value of $SSF(O_{ij})$ is 1. $PT(O_{ij})$ indicates the processing time of O_{ij}. If O_{ij} has been scheduled, $PT(O_{ij}) = P_{ijk}$; otherwise, $PT(O_{ij}) = PT(O_{ij-1}) + \min(P_{ijl})$, where $M_l \in M_{ij}$. $\left|NM_t(\mathscr{O}_{ij})\right|$ indicates the number of the neighbor machine nodes of $\mathscr{O}_{ij}$ at timestep t. $\left|RON_t(J_n)\right|$ indicates the number of operations in the job J_n that have not been scheduled at timestep t.

Action: A_t^o represents the eligible operation sequencing action space. At each timestep, the operation agent controls the operation sequencing by executing the action $a_t^o \in A_t^o$.

Transition: At timestep t, and the operation agent executes the action a_t^o, the operation state will be transformed from s_t^o to s_{t+1}^o, indicating that the operation agent selects an operation for processing.

Reward: Because the whole FJSP task requires the cooperation of operation and machine agents, the two agents use the same joint reward function to calculate the reward. The joint reward function is defined as $r_t(s_t, a_t, s_{t+1}) = C_{\max}(s_t) - C_{\max}(s_{t+1})$.

Strategy: The operation sequencing action a_t^o is trained with the operation sequencing strategy $\pi_{\phi_o}\left(a_t^o \middle| s_t^o\right)$.

3.1.2. MDP of the machine agent

State: The local state s_t^m indicates the scheduling state of machine M_k at timestep t. The machine node $\mathscr{M}_k$ contains three features, which are $\left|NO_t(M_k)\right|$, $CT_t(M_k)$, and $\hat{P}_{ijk}$. $\left|NO_t(M_k)\right|$ indicates the number of neighbor operation nodes of $\mathscr{M}_k$ at timestep t. $CT_t(M_k)$ indicates the completion time of M_k at timestep t, that is, the time when M_k has finished processing all the operations assigned to it and started the next new operation. If the operation O_{ij} can be processed on $M_k \in M_{ij}$, $\hat{P}_{ijk}$ represents the actual processing time of O_{ij}, that is, $\hat{P}_{ijk} = P_{ijk}$; otherwise, $\hat{P}_{ijk}$ is the estimated processing time of O_{ij}, that is, $\hat{P}_{ijk} = \frac{1}{|M_{ij}|}\sum_{M_l \in M_{ij}} P_{ijl}$.

Action: A_t^m represents the available machine assignment action space. At each timestep, the machine agent controls the machine assignment by executing the actions $a_t^m \in A_t^m$.

Transition: At timestep t, the machine agent executes the action a_t^m, and the machine state will be transformed from s_t^m to s_{t+1}^m, indicating that the machine agent has assigned the available machine to process the operation selected through the operation agent.

Reward: The joint reward function $r_t(s_t, a_t, s_{t+1}) = C_{\max}(s_t) - C_{\max}(s_{t+1})$ is used to calculate the reward.

Strategy: The machine assignment action a_t^m is trained with the machine assignment strategy $\pi_{\phi_m}\left(a_t^m \middle| s_t^m\right)$.

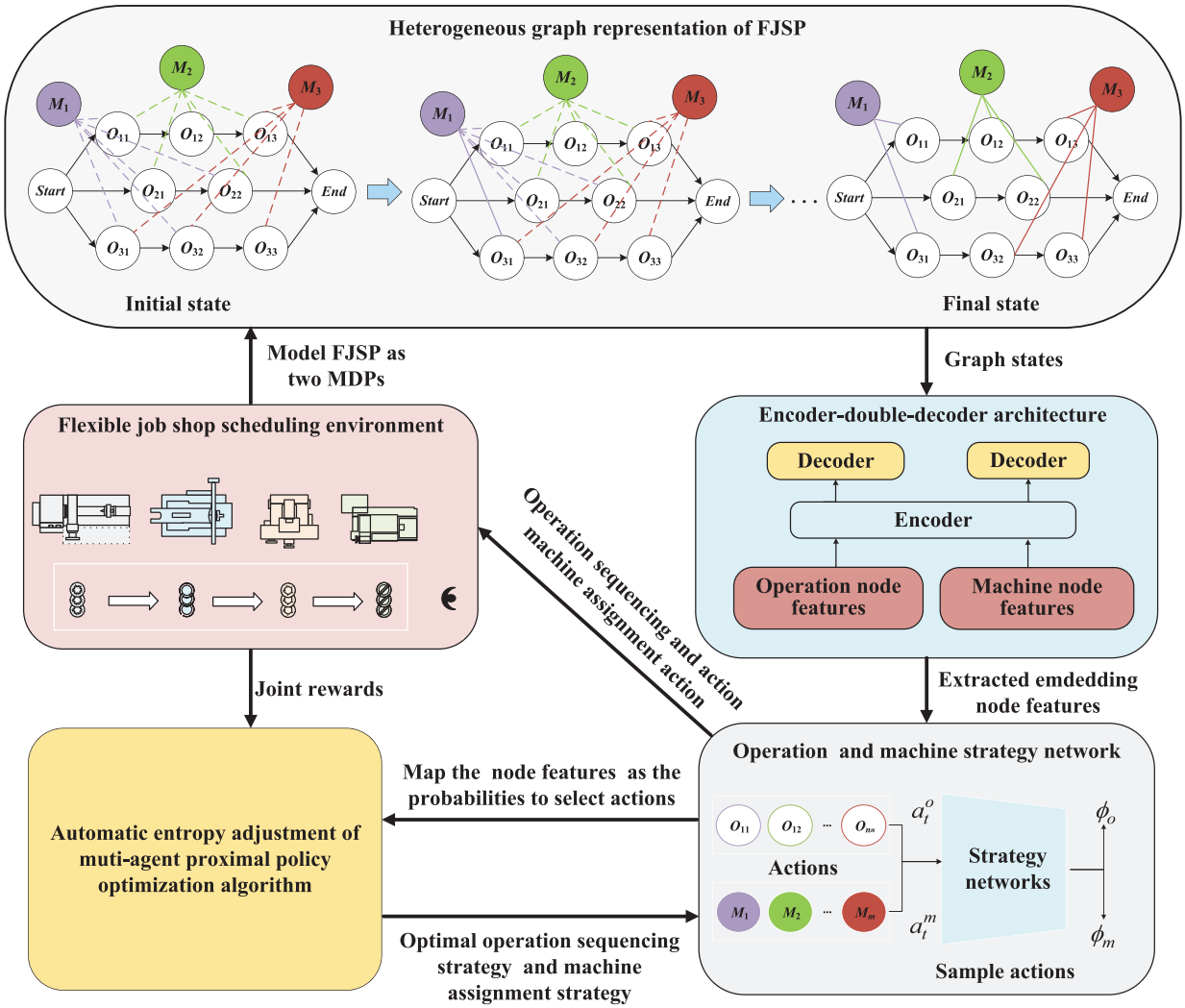


Fig. 2. Overall process of solving FJSP with MAGRL.

3.2. Encoder-double-decoder architecture to solve FJSP

To better solve the FJSP task, an encoder-double-decoder architecture is designed, as shown in Fig. 3. The operation and machine nodes are encoded using the IGAT-based encoder. The operation strategy network-based decoder and machine strategy network-based decoder transform the output of the encoder into the probabilities of the operation sequencing and machine assignment actions respectively, predicting the operation sequencing and machine assignment strategies.

3.3. IGAT-based encoder

GAT (Veličković et al., 2017) is a powerful GNN model that can handle the complex graph structure data. By introducing the attention mechanism, it improves the ability to extract the node features. However, for FJSP, there are many different types of relationships between operation nodes and between machine and operation nodes. The conventional GAT is not suitable to model the complex relationship between nodes. Therefore, an IGAT is proposed to better balance the modeling of different nodes and extract more key features. The features between operation nodes and those of between machine and operation nodes are extracted using IGAT, which is convenient for the subsequent optimal operation sequencing and machine assignment.

3.3.1. Operation nodes encoding based on IGAT

In the heterogeneous graph, the target operation node $\mathcal{O}_{ij}$, its direct predecessor $\mathcal{O}_{ij-1}$, and its direct successor $\mathcal{O}_{ij+1}$ are connected by a directed arc to represent the sequence constraint of the operation processing. The features of $\mathcal{O}_{ij}$ are closely related to those of $\mathcal{O}_{ij-1}$ and $\mathcal{O}_{ij+1}$. Therefore, the original features of $\mathcal{O}_{ij}$, $\mathcal{O}_{ij-1}$, and $\mathcal{O}_{ij+1}$ are processed by IGAT to complete the embedding of the features of $\mathcal{O}_{ij}$. The process of embedding $\mathcal{O}_{ij}$ into the heterogeneous graph is as follows.

Step 1: By processing the original features of $\mathcal{M}_k$ and its neighbor operation node $\mathcal{O}_{ij} \in NO_t\left(\mathcal{M}_k\right)$ to obtain the attention coefficient e_{ijk} of $\mathcal{O}_{ij}$ to $\mathcal{M}_k$:

$$e_{iju} = \text{LeakyReLU}\left(a^T\left[W_m F_{\mathcal{M}_u} \middle\| W_o F_{\mathcal{O}_{ij}}\right]\right), \tag{1}$$

where a^T is the transposition of the attention vector a, W_m and W_o are two learnable linear transformation matrices, and $F_{\mathcal{M}_u}$ and $F_{\mathcal{O}_{ij}}$ are the original feature vectors of $\mathcal{M}_u$ and $\mathcal{O}_{ij}$ respectively.

