Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.083100

ARTICLE

An Intelligent Algorithm for Dynamic Scheduling of Parallel Machines Considering Multi-Task Collaboration in Order Processing

Pei Xie[1], Xiaoying Yang[1,*], Bo Li[1], Zhijie Pei[1] and Fenghai Yang[2]

[1]School of Mechatronics Engineering, Henan University of Science and Technology, Luoyang, 471003, China
[2]Luoyang Bearing Research Institute Co., Ltd., Luoyang, China

*Corresponding Author: Xiaoying Yang. Email: lyyxy111@163.com

Received: 29 March 2026; Accepted: 28 May 2026; Published: 23 July 2026

ABSTRACT: To address the critical requirements for collaborative delivery of multiple tasks within each order in personalized mass customization, this paper develops a dynamic parallel machine scheduling model that accounts for stochastic machine failures and order priorities, thereby more accurately reflecting the uncertainties and complexities of real-world production environments. A dual-objective optimization framework is adopted to minimize both the makespan (maximum task completion time) and the variance of task completion times, aiming to improve the coordination and reliability of intra-order task delivery. An adaptive weighted reward function is designed to balance overall scheduling efficiency with consistency among tasks during reinforcement learning training. To tackle the challenges posed by partially observable Markov decision processes (POMDP) induced by unexpected machine breakdowns, a Gated Recurrent Unit (GRU)-embedded Proximal Policy Optimization (PPO) intelligent scheduling algorithm is proposed. The algorithm incorporates an Action Masking mechanism to prevent invalid scheduling actions, while the GRU module captures historical state sequences to enhance perception of dynamic production environments. Extensive validation on benchmark datasets, along with comparisons against traditional heuristic algorithms, metaheuristic algorithms, and other deep reinforcement learning methods, demonstrates that the proposed approach achieves robust convergence, high resilience, and strong generalization across both static and dynamic scenarios, significantly improving coordinated delivery performance of order tasks. Overall, the proposed method not only provides an efficient and scalable real-time decision-making solution for Parallel Machine Scheduling Problems (PMSP) but also offers new theoretical and practical insights for optimizing complex production scheduling in intelligent manufacturing systems.

KEYWORDS: Parallel machine scheduling problems; dynamic scheduling; gated recurrent unit; proximal policy optimization; coordinated delivery

1 Introduction

With the rapid development of Industry 4.0 and intelligent manufacturing, manufacturing enterprises are gradually shifting from traditional mass production toward more flexible Make-to-Order (MTO) production modes. In MTO environments, production activities must be organized according to customers' personalized requirements, where different orders often involve heterogeneous process routes, delivery deadlines, and resource demands. As a critical decision-making process that bridges customer orders and manufacturing resources, production scheduling directly affects machine utilization, on-time delivery performance, and overall operational efficiency. In discrete manufacturing systems, multiple tasks from different orders are typically processed on parallel machines. When a customer order contains multiple subtasks, the order can only be delivered after all subtasks are completed. Excessive differences in the

completion times of these subtasks may not only increase work-in-process (WIP) inventory costs, but also reduce customer satisfaction. For example, in bearing testing scenarios, multiple samples belonging to the same order must be completed in a coordinated manner to achieve synchronized delivery. These samples may be processed simultaneously on different testing machines and are constrained by machine availability and testing sequences. Early completion or delays may increase WIP inventory and management costs [1], and further create a "bucket effect," where the completion delay of one task postpones the delivery of the entire order. Similarly, in wind turbine beam plate manufacturing, different components within the same order must be completed collaboratively across multiple parallel machines. Such scheduling problems may increase buffer occupancy and even lead to downstream assembly line stagnation [2]. Therefore, achieving efficient parallel machine scheduling while ensuring coordinated order delivery is of significant theoretical importance and practical value.

However, real-world production environments are highly dynamic and uncertain. Random disruptions such as machine failures, urgent order arrivals, and fluctuations in processing times frequently disturb the original scheduling plan, making static scheduling strategies difficult to maintain effectively [3,4]. In addition, collaborative delivery among multiple tasks within the same order requires not only minimizing the overall completion time but also balancing the completion synchronization among different tasks, which further extends the scheduling objective from traditional single-objective optimization to a more complex multi-objective optimization problem. Meanwhile, as the order scale increases, the scheduling decision space grows exponentially, making it difficult for traditional exact algorithms to obtain high-quality solutions within a reasonable computational time.

For parallel machine scheduling problems (PMSP), existing scheduling methods still face significant challenges in dynamic manufacturing environments. Traditional optimization approaches typically rely on offline solution mechanisms and require frequent rescheduling when random disruptions such as machine failures occur, resulting in limited real-time responsiveness. Although deep reinforcement learning has gradually been applied to scheduling problems in recent years, it still suffers from several limitations in modeling collaborative order delivery, capturing historical state information, and handling invalid action constraints.

To address the above issues, this study develops a dynamic parallel machine scheduling model that aims to minimize both the maximum completion time of orders and the completion time variance among multiple tasks within the same order. To effectively balance these two objectives, an adaptive weighted reward function is designed to enable the agent to dynamically learn the trade-off between objectives, thereby avoiding the computational complexity of solving the Pareto frontier. Furthermore, an improved intelligent scheduling algorithm integrating temporal feature extraction and action masking is proposed. Specifically, a Gated Recurrent Unit (GRU) is embedded into the Proximal Policy Optimization (PPO) policy network to capture temporal dependencies in historical state information. In addition, an action masking mechanism is introduced to filter physically infeasible actions, effectively reducing the action space and significantly improving both policy update efficiency and decision feasibility.

