Journal of Manufacturing Systems 71 (2023) 257–273

Contents lists available at ScienceDirect

Journal of Manufacturing Systems

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

Technical paper

Deep reinforcement learning for dynamic flexible job shop scheduling problem considering variable processing times

Lu Zhang [a], Yi Feng [a], Qinge Xiao [b], Yunlang Xu [c], Di Li [d], Dongsheng Yang [e], Zhile Yang [b,*]

[a] School of Control Science and Engineering, Dalian University of Technology, Dalian, China
[b] Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China
[c] State Key Laboratory of ASIC and System, School of Microelectronics, Fudan University, Shanghai, China
[d] School of Mechanical and Automotive Engineering, South China University of Technology, Guangzhou, China
[e] Intelligent Electrical Science and Technology Research Institute, Northeastern University, Shenyang, China

ARTICLE INFO

Keywords:
Dynamic scheduling flexible job shop problem
Deep reinforcement learning
Variable processing times
Proximal policy optimization
Makespan

ABSTRACT

In recent years, the uncertainties and complexity in the production process, due to the boosted customized requirements, has dramatically increased the difficulties of Dynamic Flexible Job Shop Scheduling (DFJSP). This paper investigates a new DFJSP model taking into account the minimum completion time under the condition of machine processing time uncertainty, e.t. VPT-FJSP problem. In the formulated VPT-FJSP process, each workpiece needs to be processed by required machine at a certain time slot where Markov decision process (MDP) and reinforcement learning methods are adopted to solve VPT-FJSP. The agent designed in this paper employs the Proximal Policy Optimization(PPO) algorithm in deep reinforcement learning, which includes the Actor-Critic network. The input of the network is to extract the processing information matrix and to embed some advanced states in the workshop by graph neural network, which enables the agent to learn the complete state of the environment. Finally, we train and test the proposed framework on the canonical FJSP benchmark, and the experimental results show that our framework can make agent better than genetic algorithm and ant colony optimization in most cases, 94.29% of static scheduling. It is also shown superiority compared to the scheduling rules in dynamic environment and has demonstrated strong robustness in solving VPT-FJSP. Furthermore, this study conducted tests to assess the generalization capability of the agent on VPT-FJSP at different scales. In terms of exploring Makespan minimization, the agent outperformed four priority scheduling rules. These results indicate that the proposed dynamic scheduling framework and PPO algorithm are more effective in achieving superior solutions.

1. Introduction

With the continuous development of the manufacturing, production, and energy industries, scheduling problems have received increasing attention in recent years [1,2]. The dynamic flexible job-shop scheduling problem is a well-known combinatorial optimization problem in operations research and computer science [3,4], which can be highly abstracted into a real production environment and widely used in manufacturing, transportation, and energy industries [5–9]. Similarly to flexible job shop scheduling problems (FJSP), DFJSP is also a strong NP-hard problem [10] and consists of two parts, i.e., machine selection (MS) and sequence of operations (OS) [11]. Furthermore, DFJSP considers more dynamic events compared to other scheduling problems [12–14], and thus DFJSP is more closer to practical applications where unexpected events, e.g., workpiece insertion, urgent order, machine failure, variable processing times, etc., may occur. Therefore, it is of great significance to study the modeling and optimization of DFJSP, so as to improve the production efficiency, reduce production cost, and enhance production robustness.

In the past decades, many methods have been studied to solve the modeling and optimization of DFJSP. These methods can be mainly divided into two categories [15–17], i.e., rescheduling (pre-reaction scheduling and robust pro-active scheduling) and dynamic real-time scheduling.

The first kind of methods usually use meta-heuristic algorithms (MAs) to obtain an optimal rescheduling scheme when dynamic events occur. Nie et al. [18] proposed a reactive scheduling approach named Gene Expression Programming (GEP) to solve DFJSP considering new job insertion. Shen and Yao [19] developed a multi-objective evolutionary algorithm (MOEA)-based proactive-reactive method. A new rescheduling method based on MOEA is proposed to regenerate the

* Corresponding author.
E-mail address: zl.yang@siat.ac.cn (Z. Yang).

https://doi.org/10.1016/j.jmsy.2023.09.009
Received 2 March 2023; Received in revised form 18 August 2023; Accepted 17 September 2023
Available online 27 September 2023

stochastic events in DFJSP. Ning et al. [20] proposed an improved hybrid multi-phase quantum particle swarm algorithm (HQPSA) for flexible job shop with new job insertions, breakdowns of the machine, and delivery changes. Zadeh et al. [21] presented artificial bee colony (ABC) algorithm to solve DFJSP with variable processing times. In the method, firstly, the original scheduling scheme is generated by the estimated processing time, and then a new scheme is planned after the processing time changes. Nasr et al. [22] proposed a two-stage Hybrid Genetic Algorithm (HGA) to generate the predictive and robust schedule with random breakdowns machine. Buddala and Mahapatra [23] used two-stage teaching-learning-based optimization (2S-TLBO) method to solve FJSP under machine breakdown. Li et al. [24] proposed a heuristic algorithm based on Average Processing Time and Leverage (APT-LVR). By combining a closed-loop feedback control scheme, adaptive production scheduling and control were achieved in a flow shop production, transforming variable processing times into average time to address the issue. Ye et al. [25] introduced a novel approach based on real-time machine age to dynamically update maintenance intervals, maximizing the coordination between maintenance activities and job scheduling, thereby reducing overall costs. These methods can give higher quality scheduling scheme, but maybe more time-consuming and cannot meet the real-time scheduling requirements in some dynamic event frequent scheduling scenarios.

The second type of methods typically use priority dispatch rules (PDRs) [26,27] to solve dynamic events in scheduling. Lawrence and Sewell [28] have shown that simple heuristic scheduling rules provide better performance than more complex scheduling algorithms as the uncertainty of processing time increases. Gabel and Riedmiller [29] modeled the production scheduling problem as a continuous multi-agent decision-making process and trained it repeatedly by reinforcement learning (RL) algorithm. Aydin et al. [30] considered the insertion of a new job and proposes an improved Q-learning method to train agents and select scheduling rules. Shahrabi et al. [31] developed a Q-factor algorithm to improve the performance of dynamic JSP and took into account the two dynamic events of random arrival of artifacts and machine failures. Many of the above methods based on PDRs can respond to dynamic events immediately, but the selection of scheduling rules is short-sighted, and it is difficult to meet the requirements of adaptive scheduling.

To address the above issues, many scholars introduce deep reinforcement learning (DRL) into job-shop scheduling [32]. For solving permutation flow-shop scheduling problem (PFSP), Pan and Wang et al. [33] proposed an efficient optimization algorithm based on DRL to minimize the maximum completion time. Zhang et al. [34] used graph neural network to model job-shop scheduling problem, and proposed reinforcement learning method to generate adaptive scheduling. For solving dynamical flexible job shop scheduling with new job insertion, Luo [35] proposed six composite dispatching rules as the actions and adopted deep Q-network (DQN) to minimize the total tardiness.

Drawing from the literature reviewed, dynamic scheduling for flexible job shop operations is of immense significance. Under infrequent disturbances, predictive reactive scheduling and robust proactive scheduling perform well; however, they are constrained when disturbances are frequent and uncertain. Complete reactive scheduling relies on rules for job processing, offering good real-time performance but lacking adaptability to dynamic environments. Therefore, this paper investigates the dynamic flexible job shop scheduling problem considering variable processing times(VPT-FJSP). To achieve real-time scheduling effects, the paper models VPT-FJSP as a sequential decision problem and employs the Proximal Policy Optimization algorithm in deep reinforcement learning to train the agent, thus achieving the objectives of offline training and online application. Specifically, the main contributions of this article are as follows:

1. Develop a two-stage DRL dynamic scheduling framework based on scheduling experience for training the agent.
2. Present the mathematical model of VPT-FJSP and model scheduling problem as a sequential decision process using disjunctive graphs and Gantt charts.
3. Considering the high complexity of VPT-FJSP, this paper designs six state transition functions, eight intelligent agent actions, and reward functions that encompass both immediate and long-term aspects, which can effectively represent the reinforcement learning environment and speed up the convergence of the algorithm.
4. Test the feasibility of the PPO algorithm in FJSP instances of different scale sizes, and add different degrees of VPT in dynamic scheduling to test the effectiveness of the algorithm and the generalization ability of the agent.

Table 1
An example of 3 × 3 VPT-FJSP.

Job	Operation	Machines and processing time		
		M_1	M_2	M_3
J_1	$O_{1,1}$	2	7	–
	$O_{1,2}$	–	3	$6 + \Delta_{123}$
	$O_{1,3}$	7	–	$5 + \Delta_{133}$
J_2	$O_{2,1}$	3	8	–
	$O_{2,2}$	–	9	$3 + \Delta_{223}$
	$O_{2,3}$	7	3	–
J_3	$O_{3,1}$	–	3	$8 + \Delta_{313}$
	$O_{3,2}$	4	8	–
	$O_{3,3}$	–	8	$3 + \Delta_{333}$

The remainder of this paper is structured as follows. Section 2 presents the mathematical formulation of the VPT-FJSP and provides background information on PPO. Section 3 details the two-stage dynamic framework of DRL, establishes a scheduling environment for the DRL agent, which includes the design of the state, action, and reward components, and subsequently applies the PPO algorithm to address the VPT-FJSP. Section 4 presents the experimental results obtained in both static and dynamic environments. Finally, Section 5 provides the conclusions and outlines directions for future research.

2. The VPT-FJSP and PPO

This work is focused on VPT-FJSP and recommends the use of PPO to address this issue. Therefore, this section will briefly describe VPT-FJSP in mathematical formulae and introduce the principles of RL and PPO algorithms.

2.1. Mathematical formula description of VPT-FJSP

The VPT-FJSP in this paper can be defined as follows: There are n jobs on m machines which will be scheduled. Each job J_i has n_i operations, like $O_{i,1}, O_{i,2}, \ldots, O_{i,n_i}$, will be dispatched a machine, one of $M = \{M_1, M_2, \ldots, M_m\}$. For each operation $O_{i,j}$, there are many machines to choose for processing and only assigned to one of these machines, but the processing time of the operation $O_{i,j}$ is different on each machine. Even with the same machine, the processing time of the workpiece operation is variable, creating dynamic and complex production environments. To solve the VPT-FJSP, it is necessary to arrange an appropriate processing time and select a suitable machine for each operation of the workpiece.

Table 1 is an example of 3 × 3 (3 jobs and 3 machines) VPT-FJSP, where the processing time of machine M_3 is variable.

To give a more exhaustive description, we use the following symbols in Table 2.

Based on the symbols and referring the model developed in Lu et al. [36], the objective function can be expressed, where Makespan is maximum completion time, namely,

$$f = \min(\text{ Makespan }) = \min \max_{1 \le k \le m} \left\{ \max_{1 \le i \le n} C_i \right\} \tag{1}$$

Table 2
The definition of symbols.

Symbols	Definition
n	the number of jobs
m	the number of machines
J_i	the ith job
n_i	the total number of operations to job J_i
M_k	the kth machine
$M_{i,j}$	the available machine set of $O_{i,j}$
$O_{i,j}$	the jth operation of the ith job
$S_{i,j,k}$	start time of $O_{i,j}$ on M_k
$t_{i,j,k}$	processing time of $O_{i,j}$ on M_k
$t'_{i,j,k}$	variable processing times of $O_{i,j}$ on M_k
a	variable coefficient of processing time
$E_{i,j,k}$	end time of operation $O_{i,j}$ on machine k
C_i	completion time for job J_i
$C_{i,j}$	completion time for operation $O_{i,j}$
$B_{i,j}$	beginning time of operation $O_{i,j}$
$X_{i,j,k}$	whether $O_{i,j}$ is assigned on machine M_k
$Y_{i,j,h,g} = 1$	$O_{i,j}$ is a predecessor of $O_{h,g}$ in M_k
$Y_{i,j,h,g} = -1$	$O_{i,j}$ is a successor of $O_{h,g}$ in M_k

In the VPT-FJSP, the following constraints need to be met:

1. All machines are available at time zero, and all jobs can start at time zero.

$$B_{i,0} \geq 0, \quad C_{i,j} > 0, \quad \forall i, j \tag{2}$$

2. The operation of each job can only select one machine to process from machines set.

$$\sum_{k \in \boldsymbol{M}_{i,j}} X_{i,j,k} = 1, \quad \forall i, j \tag{3}$$

3. In eq (4), the processing time $t'_{i,j,k}$ for each operation on the corresponding machine is variable, which is related to the modeling of the actual production process, as in eq (5), where $t'_{i,j,k}$ follows the uniform distribution with $a \in (0,1)$, and the transportation time is ignored.

$$S_{i,j,k} + t'_{i,j,k} = E_{i,j,k} \tag{4}$$

$$t'_{i,j,k} \sim U\left[(1-a) \cdot t_{i,j,k}, (1+a) \cdot t_{i,j,k}\right], \quad \exists k \tag{5}$$

4. Sequence of the operation job J_i should be guaranteed in eq (6).

$$\left(C_{i,j} - t_{i,j,k} - C_{i,j-1}\right) X_{i,j,k} \geq 0, \quad \forall i, j, k \tag{6}$$

5. Processing of different jobs on the same machine is sequential and constraints (7) should be satisfied.

$$\begin{aligned} &\left(C_{h,g} - t_{h,g,k} - C_{i,j}\right) X_{i,j,k} X_{h,g,k} \left(Y_{i,j,h,g} + 1\right) + \\ &\left(C_{i,j} - t_{i,j,k} - C_{h,g}\right) X_{i,j,k} X_{h,g,k} \left(1 - Y_{i,j,h,g}\right) \geq 0, \\ &\qquad \forall i, j, h, g, k. \end{aligned} \tag{7}$$

These formulas and notations describe the VPT-FJSP in detail and will be used to model the DRL environment in Section 3.

2.2. Background to reinforcement learning

RL mainly consists of five parts: Agent, Environment, State, Action and Reward. Fig. 1 shows the process where the agent interacts with the environment. In detail, after an agent executes an action a_t according to the current state s_t, the environment will accept the action a_t, transfer to the next state s_{t+1}, and output a reward r_{t+1} to the agent.