Step 2: Normalize the attention coefficient e_{iju} using the softmax function:

$$\alpha_{iju} = \text{softmax}_u\left(e_{iju}\right). \tag{2}$$

Step 3: By processing the original features of $\mathcal{O}_{ij}$, $\mathcal{O}_{ij-1}$, and $\mathcal{O}_{ij+1}$ to obtain the attention coefficient e_{ij} of $\mathcal{O}_{ij}$:

$$e_{ij} = \text{LeakyReLU}\left(a^T\left[W_o F_{\mathcal{O}_{ij}} \middle\| W_o F_{\mathcal{O}_{ij-1}} \middle\| W_o F_{\mathcal{O}_{ij+1}}\right]\right), \tag{3}$$

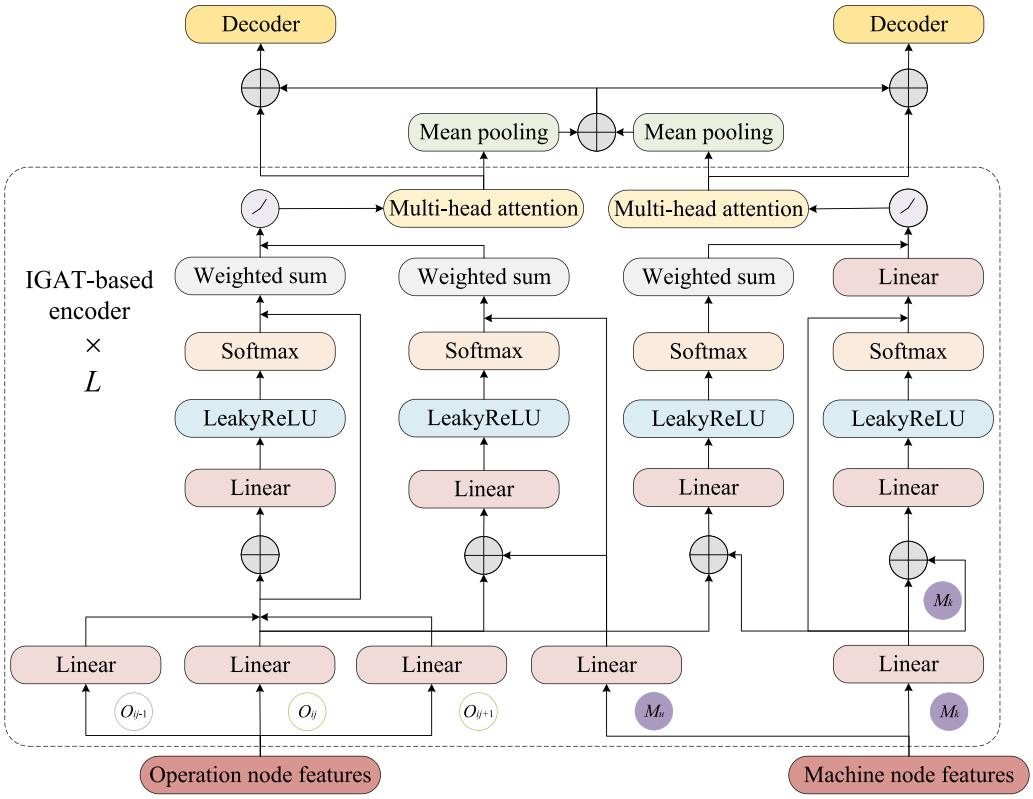


Fig. 3. Design of the encoder-double-decoder architecture to solve FJSP.

where $F_{\mathcal{O}_{ij-1}}$ and $F_{\mathcal{O}_{ij+1}}$ are the original feature vectors of $\mathcal{O}_{ij-1}$ and $\mathcal{O}_{ij+1}$, respectively.

Step 4: Normalize the attention coefficient e_{ij} using the softmax function:

$$\alpha_{ij} = \text{softmax}_j\left(e_{ij}\right). \tag{4}$$

Step 5: Firstly, the weighted linear sums of the original feature vectors $F_{\mathcal{O}_{ij}}$, $F_{\mathcal{O}_{ij-1}}$, and $F_{\mathcal{O}_{ij+1}}$ of $\mathcal{O}_{ij}$, $\mathcal{O}_{ij-1}$, and $\mathcal{O}_{ij+1}$ are calculated. Secondly, the weighted linear sum of the original feature vector $F_{\mathcal{M}_u}$ of $\mathcal{M}_u$ is calculated. Thirdly, the sum of the results obtained in the previous two steps is sent to a nonlinear activation function $\sigma(\bullet)$ for processing. Finally, the embedding feature vector $h_{\mathcal{O}_{ij}}$ of $\mathcal{O}_{ij}$ is calculated using the multi-head attention mechanism:

$$h_{\mathcal{O}_{ij}} = \sum_{k=1}^{\mathcal{K}} \sigma\left(\sum_{y=j-1}^{j+1} \alpha_{ij}^{k} W_o^{k} F_{\mathcal{O}_{iy}} + \sum_{\mathcal{M}_u \in NM_t(\mathcal{O}_{ij})} \alpha_{iju}^{k} W_m^{k} F_{\mathcal{M}_u}\right), \tag{5}$$

where $\mathcal{K}$ denotes the number of attention heads.

3.3.2. Machine nodes encoding based on IGAT

In the heterogeneous graph, the undirected O-M arc is used to connect the machine node $\mathcal{M}_k$ with its multiple neighbor operation nodes, indicating that the operations corresponding to these operation nodes can be processed on the machine M_k corresponding to $\mathcal{M}_k$. Therefore, when IGAT is used to encode the features of $\mathcal{M}_k$, the original features of $\mathcal{M}_k$ and its neighbor operation nodes should be considered. The process of embedding $\mathcal{M}_k$ into the heterogeneous graph is as follows.

Step 1: By processing the original features of $\mathcal{M}_k$ and its neighbor operation node $\mathcal{O}_{ij} \in NO_t\left(\mathcal{M}_k\right)$ to obtain the attention coefficient e_{ijk} of $\mathcal{O}_{ij}$ to $\mathcal{M}_k$:

$$e_{ijk} = \text{LeakyReLU}\left(b^T\left[W_m F_{\mathcal{M}_k} \middle\| W_o F_{\mathcal{O}_{ij}}\right]\right), \tag{6}$$

where b^T is the transposition of the attention vector b and $F_{\mathcal{M}_k}$ is the original feature vector of $\mathcal{M}_k$.

Step 2: Normalize the attention coefficient e_{ijk} using the softmax function:

$$\alpha_{ijk} = \text{softmax}_k\left(e_{ijk}\right). \tag{7}$$

Step 3: By processing the original features of $\mathcal{M}_k$ to obtain the attention coefficient e_{kk} of $\mathcal{M}_k$:

$$e_{kk} = \text{LeakyReLU}\left(b^T\left[W_m F_{\mathcal{M}_k} \parallel W_m F_{\mathcal{M}_k}\right]\right). \tag{8}$$

Step 4: Normalize the attention coefficient e_{kk} using the softmax function:

$$\alpha_{kk} = \text{softmax}_k\left(e_{kk}\right). \tag{9}$$

Step 5: Firstly, the weighted linear sum of the original feature vector $F_{\mathcal{O}_{ij}}$ of the neighbor operation node $\mathcal{O}_{ij}$ of $\mathcal{M}_k$ is calculated. Secondly, the weighted linear transformation of the original feature vector $F_{\mathcal{M}_k}$ of $\mathcal{M}_k$ is calculated. Thirdly, the sum of the results obtained in the previous two steps is sent to a nonlinear activation function for processing. Finally, the embedding feature vector $h_{\mathcal{M}_k}$ of $\mathcal{M}_k$ is calculated using the multi-head attention mechanism:

$$h_{\mathcal{M}_k} = \sum_{k=1}^{\mathcal{K}} \sigma\left(\sum_{\mathcal{O}_{ij} \in NO_t(\mathcal{M}_k)} \alpha_{ijk}^{k} W_o^{k} F_{\mathcal{O}_{ij}} + \alpha_{kk}^{k} W_m^{k} F_{\mathcal{M}_k}\right). \tag{10}$$

3.3.3. Mean pooling

$h_{\mathcal{O}_{ij}}$ and $h_{\mathcal{M}_k}$ are the feature vectors obtained from the embedding of $\mathcal{O}_{ij}$ and $\mathcal{M}_k$ on a single IGAT layer, respectively. To boost the ability to extract features from different nodes, the mean pooling of the feature vector sets $h_{\mathcal{O}_{ij}}^{(L)}$ of operation nodes and the feature vector sets $h_{\mathcal{M}_k}^{(L)}$

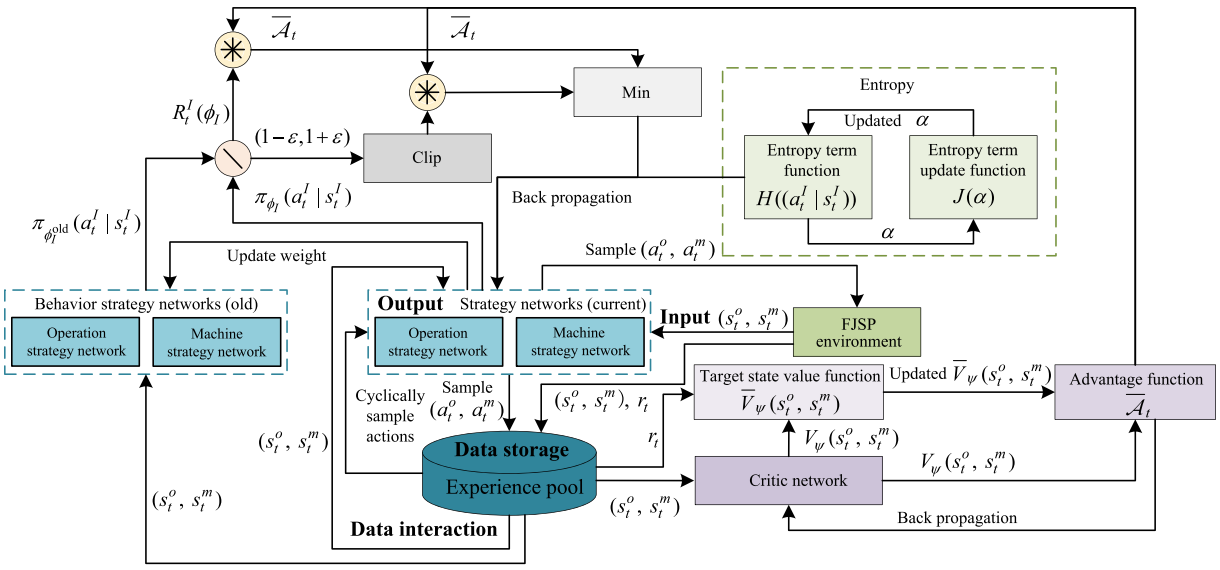


Fig. 4. Process of training the scheduling model using the proposed AEA-MAPPO algorithm.