The main contributions of this paper are as follows:

(1) A dynamic parallel machine scheduling model was developed by considering machine failures and order priorities, while incorporating the variance of task completion times within each order into the optimization objective.
(2) An improved scheduling method integrating GRU and PPO was developed, and an Action-Masking mechanism was incorporated to enhance decision quality and training efficiency in partially observable environments.

(3) Test instances with coordinated delivery constraints were reconstructed based on classical benchmark datasets, and the effectiveness and generalization capability of the proposed method were validated through multiple comparative experiments.

The remainder of this paper is organized as follows. Section 2 reviews the related literature, including studies on parallel machine scheduling, deep reinforcement learning-based scheduling, and optimization under complex constraints. Section 3 presents the parallel machine scheduling model with multi-task collaborative delivery constraints. Section 4 describes the proposed GRU-PPO algorithm for the PMSP. Section 5 provides the experimental design, result analysis, and comparative studies. Finally, Section 6 concludes the paper.

2 Literature Review

For PMSP and its variants, traditional solution paradigms mainly rely on exact algorithms and metaheuristic methods. Researchers primarily design sophisticated local search mechanisms or multi-resource constrained models to obtain near-optimal solutions. For example, Hu et al. [5] solved the unrelated parallel machine scheduling problem based on neighborhood search; Fan et al. [6] introduced a hybrid Jaya algorithm; Hu et al. [7] and Xie et al. [8] proposed Variable Neighborhood Search (VNS) and Whale Optimization Algorithm (WOA) frameworks, respectively. With the increasing complexity of manufacturing scenarios, more studies have begun to incorporate real-world production constraints into parallel machine scheduling models. Wang and Qi [9] studied an energy-aware parallel machine scheduling problem considering job deterioration and learning effects, optimizing makespan and energy consumption using a hybrid evolutionary approach. Subsequently, Lu et al. [10] investigated a two-stage hybrid flow shop scheduling problem in semiconductor manufacturing with sequence-dependent setup times and repetitive processing constraints, and proposed a customized variable neighborhood search algorithm combined with heuristic scheduling rules to improve makespan performance and resource utilization. These studies indicate that parallel machine scheduling is gradually evolving from traditional efficiency-oriented optimization toward real-world scenarios driven by complex industrial constraints.

In terms of multi-objective handling, Chen et al. [11] and Fan et al. [12] demonstrated notable effectiveness using a simulated annealing (SA) algorithm and a grey wolf optimizer (GWO), respectively. In addition, research on intra-order multi-task coordinated delivery has gradually increased. Qiu et al. [13] proposed a two-stage flexible assembly scheduling model for centralized delivery, highlighting that asynchronous task completion can significantly reduce system coordination efficiency, and employed reinforcement learning to alleviate synchronization waiting issues. Fu and Zhu [14] further incorporated dynamic material kitting constraints into a flexible assembly scheduling framework and verified the critical impact of kitting constraints on overall system makespan efficiency. However, the above studies still exhibit clear limitations when facing dynamic disturbances in real industrial environments. First, the computational cost is relatively high; when unexpected machine breakdowns occur, requiring rescheduling, these methods struggle to meet real-time decision-making requirements on the shop floor. Second, under multi-objective conflicts, traditional algorithms mostly rely on static and fixed weight configurations, lacking adaptability to stochastic environmental disturbances, which can easily lead to scheduling solutions becoming infeasible during execution. Third, existing research mainly focuses on assembly systems, while studies considering parallel machine environments with both stochastic failure disturbances and order-level coordinated delivery constraints remain relatively scarce.

In recent years, Deep Reinforcement Learning (DRL) has provided a new paradigm for complex dynamic combinatorial optimization problems by modeling scheduling processes as a Markov Decision Process (MDP), thereby overcoming the limitations of traditional methods that heavily rely on handcrafted

rules [15]. Paeng et al. [16] proposed a DRL-based scheduling framework for unrelated parallel machine scheduling with sequence-dependent family setups, aiming to minimize total tardiness while adapting to changes in production requirements and due dates. Subsequently, a series of advanced approaches have been proposed, including frameworks based on Double Deep Q-Networks (DDQN) [17,18], cooperative optimization with memetic algorithms [19], and high-level network architectures incorporating self-attention mechanisms [20]. Although these studies demonstrate strong real-time performance, existing DRL-based approaches for dynamic parallel machine scheduling with "coordinated delivery" constraints remain limited. On the one hand, at the objective modeling level, current DRL scheduling models rarely explicitly incorporate coordinated delivery requirements. Most studies focus on minimizing makespan [21], total tardiness [16–18], or energy consumption [2], while rarely considering the use of completion time variance to quantify task dispersion. As a result, these approaches struggle to ensure synchronized delivery of multi-task orders at the algorithmic level. On the other hand, in terms of state perception, existing methods often neglect the partially observable Markov decision process (POMDP). Most existing studies idealize the dynamic scheduling environment as a fully observable MDP [16], and frequently overlook both physical constraints in parallel machine scheduling and the problem of action space explosion.

In addition, PPO has been widely applied to dynamic scheduling problems due to its strong training stability and high sample efficiency [22]. At present, PPO and its variants have been successfully applied to dynamic rescheduling in job shop environments [23], real-time multi-objective scheduling in flexible manufacturing systems [24], and other continuous and discrete production scheduling tasks [25], demonstrating strong adaptability to dynamic environments. However, the standard PPO framework typically employs a feedforward network and has limited capability in capturing temporal dependencies under dynamic and partially observable environments. Therefore, incorporating GRU into PPO has the potential to improve historical state feature extraction and decision-making performance. Nevertheless, few studies have combined GRU with PPO for parallel machine scheduling problems with coordinated delivery constraints. Furthermore, machine availability may change over time due to adaptive maintenance activities and unplanned shutdowns, and certain scheduling or maintenance actions may become infeasible at specific decision time steps [26]. Existing research has paid limited attention to this issue and lacks effective mechanisms for handling action constraints. Table 1 presents a comparison between representative studies and the proposed study.