This is a sequential decision-making process that can be described by tuples $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ represent states and action spaces, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ is the state transition probability distribution, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ is a reward function, and $\gamma \in (0,1)$ is a discount factor. Given a policy $\pi(a_t \mid s_t) : \mathcal{S} \times \mathcal{A} \in (0,1)$, the agent can generate a trajectory $\tau = \{s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_T\}$. The goal of RL is to find an optimal strategy $\pi^*(a_t \mid s_t)$, which can get the maximum $V^\pi(s)$ for all state $s \in S$ as follows:

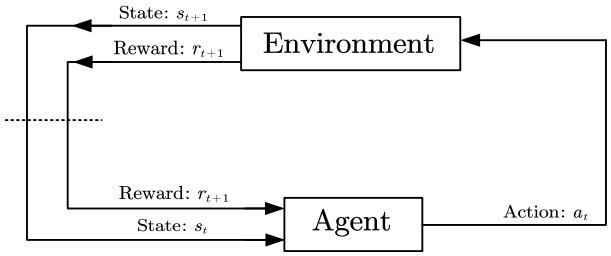


Fig. 1. Agent and environment in reinforcement learning.

$$V^\pi(s) = \mathop{\mathbb{E}}_{\tau \sim \pi}(G_t) = \mathop{\mathbb{E}}_{\tau \sim \pi}\left\{\sum_{n=0}^{\infty} \gamma^n r_{t+n} \mid s_t = s\right\}, \tag{8}$$

where $V^\pi(s)$ is state value function, standing for the expectation of long-term cumulative return G_t, and G_t is calculated from a series of reward r_t. When the agent executes action a_t in state s_t, it can also give the definition of state–action value function $Q^\pi(s,a)$ as follows:

$$Q^\pi(s,a) = \mathop{\mathbb{E}}_{\tau \sim \pi}\left\{\sum_{n=0}^{\infty} \gamma^n r_{t+n} \mid s_t = s, a_t = a\right\}. \tag{9}$$

Finally, the relationship between $V^\pi(s)$ and $Q^\pi(s,a)$ can be obtained by the formula (10) of total probability.

$$V^\pi(s) = \sum_{a \in \mathcal{A}} \pi(a \mid s) Q^\pi(s,a) = \mathbb{E}[Q^\pi(s,a)]. \tag{10}$$

2.3. Proximal policy optimization algorithm

In this paper, we use PPO algorithm to train the agent, which mainly includes three parts, namely, policy optimization method, advantage function estimation, and objective function transformation.

2.3.1. Policy optimization method

In an RL environment with continuous states, it is impractical to find the optimal action for all states. We usually use the parameterized probability function $\pi_\theta(a \mid s)$ to express the policy, and use the optimization method to seek the optimal vector parameter θ^* as follows:

$$J(\theta) = \sum_{\tau} G(\tau) \pi_\theta(\tau) = \mathop{\mathbb{E}}_{\tau \sim \pi_\theta}[G(\tau)], \tag{11}$$

$$\theta^* = \arg\max_\theta J(\theta), \tag{12}$$

where $J(\theta)$ is the objective function, representing the expected value of all returns $G(\tau)$, and $\pi_\theta(\tau)$ is the probability of the trajectory τ occurring, which can be calculated from the following formula.

$$\pi_\theta(\tau) = p\left(s_0\right) \prod_{t=0}^{T-1} \pi_\theta\left(a_t \mid s_t\right) p(s_{t+1} \mid s_t, a_t), \tag{13}$$

where $p(s_0)$ obeys the initial state distribution in the environment, and $p\left(s_{t+1} \mid s_t, a_t\right)$ is the probability of state transition.

We usually use the gradient method $\theta^{(k+1)} = \theta^{(k)} + \alpha \cdot \nabla J\left(\theta^{(k)}\right)$ to optimize θ, and then get the derivative:

$$\nabla_\theta J(\theta) = \sum_{\tau} G(\tau) \pi_\theta(\tau) \sum_{t=0}^{T-1} \nabla_\theta \ln \pi_\theta\left(a_t \mid s_t\right) \tag{14}$$

$$= \mathop{\mathbb{E}}_{\tau \sim \pi_\theta}\left\{\sum_{t=0}^{T-1} G(\tau) \cdot \nabla_\theta \ln \pi_\theta\left(a_t \mid s_t\right)\right\}. \tag{15}$$

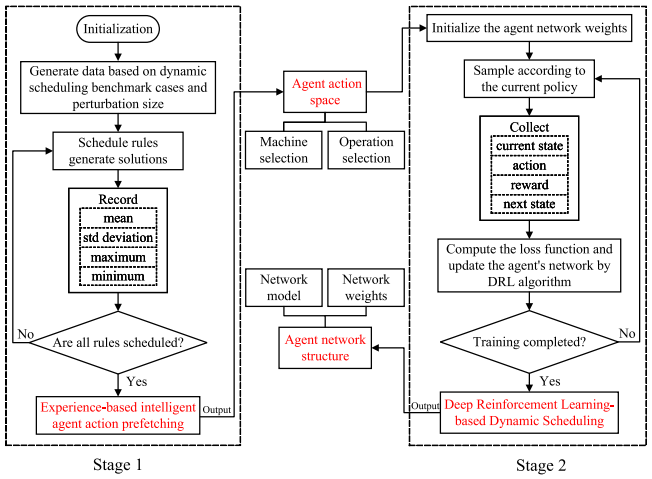


Fig. 2. Two-stage dynamic scheduling framework.

2.3.2. Advantage function estimation

Gradient optimization is similar to the maximum likelihood estimation method with weight G. However, there are two main disadvantages: G is too large to update the parameters of instability, and G has been positive probability increases is not obvious [37].

In order to overcome the above shortcomings, the advantage function is usually used as the weight, i.e.,

$$\hat{A}^{\pi_\theta}(s,a) = \hat{Q}^{\pi_\theta}(s,a) - V_\phi(s), \tag{16}$$

where the action-value function $\hat{Q}^{\pi_\theta}(s,a)$ is unbiased estimation of return G, and state-value function $V_\phi(s)$ is the expectation of $\hat{Q}^{\pi_\theta}(s,a)$ (10), representing the baseline. We usually use neural networks to fit $V_\phi(s)$, or the so-called Critic network.

There are several ways to estimate $\hat{A}^{\pi_\theta}(s,a)$, mainly the Monte Carlo (MC) methods and the time difference (TD) methods, such as formulas (17) and (18)

$$\hat{Q}^{\pi_\theta}\left(s_t,a_t\right) = \sum_{n=0}^{\infty}\gamma^n r_{t+n}. \tag{17}$$

$$\hat{Q}^{\pi_\theta}\left(s_t,a_t\right) = r_t + \gamma V_\phi\left(s_{t+1}\right). \tag{18}$$

Because the trajectory data of job-shop scheduling is relatively simple, the MC sampling method takes less time and can also reduce the estimation error. This paper will estimate $\hat{A}^{\pi_\theta}(s,a)$ using MC sampling.

2.3.3. Objective function transformation

In formula (11), policy π_θ is optimized as well as sampled online and thus a trajectory data can only be used once, resulting in extremely inefficient sampling. This method is called online policy. However, the importance sampling method [38] can be used to realize the offline sampling and improve the data utilization efficiency, i.e.,

$$J(\theta) = \mathop{\mathbb{E}}_{\tau\sim\pi_\theta} A(s,a) = \mathop{\mathbb{E}}_{\tau\sim\pi_{\theta_{old}}}\left\{\frac{\pi_\theta(a\mid s)}{\pi_{\theta_{old}}(a\mid s)}A^{\pi_{\theta_{old}}}(s,a)\right\}, \tag{19}$$

where $\pi_{\theta_{old}}$ is used for sampling, and π_θ is waiting for updates. Importance sampling requires that both policies have similar probability distributions that can be constrained to formula (20).

$$\text{s.t. KL}\left(\pi_{\theta_{\text{old}}}(\cdot\mid s)\,||\,\pi_\theta(\cdot\mid s)\right) \le \delta, \tag{20}$$

where δ is a bound on the KL divergence between π_θ and $\pi_{\theta_{old}}$. This is trust region policy optimization (TRPO) [39]. In order to solve this constrained optimization problem, the authors used the conjugate gradient method, but it is time-consuming and increases the complexity of the algorithm.

Compared with TRPO, PPO uses the clip function [40] to guarantee the constraints of formula (20), and constructs an unconstrained surrogate objective function to limit large policy updates, i.e.,

$$J(\theta) = \mathop{\mathbb{E}}_{(s,a)}\left\{\min\left(\begin{array}{c}\text{clip}\left(r(\theta),1-\varepsilon,1+\varepsilon\right)A^{\pi_{\text{old}}}(s,a),\\ r(\theta)\,A^{\pi_{\text{old}}}(s,a)\end{array}\right)\right\}, \tag{21}$$

where $r(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)}$, and $\varepsilon \in (0,1)$ is a clipping coefficient, roughly reflecting the similarity between π_θ and $\pi_{\theta_{old}}$.

3. Proposed methods for the VPT-FJSP

This section introduces a two-stage DRL dynamic scheduling framework for VPT-FJSP. Subsequently, the components of the framework are elaborated in detail.

3.1. Two-stage dynamic scheduling framework

The dynamic scheduling environment of flexible job shop scheduling is inherently more complex than static scheduling, and the size of the agent's action space is a crucial factor influencing algorithm convergence. An excessively large action space increases the complexity of algorithmic search, while a too small action space limits the decision-making capability of the agent. Therefore, this study proposes an Action Space Prefetching Strategy (ASPS) based on scheduling experience, which pre-extracts actions with potential advantages from previous scheduling experiences and incorporates them into the agent's action space. The extracted action space is then used as input for the deep reinforcement learning framework. This approach offers the advantage of reducing the algorithm's search space, thereby lowering computational complexity and enhancing algorithm convergence speed and stability.

Fig. 2 illustrates the two-stage dynamic scheduling process. In the first stage, different scheduling rules are employed to generate scheduling solutions for dynamic scheduling data, thereby obtaining performance indicators related to Makespan. The entropy weight method is then applied to extract the agent's action space. In the second stage, after determining the agent's action space, the agent interacts with the scheduling environment based on a sampling strategy, collecting relevant data. Subsequently, the data is utilized to compute the loss function and update the network using deep reinforcement learning algorithms, ultimately outputting the network model and its weights.

3.2. Stage 1: Experience-based agent action prefetching strategy

The size of the agent's action space significantly affects the convergence of the DRL algorithm. To reduce additional computational costs, a method based on scheduling rule experience is proposed for action space prefetching. Different scheduling rules are applied to a large set of dynamic scheduling data, resulting in performance indicators related to Makespan (mean, variance, maximum, minimum). The entropy weight method is then employed to standardize the scales and determine the weight coefficients. The specific calculation steps are as follows:

1. Identify the evaluation objects and evaluation indicators. With n evaluation samples and m evaluation indicators, a sample indicator matrix $X = \left(x_{ij}\right)_{n\times m}$ is formed, where $i = 1,2,\ldots,n$, $j = 1,2,\ldots,m$.
2. Normalize the values of each indicator to bring them to the same scale. The range method is used in this study. For performance indicators where higher values are better, the normalization formula is expressed as follows:

$$x'_{ij} = \frac{x_{ij} - x_{\min}}{x_{\max} - x_{\min}} \tag{22}$$

For performance indicators where lower values are better, the normalization formula is expressed as follows:

$$x'_{ij} = \frac{x_{\max} - x_{ij}}{x_{\max} - x_{\min}} \tag{23}$$

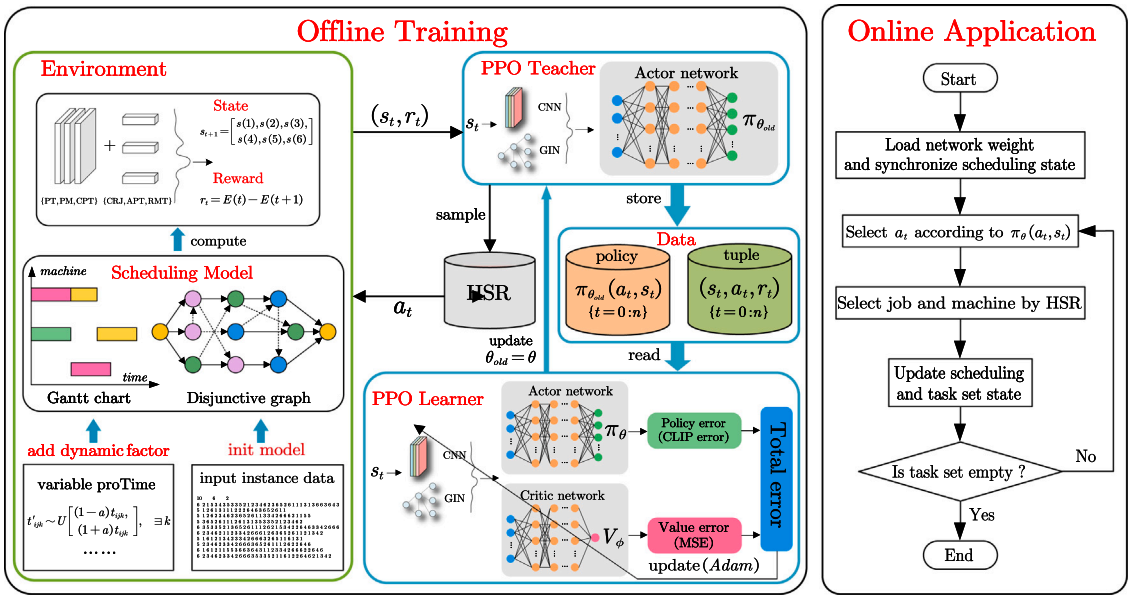


Fig. 3. PPO algorithm to solve the VPT-FJSP.

Here, $x_{\min}$ and $x_{\max}$ represent the minimum and maximum values of all samples under the jth indicator, and x'_{ij} denotes the normalized sample value.

3. Calculate the information entropy for each indicator. First, calculate the proportion of each sample under that indicator, as shown in eq (24).

$$r_{ij} = \frac{x'_{ij}}{\sum_{i=1}^{n} x'_{ij}} \tag{24}$$

Then, calculate the entropy for each evaluation indicator. The entropy formula for the ith evaluation indicator is given by eq (25).

$$E_j = -\frac{\sum_{i=1}^{n} r_{ij} \ln r_{ij}}{\ln n} \tag{25}$$

4. Calculate the weights for each indicator. Once the entropy for each indicator is obtained, the weights can be determined using the following formula:

$$w_j = \frac{1 - E_j}{m - \sum_{j=1}^{m} E_j} \tag{26}$$

5. Multiply the weights of each indicator by the normalized indicators to obtain the comprehensive scores of each sample. Based on the comprehensive scores, make decisions as outlined in eq (27).

$$s_i = \sum_{j=1}^{m} w_j \cdot x'_{ij} \tag{27}$$

Finally, the score of a specific scheduling rule is chosen as the baseline, and the rules with comprehensive scores surpassing the baseline are extracted as actions for the agent.

3.3. Stage 2: PPO algorithm for VPT-FJSP

This paper presents a comprehensive PPO-based approach to tackle dynamic scheduling problems, comprising two main phases: offline training and online deployment, as depicted in Fig. 3.