of machine nodes obtained from embedding on L IGAT layers are performed to obtain the pooling vectors $\bar{h}_{\mathcal{O}_{ij}}$ and $\bar{h}_{\mathcal{M}_k}$, respectively:

$$\bar{h}_{\mathcal{O}_{ij}} = \frac{1}{|\mathcal{O}|}\sum_{\mathcal{O}_{ij}\in\mathcal{O}} h^{(L)}_{\mathcal{O}_{ij}} \tag{11}$$

and

$$\bar{h}_{\mathcal{M}_k} = \frac{1}{|\mathcal{M}|}\sum_{\mathcal{M}_k\in\mathcal{M}} h^{(L)}_{\mathcal{M}_k}. \tag{12}$$

3.4. Decoding of the operation strategy and machine strategy networks

The operation strategy network MLP_{ϕ_o} and the machine strategy network MLP_{ϕ_m} map the operation sequencing action a_t^o under the local state s_t^o and the machine assignment action a_t^m under the local state s_t^m into the two probability distributions. In addition, the operation agent and machine agent select the operation sequencing action a_t^o and machine assignment action a_t^m according to these two different probability distributions, respectively. The prediction process of the operation sequencing action a_t^o and machine assignment action a_t^m mainly includes the following two steps.

Step 1: Calculate the probability distributions $p^o\left(a_t^o,\ s_t^o\right)$ and $p^m\left(a_t^m,\ s_t^m\right)$ for the operation sequencing action a_t^o and machine assignment action a_t^m, respectively:

$$p^o\left(a_t^o,\ s_t^o\right) = \mathrm{MLP}_{\phi_o}\left(h_{\mathcal{O}_{ij}}\left\|\bar{h}_{\mathcal{O}_{ij}}\right\|\bar{h}_{\mathcal{M}_k}\right) \tag{13}$$

and

$$p^m\left(a_t^m,\ s_t^m\right) = \mathrm{MLP}_{\phi_m}\left(h_{\mathcal{M}_k}\left\|\bar{h}_{\mathcal{O}_{ij}}\right\|\bar{h}_{\mathcal{M}_k}\right). \tag{14}$$

Step 2: Use the softmax function to normalize the probability distributions $p^o\left(a_t^o,\ s_t^o\right)$ and $p^m\left(a_t^m,\ s_t^m\right)$ to obtain the operation sequencing strategy $\pi_{\phi_o}\left(a_t^o\middle|s_t^o\right)$ and machine assignment strategy $\pi_{\phi_m}\left(a_t^m\middle|s_t^m\right)$, respectively:

$$\pi_{\phi_o}\left(a_t^o\middle|s_t^o\right) = \frac{\exp\left(p^o\left(a_t^o,\ s_t^o\right)\right)}{\sum_{a_t^{o\prime}\in A_t^o}\exp\left(p^o\left(a_t^{o\prime},\ s_t^o\right)\right)} \tag{15}$$

and

$$\pi_{\phi_m}\left(a_t^m\middle|s_t^m\right) = \frac{\exp\left(p^m\left(a_t^m,\ s_t^m\right)\right)}{\sum_{a_t^{m\prime}\in A_t^m}\exp\left(p^m\left(a_t^{m\prime},\ s_t^m\right)\right)}. \tag{16}$$

3.5. Training the scheduling model using the AEA-MAPPO algorithm

To better train and optimize the scheduling model, and improve the efficiency of collaboratively solving FJSP by the operation and machine agents, an AEA-MAPPO algorithm is proposed. The PPO algorithm (Schulman et al., 2017) has gone through the following improvements. Firstly, the operation strategy and machine strategy networks are constructed to learn the operation sequencing and machine assignment actions, respectively. Secondly, the entropy objective function is introduced to balance the randomness of strategies and actions, improving the ability to find better strategies. Finally, to improve the stability of training and accelerate the convergence of the model, the entropy objective update function is introduced, which can dynamically adjust the exploration degree of actions and strategies to better adapt to the changes of the scheduling environment. Fig. 4 presents the training process of the scheduling model using the proposed AEA-MAPPO algorithm.

3.5.1. Entropy objective function and entropy objective update

In the proposed AEA-MAPPO algorithm, the calculation of the entropy objective function is as follows:

$$H\left(\left(a_t^I\middle|s_t^I\right)\right) = -\alpha\log\left(\pi_{\phi_I}\left(a_t^I\middle|s_t^I\right)\right), \tag{17}$$

where $H\left(\left(a_t^I\middle|s_t^I\right)\right)$ denotes the entropy of the strategy $\pi_{\phi_I}\left(a_t^I\middle|s_t^I\right)$, α denotes the entropy weight parameter, and $I \in \{o,\ m\}$.

The entropy objective update function $J(\alpha)$ is minimized to update α. $J(\alpha)$ is calculated by

$$J(\alpha) = \mathbb{E}_t\left[-\alpha\log\left(\pi_{\phi_I}\left(a_t^I\middle|s_t^I\right)\right) - \alpha\bar{\mathscr{H}}\right], \tag{18}$$

where $\bar{\mathscr{H}}$ is a threshold of the minimum strategy entropy.

3.5.2. Strategy networks

The operation strategy and machine strategy networks are adopted to learn the operation sequencing and machine assignment strategies, respectively. The operation sequencing and machine assignment strategies are trained toward the direction of maximizing the joint reward at timestep t. The strategy network loss $L^I_{\mathrm{CLIP}}(\phi_I)$ is minimized to train these two strategy networks, and $L^I_{\mathrm{CLIP}}(\phi_I)$ is calculated by

$$L^I_{\mathrm{CLIP}}(\phi_I) = \mathbb{E}_t\left[\min\left(R_t^I(\phi_I)\bar{\mathscr{A}}_t, \mathrm{clip}\left(R_t^I(\phi_I), 1-\varepsilon, 1+\varepsilon\right)\bar{\mathscr{A}}_t\right) + H\left(\left(a_t^I\middle|s_t^I\right)\right)\right]. \tag{19}$$

In Eq. (19), $R_t^I(\phi_I) = \frac{\pi_{\phi_I}\left(a_t^I \middle| s_t^I\right)}{\pi_{\phi_I^{\text{old}}}\left(a_t^I \middle| s_t^I\right)}$ is the ratio between the probability of actions obtained with the current strategy $\pi_{\phi_I}\left(a_t^I \middle| s_t^I\right)$ and that obtained with the old strategy $\pi_{\phi_I^{\text{old}}}\left(a_t^I \middle| s_t^I\right)$, which is used to measure the difference between the current and old strategies (i.e., the extent of strategy update). Clip(•) is an operation that controls the clip range, while ε denotes the hyperparameter that controls the clip range. $\bar{\mathscr{A}}_t$ represents the advantage of taking an action relative to the average expected reward, which is calculated by

$$\bar{\mathscr{A}}_t = -V_\psi\left(s_t^o, s_t^m\right) + r_t + \gamma r_{t+1} + \cdots + \gamma^{T-t+1} r_{T-1} + \gamma^{T-t} V_\psi\left(s_{t+1}^o, s_{t+1}^m\right), \quad (20)$$

where $V_\psi\left(s_t^o, s_t^m\right)$ is the estimated state value function, T is the number of timesteps, and γ denotes the discount factor.

3.5.3. Critic network

The critic network guides the strategy networks to perform strategy training by evaluating the quality of actions selected by the strategy networks in the training process. The critic network loss $L_t^{\text{MSE}}(\psi)$ is minimized to update the critic network. $L_t^{\text{MSE}}(\psi)$ is calculated by

$$L_t^{\text{MSE}}(\psi) = \mathbb{E}_t\left[\left(V_\psi\left(s_t^o, s_t^m\right) - \bar{V}_\psi\left(s_t^o, s_t^m\right)\right)^2\right], \quad (21)$$

where $\bar{V}_\psi\left(s_t^o, s_t^m\right)$ represents the objective state value function and is calculated by

$$\bar{V}_\psi\left(s_t^o, s_t^m\right) = r_t + \gamma V_\psi\left(s_{t+1}^o, s_{t+1}^m\right). \quad (22)$$