Table 1: Comparison between representative studies and the proposed study.

Studies	Problem Setting/Method	Main Focus	Limitations/Gaps
Traditional PMSP studies [5–8]	PMSP; neighborhood search, Jaya, VNS, WOA	Makespan, tardiness, workload balance	Mainly designed for static/offline scheduling; limited responsiveness to stochastic machine failures
Energy-aware and constraint-based PMSP studies [9,10]	Unrelated PMSP and semiconductor testing; hybrid evolutionary algorithm, improved VNS	Energy, learning effects, deterioration, and power constraints	Lack order-level coordinated delivery and dynamic failure handling

(Continued)

Table 1 (continued)

Studies	Problem Setting/Method	Main Focus	Limitations/Gaps
Multi-objective PMSP studies [11,12]	Multi-objective parallel machine scheduling; SA, GWO	Makespan, energy consumption, tardiness, workload balance	Usually rely on fixed trade-off strategies; intra-order task completion synchronization is rarely modeled
Coordinated delivery studies [13,14]	Flexible assembly scheduling	Centralized delivery and synchronization	Mainly focus on assembly systems rather than PMSP
DRL-based dynamic scheduling [16–25]	Dynamic scheduling; DQN, DDQN, PPO, DRL	Real-time scheduling and rescheduling	Limited modeling of intra-order synchronization and POMDP
Wang et al. [26]	Dynamic PMSP; DRL, action masking	Order–maintenance scheduling	Limited task; process synchronization
This study	Dynamic PMSP; GRU-PPO with action masking	Coordinated delivery, failures, task completion variance, adaptive reward	Integrates machine failures, task completion variance, GRU-PPO, and action masking

As summarized in Table 1, previous studies have made important contributions to PMSP optimization, coordinated delivery, and DRL-based dynamic scheduling. However, few studies have simultaneously considered stochastic machine failures, order-level coordinated delivery, partially observable system states, and dynamically infeasible actions. In particular, intra-order task completion variance has rarely been used to quantify coordinated delivery performance in dynamic PMSP. To address these gaps, this study develops a dynamic PMSP model considering machine failures, order priorities, and task completion variance. Furthermore, a GRU-embedded PPO algorithm with action masking is proposed to enhance historical state perception and improve scheduling feasibility.

3 Parallel Machine Scheduling Model for Multi-Task Coordination within Orders

3.1 Problem Description

This study focuses on the PMSP with order-level batch delivery constraints [27]. This problem generally involves n orders $I = \{O_1, O_2, \cdots, O_n\}$, each of which consists of m_i independent order tasks, as shown in Fig. 1, and the set of these tasks is denoted by $J_i = \{J_{i1}, J_{i2}, \ldots, J_{im_i}\}$. The production shop contains k available parallel machines, which constitute the machine set $M = \{M_1, M_2, \cdots, M_k\}$. In the parallel machine scheduling process, all order tasks are treated as single-operation jobs. The machines are selectable, and each order task J_{ij} can be processed on any machine within the subset $M_{ij} \subseteq M$ consisting of multiple eligible machines. Due to the heterogeneity of parallel machines, the processing time of the same task may vary across different machines. Therefore, during scheduling, it is necessary not only to determine the processing sequence of order tasks, but also to select an appropriate machine for each order task so as to achieve the coordinated delivery of multiple tasks within an order.

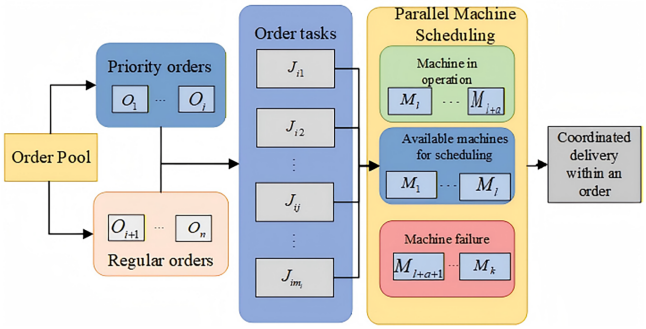

Figure 1: Dynamic parallel machine scheduling problem model.

The following basic assumptions are made:

(1) At time zero, all order tasks are available for processing, and all machines are available for operation.
(2) Each machine can process only one order task at a time.
(3) Once a task has started, it cannot be interrupted midway.
(4) All operators are assumed to have the same level of proficiency.

3.2 Mathematical Formulation of the Parallel Machine Scheduling Model

As shown in Table 2, the list of variables provides the definition of each variable used in the equations.