During the offline training phase, the scheduling environment is established. Firstly, the scheduling problem is formulated as a sequential decision problem using a disjunctive graph, and the scheduling state information is stored using a Gantt graph. Secondly, the dynamic scheduling environment is constructed by incorporating continuous disturbance factors. Lastly, when the environment receives an action from the agent, it undergoes state transition and reward calculation. Apart from the environment, there is the PPO teacher (agent), composed of a feature extraction network and an actor network. The former employs GNN and CNN architectures proposed by Zhang et al. [34], which have demonstrated superior ability in extracting scheduling states. The latter uses a multi-layer perceptron (MLP) to output sampling action policy $\pi_{\theta_{old}}(a_t, s_t)$. Interacting with the environment, PPO teacher generates experience data $\{\pi_{\theta_{old}}(a_t, s_t)\}_{t=0:n}$ and $\{(s_t, a_t, r_t)\}_{t=0:n}$. PPO learner then uses these data to update by gradient descent method, and synchronize the updated parameters to the teacher for the next iteration.

Regarding the online application module, it directly loads the trained network weights to the agent and synchronizes the initial scheduling state. Subsequently, the action a_t is chosen based on the probability distribution $\pi_\theta(a_t, s_t)$, and a_t is mapped to the appropriate hybrid scheduling rules (HSR) for job and machine selection. Finally, the scheduling state and task set are updated iteratively until all tasks are dispatched.

3.4. Setup dynamic scheduling environment

In order to create a reinforcement learning environment for workshop scheduling, it is necessary to model the scheduling problem as a Markov Decision Process and break down the scheduling tasks into individual nodes using disjunctive graphs. Processing information can then be stored in a Gantt chart. Additionally, since actual workshop scheduling involves various dynamic events, perturbation events should be added to the environment to create a dynamic workshop environment. Furthermore, the environment should include a state transition function, a reward function, and an agent action decoder.

3.4.1. Markov decision process establishment

Disjunctive graph model [41] and Gantt chart model [42] are well-known used to describe the shop scheduling problem. We typically use $\mathcal{N} = \{O_{ij} \mid \forall i, j\} \cup \{S, E\}$ to represent the set of all operations, where S and E represent the dummy start and end node with zero processing time. Then the disjunctive graph can be described with $G = (\mathcal{N}, C, D)$, where $\mathcal{N}$ is task set, C is a set of directed arcs (conjunctions) representing precedence constraints between operations on the same job, and D is a set of undirected arcs (disjunctions), each of which

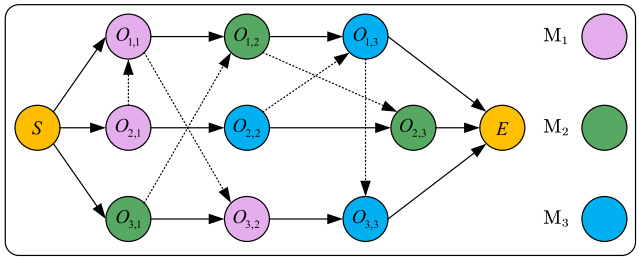


Fig. 4. Disjunctive graph model.

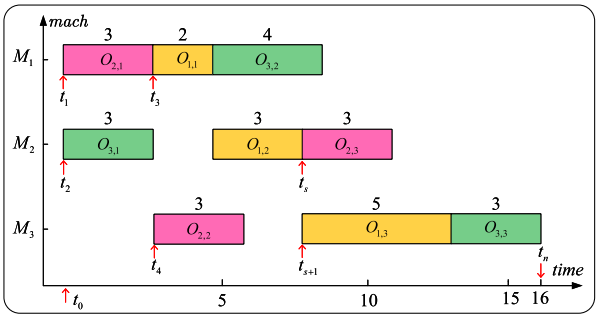


Fig. 5. Gantt chart model.

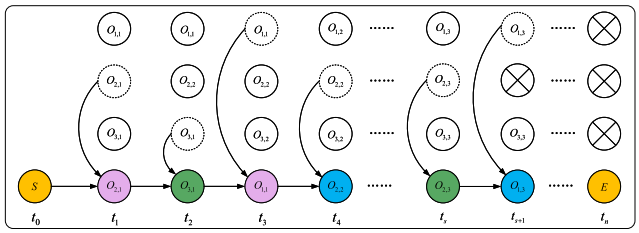


Fig. 6. Building DFJSP as MDP Model.

connects a pair of operations that require processing by the same machine.

Therefore, solving a job shop scheduling problem is equivalent to fixing the direction for each disjunction D (the dotted arrow in Fig. 4) on each machine, as shown in Fig. 4, representing the solution for the Table 1 example. In Fig. 4, each row represents an job, directed arc C is represented by solid arrows, and nodes with the same color represent processing on the same machine. For simplicity and convenience, some processing information, e.g., the starting time $S_{i,j,k}$ and ending time $E_{i,j,k}$ of a machining operation, the starting time $B_{i,j}$ and completed time $C_{i,j}$ of the operation $O_{i,j}$, are not indicated in disjunctive graph, but will be reflected on the Gantt chart, Fig. 5.

With the disjunctive graph model, the job shop scheduling problem can be abstracted into a sequential decision-making process known as the Markov Decision Process (MDP), as illustrated in Fig. 6. Initially, at time t_1, the set of operations waiting to be processed is $\{O_{11}, O_{21}, O_{31}\}$. The agent selects an operation, such as O_{21}, using the policy $\pi_\theta(s,a)$, as depicted in the figure. This selection transforms the set of operations to $\{O_{11}, O_{22}, O_{31}\}$, leading to the next decision point at t_2. As the last operation of a job is completed, the operations set decreases by one, resulting in options like $\{O_{13}, O_{33}\}$ at time t_{s+1}. This process continues until the operations set becomes empty, and the direction of each disjunction is established.

In short, the mathematical constraint formula in Section 2 is satisfied by the disjunctive graph, some scheduling state information is saved by gantt graph, and VPT-FJSP is also defined as MDP, which makes it possible to solve it next by PPO.

Algorithm 1: Using PPO to train agent and save model

Input: initial actor network π_θ and critic network υ_ϕ with normalized parameter. Initial $\pi_{\theta_{old}}$ by $\theta_{old} = \theta$. Set the following super parameters: episodes M of training; epochs K of updating $\theta_{old} = \theta$; episodes j of validating; clipping ratio ϵ; discounting factor γ; policy error and value error coefficient c_p, c_υ.

Output: the actor-critic network weight θ, ϕ of the min-makespan on the validation set.

for $m = 1, 2, ..., M$ **do**
  Collect a trajectory τ by running policy $\pi_{\theta_{old}}$;
  while *Ture* **do**
    Sample a_t according to $\pi_{\theta_{old}}(a_t, s_t)$;
    The agent receives immediate reward r_t and environment transfers to the next state s_{t+1};
    To store (s_t, a_t, r_t) and $\pi_{\theta_{old}}(a_t, s_t)$;
    if s_{t+1} *is the last state* **then**
      break;
    end
  end
  Compute rewards-to-go $\hat{Q}_t = r_t + \gamma r_{t+1} + \ldots + \gamma^N r_{t+N} = \sum_{n=0}^{N} \gamma^n r_{t+n}$;
  for $k = 1, 2, ..., K$ **do**
    Compute advantage estimates $\hat{A}_t = \hat{Q}_t - V_\phi(s_t)$;
    Compute loss error:
    $J^{CLIP}(\theta) = \sum_{t=0}^{T} \min\left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)$, where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$;
    $L^V(\phi) = \sum_{t=0}^{T}\left(V_\phi\left(s_t\right) - \hat{A}_t\right)^2$;
    Total error: $L(\theta, \phi) = c_p J^{CLIP}(\theta) - c_\upsilon L^V(\phi)$;
    Update θ, ϕ using back-propagation algorithm: $\theta, \phi =$ **argmax** $L(\theta, \phi)$
  end
  Update weights to the old actor network: $\theta_{old} = \theta$;
  if m *mod* $j == 0$ **then**
    Run model many times on validation set and save average value of makespan C_{avg};
    if $C_{avg} < C_{best}$ **then**
      $C_{best} = C_{avg}$;
      Save the actor-critic network weight θ, ϕ;
    end
  end
end

3.4.2. Construction of state transition function

When constructing a state transition function in RL, it is essential to adhere to the following principles. Firstly, it should provide a comprehensive depiction of the deep reinforcement learning environment, effectively capturing the environmental dynamics across diverse scenarios. Secondly, the chosen states should possess a certain level of generality, enabling the agent to exhibit a degree of generalization within the same scheduling problem category. Lastly, the design of states should prioritize computational efficiency, avoiding excessive complexity and minimizing the computational burden on the agent.

To ensure a comprehensive representation of the workshop environment, this article meticulously devised two distinct types of states.

The first type is the processing information state, comprising three matrices of size $|\mathcal{J}| \times |\mathcal{M}|$, where $|\mathcal{J}|$ represents the number of workpieces to be processed, and $|\mathcal{M}|$ represents the total number of available machines. These three matrices are concatenated into a three-channel high-dimensional vector, facilitating computation with convolutional neural networks. The following is a detailed description of the three states, namely $\{PT, PM, CPT\}$.

1. State $s(1)$ represents the processing time (PT) matrix, with each row denoting an operation O_{ij} of a job, and each column representing a machine M_k. The value of each element in the matrix represents the processing time required for the corresponding operation on the designated machine. A value of 0 indicates that the operation cannot be processed on the respective machine. To facilitate improved learning of the agent, the processing time can be scaled using the following formula.

$$PT_{i,j,k} = \frac{t_{ij,k}}{\{t_{ij,k}, \forall i,j,k\}_{\max}} \quad \forall i,j,k \tag{28}$$

2. State $s(2)$ is the processing mark (PM) matrix, where the elements are of boolean type, indicating whether a job operation $O_{i,j}$ can be processed on the corresponding machine.

$$PM_{ij,k} = \begin{cases} 1 & \text{can be processed on } M_k \\ 0 & \text{cannot be processed on } M_k \end{cases} \quad \forall i,j,k \tag{29}$$

3. State $s(3)$ is the cumulative processing time (CPT) matrix, which represents the accumulated processing time for each job on different machines at the current decision point. $X_{i,j,k}$ is a decision variable that indicates whether operation O_{ij} can be processed on machine M_k.

$$CPT_{i,k} = \sum_{j=0}^{j} t_{ijk} * X_{i,j,k} \quad \forall i,k \tag{30}$$

The second type of state is the semantic information state, comprising three vectors of size $|\mathcal{J}| \times |1|$. These three vectors can be incorporated into a disjunctive graph, facilitating the calculation of a graph embedding network. The following are detailed descriptions of these three states.

1. State $s(4)$ indicates the percentage of operations completed for each job (CRJ), i.e.,

$$CRJ_i(t) = \frac{\sum_{j=0}^{t}\sum_{k=0}^{m} X_{i,j,k}}{n_i}, \quad \forall i. \tag{31}$$

2. State $s(5)$ is the average processing time of operation for each job (APT), i.e.,

$$APT_{ij} = \frac{\sum_{k=0}^{m} l_{i,j,k} * t_{i,j,k}}{\sum_{k=0}^{m} l_{i,j,k}}, \quad \forall i,j. \tag{32}$$

where $APT_{i,j}$ should be scaled to $[0,1]$ by the longest processing time, $l_{i,j,k}$ indicates the operation O_{ij} can be processed on the machine M_k.

3. State $s(6)$ represents the total remaining completion time of the job (RMT), scaled to $[0,1]$ by the maximum completion time of the job, i.e.,

$$RMT_i(t) = \frac{\sum_{j=t}^{n_i} APT_{i,j}}{\{RMT_i\}_{max}}, \quad \forall i. \tag{33}$$

Thus, this paper has designed six states $\{PT, PM, CPT, CRJ, APT, RMT\}$ to represent the reinforcement learning environment and used two network structures to extract features from the job shop state information.

3.4.3. Design of action space

In VPT-FJSP, the agent's action is to select the appropriate workpiece at the decision point and assign it to a feasible machine for processing. Therefore, the action consists of two parts, namely, operation sequence (OS) and machine selection (MS). Usually, two methods can be used to encode the action space.

One method is to use hard coding to encode the machines and workpieces to be selected into a matrix of size $|\mathcal{J}| \times |\mathcal{M}|$. The policy function is used to directly select the value at a certain coordinate in the matrix, which decodes the workpiece operation and the corresponding machine to be processed. However, the search space complexity of the agent is $O(n^2)$, and the training process is unstable, making it difficult for the network to converge. The other method is to use soft coding, which defines the agent's action space based on the form of priority dispatching rules (PDR). The agent does not directly output the corresponding workpiece operation and machine, but outputs a coded heuristic scheduling rule. Based on the current state information and processing information matrix, the heuristic scheduling rule is used to decode the corresponding workpiece operation and machine. At this time, the search space complexity of the agent is $O(k)$, which is of constant complexity and beneficial for the convergence of the network.

Table 3
The meaning of priority dispatching rules.

PDR	Meaning of priority dispatching rules
MWKR	most average work time remaining
MOPNR	most operations remaining
SPT	shortest average processing time
LPT	longest average processing time
LWT	less total processing time of the machine
SPTM	shortest processing time of the machine

To minimize makespan as the objective on VPT-FJSP, this article defined four scheduling rules for the agent to select workpiece operations, which are MWKR, MOPNR, SPT, and LPT. Two rules are used to select machines, which are LWT and SPTM. The specific meanings are shown in Table 3. These rules are combined into 8 hybrid scheduling rules (HSR) as the action space of the agent (see Fig. 8). Dashed lines of the same color represent the combination of two rules. The specific combination method is shown in Fig. 12.

3.4.4. Reward function setting

The reward function is the short-term reward obtained by taking an action in the current state. In RL, there is no target value label like in supervised learning, so the reward function is crucial in guiding the algorithm training. Therefore, the definition of the reward function is very important in reinforcement learning and should satisfy the following three principles: Firstly, the calculation of the reward function should be immediate and be able to evaluate the goodness or badness of the agent's current action; secondly, the definition of the reward function should have long-term considerations, meaning that the optimization objective function and the maximum expected cumulative reward should be consistent; finally, the design of the reward function should be general, considering not only the general situations in the environment, but also some special cases in the environment.

This paper takes the minimization of Makespan as the scheduling objective, formula (1), so the expectation of maximizing the cumulative reward value G_t (the goal of RL) should be consistent with the minimization of Makespan. With this in mind, the definition of reward function is given, i.e.,

$$R(s_t, a_t) = E(t) - E(t+1) \tag{34}$$

where $R(s_t, a_t)$ is an immediate reward for performing an action a_t in state s_t, $E(t)$ is maximum end time on all machines at s_t, and the same $E(t+1)$ at s_{t+1}. Such a reward function enables the agent to select a operation with short processing time to fill the gap time, while balancing the load between the machines.