Algorithm 1 Training procedure of the scheduling model with AEA-MAPPO algorithm

Input: The maximum number of iterations $\mathscr{J}$, and the entropy weight parameter α, the discount factor γ, the clip coefficient ε, and the update epoch E.
Output: Two trained strategy networks.

```
Initialize the strategy network parameter φ_I, the behaviour strategy network parameter φ_I^old = φ_I, and the critic network parameter ψ;
for 𝒿 = 1 to 𝒥 do
  Get 𝒳 random FJSP instances;
  for 𝓍 = 1 to 𝒳 do
    while s_t is not terminal do
      Get a_t^I according to π_{φ_I}(a_t^I | s_t), where I ∈ {o, m};
      Get the joint reward r_t and next state s_{t+1};
      Calculate Ā_t by Eq. (20);
      Calculate R_t^I(φ_I) = π_{φ_I}(a_t^I|s_t^I) / π_{φ_I^old}(a_t^I|s_t^I);
      s_t ← s_{t+1};
    end while
    Calculate the entropy term function H((a_t^I|s_t^I)) by Eq. (17);
    Calculate the strategy network loss L_CLIP^I(φ_I) by Eq. (19);
    Calculate the critic network loss L_t^MSE(ψ) by Eq. (21);
  end for
  for e = 1 to E do
    Update φ_I and ψ by Adam optimizer to calculate the gradients of L_CLIP^I(φ_I) and L_t^MSE(ψ), respectively;
    Update α by Eq. (18);
  end for
  φ_I^old ← φ_I;
end for
```

Algorithm 1 presents the procedure of training the scheduling model using the proposed AEA-MAPPO algorithm. During each iteration of training, the $\mathscr{X}$ randomly generated FJSP instances are utilized for training the two strategy networks. Firstly, at timestep t, the two behavioral strategy networks respectively execute the operation sequencing and machine assignment actions to obtain the joint reward r_t and next state s_{t+1}. Secondly, at timestep t, the advantage function $\bar{\mathscr{A}}_t$ and the ratio $R_t^I(\phi_I)$ between the probability of actions obtained with the current strategy and that obtained with the old strategy are calculated to update the two behavioral strategy networks. These two processes are repeated until all operations are assigned to the available machines for processing. Thirdly, the entropy objective function $H\left(\left(a_t^I \middle| s_t^I\right)\right)$, the strategy network loss $L_{\text{CLIP}}^I(\phi_I)$, and the critic network loss $L_t^{\text{MSE}}(\psi)$ are calculated. Fourthly, at each update step E, the gradient of $L_{\text{CLIP}}^I(\phi_I)$ is calculated to update the two strategy networks, the gradient of $L_t^{\text{MSE}}(\psi)$ is calculated to update the critic network, and the entropy objective update function $J(\alpha)$ is minimized to update the entropy weight parameter α. Finally, the strategy network parameter ϕ_I is assigned to the behavior strategy network parameter ϕ_I^{old} to update the parameter weight. After the iterative training is completed, the two trained strategy networks are finally obtained.

Table 2
Settings of the core parameters for the proposed MAGRL method.

Parameter	Value
Number of IGAT layers (L)	2
Number of multi-attention heads ($\mathscr{K}$)	3
Optimizer	Adam
Clip coefficient (ε)	0.2
Entropy weight parameter (α)	0.01
Discount factor (γ)	1
Update epoch (E)	4
Learning rate	0.0001
Number of iterations ($\mathscr{J}$)	1000

4. Experiments

4.1. Experimental settings

In this experiment, the training set is composed of 400 randomly generated FJSP instances. It includes 100 instances of 10 × 5, 100 instances of 10 × 10, 100 instances of 20 × 10, and 100 instances of 30 × 10, which are used to train the scheduling models of 10 × 5, 10 × 10, 20 × 10, and 30 × 10, respectively. The test set includes 600 randomly generated FJSP instances (i.e., 100 instances of 10 × 5, 100 instances of 10 × 10, 100 instances of 20 × 10, 100 instances of 30 × 10, 100 instances of 50 × 20, and 100 instances of 100 × 20) along with some FJSP instances from Hurink (Hurink et al., 1994) and Brandimarte (Brandimarte, 1993) datasets. To better verify the effectiveness of the proposed MAGRL method, a series of comparative experiments are carried out on the Hurink and Brandimarte datasets. Hurink dataset contains 40 instances, which are divided into eight groups according to different scales. Each group contains five different FJSP instances of the same scale. The scale of each group of instances is 10 × 5, 10 × 10, 15 × 5, 15 × 10, 15 × 15, 20 × 5, 20 × 10, and 30 × 10 in order. Brandimarte dataset contains 10 instances. The scale of each instance is 10 × 6, 10 × 6, 15 × 8, 15 × 8, 15 × 4, 10 × 10, 20 × 5, 20 × 10, 20 × 10, and 20 × 15 in order. The settings of the core parameters for the proposed MAGRL method are shown in Table 2.

The software configurations of the experimental platform are PyTorch 1.6 and Python 3.9. The hardware configurations of the experimental platform are an RTX 2070 Super GPU with 2560 CUDA cores and 8 GB device memory, an eight-core Intel i7-9700K CPU, and 64 GB host memory.

4.2. Baseline methods

To validate the efficiency and robustness of the proposed MAGRL method to solve different-sized FJSP instances, the proposed MAGRL method is compared with the four scheduling rules, the improved particle swarm optimization (IPSO) algorithm (Ding and Gu, 2020),

Table 3
The comparison of the experimental results obtained adopting FIFO, SPT, MOPNR, MWKR, MAGRL, and OR-Tools on the randomly generated FJSP instances of different sizes.

Size		FIFO	SPT	MOPNR	MWKR	MAGRL	OR-Tools
	C_{max}	286.18	251.42	281.27	274.66	181.69	**163.75**
10 × 5	Time (s)	0.05	0.05	0.05	0.05	0.47	3.93
	Ave. gap	74.96%	53.71%	71.96%	67.92%	11.08%	**0.00%**
	C_{max}	356.73	272.12	359.11	355.23	235.43	**169.73**
10 × 10	Time (s)	0.10	0.10	0.10	0.10	1.29	17.73
	Ave. gap	110.17%	60.33%	111.58%	109.29%	38.71%	**0.00%**
	C_{max}	554.91	435.25	540.23	533.57	297.75	**244.83**
20 × 10	Time (s)	0.21	0.21	0.21	0.20	3.71	594.49
	Ave. gap	126.65%	77.78%	120.66%	117.93%	21.61%	**0.00%**
	C_{max}	798.70	564.07	792.04	786.19	396.60	**363.09**
30 × 10	Time (s)	0.33	0.32	0.33	0.33	6.90	600.00
	Ave. gap	119.97%	55.35%	118.14%	116.53%	9.23%	**0.00%**
	C_{max}	1328.68	787.97	1306.87	1298.64	**517.35**	–
50 × 20	Time (s)	1.63	1.63	1.67	1.63	10.48	–
	Ave. gap	156.82%	52.31%	152.61%	151.02%	**0.00%**	–
	C_{max}	2580.95	1320.66	2577.86	2572.16	**895.63**	–
100 × 20	Time (s)	4.57	4.48	4.55	4.52	42.96	–
	Ave. gap	188.17%	47.46%	187.83%	187.19%	**0.00%**	–

the self-learning genetic algorithm (SLGA) (Chen et al., 2020), the self-learning artificial bee colony (SLABC) method (Long et al., 2022), the multi-proximal policy optimization (MPPO) method (Lei et al., 2022), and the graph neural network and deep reinforcement learning (GNN-DRL) method (Song et al., 2022). The four scheduling rules include first in first out (FIFO), shortest processing time (SPT), most operations remaining (MOPNR), and most work remaining (MWKR). IPSO is the state-of-the-art MHA to solve FJSP. SLGA and SLABC are the two state-of-the-art methods combining MHA with RL for solving FJSP. MPPO and GNN-DRL are the two state-of-the-art GRL methods for solving FJSP. It should be noted that the parameters of all methods are set to their respective optimal parameter values to ensure the fairness and effectiveness of the comparison experiments.

In this experiment, the percentage deviation namely gap and the advantage rate are used as the performance metrics to evaluate the different methods to solve FJSP instances. The gap is calculated by

$$\text{Gap} = \frac{C_{\max} - \text{UB}}{\text{UB}} \times 100\%, \tag{23}$$

where UB (Behnke and Geiger, 2012) represents the optimal known solution and $C_{\max}$ is the makespan of a FJSP instance solved by different methods, including the scheduling rules, MPPO, GNN-DRL, IPSO, SLGA, SLABC, and MAGRL. It should be noted that since each randomly generated FJSP instance does not have the optimal known solution, the minimum value of $C_{\max}$ obtained using different methods to solve the randomly generated FJSP instance will be taken as UB in this experiment.

Compared with the optimal solution UB, the advantage rate of the scheduling rules, MPPO, GNN-DRL, IPSO, SLGA, SLABC and MAGRL for solving a FJSP instance is calculated by

$$O_{\text{rate}} = \frac{\text{UB}}{C_{\max}} \times 100\%. \tag{24}$$

4.3. Parameter sensitivity analysis

In the training process of the scheduling model, sensitivity analyses are conducted on the four most key parameters, namely, the number of IGAT layers, the number of multi-attention heads, the entropy weight parameter, and the learning rate, as shown in Fig. 5. The results show that when the number of IGAT layers, the number of multi-attention heads, the entropy weight parameter, and the learning rate are set to 2, 3, 0.01, and 0.0001 respectively, the scheduling model can converge faster and achieve the optimal makespan.

Moreover, to better control the extent of updating the scheduling strategy through $R_t^I(\phi_I)$ in Eq. (19), the clip coefficient ε is set to 0.2, thereby maintaining the stability of the model training. To approach the optimal solution more quickly in the model training, the update epoch E is set to 4, thereby accelerating the training process. To better calculate the joint reward function, the discount factor γ is set to 1.