In practical make-to-order production environments, an order i usually consists of multiple independent order tasks $\{J_{i1}, J_{i2}, \ldots, J_{im_i}\}$. Owing to the physical attribute of coordinated delivery, the final delivery time of an order depends on the latest completion time among all its subtasks. The excessively early completion of certain tasks within the same order leads to the accumulation of work-in-process inventory, whereas delays in some tasks postpone the delivery of the entire order. Therefore, in addition to the conventional objective of minimizing the maximum completion time, as shown in Eq. (2), this model places particular emphasis on mathematically quantifying "multi-task coordination within an order." Specifically, the variance of task completion times within an order is introduced as the second optimization objective, as shown in Eq. (3). By minimizing the dispersion of the completion times of individual tasks within the same order relative to the average completion time AVG_i of that order, the agent is encouraged to schedule tasks belonging to the same order in a temporally clustered manner, thereby ensuring the coordinated delivery of multiple tasks within the order at the modeling level.

$$C_{\max} = \max_{j \in J_i} \{E_{ij}\} \quad \forall i \in I, j \in J_i \tag{1}$$

$$f_1 = \min C_{\max} \tag{2}$$

$$f_2 = \min \sum_{i \in I} \left(\frac{1}{m_i} \sum_{j \in J_i} \left(E_{ij} - AVG_i\right)^2 \right) \tag{3}$$

The average completion time of order O_i is defined as

$$AVG_i = \frac{1}{m_i} \sum_{j \in J_i} E_{ij}, \quad \forall i \in I \tag{4}$$

Table 2: Meaning of the notation.

Notation	Meaning
$I = \{O_1, O_2, \cdots, O_i\}$	Set of orders, indexed by $i \in \{1, 2, \ldots, n\}$
J_{ij}	Set of tasks belonging to order O_{ij}, indexed by $j \in \{1, 2, \ldots, m_i\}$
J	Global set of all order tasks
$M = \{M_1, M_2, \cdots, M_k\}$	Set of all machines, indexed by $l \in \{1, 2, \ldots, k\}$
$M_{ij} \subseteq M$	Set of eligible machines for task J_{ij}
P_{ijl}	Processing time of task J_{ij} on machine l
D_i	Order type indicator: $D_i = 2$ if O_i is a priority order, $D_i = 1$ if O_i is an urgent order, and $D_i = 0$ if O_i is a regular order.
m_i	Number of tasks in order O_i
x_{ijl}	Binary decision variable indicating whether task J_{ij} is processed on machine M_l
$y_{ij,i'j'l}$	Binary decision variable indicating whether task J_{ij} is processed before task $J_{i'j'}$ on machine M_l
S_{ij}	Start time of task J_{ij}
E_{ij}	Completion time of task J_{ij}
$C_{\max}$	Maximum completion time among all tasks in order i
AVG_i	Average completion time of order i

The constraints are formulated as follows:

s.t.

$$S_{ij} \geq 0, P_{ijl} \geq 0 \quad \forall i \in I, j \in J_i, l \in M \tag{5}$$

$$\sum_{l \in M_{ij}} x_{ijl} = 1 \quad \forall i \in I, j \in J_i \tag{6}$$

$$E_{ij} \leq C_{\max} \quad \forall i \in I, j \in J_i \tag{7}$$

$$E_{ij} \leq S_{i'j'} + H\left(3 - x_{ijl} - x_{i'j'l} - y_{ij,i'j'l}\right) \quad \forall l \in M, i, i' \in I, j \in J_i, j' \in J_{i'} \tag{8}$$

$$E_{i'j'} \leq S_{ij} + H\left(2 - x_{ijl} - x_{i'j'l} + y_{ij,i'j'l}\right) \quad \forall l \in M, i, i' \in I, j \in J_i, j' \in J_{i'} \tag{9}$$

$$E_{ij} \leq S_{i'j'} + H\left(2 - x_{ijl} - x_{i'j'l}\right) \quad \forall l \in M, i, i' \in I, j \in J_i, j' \in J_{i'}, D_i > D_{i'} \tag{10}$$

$$E_{ij} = S_{ij} + \sum_{l \in M} P_{ijl} \cdot x_{ijl} \quad \forall i \in I, j \in J_i \tag{11}$$

Specifically, Eq. (5) indicates that the start time and processing time of all operations are non-negative; Eq. (6) ensures that each task can be assigned to only one machine; Eq. (7) states that the completion time of every task does not exceed the maximum completion time; Eqs. (8) and (9) ensures that any two tasks assigned to the same machine cannot be processed simultaneously. The binary variable y determines the processing order between two tasks on the same machine; Eq. (10) indicates that, for all priority tasks J_{ij} and regular tasks $J_{i'j'}$, if both tasks are assigned to machine M_l (i.e., $x_{ijl} = x_{i'j'l} = 1$), the priority task must be processed before the regular task; and Eq. (11) represents the task completion time constraint, where the completion time is determined by the start time and the processing time.

4 GRU-PPO-Based Dynamic Scheduling Algorithm for Parallel Machines

4.1 Framework of the GRU-PPO Algorithm

The scheduling agent interacts with the state environment through the GRU-PPO algorithm, and its overall framework is shown in Fig. 2. The environmental state sequence s_t is fed into the GRU network as input to capture the temporal dependencies in the scheduling process. The hidden state $s_t^{'}$ output by the GRU is then passed to the policy network and mapped into the logits of all candidate actions. On this basis, the Action Masking mechanism sets the logits corresponding to currently infeasible actions to negative infinity, after which the Softmax function is applied to normalize them into the probability distribution over feasible actions. The agent samples and selects an action according to this probability distribution. After execution, the environment transitions to the next state and returns an immediate reward, while the experience tuple (s_t, a_t, r_t, s_{t+1}) is stored in the experience replay buffer. During training, a batch of experiences is randomly sampled from the buffer at fixed intervals, and the advantage function is computed using Generalized Advantage Estimation (GAE). The loss functions of the policy network and the value network are then constructed based on the clipped objective of PPO, and the network parameters are updated through gradient descent. After each update, the parameters of the old policy network are synchronized with those of the current policy network for the next round of interaction.