Taking the decision point t_3 in Fig. 6 as an example, the end time $E(3)$ of the current node is 5. As shown in Fig. 7, the process inside the dashed box represents the process waiting for processing, i.e., $\{O_{1,2}, O_{2,2}, O_{3,2}\}$, and the agent can only select one process from it for processing, thus transitioning to $E(4)$. The timely reward $\{-1, -3, -4\}$ for the three options can be calculated. Assuming that the action selected by the agent is process $O_{2,2}$, the environment undergoes

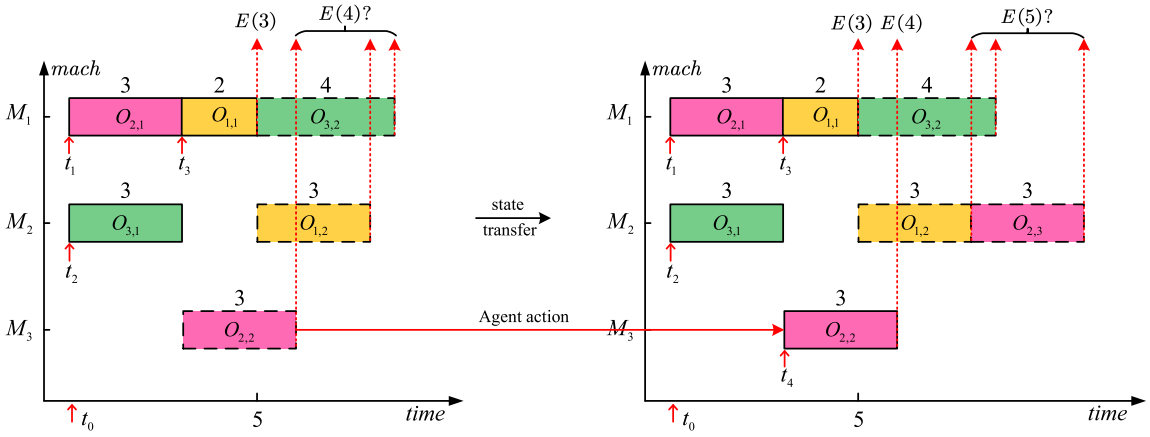


Fig. 7. Reward function under state transition..

Table 4
The result of static experiments.

Instance	Metric	OPT	$Rule_1$	$Rule_2$	$Rule_3$	$Rule_4$	$Rule_5$	$Rule_6$	$Rule_7$	$Rule_8$	ACO	GA	PPO
mk01	Makespan	40	51	49	71	73	87	81	67	82	**42**	**42**	**42**
(10 × 6)	Scheduling (%)	–	78.4	81.6	56.3	54.8	46.0	49.4	59.7	48.8	**95.2**	**95.2**	**95.2**
mk03	Makespan	204	230	219	330	330	428	332	362	392	**204**	216	**204**
(15 × 8)	Scheduling (%)	–	88.7	93.2	61.8	61.8	49.5	61.4	56.2	52.0	**100.0**	94.4	**100.0**
mk05	Makespan	172	192	201	209	210	248	211	260	250	**176**	180	182
(15 × 4)	Scheduling (%)	–	89.6	85.6	82.3	81.9	69.4	81.5	66.2	68.8	**97.7**	95.6	94.5
mk06	Makespan	58	118	107	104	101	115	117	145	180	**75**	106	88
(10 × 15)	Scheduling (%)	–	49.2	54.2	55.8	57.4	50.4	49.6	40.0	32.2	**77.3**	54.7	65.9
mk07	Makespan	139	227	220	217	217	225	234	311	258	160	171	**159**
(20 × 5)	Scheduling (%)	–	61.2	63.2	64.1	64.1	61.8	59.4	44.7	53.9	86.9	81.3	**87.4**
mk08	Makespan	523	551	537	597	603	711	689	711	637	**523**	**523**	529
(20 × 10)	Scheduling (%)	–	94.9	97.4	87.6	86.7	73.6	75.9	73.6	82.1	**100.0**	**100.0**	98.9
la01	Makespan	570	635	743	805	729	930	956	784	825	603	589	**578**
(10 × 5)	Scheduling (%)	–	89.8	76.7	70.8	78.2	61.3	59.6	72.7	69.1	94.5	96.8	**98.6**
la06	Makespan	799	830	981	1101	1050	1167	1166	1311	1025	824	810	**809**
(15 × 5)	Scheduling (%)	–	96.3	81.4	72.6	76.1	68.5	68.5	60.9	75.0	97.0	98.6	**98.8**
la11	Makespan	1071	1114	1179	1227	1276	1652	1329	1549	1379	1095	**1074**	1075
(20 × 5)	Scheduling (%)	–	96.1	90.8	87.3	83.9	64.8	80.6	69.1	77.7	97.8	**99.7**	99.6
la16	Makespan	717	840	791	1186	1140	1539	1190	1228	1186	761	787	**730**
(10 × 10)	Scheduling (%)	–	85.4	90.6	60.5	62.9	46.6	60.3	58.4	60.5	94.2	91.1	**98.2**
la21	Makespan	804	931	1066	1270	1209	1658	1518	1330	1550	999	894	**891**
(15 × 10)	Scheduling (%)	–	86.4	75.4	63.3	66.5	48.5	53.0	60.5	51.9	80.5	89.9	**90.2**
la26	Makespan	1053	1150	1245	1575	1528	2060	2444	1762	1664	1212	1117	**1103**
(20 × 10)	Scheduling (%)	–	91.6	84.6	71.0	68.9	51.1	43.1	59.8	63.3	86.9	94.3	**95.5**
la31	Makespan	1520	1582	1668	2082	2004	2667	2478	2292	2303	1628	1552	**1538**
(30 × 10)	Scheduling (%)	–	96.1	91.1	73.0	75.8	57.0	61.3	66.3	66.0	93.4	97.9	**98.8**
la36	Makespan	948	1222	1084	1698	1653	1962	2130	1805	1676	1187	1260	**1018**
(15 × 15)	Scheduling (%)	–	77.6	87.5	55.8	57.4	47.8	44.5	52.5	56.6	79.9	75.2	**93.1**
mt06	Makespan	47	56	60	99	98	100	117	75	73	**47**	**47**	**47**
(6 × 6)	Scheduling (%)	–	83.9	78.3	47.5	48.0	47.0	40.2	62.7	64.4	**100.0**	**100.0**	**100.0**
mt20	Makespan	1022	1052	1222	1264	1243	1654	1530	1561	1388	1058	**1027**	1040
(20 × 5)	Scheduling (%)	–	97.1	83.6	76.3	82.2	61.8	66.8	65.5	73.6	96.6	**99.5**	98.3
Average	Scheduling (%)	100	85.14	82.20	67.88	69.16	56.57	59.69	60.55	62.24	92.37	91.51	**94.29**

a state transition to the decision point t_4, and an immediate reward of −1 is output to the agent.

From Eq. (34), it can be concluded that the designed reward function can evaluate the goodness of the agent's current action, satisfying the immediacy design principle of the reward function. When setting the discount factor $\gamma = 1$, the calculation of the cumulative reward G_t can be obtained as follows.

$$\begin{aligned} G_t &= \sum_{t=0}^{N} R_(s_t, a_t) = \sum_{t=0}^{N} E(t) - E(t+1) \\ &= E(0) - E(1) + E(1) - E(2) + \cdots - E(N+1) \\ &= E(0) - E(N+1) \\ &= -Makespan \end{aligned} \tag{35}$$

which indicates that maximizing G_t is equivalent to minimizing Makespan, which conforms to the principle of long-term reward function design.

3.5. Using PPO to solve VPT-FJSP

Algorithm 1 gives the steps and details of PPO training agent and saving model (lines 1-28). It consists mainly of the following three

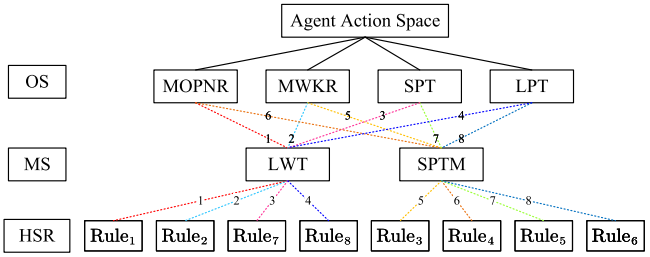


Fig. 8. Agent Action Space.

parts: PPO teacher generates experience data (lines 2-10), the network updates weight (lines 11-20) and the agent tests on the verification set (lines 21-27).

(1) PPO teacher samples a_t through policy $\pi_{\theta_{old}}$, interacts with the environment to generate data $\pi_{\theta_{old}}(a_t, s_t)$ and (s_t, a_t, r_t), and stores them to memory units. Repeat the process until the termination state (task set is empty).

(2) Rewards-to-go $\hat{Q}_t$ is calculated by MC method (line 11). Then the estimation of advantage function $\hat{A}_t$ is calculated (line 13), and the CLIP error $J^{CLIP}(\theta)$ is calculated in line 15 and the error of critic network $L^V(\phi)$ is calculated in line 16. Finally, the total error is obtained by weighting two error coefficients (c_p, c_v) and the PPO learner is updated by gradient ascent method (line 18). After updating the learner K times, copy its weights to the teacher (line 20).

(3) We arrange the agent to test on the validation set every j episodes, and save the weight of the agent if the average makespan is better than last time (lines 23-25).

4. Experiments

In this section, the relevant experimental configurations are presented. Subsequently, experiments without variable processing times (VPT) are designed in this paper and compared against a single priority scheduling rule and well-known meta-heuristic algorithms (MAs). Following that, the proposed two-stage dynamic scheduling framework is employed in the experiments considering VPT. In the stage 1, the action space of the agent is pre-extracted using the example of la36 to accelerate training. In the stage 2, the agent is trained and tested on instances with different levels of VPT. Finally, the trained model is loaded and tested on various dynamic benchmark tests to verify the generalization of the agent.

4.1. Experimental configurations

In this article, we utilize benchmark instances from the OR library as the basis for initializing the scheduling model and creating the scheduling environment. The source of these instances is indicated in Table 5. These instances are static and have fixed processing time (without VPT). However, to address the VPT-FJSP, we employ formula ((4), (5)) to fluctuate the processing time $t_{i,j,k}$ of each operation $O_{i,j}$. This enables the generation of a significant amount of training data from each instance. The same approach can be applied to construct the validation set.

In addition to introducing the data source in the experiment, the experiment parameters need to be configured, as shown in Table 6, where algorithm 1 gives the meaning of most hyperparameters. Besides, M_1 is the episodes in static experiment (without VPT), while M_2 is in dynamic experiment (with VPT). a is the volatility factor that determines the percentage of variation in processing time.

Table 5
The source of instance in Benchmarks.

Instance	Source
mk01(10 × 6), mk03(15 × 8), mk05(15 × 4) mk06(10 × 15), mk07(20 × 5), mk08(20 × 10)	Brandimarte [43]
la01-05(10 × 5), la06-10(15 × 5), la11-15(20 × 5) la16-20(10 × 10), la21-25(15 × 10), la26-30(20 × 10) la31-35(30 × 10), la36-40(15 × 15), abz7-9(20 × 15) mt06(6 × 6), mt20(20 × 5)	Vdata of Hurink [44]

Table 6
The hyperparameters of experiments.

Parameters	Value
Episodes M_1	200
Episodes M_2	20000
Clip ratio ϵ	0.1
Policy error coefficient c_p	2
Value error coefficient c_v	1
Epochs K	12
Discount factor γ	0.99
Episodes j	100
Optimizer	Adam
Learning rate	1e-4 → 0
Variable factor a	{0, 10, 30, 50}

Table 7
The parameters of GA and ACO.

Parameters	Value
GA	
Total population	400
Max generation	200
Mutation probability	0.1
Permutation probability	0.3
ACO	
Pheromone factor α	2
Expectation factor β	5
Evaporation coefficient ρ	0.1
Number of iterations N	200
Number of ant colony S	150

4.2. Comparison of results on static experiments

In this experiment, we trained the agent using PPO algorithm without considering VPT (set $a = 0$). To validate the effectiveness of the proposed PPO, we compared its performance against the proposed 8 hybrid scheduling rules. To ensure more conclusive results, we also compared PPO with other MAs, specifically the Genetic Algorithm (GA) and the Ant Colony Optimization (ACO). The parameters for GA and ACO were set as (see Table 7).

Table 4 presents the makespan results of these methods on static instances of different sizes, where OPT represents the best solution [12], Scheduling indicates the optimization rate, and the best results are highlighted in boldface for better visibility. Additionally, the instances are categorized into different scales based on the value of $n \times m$. Specifically, instances with $n \times m \leqslant 100$ are considered small-scale, instances with $100 < n \times m \leqslant 200$ are considered medium-scale, and instances with $n \times m > 200$ are considered large-scale. Therefore, the statistical results for the optimization rate are obtained as Fig. 9.

From the results presented in Table 4, it is evident that PPO consistently outperforms the single HSR in all tests, demonstrating superior scheduling performance compared to GA and ACO in most instances. The statistical analysis reveals that PPO achieves a high optimization rate across all scales, achieving a remarkable 94.29% on all benchmark instances.

These findings lead us to conclude that the single HSR exhibits subpar scheduling performance, emphasizing the significance of careful

Table 8
The result of dynamic experiments.