4.4. Comparison results on randomly generated FJSP instances

In this experiment, the test set includes the following randomly generated different-sized FJSP: 100 instances of 10 × 5, 100 instances of 10 × 10, 100 instances of 20 × 10, 100 instances of 30 × 10, 100 instances of 50 × 20, and 100 instances of 100 × 20. The proposed MAGRL method is compared with FIFO, SPT, MOPNR, MWKR, and OR-Tools.[1] Note that OR-Tools is a solver developed by Google to solve combinatorial optimization problems, which can obtain the optimal makespan to solve FJSP instances. The upper limit of the solution time of OR-Tools is set to 600 s, which means to stop the calculation if it exceeds 600 s. The scheduling models of 10 × 5, 10 × 10, 20 × 10, and 30 × 10 trained by MAGRL are used to solve the corresponding same-sized FJSP instances, respectively. In addition, the scheduling model of 30 × 10 is used to solve the large-sized FJSP instances of 50 × 20 and 100 × 20. For different methods, the average makespan and the average solution time obtained from solving the 100 same-sized FJSP instances are taken as the experimental results of solving the FJSP instances of this size. Table 3 shows the comparison of the experimental results obtained adopting FIFO, SPT, MOPNR, MWKR, MAGRL, and OR-Tools on the randomly generated FJSP instances of different sizes.

As shown in Table 3, when solving FJSP instances of different sizes, the four scheduling rules have certain advantages compared with MAGRL in solution time, but their solution quality (i.e., $C_{\max}$) is not ideal. Obviously, the solution quality of MAGRL is far better than that of these four scheduling rules on different-sized FJSP instances. This is because the scheduling rules are usually designed according to the specific FJSP scenarios and solution requirements. Although the scheduling rules have shorter solution time when solving FJSP instances of different sizes, they may not achieve good solution quality, that is, there may be a large difference in the solution quality of the scheduling rules for FJSP instances. When solving the small and medium-sized (e.g., 10 × 5, 10 × 10, 20 × 10, and 30 × 10) FJSP instances, OR-Tools achieves better solution quality than MAGRL. However, with the increase of the size of FJSP instances, the gap value between MAGRL and OR-Tools is gradually decreased, and the solution time of OR-Tools is much longer than that of MAGRL. When solving the large-sized (such as 50 × 20 and 100 × 20) FJSP instances, OR-Tools has failed to solve these FJSP instances in the reasonable time, but the solution quality achieved with

[1] [Online]. Available: https://developers.google.com/optimization.

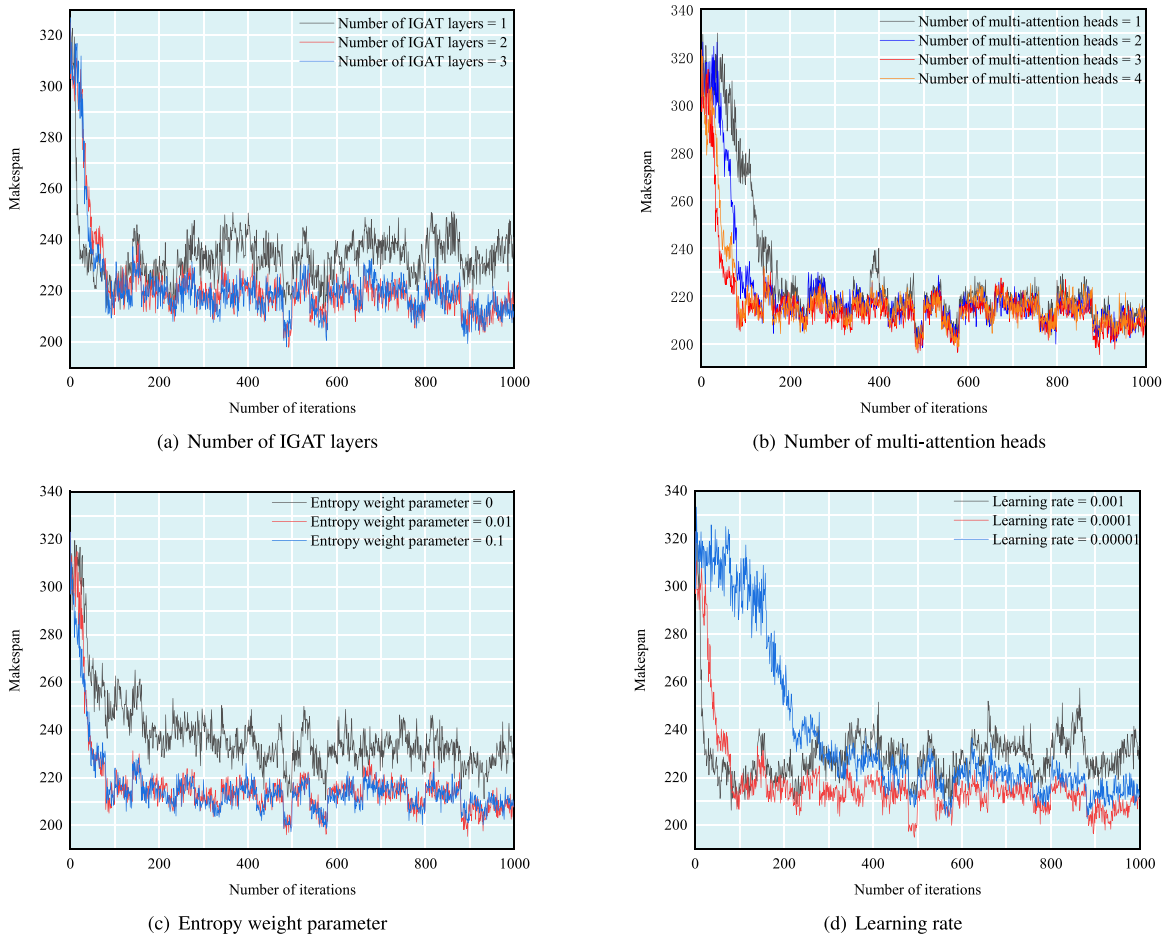


Fig. 5. Sensitivity analyses of the four most key parameters.

MAGRL is getting better and better. It is not difficult to see that with the increase of the size of FJSP instances, OR-Tools as a solver based on the exact algorithm, its solution time has become a significant limiting factor, whereas MAGRL has considerable obvious advantages compared with OR-Tools. The above results show that MAGRL can efficiently solve different-sized FJSP instances. Moreover, its advantages are more significant especially when solving large-sized FJSP instances.

4.5. Comparison results on benchmark instances

4.5.1. Comparison results on Hurink's instances

In this experiment, the 40 FJSP instances in the sub-dataset Rdata of Hurink dataset are selected as the test set. These 40 FJSP instances are divided into eight groups according to different sizes, and each group contains five FJSP instances of the same size. The scheduling model of 10×10 trained by MAGRL is compared with FIFO, SPT, MOPNR, MWKR, MPPO (Lei et al., 2022), and GNN-DRL (Song et al., 2022), where MPPO and GNN-DRL are the two state-of-the-art GRL methods for solving FJSP. For different methods, the average makespan and the average solution time obtained from solving the five same-sized FJSP instances are taken as the experimental results of solving the FJSP instances of this size. As shown in Tables 4 and 5, the comparisons of the solution quality and solution time achieved with FIFO, SPT, MOPNR, MWKR, MPPO, GNN-DRL, and MAGRL on Hurink's instances, respectively. Meanwhile, as shown in Fig. 6, the comparisons of the overall performance achieved with FIFO, SPT, MOPNR, MWKR, MPPO, GNN-DRL, and MAGRL on Hurink's instances.

As shown in Tables 4 and 5, although the solution speed of MAGRL is slower than those of the four scheduling rules of FIFO, SPT, MOPNR, and MWKR, the maximum difference does not exceed 4.65 s. Moreover, MAGRL achieves significantly better solution quality than FIFO, SPT, MOPNR, and MWKR on the FJSP instances of different sizes. Compared with MPPO, on the different-sized FJSP instances, the maximum difference between the solution time of MAGRL and that of MPPO does not exceed 3.49 s, but MAGRL achieves significant better solution quality. Compared with GNN-DRL, MAGRL gets better solution quality and consumes less solution time on most different-sized FJSP instances. As depicted in Fig. 6, the average gap value of MAGRL is significantly better than those of FIFO, SPT, MOPNR, MWKR, and MPPO and better than that of GNN-DRL. Compared with the optimal solution UB, the advantage rates of the scheduling rules, MPPO, GNN-DRL, and MAGRL are shown in Fig. 7. As depicted in Fig. 7, MAGRL consistently achieves the highest advantage rate on each Hurink's instances and its advantage rate exceeds 90%. In contrast, the lowest advantage rate of the other methods is only 75%. Taken together, the results indicate that MAGRL is superior to these comparison methods in solving FJSP instances of different sizes, which confirms the effectiveness of MAGRL.

4.5.2. Comparison results on Brandimarte's instances

In this experiment, the top 10 FJSP instances in Brandimarte dataset are selected as the test set, and the scheduling model of 20×10 trained by MAGRL is used to solve these 10 FJSP instances. The proposed MAGRL method is compared with MOPNR, MPPO (Lei et al., 2022), GNN-DRL (Song et al., 2022), IPSO (Ding and Gu, 2020), SLGA (Chen et al., 2020), and SLABC (Long et al., 2022). MOPNR is the best-performing scheduling rule among FIFO, SPT, MOPNR, and MWKR on

Table 4
The comparisons of the solution quality achieved with FIFO, SPT, MOPNR, MWKR, MPPO, GNN-DRL, and MAGRL on Hurink's instances.