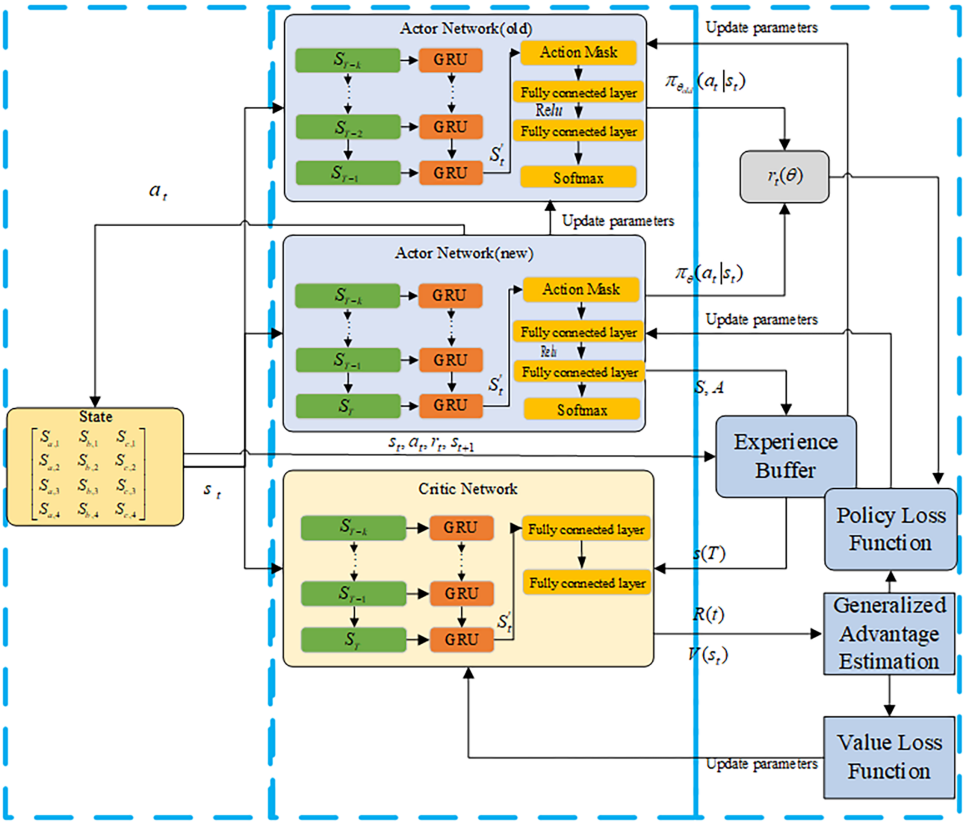

Figure 2: Framework of the GRU-PPO algorithm model.

GRU is used to encode historical state sequences and enhance policy learning under partial observability. Its core equations are as follows:

$$r_t = \sigma\left(W_r\left[h_{t-1}, x_t\right] + b_r\right) \tag{12}$$

$$\widetilde{h_t} = \tanh\left(W\left[r_t * h_{t-1}, x_t\right]\right) \tag{13}$$

$$z_t = \sigma\left(W_z\left[h_{t-1}, x_t\right]\right) \tag{14}$$

$$h_t = \left(1 - z_t\right) * h_{t-1} + z_t * \tilde{h}_t \tag{15}$$

During the training phase, GRU-PPO follows the core training framework of PPO:

The advantage function is estimated using GAE. The calculation formula is given as follows:

$$A_t^{GAE}\left(s_t, a\right) = \sum_{b=0}^{\infty} \left(\gamma\lambda\right)^b \delta_{t+b}^{v} \tag{16}$$

$$\delta_t^{v} = r_t + rV_\theta\left(s_{t+1}\right) - V_\theta\left(s_t\right) \tag{17}$$

The clipped objective function used in PPO is given in its basic form in Eq. (18):

$$L^{CLIP}\left(\theta\right) = \hat{E}_t\left[\min\left(r_t\left(\theta\right)A_t, clip\left(r_t\left(\theta\right), 1-\varepsilon, 1+\varepsilon\right)A_t\right)\right] \tag{18}$$

For the value network, the Temporal Difference (TD) error is adopted as the value loss function, as shown in Eq. (19).

$$\delta_t^{v} = r_t + rV_\theta\left(s_{t+1}\right) - V_\theta\left(s_t\right) \tag{19}$$

In this study, an action is defined as a composite dispatching rule rather than a direct job-machine assignment. Specifically, the action space is composed of job-selection rules and machine-selection rules. At each decision point, the agent selects one composite rule, which is then used to choose a candidate order task and a feasible machine. However, under dynamic disturbances such as machine failures or machine occupancy, some composite actions may become infeasible. To handle this issue, an action masking mechanism is applied at the output of the policy network. A binary mask is generated according to the current system state, and the logits of infeasible actions are set to negative infinity before Softmax normalization. In this way, the agent explores only feasible composite actions, thereby satisfying physical constraints, reducing invalid exploration, and improving training efficiency and convergence stability.

4.2 State Space Design

In deep reinforcement learning, the design of the state space is of critical importance, because, as the basis for the agent's decision-making, it can directly affect both efficiency and generalization capability. In the dynamic parallel machine scheduling problem, the state space is designed as a three-level representation $S = [S_a, S_b, S_c]$.