Instance	a	PPO	MOPNR+LWT	MWKR+LWT	MWKR+SPTM	MOPNR+SPTM	Random
la01(10 × 5)	10	**0.561e+3/36.94***	0.564e+3/40.03	0.617e+3/56.90	0.748e+3/84.86	0.725e+3/85.70	0.675e+3/87.19
	20	**0.556e+3/49.59***	0.564e+3/50.79	0.623e+3/61.39	0.719e+3/83.61	0.695e+3/79.87	0.646e+3/66.85
	30	**0.562e+3/55.37***	0.570e+3/53.66	0.614e+3/65.24	0.685e+3/81.77	0.666e+3/83.82	0.644e+3/72.63
	50	**0.585e+3/59.09***	0.598e+3/60.50	0.635e+3/69.01	0.628e+3/85.63	0.604e+3/76.64	0.611e+3/79.52
la06(15 × 5)	10	**0.797e+3/55.11***	0.802e+3/55.35	0.878e+3/71.44	1.075e+3/132.31	1.043e+3/133.59	0.914e+3/90.79
	20	**0.798e+3/57.62**	0.802e+3/57.57	0.864e+3/76.63	0.993e+3/119.52	0.965e+3/115.90	0.890e+3/81.78
	30	**0.798e+3/52.30**	0.801e+3/52.64	0.878e+3/72.61	0.963e+3/107.64	0.932e+3/103.21	0.888e+3/99.22
	50	**0.805e+3/57.91***	0.810e+3/59.54	0.893e+3/79.80	0.886e+3/106.22	0.860e+3/107.27	0.851e+3/82.20
la11(20 × 5)	10	**1.040e+3/57.07**	1.041e+3/57.23	1.095e+3/77.62	1.267e+3/127.16	1.237e+3/130.49	1.128e+3/91.71
	20	**1.039e+3/51.28**	1.040e+3/56.16	1.097e+3/85.91	1.201e+3/99.53	1.186e+3/100.45	1.085e+3/81.49
	30	**1.042e+3/68.09**	1.044e+3/70.94	1.089e+3/74.07	1.163e+3/133.35	1.145e+3/120.58	1.066e+3/83.97
	50	1.034e+3/81.98	**1.047e+3/68.88***	1.106e+3/78.94	1.078e+3/109.80	1.061e+3/111.81	1.027e+3/85.83
la16(10 × 10)	10	**0.736e+3/54.80***	0.783e+3/47.14	0.771e+3/66.28	1.032e+3/104.02	0.983e+3/94.93	0.892e+3/71.23
	20	**0.727e+3/51.60***	0.790e+3/52.73	0.767e+3/60.44	0.946e+3/94.37	0.918e+3/94.30	0.862e+3/77.80
	30	**0.740e+3/62.08***	0.798e+3/58.08	0.774e+3/72.08	0.866e+3/79.57	0.841e+3/83.76	0.829e+3/76.94
	50	0.738e+3/70.95	0.839e+3/62.44	0.801e+3/63.43	0.765e+3/80.79	**0.735e+3/67.40***	0.786e+3/62.88
la21(15 × 10)	10	0.901e+3/55.25	**0.901e+3/41.92**	0.929e+3/62.19	1.288e+3/128.67	1.237e+3/116.94	1.090e+3/81.08
	20	**0.894e+3/49.03***	0.914e+3/47.22	0.947e+3/67.03	1.173e+3/95.69	1.132e+3/102.89	1.038e+3/85.92
	30	**0.907e+3/48.55***	0.931e+3/50.56	0.948e+3/67.08	1.091e+3/109.56	1.052e+3/93.70	1.014e+3/82.31
	50	**0.922e+3/56.78**	0.968e+3/54.71	0.975e+3/72.07	0.931e+3/77.51	0.908e+3/80.15	0.952e+3/67.74
la26(20 × 10)	10	**1.079e+3/43.79***	1.089e+3/43.87	1.160e+3/58.92	1.523e+3/118.00	1.489e+3/119.44	1.289e+3/84.39
	20	**1.068e+3/41.63***	1.088e+3/44.15	1.170e+3/57.45	1.408e+3/108.59	1.384e+3/107.13	1.252e+3/81.18
	30	**1.081e+3/45.21***	1.102e+3/50.78	1.179e+3/70.89	1.304e+3/105.58	1.277e+3/96.61	1.207e+3/77.91
	50	**1.114e+3/48.65***	1.136e+3/54.54	1.194e+3/74.88	1.146e+3/101.24	1.118e+3/98.31	1.132e+3/70.53
la31(30 × 10)	10	1.519e+3/50.22	**1.531e+3/45.55**	1.641e+3/70.00	2.122e+3/173.01	2.074e+3/175.70	1.703e+3/86.78
	20	**1.520e+3/51.42***	1.526e+3/50.00	1.630e+3/66.97	1.887e+3/134.26	1.845e+3/139.49	1.631e+3/82.73
	30	**1.516e+3/53.21***	1.522e+3/53.36	1.645e+3/67.44	1.752e+3/125.42	1.702e+3/123.12	1.593e+3/84.12
	50	1.471e+3/82.49	1.545e+3/61.51	1.654e+3/86.58	1.499e+3/102.41	1.475e+3/103.58	**1.477e+3/73.63**
la36(15 × 15)	10	**1.029e+3/53.13***	1.153e+3/44.55	1.072e+3/59.42	1.534e+3/98.16	1.479e+3/84.72	1.316e+3/76.91
	20	**1.038e+3/51.39***	1.166e+3/48.43	1.085e+3/68.77	1.394e+3/95.27	1.345e+3/90.58	1.249e+3/77.26
	30	**1.044e+3/52.02***	1.172e+3/54.93	1.088e+3/61.66	1.255e+3/83.72	1.215e+3/78.85	1.209e+3/70.68
	50	**1.011e+3/67.66**	1.231e+3/66.65	1.149e+3/83.00	1.043e+3/77.16	1.012e+3/74.00	1.126e+3/70.76
mt06(6 × 6)	10	**0.479e+3/57.04***	0.505e+3/53.73	0.501e+3/58.77	0.598e+3/92.16	0.584e+3/84.76	0.545e+3/63.87
	20	**0.477e+3/58.13***	0.501e+3/52.77	0.504e+3/57.31	0.572e+3/75.56	0.550e+3/69.98	0.535e+3/70.93
	30	**0.489e+3/67.21***	0.507e+3/57.25	0.503e+3/65.43	0.547e+3/72.02	0.528e+3/67.22	0.525e+3/65.05
	50	**0.493e+3/65.36**	0.526e+3/59.55	0.525e+3/79.80	0.513e+3/74.23	0.489e+3/69.85	0.508e+3/66.96
abz7(20 × 15)	10	**1.237e+3/44.68***	1.263e+3/45.71	1.253e+3/64.76	1.805e+3/123.06	1.753e+3/117.03	1.514e+3/87.61
	20	**1.241e+3/40.21**	1.274e+3/42.26	1.254e+3/58.15	1.616e+3/123.12	1.553e+3/116.89	1.432e+3/69.69
	30	**1.252e+3/46.86***	1.291e+3/49.47	1.260e+3/69.39	1.480e+3/96.28	1.427e+3/96.24	1.376e+3/81.72
	50	1.273e+3/66.41	1.362e+3/57.00	1.315e+3/74.05	1.222e+3/68.73	**1.180e+3/71.17***	1.289e+3/76.42

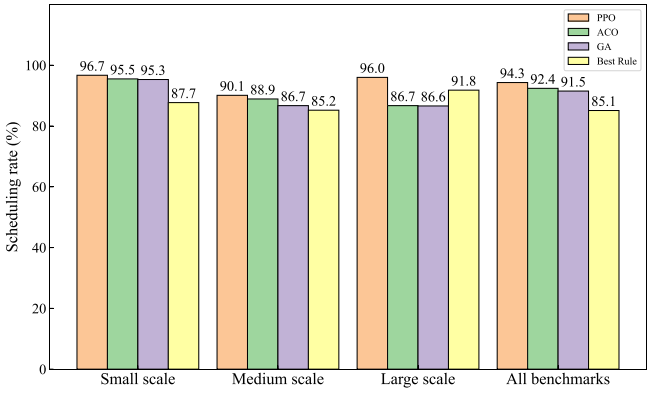


Fig. 9. Comparison of optimization rates among all methods at different scales.

design and selection of scheduling rules for achieving optimal results. The proposed PPO algorithm effectively addresses this issue by dynamically selecting the appropriate HSR based on the decision point's state and gradually optimizing scheduling objectives. This validation confirms the feasibility and effectiveness of utilizing the PPO algorithm, based on deep reinforcement learning, for solving the flexible job shop scheduling problem, as presented in this paper.

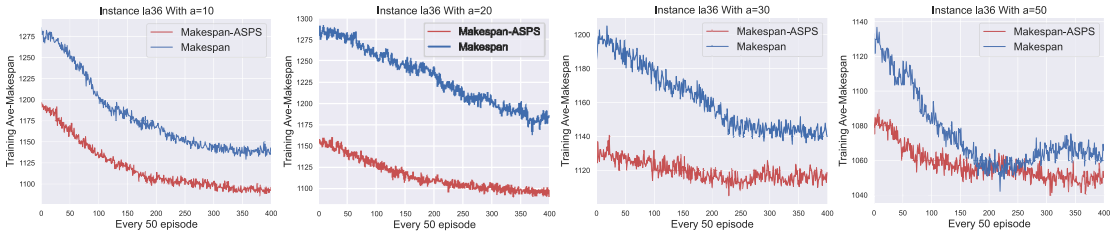
4.3. Comparison of results on dynamic experiments

In this experiment, we first employed the action space prefetching strategy (ASPS), proposed in Stage 1, to extract a set of potentially advantageous actions based on scheduling experience. These actions were then utilized as the action space for the agent. The effectiveness of this strategy was subsequently analyzed through comparative experiments. Subsequently, the agent was trained using the PPO algorithm, and models were saved at different VPT levels (specifically, for the set $a = 10, 20, 30, 50$). Finally, the scheduling results of the trained models were compared with those obtained from hybrid scheduling rules on the test set.

4.3.1. Validation of ASPS effectiveness

Taking instance la36 (15 × 15) as an example, we generated a total of 2000 VPT-FJSP data points under different parameter (settings $a = 10, 20, 30, 50$). We conducted separate tests for the proposed 8 rules, and the results are presented in Table 9. In the table, the Random rule indicates randomly selecting one rule from all available HSRs at each decision point to schedule the operations. To visually illustrate the scheduling differences among the rules, a boxplot in Fig. 11 was provided.

Upon analyzing the data in Table 9, it is evident that the mean, standard deviation, maximum, and minimum values of the Makespan should all be minimized for optimal results. These values can be normalized using eq (23). By applying eq (24)–(27), composite scores



(a) Variable factor $a = 10$ (b) Variable factor $a = 20$ (c) Variable factor $a = 30$ (d) Variable factor $a = 50$

Fig. 10. Comparison of Makespan convergence results in different action spaces of instance la36.

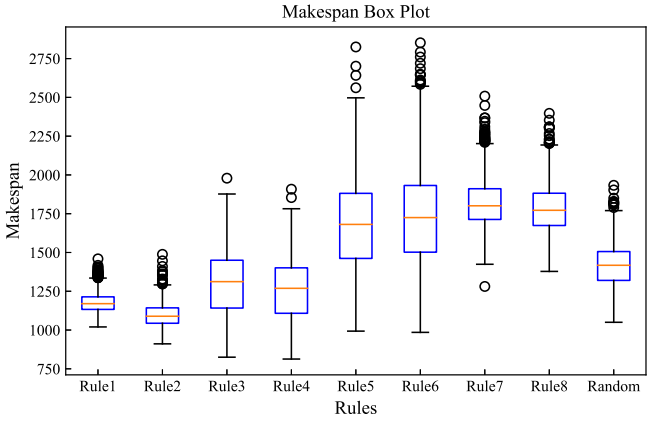


Fig. 11. Makespan under Different Single Rules.

Table 9
Performance metrics of Makespan under the single scheduling rule.

Rule	Mean	Std deviation	Minimum	Maximum
Rule1	1177	62.64	1020	1459
Rule2	1096	75.16	911	1488
Rule3	1298	203.28	825	1979
Rule4	1259	196.27	813	1908
Rule5	1678	288.53	993	2825
Rule6	1728	304.32	985	2852
Rule7	1818	149.49	1281	2508
Rule8	1783	150.84	1378	2397
Random	1417	133.45	1050	1933

were computed for each rule, resulting in values of 0.902, 0.954, 0.671, 0.712, 0.206, 0.171, 0.261, 0.265, and 0.632, respectively. Taking the composite score of 0.632 for the Random rule as the baseline for selecting agent actions, the chosen action space consists of { $Rule_1$, $Rule_2$, $Rule_3$, $Rule_4$ }.

To validate the effectiveness of the proposed Action Space Prefetching Strategy (ASPS) based on scheduling experience, this section conducts experiments on the la36 benchmark instance with different coefficients of processing time fluctuations: 10, 20, 30, and 50. The original action space of the agent is denoted as $Rule_1$ to $Rule_8$ ($Action_1$ to $Action_8$), while the action space after applying ASPS is limited to $Rule_1$ to $Rule_4$ ($Action_1$ to $Action_4$). Fig. 10 illustrates the convergence curves of the average maximum completion time during the training process. The red line represents the convergence curve of the average maximum completion time with ASPS applied, while the blue line corresponds to the curve without ASPS.

The experimental results demonstrate that as the number of training steps increases, the average maximum completion time with ASPS gradually decreases and converges to lower values. Therefore, compared to the original action space, the proposed ASPS in this chapter enables the agent to exhibit superior performance in terms of convergence speed and stability.

4.3.2. Analysis of training results

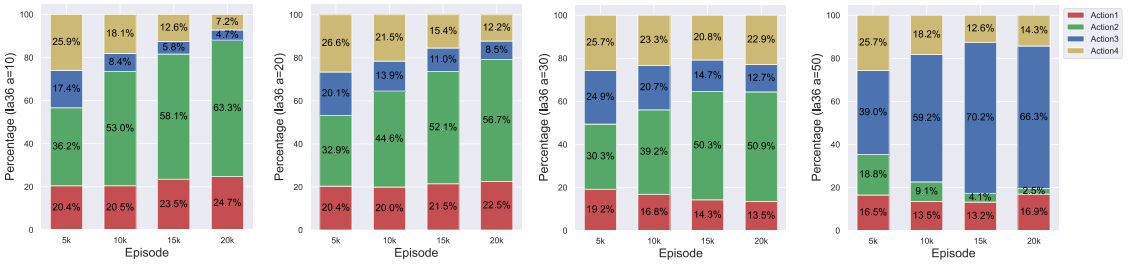
Analyzing the percentage of different actions selected by the agent during various training stages on the la36 case provides insights into the behavioral trends of the agent. Fig. 12 illustrates the percentage of action outputs during different training stages for fluctuation coefficients of 10, 20, 30, and 50. It is evident that the agent's action selection trends align with the comprehensive scores of the scheduling rules. Additionally, the agent exhibits varying biases towards specific rules at different fluctuation coefficients. For instance, the weight of action $Action_3$ steadily increases as the fluctuation coefficient increases. This observation validates the effectiveness of the proposed framework and algorithm, demonstrating that the agent can adaptively select actions based on learned environmental dynamics to solve dynamic shop scheduling problems.

For the reward function, eq (22) is employed in this chapter to normalize the reward, aiming to unify the scale of maximum makespan across different instances. Fig. 13 illustrates the convergence of average returns for cases with varying scales and processing time fluctuations, with the objective of minimizing makespan. It can be observed that under smaller fluctuation coefficients, the average return remains relatively stable and the curve appears smoother. However, when the fluctuation coefficient of processing time increases, the average returns exhibit certain instability, which aligns with the dynamics of the environment. Furthermore, it is noteworthy that the average returns for all cases eventually converge near 1.