Size	FIFO		SPT		MOPNR		MWKR		MPPO (Lei et al., 2022)		GNN-DRL (Song et al., 2022)		MAGRL		UB
	C_{max}	Gap	C_{max}	Gap	C_{max}	Gap	C_{max}	Gap	C_{max}	Gap	C_{max}	Gap	C_{max}	Gap	
10 × 5	589.48	16.13%	672.96	32.58%	574.48	13.18%	593.64	16.95%	557.80	9.89%	523.40	3.11%	**521.80**	**2.80%**	507.60
10 × 10	852.56	22.32%	877.80	25.94%	858.40	23.16%	823.40	18.13%	835.80	19.91%	**757.40**	**8.67%**	758.00	8.75%	697.00
15 × 5	855.96	7.78%	994.20	25.18%	847.84	6.75%	843.36	6.19%	856.80	7.88%	808.80	1.84%	**804.40**	**1.28%**	794.20
15 × 10	998.80	23.74%	1104.80	36.87%	983.40	21.83%	980.00	21.41%	992.60	22.97%	896.40	11.05%	**886.00**	**9.76%**	807.20
15 × 15	1246.44	23.09%	1299.92	28.37%	1245.80	23.03%	1237.36	22.20%	1216.00	20.09%	1127.20	11.32%	**1120.00**	**10.61%**	1012.60
20 × 5	1125.64	8.13%	1203.92	15.65%	1130.72	8.62%	1093.68	5.06%	1093.00	5.00%	1051.20	0.98%	**1046.80**	**0.56%**	1041.00
20 × 10	1254.36	18.16%	1441.56	35.79%	1219.16	14.84%	1196.84	12.74%	1192.40	12.32%	1135.00	6.91%	**1129.40**	**6.39%**	1061.60
30 × 10	1684.04	8.44%	1881.00	21.12%	1669.00	7.47%	1650.24	6.26%	1669.60	7.51%	1599.80	3.01%	**1578.20**	**1.62%**	1553.00

Table 5
The comparisons of the solution time achieved with FIFO, SPT, MOPNR, MWKR, MPPO, GNN-DRL, and MAGRL on Hurink's instances.

Size	FIFO Time (s)	SPT Time (s)	MOPNR Time (s)	MWKR Time (s)	MPPO (Lei et al., 2022) Time (s)	GNN-DRL (Song et al., 2022) Time (s)	MAGRL Time (s)
10 × 5	0.05	0.05	0.05	0.05	0.22	0.76	0.51
10 × 10	0.10	0.10	0.10	0.10	0.42	1.19	1.33
15 × 5	0.07	0.07	0.08	0.08	0.33	1.68	0.83
15 × 10	0.16	0.16	0.16	0.16	0.66	1.85	2.38
15 × 15	0.25	0.24	0.25	0.25	1.04	3.24	4.19
20 × 5	0.10	0.09	0.10	0.10	0.44	5.03	1.19
20 × 10	0.21	0.21	0.21	0.21	0.86	10.14	3.71
30 × 10	0.33	0.33	0.34	0.34	1.49	6.58	4.98

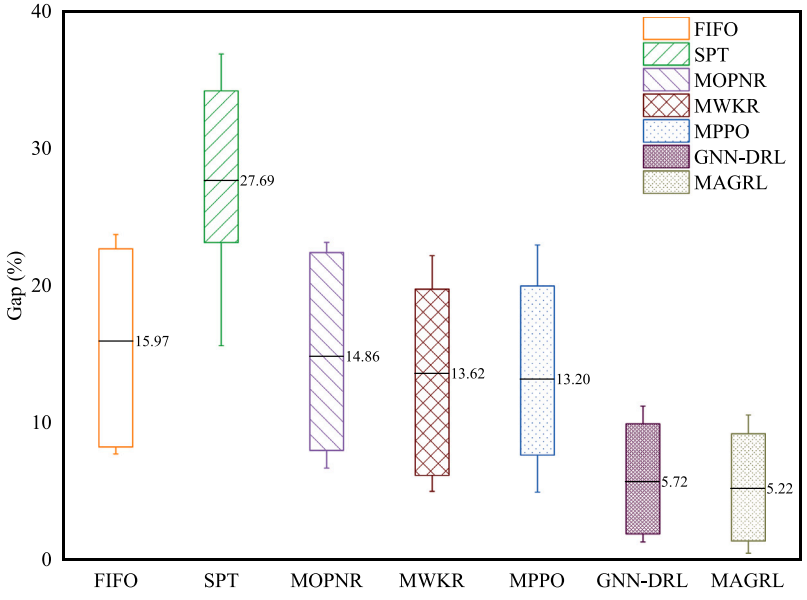


Fig. 6. The comparisons of the overall performance achieved with FIFO, SPT, MOPNR, MWKR, MPPO, GNN-DRL, and MAGRL on Hurink's instances.

the Brandimarte dataset. The comparisons of the solution quality and solution time achieved with MOPNR, MPPO, GNN-DRL, IPSO, SLGA, SLABC, and MAGRL on Brandimarte's instances are shown in Tables 6 and 7, respectively. Fig. 8 presents the comparisons of the overall performance achieved with MOPNR, MPPO, GNN-DRL, IPSO, SLGA, SLABC, and MAGRL on Brandimarte's instances.

As shown in Tables 6 and 7, the solution time of MOPNR is slightly better than that of MAGRL, but the solution quality of MAGRL is much better than that of MOPNR on all FJSP instances. Compared with MPPO and GNN-DRL, the solution time of MAGRL, MPPO, and GNN-DRL are relatively similar on different FJSP instances, but the solution quality of MAGRL outperforms that of MPPO and GNN-DRL on most FJSP instances. Compared with IPSO, SLGA, and SLABC, MAGRL achieves better solution quality on most FJSP instances, and there is little difference between the solution quality achieved with MAGRL and that achieved with SLGA on all FJSP instances. However, as the size of FJSP increases, these solution methods based on the MHAs including IPSO, SLGA, and SLABC need a long iteration time to get better solution quality, which leads to a significant increase in the solution time of IPSO, SLGA, and SLABC. As shown in Tables 6 and 7, MAGRL has a very short solution time on all FJSP instances, but achieves better solution quality, which shows the advantages of MAGRL over IPSO, SLGA, and SLABC. As depicted in Fig. 8, the average gap value of MAGRL is significantly better than those of MOPNR, MPPO, GNN-DRL, and SLABC, and the maximum differences between the gap values of MOPNR, MPPO, GNN-DRL, and SLABC and that of MAGRL are 53.45%, 15.39%, 32.76%, and 29.94% on the same FJSP instance, respectively. In addition, the average gap value of MAGRL is similar to those of IPSO and SLGA. Compared with the optimal solution UB, the advantage rates of MOPNR, MPPO, GNN-DRL, IPSO, SLGA, SLABC, and MAGRL to solve Brandimarte's instances are shown in Fig. 9. As seen in Fig. 9, on all Brandimarte's instances, the advantage rates of MAGRL are significantly higher than those of the scheduling rule MOPNR. On most Brandimarte's instances, the advantage rates of MAGRL are superior to

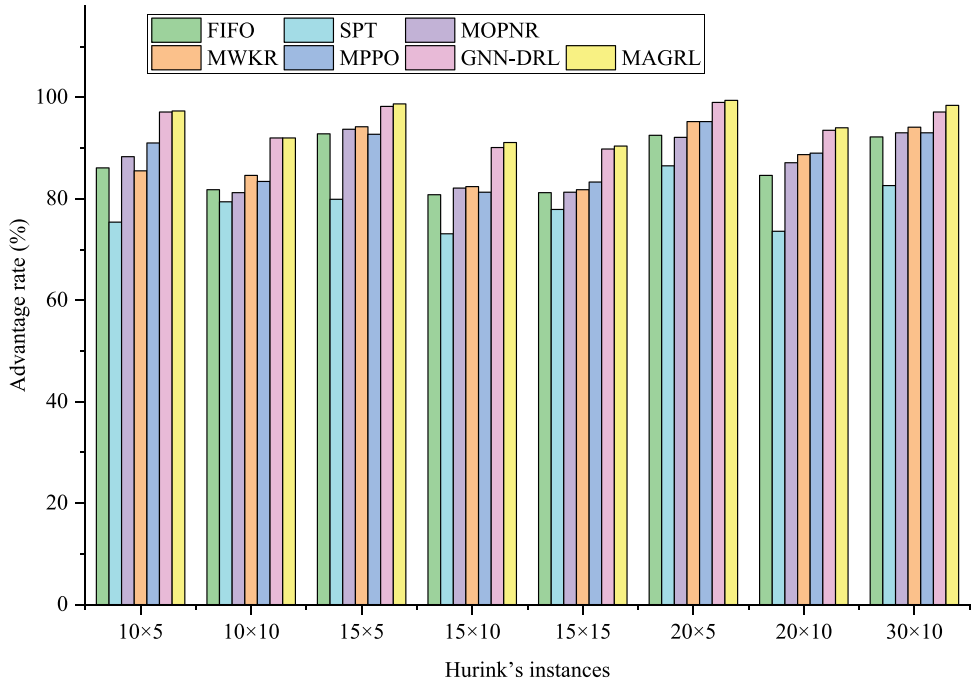


Fig. 7. The advantage rates of different methods to solve Hurink's instances.

Table 6
The comparisons of the solution quality achieved with MOPNR, MPPO, GNN-DRL, IPSO, SLGA, SLABC, and MAGRL on Brandimarte's instances.