Here, $S_a = [S_{a,1}, S_{a,2}, S_{a,3}, S_{a,4}]$ denotes the state space associated with order-related features. $S_{a,1}$ represents the slack time of the current order, as defined in Eq. (20), where d_i denotes the due date of order i, t denotes the current decision time, and p_{avgi} denotes the average processing time of the tasks in that order up to the current time. $S_{a,2}$ denotes the order-type identifier for each order. In defining the order-type labels, the order-priority principle is taken into consideration, and three priority tiers are established. The first tier corresponds to inherently priority orders, whose priority is independent of time; these are predefined high-priority orders and are labeled as 2. The second tier corresponds to urgent orders, which are granted priority because of time urgency and are labeled as 1; the triggering condition is that the slack time is smaller than a preset threshold T (T = 2). The third tier corresponds to regular orders, which are neither inherently priority orders nor urgent orders and can therefore be processed in the normal sequence; these are labeled as 0. $S_{a,3}$

denotes the start time of the current order-task operation, recorded from the release time of the order task. If its value is 0, it indicates that the order task has not yet begun processing on any machine. $S_{a,4}$ denotes the completion time of the order-task operation. If its value is 0, it indicates that the order task has not yet begun processing on any machine.

$$T_S = d_i - t - p_{avgi} \tag{20}$$

$S_b = [S_{b,1}, S_{b,2}, S_{b,3}, S_{b,4}]$ denotes the state space associated with machine-related features. Specifically, $S_{b,1}$ represents the status of each machine: a value of 1 indicates that the machine is in operation, a value of 0 indicates that the machine is idle, and a value of 2 indicates that the machine has failed. $S_{b,2}$ denotes the cumulative operating time of the machine up to the current moment. $S_{b,3}$ denotes the type of task being processed on the machine. $S_{b,4}$ denotes the utilization rate of each machine prior to the current time, as defined in Eq. (21), where T_{act} represents the actual processing time of each machine and T_{avl} represents the cumulative available processing time of each machine up to the current moment. In addition, according to the bathtub-curve law governing machine failure rates, this study focuses on the random failure period during stable equipment operation, with the average machine failure rate set to 0.03. The failure duration is determined immediately after the occurrence of the failure, and the possible failure time in the experiments is set from 1 to 45.

$$M_{U_r} = \frac{T_{act}}{T_{avl}} \tag{21}$$

$S_c = [S_{c,1}, S_{c,2}, S_{c,3}]$ denotes the global state space. Specifically, $S_{c,1}$ represents the priority-order load. When the proportion of inherently priority orders among all orders approaches 1, the agent only needs to optimize scheduling within the priority orders; when this value is low, the number of inherently priority tasks is relatively small, and the agent can adopt a strategy that places greater emphasis on efficiency. $S_{c,2}$ denotes the average cumulative working time of all machines, which is used to balance the machine load allocation strategy. $S_{c,3}$ denotes the total tardiness of all completed orders, as shown in Eq. (22).

$$T_{totalla} = \sum_{i=1}^{I} (d_i - t) \tag{22}$$

4.3 Action Space and Masking Mechanism Design

The action space consists of four job-selection rules and four machine-selection rules, yielding 16 composite actions. The agent outputs rule combinations instead of explicit job-machine indices. However, in dynamic scheduling environments, due to machine failures, machine availability fluctuations, and task processability constraints, certain rule combinations may correspond to infeasible actions at specific decision-making time steps. If an agent performs unrestricted exploration within an action space that includes invalid actions, it not only reduces the efficiency of policy learning but may also generate infeasible scheduling solutions. Therefore, an action masking mechanism is introduced, which dynamically generates a binary mask at each decision step to proactively filter out illegal actions, thereby restricting the agent's exploration strictly within the feasible action space and improving both decision efficiency and the feasibility of scheduling solutions. An action masking mechanism is applied at each decision step, and the action space is illustrated in Fig. 3. The action space designed in this study is as follows:

In the job-selection action space, four commonly used job-selection rules are adopted:

(1) EDD: Select the order task with the earliest due date among the current orders.
(2) SPT: Select the order task with the shortest current processing time.

(3) LPT: Select the order task with the longest current processing time.
(4) SLACK: Select the order task with the minimum slack time.

In the machine-selection action space, four commonly used rules are adopted:

(1) LUR: Select the machine with the lowest current utilization rate.
(2) LB: Select the machine with the lightest current workload.
(3) ECT: Select the machine that yields the earliest completion time for the order task.
(4) WINQ: Select the machine with the shortest workload in the subsequent queue.

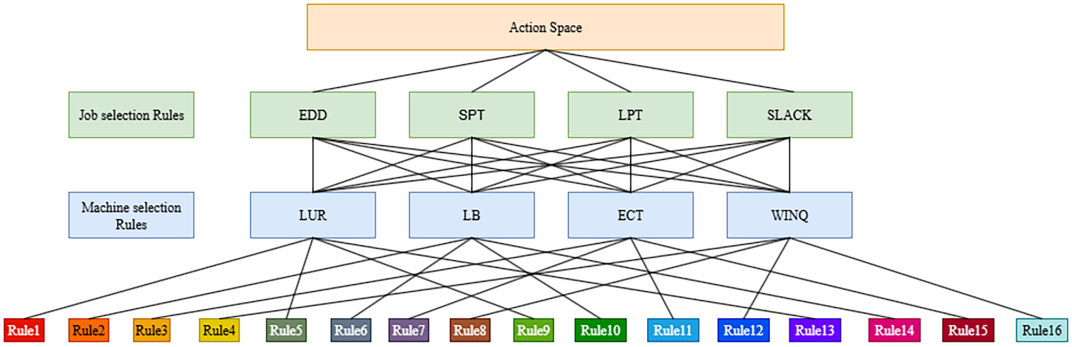

Figure 3: Rules in the action space.