4.3.3. Analysis of test set results

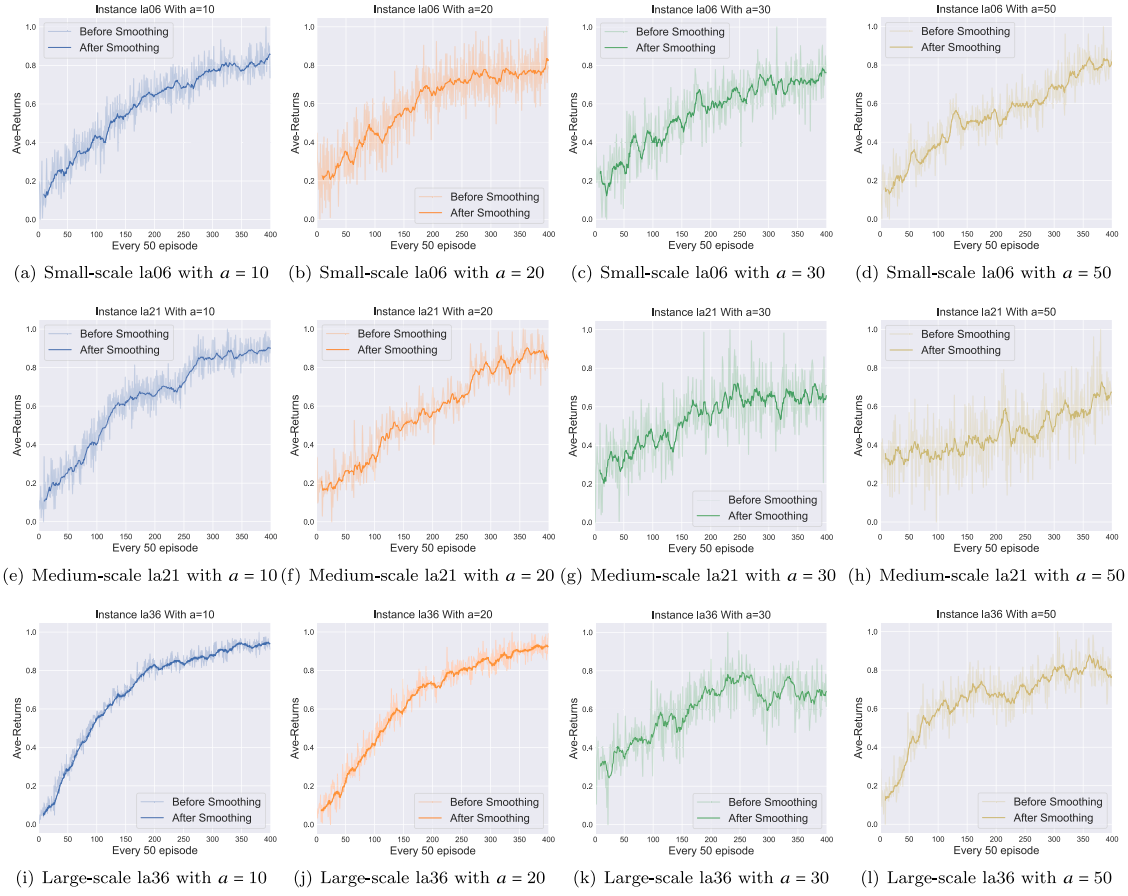
To evaluate the effectiveness and generality of the trained PPO agent, this experiment employed various standard cases with added fluctuations to generate a large training dataset and initialize the scheduling environment. The agent was trained for 20,000 episodes in each environment and subsequently subjected to 100 tests on the validation set with the same data distribution after each training. The specific Makespan test results are presented in Table 8 and Table 10. Traditional meta-heuristic algorithms, such as GA and ACO, are time-consuming and slow in response due to the unpredictability of task processing times, relying on discretization and classification methods to tackle the problem [21,45–47]. Therefore, we compared the proposed PPO method with the hybrid scheduling rules introduced in this paper and also with a random action selection strategy (labeled as Random in the table).

In Tables 8 and 10, the term "Instance" represents the standard cases used for generating training and testing data, where a is the fluctuation parameter. The symbol "/" on the left side indicates the average value over 100 tests, while on the right side it represents the standard deviation of the 100 tests. The best results are highlighted in bold font. Additionally, for each test case, a paired t-test was conducted at a significance level of 5% to determine if there is a significant difference between the best and second-best results. The null hypothesis states that, in 100 runs, the two data vectors representing the maximum completion time obtained through the two comparison methods are from populations with equal means. The superscript "*" indicates that the corresponding best result is significantly superior (i.e., the average of



(a) Variable factor $a = 10$ (b) Variable factor $a = 20$ (c) Variable factor $a = 30$ (d) Variable factor $a = 50$

Fig. 12. The percentage of agent's action outputs in different training stages for the instance la36.



(a) Small-scale la06 with $a = 10$ (b) Small-scale la06 with $a = 20$ (c) Small-scale la06 with $a = 30$ (d) Small-scale la06 with $a = 50$

(e) Medium-scale la21 with $a = 10$ (f) Medium-scale la21 with $a = 20$ (g) Medium-scale la21 with $a = 30$ (h) Medium-scale la21 with $a = 50$

(i) Large-scale la36 with $a = 10$ (j) Large-scale la36 with $a = 20$ (k) Large-scale la36 with $a = 30$ (l) Large-scale la36 with $a = 50$

Fig. 13. The average returns of different scale instances under different volatility coefficients.

the derived population is significantly lower) than all results obtained through other methods. To visualize the differences between these methods, the test results were normalized using entropy weighting to a range of 0 to 1. Fig. 14 illustrates the Makespan evaluation scores of the various methods on different standard cases, with "Other" representing the best result among all hybrid scheduling rules. The statistics reveal that PPO achieved an average of 94.65%, while "Other" only reached 87.87%.

Based on the experimental results, it can be concluded that different variations of the Variable Processing Times (VPT) significantly affect the performance of the PPO algorithm and the hybrid scheduling rules. As the fluctuation coefficient a increases, the standard deviation of Makespan also increases. In comparison to the random action selection strategy, the PPO algorithm demonstrates lower mean and variance across all test cases, indicating greater stability. Compared to a single hybrid scheduling rule, the PPO algorithm outperforms in the majority of test cases. T significant analysis reveals that when the PPO algorithm produces the best result for a particular case, it significantly outperforms the results obtained from other methods. Conversely, when other methods achieve the best result for a specific case, their performance is not significantly better than that of the PPO algorithm. Additionally, the hybrid scheduling rules $Rule_3$ and $Rule_4$ are better suited for larger-scale scenarios, while the combined rules $Rule_1$ and $Rule_2$ are more suitable for smaller-scale scenarios. In summary, the PPO agent dynamically selects the appropriate scheduling rule to allocate machines and operations based on the current production situation, making it more efficient and versatile compared to a single hybrid scheduling rule.

4.4. Generalization of the model

This experiment aims to verify the generalization capability of the model. Typically, the model's architecture and the creation of scheduling environments depend on the scale of standard instances. In order to assess the generalization capability, this section uses the trained

Table 10
The result of dynamic experiments.

Instance	a	PPO	SPT+SPTM	LPT+SPTM	SPT+LWT	LPT+LWT	Random
la01(10 × 5)	10	**0.561e+3/36.94***	0.911e+3/125.66	0.902e+3/130.4	0.838e+3/88.08	0.782e+3/95.14	0.798e+3/98.20
	20	**0.556e+3/49.59***	0.864e+3/116.33	0.878e+3/123.57	0.833e+3/94.86	0.791e+3/100.28	0.798e+3/99.88
	30	**0.562e+3/55.37***	0.821e+3/106.47	0.831e+3/126.72	0.844e+3/98.97	0.791e+3/100.73	0.762e+3/94.84
	50	**0.585e+3/59.09***	0.753e+3/96.39	0.740e+3/106.47	0.862e+3/106.94	0.792e+3/95.85	0.742e+3/106.86
la06(15 × 5)	10	**0.797e+3/55.11***	1.251e+3/158.72	1.234e+3/172.03	1.127e+3/101.12	1.038e+3/104.40	1.083e+3/114.64
	20	**0.798e+3/57.62**	1.187e+3/136.18	1.154e+3/114.55	1.148e+3/112.38	1.046e+3/107.91	1.070e+3/110.24
	30	**0.798e+3/52.30**	1.128e+3/133.70	1.105e+3/123.37	1.141e+3/104.06	1.044e+3/105.58	1.040e+3/100.82
	50	**0.805e+3/57.91***	1.041e+3/114.55	1.038e+3/132.23	1.168e+3/112.30	1.049e+3/109.98	1.007e+3/93.81
la11(20 × 5)	10	**1.040e+3/57.07**	1.492e+3/144.68	1.427e+3/142.50	1.429e+3/110.50	1.282e+3/107.87	1.342e+3/105.62
	20	**1.039e+3/51.28**	1.425e+3/126.02	1.386e+3/146.54	1.419e+3/99.79	1.264e+3/97.08	1.310e+3/95.27
	30	**1.042e+3/68.09**	1.368e+3/129.10	1.315e+3/140.54	1.435e+3/112.75	1.278e+3/104.77	1.286e+3/105.34
	50	1.034e+3/81.98	1.268e+3/132.01	1.220e+3/146.84	1.451e+3/126.15	1.287e+3/115.07	1.248e+3/121.49
la16(10 × 10)	10	**0.736e+3/54.80***	1.287e+3/153.67	1.352e+3/174.27	1.166e+3/117.44	1.148e+3/128.69	1.126e+3/116.27
	20	**0.727e+3/51.60***	1.205e+3/151.59	1.246e+3/183.53	1.180e+3/118.48	1.165e+3/135.09	1.109e+3/118.82
	30	**0.740e+3/62.08***	1.110e+3/133.85	1.169e+3/164.03	1.168e+3/121.29	1.166e+3/132.89	1.055e+3/119.51
	50	0.738e+3/70.95	0.974e+3/121.40	0.982e+3/146.06	1.192e+3/131.42	1.164e+3/128.58	1.003e+3/114.40
la21(15 × 10)	10	0.901e+3/55.25	1.612e+3/161.99	1.640e+3/191.06	1.473e+3/129.07	1.414e+3/129.16	1.395e+3/111.53
	20	**0.894e+3/49.03***	1.478e+3/158.44	1.506e+3/185.21	1.493e+3/122.73	1.418e+3/125.04	1.350e+3/112.46
	30	**0.907e+3/48.55***	1.367e+3/126.51	1.383e+3/165.29	1.482e+3/120.42	1.408e+3/129.75	1.311e+3/123.60
	50	**0.922e+3/56.78**	1.182e+3/120.24	1.173e+3/120.56	1.518e+3/123.00	1.432e+3/135.59	1.247e+3/123.11
la26(20 × 10)	10	**1.079e+3/43.79***	1.890e+3/180.82	1.867e+3/179.07	1.786e+3/127.77	1.682e+3/126.29	1.681e+3/118.96
	20	**1.068e+3/41.63***	1.742e+3/142.43	1.739e+3/167.85	1.793e+3/131.74	1.689e+3/127.05	1.617e+3/114.15
	30	**1.081e+3/45.21***	1.633e+3/137.03	1.606e+3/160.39	1.783e+3/126.07	1.685e+3/123.40	1.572e+3/121.47
	50	**1.114e+3/48.65***	1.437e+3/132.60	1.391e+3/132.13	1.821e+3/143.24	1.707e+3/133.61	1.488e+3/128.64
la31(30 × 10)	10	1.519e+3/50.22	2.579e+3/190.39	2.462e+3/192.91	2.397e+3/132.63	2.203e+3/131.06	2.202e+3/126.30
	20	**1.520e+3/51.42***	2.337e+3/177.32	2.237e+3/172.42	2.386e+3/134.57	2.210e+3/127.80	2.146e+3/126.63
	30	**1.516e+3/53.21***	2.142e+3/155.20	2.058e+3/159.02	2.404e+3/152.36	2.197e+3/137.84	2.075e+3/118.77
	50	1.471e+3/82.49	1.855e+3/142.38	1.764e+3/143.92	2.445e+3/159.46	2.222e+3/133.34	1.956e+3/128.54
la36(15 × 15)	10	**1.029e+3/53.13***	1.982e+3/178.84	2.018e+3/206.74	1.804e+3/139.84	1.783e+3/142.44	1.724e+3/143.58
	20	**1.038e+3/51.39***	1.804e+3/163.86	1.831e+3/198.63	1.831e+3/155.47	1.776e+3/144.40	1.672e+3/135.67
	30	**1.044e+3/52.02***	1.623e+3/148.43	1.686e+3/202.10	1.823e+3/145.48	1.784e+3/153.24	1.613e+3/144.83
	50	**1.011e+3/67.66**	1.329e+3/130.02	1.389e+3/151.00	1.840e+3/160.22	1.801e+3/147.62	1.488e+3/126.34
mt06(6 × 6)	10	**0.479e+3/57.04***	0.733e+3/127.12	0.752e+3/131.39	0.649e+3/98.69	0.665e+3/107.10	0.647e+3/101.60
	20	**0.477e+3/58.13***	0.697e+3/102.37	0.736e+3/126.71	0.655e+3/91.10	0.672e+3/119.38	0.651e+3/99.40
	30	**0.489e+3/67.21***	0.668e+3/112.72	0.687e+3/115.97	0.655e+3/101.99	0.669e+3/112.87	0.620e+3/96.75
	50	**0.493e+3/65.36**	0.610e+3/106.70	0.624e+3/115.06	0.678e+3/101.97	0.683e+3/118.35	0.600e+3/98.65
abz7(20 × 15)	10	**1.237e+3/44.68***	2.326e+3/212.66	2.298e+3/212.49	2.133e+3/138.30	2.049e+3/145.94	2.008e+3/140.94
	20	**1.241e+3/40.21**	2.066e+3/181.96	2.077e+3/177.96	2.128e+3/162.46	2.062e+3/156.99	1.941e+3/141.63
	30	**1.252e+3/46.86***	1.885e+3/158.33	1.931e+3/173.99	2.134e+3/151.59	2.096e+3/157.17	1.880e+3/138.70
	50	1.273e+3/66.41	1.556e+3/129.14	1.564e+3/135.59	2.179e+3/171.00	2.111e+3/158.96	1.745e+3/127.81

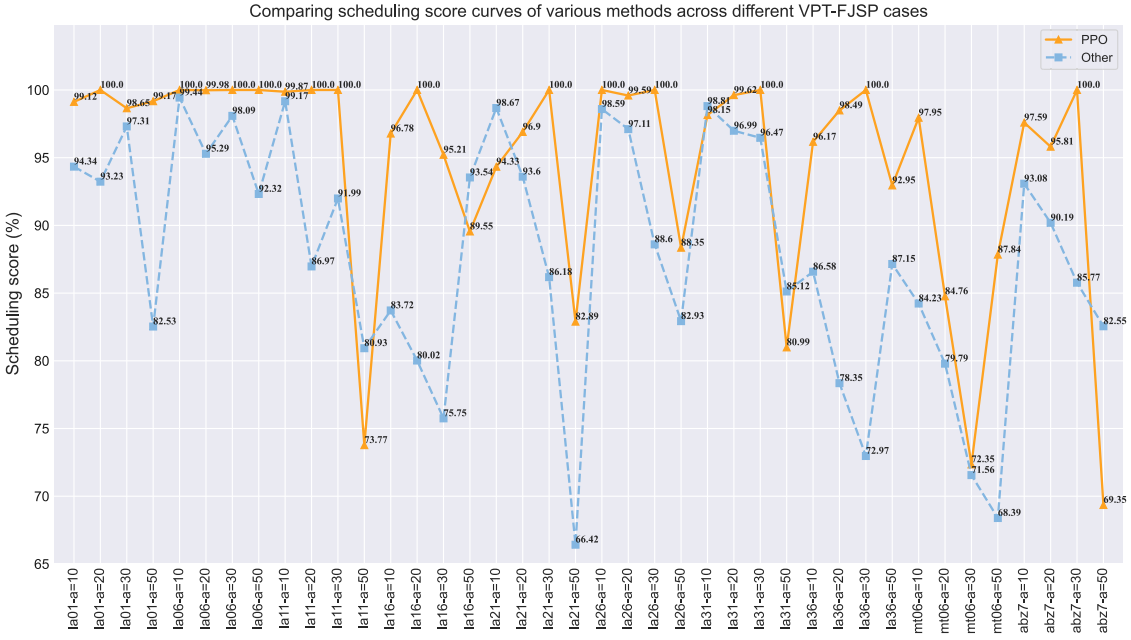


Fig. 14. The test scores of various methods on different VPT-FJSP cases.