Size	MOPNR		MPPO (Lei et al., 2022)		GNN-DRL (Song et al., 2022)		IPSO (Ding and Gu, 2020)		SLGA (Chen et al., 2020)		SLABC (Long et al., 2022)		MAGRL		UB
	C_{max}	Gap	C_{max}	Gap	C_{max}	Gap	C_{max}	Gap	C_{max}	Gap	C_{max}	Gap	C_{max}	Gap	
10 × 6	45	15.38%	44	12.85%	42	7.69%	**40**	**2.56%**	**40**	**2.56%**	42	7.69%	42	7.69%	39
10 × 6	37	42.31%	32	23.08%	35	34.62%	29	11.54%	**27**	**3.85%**	29	11.54%	28	7.69%	26
15 × 8	220	7.84%	**204**	**0.00%**	**204**	**0.00%**	**204**	**0.00%**	**204**	**0.00%**	**204**	**0.00%**	**204**	**0.00%**	204
15 × 8	75	25.00%	70	16.67%	72	20.00%	66	10.00%	**60**	**0.00%**	69	15.00%	67	11.67%	60
15 × 4	187	8.72%	182	5.81%	181	5.23%	175	1.74%	**172**	**0.00%**	175	1.74%	175	1.74%	172
10 × 10	102	75.86%	78	34.48%	90	55.17%	77	32.76%	**69**	**18.97%**	80	37.93%	71	22.41%	58
20 × 5	214	53.96%	157	12.95%	194	39.57%	145	4.32%	**144**	**3.60%**	155	11.51%	146	5.04%	139
20 × 10	531	1.53%	531	1.53%	**523**	**0.00%**	**523**	**0.00%**	**523**	**0.00%**	**523**	**0.00%**	**523**	**0.00%**	523
20 × 10	311	1.30%	331	7.82%	317	3.26%	320	4.23%	320	4.23%	368	19.87%	**312**	**1.63%**	307
20 × 15	269	36.55%	247	25.38%	252	27.92%	239	21.32%	254	28.93%	283	43.65%	**224**	**13.71%**	197

Table 7
The comparisons of the solution time achieved with MOPNR, MPPO, GNN-DRL, IPSO, SLGA, SLABC, and MAGRL on Brandimarte's instances.

Instance	Size	MOPNR	MPPO (Lei et al., 2022)	GNN-DRL (Song et al., 2022)	IPSO (Ding and Gu, 2020)	SLGA (Chen et al., 2020)	SLABC (Long et al., 2022)	MAGRL
		Time (s)	Time (s)	Time (s)	Time (s)	Time (s)	Time (s)	Time (s)
MK1	10 × 6	0.05	0.23	0.86	34.70	27.63	29.65	0.68
MK2	10 × 6	0.06	0.24	0.86	50.00	29.11	73.97	0.58
MK3	15 × 8	0.15	0.65	3.13	562.50	112.60	117.20	2.21
MK4	15 × 8	0.09	0.42	1.55	135.00	63.21	55.67	1.10
MK5	15 × 4	0.11	0.45	1.89	131.50	60.35	165.40	1.22
MK6	10 × 10	0.16	0.43	3.17	797.80	72.80	170.10	2.33
MK7	20 × 5	0.10	0.43	1.66	160.60	57.77	1189.70	1.20
MK8	20 × 10	0.26	1.09	6.32	944.80	521.70	366.30	4.39
MK9	20 × 10	0.26	1.25	7.00	1762.00	552.50	410.60	4.81
MK10	20 × 15	0.27	1.16	7.20	2675.00	1335.00	332.70	5.96

those of MPPO, GNN-DRL, and SLABC. Taken together, MAGRL is far superior to MOPNR, MPPO, GNN-DRL, and SLABC, and slightly better than IPSO and SLGA.

According to the experimental results achieved on the randomly generated FJSP instances and two common benchmarks, the performance of MAGRL is considerably stable on different test sets, and the overall performance of MAGRL is better than that of various advanced comparison methods to solve FJSP, indicating that MAGRL has good efficiency and robustness.

4.6. Computational complexity analysis

To explore the computational complexity of the proposed MAGRL method, an experimental analysis is carried out on the changes in the scale of FJSP, focusing on the impacts of the number of machines and the number of jobs on the solution efficiency. Specifically, this experiment is conducted on Hurink's instances and divided into the following two groups: (1) the number of machines is fixed to 10, while the number of jobs is gradually increased from 10 to 30; (2)

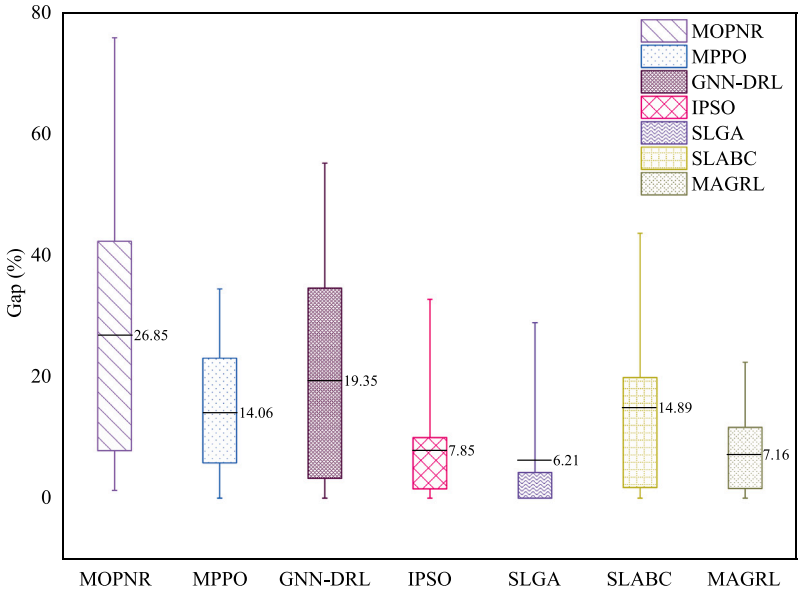


Fig. 8. The comparisons of the overall performance achieved with MOPNR, MPPO, GNN-DRL, IPSO, SLGA, SLABC, and MAGRL on Brandimarte's instances.

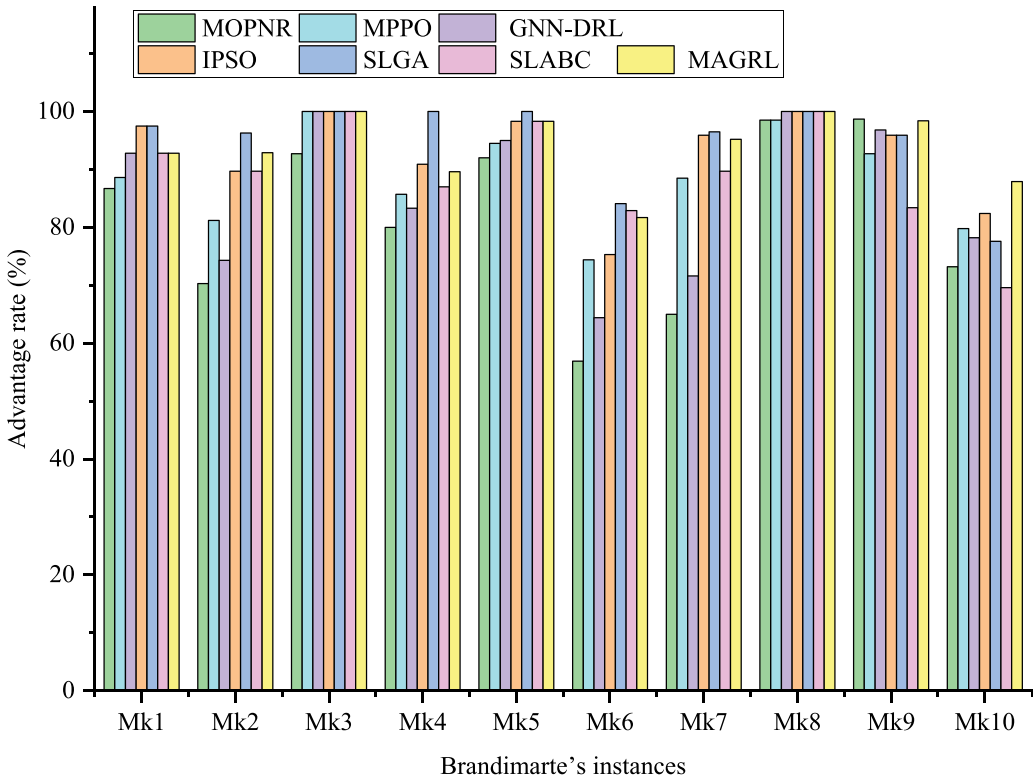


Fig. 9. The advantage rates of different methods to solve Brandimarte's instances.

the number of jobs is fixed to 15, while the number of machines is gradually increased from 5 to 15. Figs. 10 and 11 clearly show the trend of computational complexity with the number of jobs and machines. As seen in Fig. 10, when the number of machines is kept at 10, the computation time of MAGRL gradually increases with the increase of the number of jobs, but the growth trend is relatively gentle. It indicates that MAGRL increases the computation time when processing more jobs, but does not show a sharp increase, showing good scalability. As seen in Fig. 11, when the number of jobs is kept at 15, the computation time of MAGRL increases significantly with the increase of the number of machines, but it is basically a linear increase.

To sum up, the experimental results show that MAGRL has good scalability when dealing with different-scale FJSP instances. Especially, when the number of jobs changes, it shows a relatively slow increase in computation time. Whereas when the number of machines increases, the linear increase in computation time is also expected. This is because the proposed MAGRL method requires the observation of the states of the FJSP environment and the use of the operation and machine agents

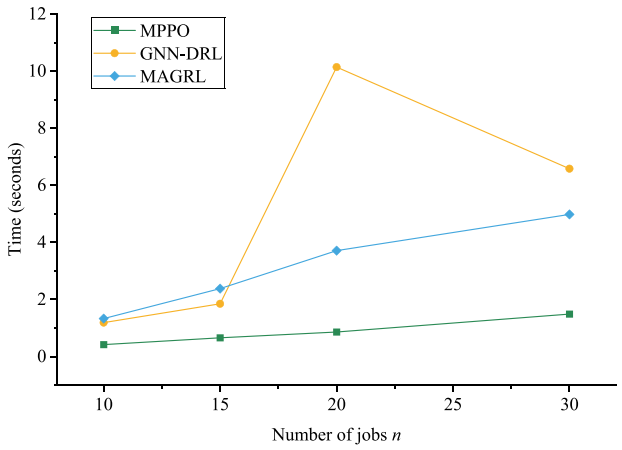


Fig. 10. Keep the number of machines $m = 10$.