4.4 Reward Function Design

In this study, our objective is for the agent to select an appropriate action in each state so as to minimize the maximum completion time and the completion-time variance. The original objectives are transformed into dimensionless normalized values to eliminate the influence of differences in scale among objectives on subsequent calculations, and the Tanh function is employed to constrain the normalized data. The two objective values are normalized as follows:

At the $t-$th decision step, the maximum completion time and the completion-time variance are normalized, respectively, and the general formula is given in Eq. (23).

$$\hat{f}_i^{(t)} = \frac{f_i^{(t)} - u_i^{(t)}}{\sigma_i^{(t)} + \omega} \tag{23}$$

where $u_i^{(t)}$ denotes the online mean of objective f_i at time step t, $\sigma_i^{(t)}$ denotes the online standard deviation of objective f_i at time step t, and ω is a small positive constant introduced to prevent the denominator from being zero.

The mean and standard deviation are dynamically updated using the exponentially weighted moving average method so as to continuously reflect the changing trend of the latest data, where $\chi \in [0.01, 0.1]$ is used to adjust the influence weight of new data relative to the original data, as shown in Eqs. (24) and (25).

$$u_i^{(t)} = (1-\chi)\, u_i^{(t-1)} + \chi f_i^{(t)} \tag{24}$$

$$\sigma_i^{(t)^2} = (1-\chi)\, \sigma_i^{(t-1)^2} + \chi \left(f_i^{(t)} - u_i^{(t)}\right)^2 \tag{25}$$

To prevent the model from being overly sensitive to the numerical range and to avoid an excessively large range of objective fluctuations, we employ the Tanh function to constrain the normalized values, ensuring that the values fall within $[-1, 1]$. Its functional expression is given in Eq. (26):

$$\hat{f}_i^{(t)'} = \tanh\left(\gamma \cdot \hat{f}_i^{(t)}\right) \tag{26}$$

Weights are designed based on task urgency considerations to adjust the reward bias, as shown in Eqs. (27) and (28).

$$\psi^{(t)} = p^{(t)} \cdot \alpha, \kappa^{(t)} = \left(1 - p^{(t)}\right) \cdot \beta \tag{27}$$

$$r^{(t)} = -\left(\psi^{(t)} \cdot \hat{f}_1^{(t)'} + \kappa^{(t)} \cdot \hat{f}_2^{(t)'}\right) \tag{28}$$

Here, α, β is the base weighting coefficient, which ensures that the two optimization objectives have equal importance in the initial state. Since the dimensional differences between the objectives have been eliminated through normalization in the previous section, its value is set to 1 in this study. The dynamic weight adjustment mechanism is defined in Eq. (27), where the weight coefficient is determined by the order urgency $p^{(t)}$, with $p \in [0,1]$ as a constant parameter. It is dynamically computed based on the remaining delivery time and the remaining processing time of the current order. When the order urgency is high, the weight of the makespan objective is increased to prioritize delivery efficiency; when the order urgency is low, the weight of the completion time variance objective is increased to enhance task synchronization and coordinated delivery performance. The dynamically weighted reward function is given in Eq. (28). It should be noted that this adaptive reward mechanism is consistently applied across all static experiments and dynamic disturbance experiments in this study.

The pseudocode of the GRU-PPO algorithm is presented below (Algorithm 1):

Algorithm 1: GRU-PPO training algorithm

Input: training steps E_t, PPO update steps E_s, batch size B, actor network π_θ, old actor network π_θ_old, critic network v_ϕ, clipping parameter ε, discount factor γ, GAE parameter λ
Initialization: Initialize θ and ϕ; θ_old ← θ; Experience Buffer ← ∅
for e = 1, . . ., E_t **do**
  Sample B PMSP instances from a dynamic distribution;
  for b = 1, . . ., B **do**
Initialize state s_{b,0}; set done ← False
for t = 1, 2, . . ., **do**
  h_{b,t} ← GRU(π_θ_old,s_{b,t});
  logits ← FullyConnectedLayer(h_{b,t});
  action_mask ← GenerateActionMask(s_{b,t});
  masked_logits ← logits + (action_mask − 1) × 1e9;
  Sample a_{b,t} from Softmax(masked_logits);
  Execute a_{b,t}; observe r_{b,t}, s_{b,t + 1};
  Store (s_{b,t}, a_{b,t}, r_{b,t}, s_{b,t + 1}, done)
if s_{b,t} is terminal then
  done ← True;
  break;
end if
end for

(Continued)

Algorithm 1 (continued)

end for
Extract (S, A, R, S', Done) from ExperienceBuffer;
V ← v_ϕ(S); V′ ← v_ϕ(S′); V′[Done] ← 0;
δ ← R + γ × V′ − V;
Â ← ComputeGAE(δ, γ, λ); R_target ← Â + V;
for p = 1, . . ., E_s do
p_θ ← π_θ(A|S); p_old = π_{θ_old}(A|S);
r_θ ← p_θ/(p_old + 1e−8);
L_CLIP(θ) ← E[min(r_θ × Â,clip(r_θ, 1 − ε, 1 + ε)Â)];
L_Hˆ(θ) ← E[Entropy(Softmax(masked_logits))];
L(θ) ← −c_pL_CLIP(θ) − c_HL_H(θ);
L_MSE(ϕ) ← E[MSE(V, R_target)];
Update θ using L(θ); update ϕ using L_MSE(ϕ);
end for
θ_old ← θ;
Clear ExperienceBuffer;
end for
Output: trained actor parameter θ;

5 Experiments and Analysis

5.1 Experimental Setup

Traditional studies on exact algorithms have mostly been limited to small-scale instances to within 6 × 30 [28], leaving large-scale scenarios insufficiently explored. To evaluate the proposed method, this study uses the classical Hurink and Brandimarte benchmark datasets. Although originally developed for the flexible job shop scheduling problem (FJSP), these datasets are converted into dynamic parallel machine scheduling instances by aggregating all operations of each job into a single total processing time. If a machine cannot process any original operation, the corresponding processing time is set to infinity. In scheduling studies where standard datasets with specific complex constraints are unavailable, cross-domain mapping and reconstruction of classical benchmark datasets have been widely adopted as an effective strategy in many frontier studies. For example, in the field of distributed flexible job shop scheduling, representative studies such as Wang et al. [26] and Gao et al. [29] transformed the classical FJSP benchmark datasets proposed by Hurink et al. into macro-level parallel-machine system scheduling instances through cross-domain mapping. The single-operation aggregation mapping adopted in this study follows this scientific paradigm.