Table 11
The result of generalization experiments.

Instance	a	$Rule_n$	Random	PPO	$Rule_n$	Random	PPO
	10	0.581e+3/42.70	0.666e+3/69.13	**0.577e+3/42.26***	0.577e+3/51.08	0.675e+3/78.90	**0.571e+3/50.07***
la02(10 × 5)	20	0.591e+3/49.41	0.645e+3/77.83	**0.583e+3/47.11***	0.572e+3/49.32	0.648e+3/76.16	**0.565e+3/47.99***
la03(10 × 5)	30	0.589e+3/61.64	0.622e+3/76.30	**0.580e+3/54.02***	0.583e+3/50.40	0.644e+3/76.19	**0.574e+3/48.93***
	50	0.596e+3/63.28	0.598e+3/70.41	**0.583e+3/58.03***	0.604e+3/68.06	0.623e+3/78.91	**0.584e+3/62.00***
	10	0.580e+3/61.47	0.666e+3/70.13	**0.572e+3/55.99***	0.598e+3/60.71	0.676e+3/79.51	**0.584e+3/56.85***
la04(10 × 5)	20	0.587e+3/52.04	0.672e+3/84.87	**0.579e+3/49.37***	0.599e+3/61.00	0.660e+3/72.53	**0.587e+3/48.76***
la05(10 × 5)	30	0.585e+3/53.48	0.647e+3/77.31	**0.576e+3/50.56***	0.606e+3/64.45	0.651e+3/74.25	**0.585e+3/52.23***
	50	0.612e+3/67.63	0.637e+3/85.94	**0.597e+3/61.38***	0.613e+3/72.25	0.647e+3/82.38	**0.596e+3/62.40***
	10	0.810e+3/54.00	0.932e+3/91.72	**0.799e+3/48.30***	0.802e+3/48.73	0.895e+3/75.54	**0.799e+3/49.09***
la07(15 × 5)	20	0.819e+3/64.51	0.922e+3/91.39	**0.807e+3/54.56***	**0.795e+3/60.43**	0.864e+3/86.35	0.793e+3/62.41
la08(15 × 5)	30	0.833e+3/60.41	0.921e+3/100.49	**0.813e+3/53.96***	0.807e+3/53.66	0.858e+3/81.55	**0.803e+3/53.01**
	50	0.840e+3/63.43	0.880e+3/96.79	**0.823e+3/60.73***	0.811e+3/68.18	0.831e+3/90.32	**0.803e+3/70.10***
	10	0.796e+3/46.48	0.882e+3/68.06	**0.792e+3/46.02***	0.793e+3/49.21	0.897e+3/75.53	**0.788e+3/46.78***
la09(15 × 5)	20	**0.790e+3/47.01**	0.870e+3/76.34	0.788e+3/47.61	0.796e+3/52.61	0.863e+3/77.22	**0.789e+3/50.35***
la10(15 × 5)	30	0.790e+3/52.48	0.849e+3/77.82	**0.785e+3/50.10***	0.807e+3/53.71	0.862e+3/81.65	**0.799e+3/47.03***
	50	0.804e+3/61.23	0.822e+3/87.63	**0.799e+3/59.44***	**0.813e+3/57.89**	0.815e+3/78.86	0.806e+3/59.47
	10	1.034e+3/57.18	1.117e+3/78.51	**1.031e+3/54.47**	1.029e+3/59.37	1.116e+3/84.69	**1.026e+3/57.99**
la12(20 × 5)	20	1.029e+3/55.53	1.106e+3/86.08	**1.028+3/55.80***	1.028e+3/61.73	1.094e+3/87.34	**1.026e+3/61.65**
la13(20 × 5)	30	1.039e+3/55.98	1.081e+3/74.15	**1.040e+3/55.13**	1.047e+3/66.73	1.087e+3/88.20	**1.047e+3/67.83**
	50	**1.039e+3/71.94**	1.046e+3/79.45	1.028e+3/74.63	1.043e+3/83.30	1.053e+3/94.88	**1.036e+3/69.00**
	10	1.031e+3/60.61	1.120e+3/79.28	**1.029e+3/60.97**	**1.035e+3/59.40***	1.111e+3/97.22	1.028e+3/60.65
la14(20 × 5)	20	**1.034e+3/61.26**	1.090e+3/79.29	1.033+3/61.52	1.038e+3/52.54	1.087e+3/75.70	**1.037e+3/52.11**
la15(20 × 5)	30	**1.041e+3/64.37***	1.067e+3/81.38	1.047e+3/68.17	**1.035e+3/54.35**	1.058e+3/75.36	1.031e+3/58.42
	50	1.035e+3/69.05	1.023e+3/79.56	**1.031e+3/72.16**	**1.041e+3/67.00**	1.031e+3/94.83	1.031e+3/83.82
	10	0.745e+3/52.64	0.875e+3/72.56	**0.724e+3/49.36***	0.769e+3/45.14	0.884e+3/75.34	**0.730e+3/54.68***
la17(10 × 10)	20	0.745e+3/53.54	0.849e+3/76.36	**0.719e+3/54.29***	0.781e+3/52.06	0.854e+3/75.31	**0.724e+3/53.60***
la18(10 × 10)	30	0.767e+3/70.47	0.818e+3/71.43	**0.742e+3/67.43***	0.797e+3/50.50	0.819e+3/71.26	**0.744e+3/59.80***
	50	**0.720e+3/68.77**	0.779e+3/70.81	0.734e+3/70.15	0.726e+3/74.40	0.787e+3/77.66	**0.732e+3/66.82**
	10	0.749e+3/59.84	0.887e+3/92.60	**0.725e+3/58.15***	0.747e+3/67.01	0.887e+3/82.08	**0.730e+3/58.24***
la19(10 × 10)	20	0.746e+3/58.43	0.845e+3/72.17	**0.726e+3/54.67***	0.734e+3/49.41	0.849e+3/71.04	**0.712e+3/38.73***
la20(10 × 10)	30	0.761e+3/63.62	0.809e+3/69.79	**0.734e+3/62.41***	0.750e+3/67.25	0.814e+3/62.57	**0.720e+3/58.10***
	50	**0.721e+3/72.59***	0.786e+3/71.09	0.736e+3/72.49	**0.716e+3/78.12***	0.770e+3/68.73	0.729e+3/73.66

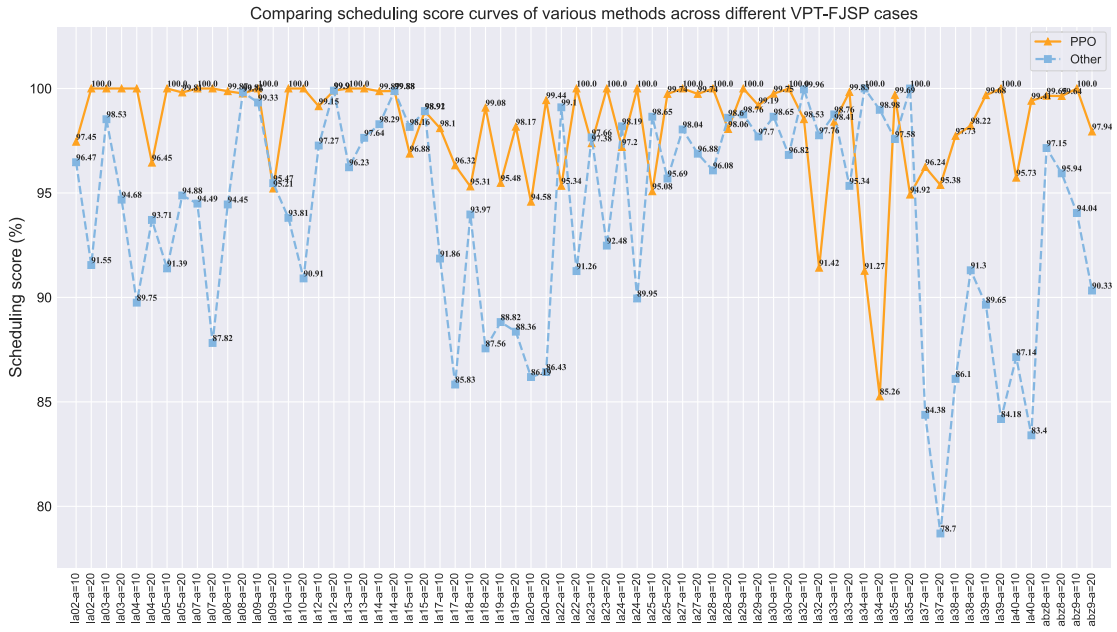


Fig. 15. Generalization ability scores of various methods on VPT-FJSP cases with fluctuation coefficients of 10 and 20.

models from Table 8 to test against instances of the same scale as listed in Tables 11 and 12. The symbol "*" indicates that the best result is significantly superior to all other results obtained through other methods, based on a paired t-test conducted at a significance level of 5%. Among the four hybrid scheduling rules, the one denoted by "$Rule_n$" represents the best-performing result, "PPO" represents the generalized outcome of the trained model, and "Random" represents the result obtained by randomly selecting a hybrid scheduling rule from the action space at decision points. The best result from the testing will be highlighted in bold font to emphasize its relative prominence.

To visualize the differences among the methods presented in Tables 11 and 12, this study normalizes the generalized results using

Table 12
The result of generalization experiments.

Instance	a	Rule$_n$	Random	PPO	Rule$_n$	Random	PPO
	10	**0.926e+3/47.47***	1.095e+3/81.72	0.915e+3/55.84	**0.913e+3/38.63**	1.085e+3/76.87	0.906e+3/44.76
la22(15 × 10)	20	0.934e+3/52.97	1.058e+3/78.54	**0.910e+3/46.33***	0.934e+3/54.77	1.052e+3/74.43	**0.915e+3/51.51***
la23(15 × 10)	30	0.948e+3/58.53	1.008e+3/78.14	**0.920e+3/46.99***	0.932e+3/45.73	1.017e+3/82.23	**0.907e+3/45.76***
	50	0.915e+3/78.98	0.955e+3/64.28	**0.931e+3/54.00***	0.904e+3/84.22	0.958e+3/72.39	**0.921e+3/59.39***
	10	**0.911e+3/40.19**	1.087e+3/81.45	0.905e+3/47.95	**0.897e+3/41.37**	1.083e+3/81.84	0.902e+3/52.74
la24(15 × 10)	20	0.908e+3/43.32	1.038e+3/76.28	**0.889e+3/41.62***	0.915e+3/40.85	1.034e+3/88.06	**0.894e+3/41.62***
la25(15 × 10)	30	0.943e+3/59.37	0.997e+3/68.36	**0.908e+3/48.40***	0.932e+3/50.71	1.005e+3/60.90	**0.908e+3/46.34***
	50	0.896e+3/80.38	0.944e+3/75.85	**0.924e+3/68.96***	0.913e+3/81.40	0.951e+3/69.80	**0.931e+3/60.22***
	10	1.093e+3/54.56	1.304e+3/93.89	**1.081e+3/51.79***	1.084e+3/45.51	1.296e+3/83.30	**1.072e+3/41.37***
la27(20 × 10)	20	1.094e+3/45.97	1.245e+3/82.13	**1.074e+3/46.80***	**1.092e+3/54.20***	1.252e+3/88.35	1.081e+3/58.55
la28(20 × 10)	30	1.110e+3/50.86	1.200e+3/83.34	**1.067e+3/48.17***	1.107e+3/51.26	1.197e+3/70.88	**1.084e+3/51.24***
	50	1.131e+3/63.30	1.140e+3/84.48	**1.108e+3/67.21***	1.136e+3/57.25	1.137e+3/76.22	**1.106e+3/54.32***
	10	1.087e+3/38.41	1.306e+3/82.57	**1.075e+3/37.18***	1.086e+3/40.00	1.293e+3/89.44	**1.074e+3/41.16***
la29(20 × 10)	20	1.095e+3/48.11	1.277e+3/93.07	**1.081e+3/48.75***	1.086e+3/48.28	1.253e+3/76.77	**1.068e+3/47.67***
la30(20 × 10)	30	1.103e+3/56.85	1.203e+3/81.76	**1.083e+3/54.33***	1.098e+3/47.05	1.213e+3/86.22	**1.078e+3/49.99***
	50	1.144e+3/57.58	1.156e+3/81.14	**1.124e+3/60.76***	1.141e+3/60.78	1.133e+3/84.15	**1.116e+3/60.92***
	10	**1.529e+3/51.02**	1.717e+3/98.73	1.535e+3/52.93	1.523e+3/52.21	1.710e+3/98.53	**1.530e+3/56.75**
la32(30 × 10)	20	**1.531e+3/47.66***	1.635e+3/84.39	1.544e+3/61.60	1.524e+3/61.96	1.643e+3/89.80	**1.526e+3/55.38**
la33(30 × 10)	30	1.536e+3/51.78	1.597e+3/85.07	**1.524e+3/53.99***	1.531e+3/50.83	1.580e+3/74.09	**1.510e+3/50.62***
	50	1.540e+3/63.11	1.476e+3/83.89	**1.471e+3/74.03**	**1.538e+3/56.67***	1.481e+3/81.14	1.500e+3/72.27
	10	**1.527e+3/56.21***	1.716e+3/100.79	1.571e+3/63.15	1.525e+3/51.38	1.699e+3/79.66	**1.531e+3/47.74**
la34(30 × 10)	20	**1.526e+3/53.11***	1.639e+3/80.40	1.587e+3/68.65	**1.537e+3/49.46**	1.635e+3/81.26	1.537e+3/52.97
la35(30 × 10)	30	**1.532e+3/55.10***	1.576e+3/83.66	1.552e+3/65.78	1.527e+3/50.84	1.584e+3/75.55	**1.518e+3/58.54***
	50	1.549e+3/62.27	**1.481e+3/76.84**	1.495e+3/77.10	1.545e+3/59.02	**1.473e+3/72.27***	1.552e+3/98.98
	10	1.142e+3/41.56	1.322e+3/77.34	**1.033e+3/53.54***	1.082e+3/68.24	1.307e+3/80.93	**1.039e+3/55.44***
la37(15 × 15)	20	1.085e+3/63.51	1.260e+3/93.31	**1.043e+3/57.41***	1.078e+3/54.20	1.257e+3/87.53	**1.036e+3/57.85***
la38(15 × 15)	30	1.087e+3/66.45	1.204e+3/82.27	**1.047e+3/55.84***	1.112e+3/69.58	1.213e+3/79.84	**1.067e+3/59.43***
	50	1.019e+3/71.63	1.136e+3/81.97	**1.008e+3/69.00**	1.031e+3/75.97	1.125e+3/76.16	**1.014e+3/73.30***
	10	1.082e+3/66.08	1.325e+3/86.54	**1.083e+3/50.98***	1.083e+3/68.88	1.315e+3/90.99	**1.049e+3/61.22***
la37(15 × 15)	20	1.084e+3/61.88	1.257e+3/74.59	**1.038e+3/48.83***	1.152e+3/56.48	1.256e+3/84.71	**1.052e+3/57.54***
la38(15 × 15)	30	1.094e+3/72.78	1.202e+3/73.14	**1.054e+3/64.18***	1.104e+3/73.64	1.208e+3/73.80	**1.060e+3/67.60***
	50	**0.996e+3/67.28***	1.118e+3/70.84	1.018e+3/71.87	1.010e+3/70.99	1.134e+3/81.44	**1.008e+3/66.01**
	10	1.251e+3/41.53	1.507e+3/81.33	**1.224e+3/42.82***	1.259e+3/45.52	1.512e+3/78.10	**1.229e+3/39.51***
abz8(20 × 15)	20	1.262e+3/43.83	1.431e+3/73.21	**1.232e+3/44.79***	1.270e+3/52.14	1.435e+3/76.25	**1.242e+3/47.60***
abz9(20 × 15)	30	1.266e+3/67.13	1.382e+3/72.59	**1.255e+3/44.58**	1.289e+3/50.92	1.370e+3/83.53	**1.254e+3/46.87***
	50	**1.168e+3/77.60***	1.275e+3/72.12	1.272e+3/77.60	**1.183e+3/84.01***	1.285e+3/71.88	1.268e+3/77.00