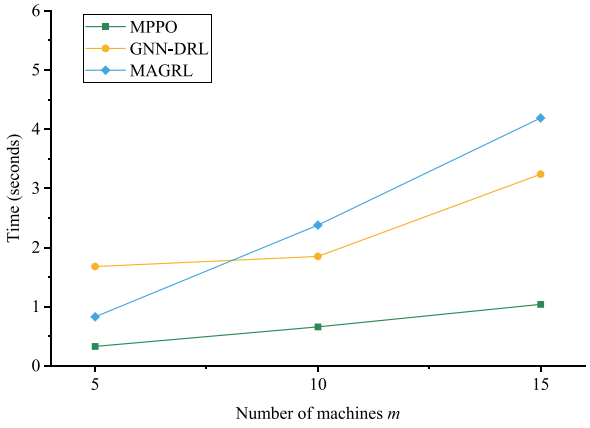


Fig. 11. Keep the number of jobs $n = 15$.

to make scheduling decisions, and its computational complexity closes to $O(nm)$.

5. Conclusions

A novel MAGRL method is studied to effectively solve FJSP. The global scheduling states of FJSP are represented by a heterogeneous graph. The operation agent is responsible for controlling the selection of the operations and the machine agent is responsible for controlling the assignment of the machines, and FJSP is modeled into two MDPs. The encoder-double-decoder architecture is constructed for handling the structural information of this heterogeneous graph. In this encoder-double-decoder architecture, IGAT is adopted as an encoder to effectively extract the features of operation and machine nodes, and the operation strategy and machine strategy networks are designed as double decoders to effectively predict the operation sequencing and machine assignment strategies. The AEA-MAPPO algorithm is designed to train these two strategy networks, which can efficiently learn the optimal operation sequencing and machine assignment strategies. A large number of experiments have been conducted on the randomly generated FJSP instances and two common benchmarks. Compared with various state-of-the-art methods to solve FJSP, these results show that MAGRL can efficiently and stably solve FJSP instances of different sizes.

Although the proposed MAGRL method performs well in solving static FJSP, it is still unable to make an optimal decision in real-time scheduling when dealing with diversified dynamic events in actual productions, such as the insertion of new jobs, machine failures, and random arrival of jobs. Furthermore, due to MAGRL is based on a multi-agent framework, the solution quality may be limited by the model complexity and computing resources as the problem size increases. Therefore, the focus of future research will be to further optimize MAGRL to improve its adaptability and solution efficiency in dealing with more complex DFJSPs, especially its real-time decision-making ability in dynamic environments.

CRediT authorship contribution statement

Lanjun Wan: Writing – review & editing, Writing – original draft, Software, Methodology, Investigation, Conceptualization. **Long Fu:** Writing – original draft, Visualization, Validation, Software, Methodology, Data curation. **Changyun Li:** Supervision, Funding acquisition. **Keqin Li:** Writing – review & editing, Methodology, Formal analysis.

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.

Acknowledgments

This work was supported by the Hunan Provincial Natural Science Foundation of China [grant number 2023JJ30217] and the National Natural Science Foundation for Young Scientists of China [grant number 61702177].

Data availability

Data will be made available on request.

References

Behnke, D., Geiger, M.J., 2012. Test Instances for the Flexible Job Shop Scheduling Problem with Work Centers. Tech. Rep. RR-12-01-01, Helmut Schmidt Univ., Hamburg, Germany.

Brandimarte, P., 1993. Routing and scheduling in a flexible job shop by tabu search. Ann. Oper. Res. 41 (3), 157–183.

Chen, R., Yang, B., Li, S., Wang, S., 2020. A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem. Comput. Ind. Eng. 149, 106778.

Dauzère-Pérès, S., Ding, J., Shen, L., Tamssaouet, K., 2024. The flexible job shop scheduling problem: A review. European J. Oper. Res. 314 (2), 409–432.

Ding, H., Gu, X., 2020. Improved particle swarm optimization algorithm based novel encoding and decoding schemes for flexible job shop scheduling problem. Comput. Oper. Res. 121, 104951.

Fan, C., Wang, W., Tian, J., 2024. Flexible job shop scheduling with stochastic machine breakdowns by an improved tuna swarm optimization algorithm. J. Manuf. Syst. 74, 180–197.

Fekih, A., Hadda, H., Kacem, I., Hadj-Alouane, A.B., 2023. Mixed-integer programming and constraint programming models for the flexible job shop scheduling problem. In: Int. Conf. Artif. Intell. Ind. Appl.. Springer, pp. 110–122.

Gheisari, M., Ebrahimzadeh, F., Rahimi, M., Moazzamigodarzi, M., Liu, Y., Dutta Pramanik, P.K., Heravi, M.A., Mehbodniya, A., Ghaderzadeh, M., Feylizadeh, M.R., Kosari, S., 2023. Deep learning: Applications, architectures, models, tools, and frameworks: A comprehensive survey. CAAI Trans. Intell. Technol. 8 (3), 581–606.

Hurink, J., Jurisch, B., Thole, M., 1994. Tabu search for the job-shop scheduling problem with multi-purpose machines. OR Spectrum 15, 205–215.

Jiang, B., Ma, Y., Chen, L., Huang, B., Huang, Y., Guan, L., 2023. A review on intelligent scheduling and optimization for flexible job shop. Int. J. Control Autom. 21 (10), 3127–3150.

Jun, S., Lee, S., Chun, H., 2019. Learning dispatching rules using random forest in flexible job shop scheduling problems. Int. J. Prod. Res. 57 (10), 3290–3310.

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 Syst. Appl. 205, 117796.

Li, X., Guo, X., Tang, H., Wu, R., Wang, L., Pang, S., Liu, Z., Xu, W., Li, X., 2022. Survey of integrated flexible job shop scheduling problems. Comput. Ind. Eng. 174, 108786.

Li, Y., Liao, C., Wang, L., Xiao, Y., Cao, Y., Guo, S., 2023a. A reinforcement learning-artificial bee colony algorithm for flexible job-shop scheduling problem with lot streaming. Appl. Soft Comput. 146, 110658.

Li, C., Zheng, P., Yin, Y., Wang, B., Wang, L., 2023b. Deep reinforcement learning in smart manufacturing: A review and prospects. CIRP J. Manuf. Sci. Technol. 40, 75–101.

Liu, J., Sun, B., Li, G., Chen, Y., 2024. Multi-objective adaptive large neighbourhood search algorithm for dynamic flexible job shop schedule problem with transportation resource. Eng. Appl. Artif. Intell. 132, 107917.

Long, X., Zhang, J., Qi, X., Xu, W., Jin, T., Zhou, K., 2022. A self-learning artificial bee colony algorithm based on reinforcement learning for a flexible job-shop scheduling problem. Concurr. Comput. Pract. Exp. 34 (4), e6658.

Luo, S., 2020. Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning. Appl. Soft Comput. 91, 106208.

Müller, D., Müller, M.G., Kress, D., Pesch, E., 2022. An algorithm selection approach for the flexible job shop scheduling problem: Choosing constraint programming solvers through machine learning. European J. Oper. Res. 302 (3), 874–891.

Munikoti, S., Agarwal, D., Das, L., Halappanavar, M., Natarajan, B., 2024. Challenges and opportunities in deep reinforcement learning with graph neural networks: A comprehensive review of algorithms and applications. IEEE Trans. Neural Netw. Learn. Syst. 35 (11), 15051–15071.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O., 2017. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

Song, W., Chen, X., Li, Q., Cao, Z., 2022. Flexible job-shop scheduling via graph neural network and deep reinforcement learning. IEEE Trans. Ind. Inform. 19 (2), 1600–1610.

Song, X., Sun, P., Song, S., Stojanovic, V., 2023. Quantized neural adaptive finite-time preassigned performance control for interconnected nonlinear systems. Neural Comput. Appl. 35 (21), 15429–15446.

Su, C., Zhang, C., Xia, D., Han, B., Wang, C., Chen, G., Xie, L., 2023. Evolution strategies-based optimized graph reinforcement learning for solving dynamic job shop scheduling problem. Appl. Soft Comput. 145, 110596.

Tao, Y., Tao, H., Zhuang, Z., Stojanovic, V., Paszke, W., 2024. Quantized iterative learning control of communication-constrained systems with encoding and decoding mechanism. Trans. Inst. Meas. Control 46 (10), 1943–1954.

Teymourifar, A., Ozturk, G., Ozturk, Z.K., Bahadir, O., 2020. Extracting new dispatching rules for multi-objective dynamic flexible job shop scheduling with limited buffer spaces. Cogn. Comput. 12, 195–205.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y., 2017. Graph attention networks. arXiv preprint arXiv:1710.10903.

Wan, H., Luan, X., Stojanovic, V., Liu, F., 2023. Self-triggered finite-time control for discrete-time Markov jump systems. Inform. Sci. 634, 101–121.

Wang, L., Pan, Z., Wang, J., 2021. A review of reinforcement learning based intelligent optimization for manufacturing scheduling. Complex Syst. Model. Simul. 1 (4), 257–270.

Yuan, E., Wang, L., Cheng, S., Song, S., Fan, W., Li, Y., 2024. Solving flexible job shop scheduling problems via deep reinforcement learning. Expert Syst. Appl. 245, 123019.

Zhang, Z.-Q., Wu, F.-C., Qian, B., Hu, R., Wang, L., Jin, H.-P., 2023. A Q-learning-based hyper-heuristic evolutionary algorithm for the distributed flexible job-shop scheduling problem with crane transportation. Expert Syst. Appl. 234, 121050.

Zhao, L., Fan, J., Zhang, C., Shen, W., Zhuang, J., 2024. A DRL-based reactive scheduling policy for flexible job shops with random job arrivals. IEEE Trans. Autom. Sci. Eng. 21 (3), 2912–2923.