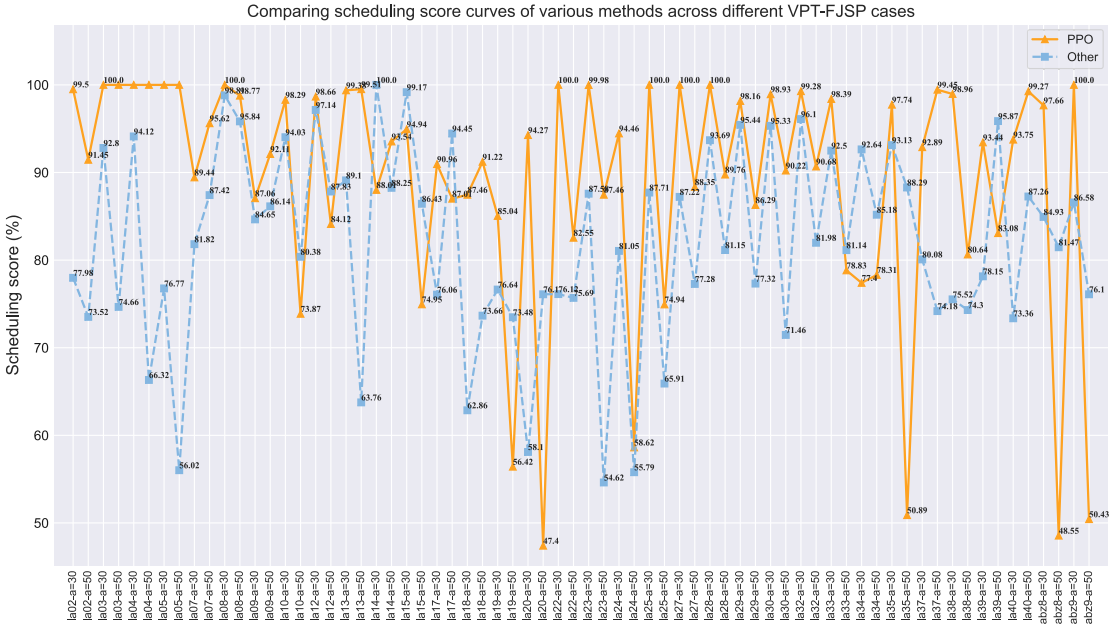


Fig. 16. Generalization ability scores of various methods on VPT-FJSP cases with fluctuation coefficients of 30 and 50.

entropy weighting, ranging from 0 to 1. Figs. 15 and 16 depict the generalized performance scores of each method on VPT-FJSP with different fluctuation coefficients.

Based on the experimental results, it can be concluded that the models trained in random environments exhibit lower average and standard deviation of Makespan, compared to $Rule_n$ and Random, in most cases of the same scale. Therefore, the models demonstrate a certain level of generalization, allowing them to effectively extend to scheduling cases of the same scale. Moreover, they outperform individual hybrid scheduling rules, saving retraining time for specific cases and improving the utilization of the models.

5. Conclusion

Previous research on dynamic scheduling has mainly focused on machine failures, job insertions, and delay time, while neglecting the impact of variability in processing times (VPT). This paper introduces a DFJSP model that considers the variability of processing time. Additionally, a two-stage dynamic scheduling framework based on scheduling experience is proposed. In Stage 1, an action space prefetching strategy is employed using entropy weighting to expedite the convergence speed of the agent. The effectiveness of the action prefetching strategy is analyzed during the experimental stage. In Stage 2, a deep reinforcement learning algorithm is utilized to train the agent, save the network model and weights, and test the generalization capability of the model during the experimental stage. In the offline training phase of Stage 2, an effective workshop scheduling environment is established, including six scheduling states, eight hybrid scheduling rules, and appropriate reward functions.

Subsequently, the PPO algorithm is applied to the proposed framework in static experiments, involving testing the algorithm using 16 different-sized FJSP instances. Under the same hyperparameters, the results of PPO are compared with those of GA and ACO. The results indicate that PPO outperforms GA and ACO in most cases, and achieves a maximum optimization rate of 94.29% in all sizes and scenarios, surpassing individual hybrid scheduling rules. In dynamic scheduling experiments, the agent is trained and tested under 10 different levels of VPT across various scales, and independent t-tests are conducted for different methods. The results demonstrate that the proposed PPO algorithm outperforms single scheduling rules in most cases and exhibits strong generalization capabilities for certain novel scenarios.

In conclusion, the proposed PPO algorithm exhibits significant advantages in addressing VPT-FJSP and provides valuable insights for users interested in utilizing reinforcement learning for scheduling problems. Future research will explore additional optimization objectives and different dynamic events.

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 research work is supported by the National Key Research and Development Project, China under Grant 2018YFB1700500 and Natural Science Foundation of Guangdong Province, China 2020A1515110541.

References

[1] Wang X, Zhang L, Liu Y, Li F, Chen Z, Zhao C, et al. Dynamic scheduling of tasks in cloud manufacturing with multi-agent reinforcement learning. J Manuf Syst 2022;65:130–45.
[2] Zhang W, Zheng Y, Ahmad R. An energy-efficient multi-objective scheduling for flexible job-shop-type remanufacturing system. J Manuf Syst 2023;66:211–32.
[3] Yang D, Wu M, Li D, Xu Y, Zhou X, Yang Z. Dynamic opposite learning enhanced dragonfly algorithm for solving large-scale flexible job shop scheduling problem. Knowl-Based Syst 2022;238:107815.
[4] Baykasoğlu A, Madenoğlu FS, Hamzadayı A. Greedy randomized adaptive search for dynamic flexible job-shop scheduling. J Manuf Syst 2020;56:425–51.
[5] Letafat A, Rafiei M, Ardeshiri M, Sheikh M, Banaei M, Boudjadar J, et al. An efficient and cost-effective power scheduling in zero-emission ferry ships. Complexity 2020;2020.
[6] Gao KZ, Suganthan PN, Chua TJ, Chong CS, Cai TX, Pan QK. A two-stage artificial bee colony algorithm scheduling flexible job-shop scheduling problem with new job insertion. Expert Syst Appl 2015;42(21):7652–63.
[7] Zhang G, Sun J, Liu X, Wang G, Yang Y. Solving flexible job shop scheduling problems with transportation time based on improved genetic algorithm. Math Biosci Eng 2019;16(3):1334–47.
[8] Destouet C, Tlahig H, Bettayeb B, Mazari B. Flexible job shop scheduling problem under industry 5.0: A survey on human reintegration, environmental consideration and resilience improvement. J Manuf Syst 2023;67:155–73.
[9] Xiao Q, Yang Z, Zhang Y, Zheng P. Adaptive optimal process control with actor-critic design for energy-efficient batch machining subject to time-varying tool wear. J Manuf Syst 2023;67:80–96.
[10] Garey MR, Johnson DS, Sethi R. The complexity of flowshop and jobshop scheduling. Math Oper Res 1976;1(2):117–29.
[11] Gao KZ, Suganthan PN, Pan QK, Tasgetiren MF, Sadollah A. Artificial bee colony algorithm for scheduling and rescheduling fuzzy flexible job shop problem with new job insertion. Knowl -Based Syst 2016;109:1–16.
[12] Li X, Gao L. An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem. Int J Prod Econ 2016;174:93–110.
[13] Pan Q-K, Gao L, Wang L, Liang J, Li X-Y. Effective heuristics and metaheuristics to minimize total flowtime for the distributed permutation flowshop problem. Expert Syst Appl 2019;124:309–24.
[14] Miyata HH, Nagano MS. Optimizing distributed no-wait flow shop scheduling problem with setup times and maintenance operations via iterated greedy algorithm. J Manuf Syst 2021;61:592–612.
[15] Vieira GE, Herrmann JW, Lin E. Rescheduling manufacturing systems: a framework of strategies, policies, and methods. J Schedul 2003;6(1):39–62.
[16] Aytug H, Lawley MA, McKay K, Mohan S, Uzsoy R. Executing production schedules in the face of uncertainties: A review and some future directions. European J Oper Res 2005;161(1):86–110.
[17] Potts CN, Strusevich VA. Fifty years of scheduling: a survey of milestones. J Oper Res Soc 2009;60(1):S41–68.
[18] Nie L, Gao L, Li P, Li X. A GEP-based reactive scheduling policies constructing approach for dynamic flexible job shop scheduling problem with job release dates. J Intell Manuf 2013;24(4):763–74.
[19] Shen X-N, Yao X. Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems. Inform Sci 2015;298:198–224.
[20] Ning T, Huang M, Liang X, Jin H. A novel dynamic scheduling strategy for solving flexible job-shop problems. J Ambient Intell Human Comput 2016;7(5):721–9.
[21] Shahgholi Zadeh M, Katebi Y, Doniavi A. A heuristic model for dynamic flexible job shop scheduling problem considering variable processing times. Int J Prod Res 2019;57(10):3020–35.
[22] Al-Hinai N, ElMekkawy TY. Robust and stable flexible job shop scheduling with random machine breakdowns using a hybrid genetic algorithm. Int J Prod Econ 2011;132(2):279–91.
[23] Buddala R, Mahapatra SS. Two-stage teaching-learning-based optimization method for flexible job-shop scheduling under machine breakdown. Int J Adv Manuf Technol 2019;100(5):1419–32.
[24] Li W, Luo X, Xue D, Tu Y. A heuristic for adaptive production scheduling and control in flow shop production. Int J Prod Res 2011;49(11):3151–70.
[25] Ye H, Wang X, Liu K. Adaptive preventive maintenance for flow shop scheduling with resumable processing. IEEE Trans Autom Sci Eng 2020;18(1):106–13.
[26] Durasevic M, Jakobovic D. A survey of dispatching rules for the dynamic unrelated machines environment. Expert Syst Appl 2018;113:555–69.
[27] Zhang L, Hu Y, Wang C, Tang Q, Li X. Effective dispatching rules mining based on near-optimal schedules in intelligent job shop environment. J Manuf Syst 2022;63:424–38.
[28] Lawrence SR, Sewell EC. Heuristic, optimal, static, and dynamic schedules when processing times are uncertain. J Oper Manage 1997;15(1):71–82.
[29] Gabel T, Riedmiller M. Adaptive reactive job-shop scheduling with reinforcement learning agents. Int J Inf Technol Intell Comput 2008;24(4):14–8.
[30] Aydin ME, Öztemel E. Dynamic job-shop scheduling using reinforcement learning agents. Robot Auton Syst 2000;33(2–3):169–78.
[31] Shahrabi J, Adibi MA, Mahootchi M. A reinforcement learning approach to parameter estimation in dynamic job shop scheduling. Comput Ind Eng 2017;110:75–82.
[32] Wang L, Pan Z, Wang J. A review of reinforcement learning based intelligent optimization for manufacturing scheduling. Complex Syst Model Simul 2021;1(4):257–70.

[33] Pan Z, Wang L, Wang J, Lu J. Deep reinforcement learning based optimization algorithm for permutation flow-shop scheduling. IEEE Trans Emerg Top Comput Intell 2021.
[34] Zhang C, Song W, Cao Z, Zhang J, Tan PS, Xu C. Learning to dispatch for job shop scheduling via deep reinforcement learning. 2020, arXiv preprint arXiv:2010.12367.
[35] Luo S. Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning. Appl Soft Comput 2020;91:106208.
[36] Lu C, Li X, Gao L, Liao W, Yi J. An effective multi-objective discrete virus optimization algorithm for flexible job-shop scheduling problem with controllable processing times. Comput Ind Eng 2017;104:156–74.
[37] Mnih V, Badia AP, Mirza M, Graves A, Lillicrap T, Harley T, et al. Asynchronous methods for deep reinforcement learning. In: International conference on machine learning. PMLR; 2016, p. 1928–37.
[38] Neal RM. Annealed importance sampling. Statist Comput 2001;11:125–39.
[39] Schulman J, Levine S, Abbeel P, Jordan M, Moritz P. Trust region policy optimization. In: International conference on machine learning. PMLR; 2015, p. 1889–97.
[40] Chen G, Peng Y, Zhang M. An adaptive clipping approach for proximal policy optimization. 2018, arXiv preprint arXiv:1804.06461.
[41] Balas E. Machine sequencing via disjunctive graphs: an implicit enumeration algorithm. Oper Res 1969;17(6):941–57.
[42] Wilson JM. Gantt charts: A centenary appreciation. European J Oper Res 2003;149(2):430–7.
[43] Brandimarte P. Routing and scheduling in a flexible job shop by tabu search. Ann Oper Res 1993;41(3):157–83.
[44] Hurink J, Jurisch B, Thole M. Tabu search for the job-shop scheduling problem with multi-purpose machines. Oper Res-Spektr 1994;15(4):205–15.
[45] Lim RJ, Yen B, Zhang W. Just-in-time scheduling with machining economics for single-machine turning process. J Manuf Syst 2000;19(4):219–28.
[46] Guo Z, Wong WK, Leung SY-S, Fan J, Chan S. Genetic optimization of order scheduling with multiple uncertainties. Expert Syst Appl 2008;35(4):1788–801.
[47] Gao Z, Peng J, Han Z, Jia M. Flow shop scheduling with variable processing times based on differential shuffled frog leaping algorithm. Int J Model Ident Control 2019;33(2):179–87.