Experiments are conducted under both static and dynamic settings. In the static setting, the model is first trained on the Hurink dataset. In the dynamic setting, random machine failures are introduced with a failure rate of 0.03 and a duration uniformly sampled from 1 to 45. To further evaluate the applicability of the proposed method, combined Hurink and Brandimarte instances are used for mixed training. All experiments are implemented on a computer running the 64-bit Windows 10 operating system, equipped with an Intel(R) Core(TM) i5-13400 processor at 2.50 GHz. The programming environment was based on Python 3.8, and PyTorch 2.1 was adopted as the deep learning framework.

To further evaluate the robustness of the proposed GRU-PPO algorithm, a sensitivity analysis was conducted on several key hyperparameters, including the learning rate, discount factor, GAE parameter, clipping factor, GRU hidden layers, and batch size. As shown in Fig. 4, different hyperparameter settings lead to different convergence behaviors and final objective values. The results indicate that the proposed

algorithm is relatively stable under appropriate parameter ranges, while unsuitable settings may lead to slower convergence or larger fluctuations.

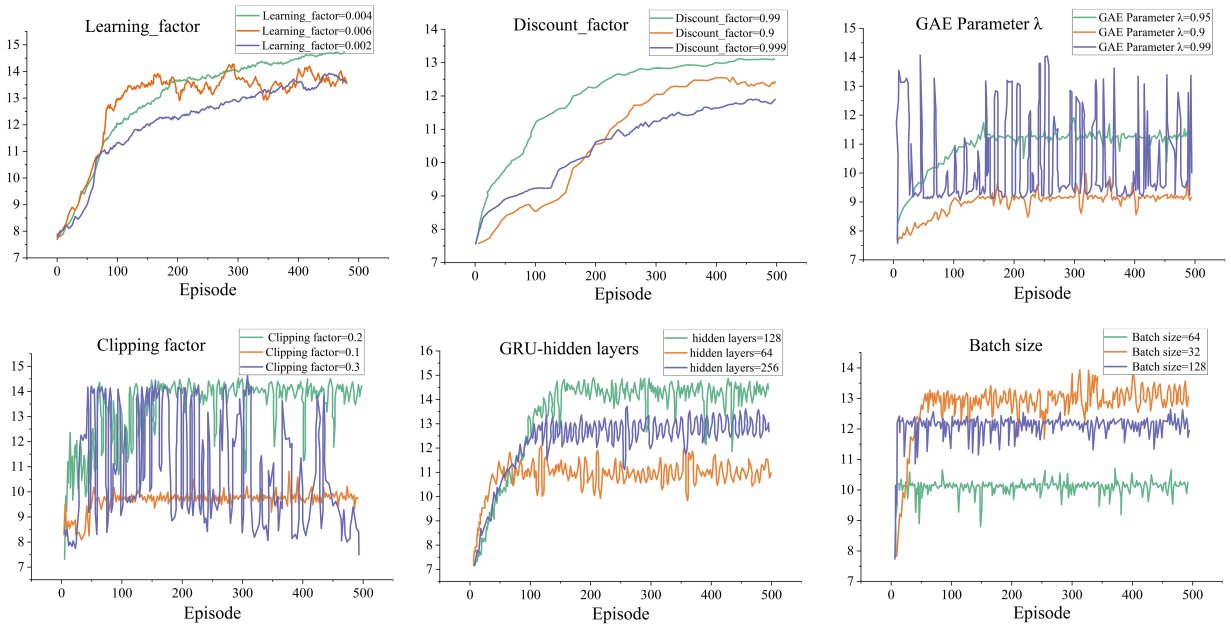

Figure 4: Sensitivity analysis of key hyperparameters.

The hyperparameter settings of the GRU-PPO algorithm are shown in Table 3:

Table 3: Some hyperparameters of the algorithm.

Parameters	Value
Actor-network learning rate	0.004
Critic-network learning rate	0.004
Discount factor	0.99
GAE Parameter λ	0.95
Clipping factor	0.2
GRU hidden layers	128
Batch size	64
Entropy weight	0.01
Value function weight	0.5

To validate the effectiveness of the proposed dual-objective scheduling framework, comparative experiments were conducted using both makespan and completion time variance as evaluation metrics. Regarding the selection of baseline algorithms, this study refers to recent advances in dynamic scheduling research. Existing studies have shown that DQN and its variants are widely applied to dynamic job shop scheduling problems; for example, DDQN has been used to address machine disruptions in dynamic flexible job shops [30]. Meanwhile, PPO, due to its strong training stability and policy optimization capability, has also been widely applied in recent years to dynamic scheduling research [31].

Based on this, DQN, DDQN, and PPO are selected as deep reinforcement learning baseline algorithms, representing the classical value-based method, the improved value-based method, and the mainstream policy gradient method, respectively. These methods make decisions based solely on the current state information and are suitable for standard MDP settings. In contrast, the proposed GRU-PPO incorporates historical state awareness to mitigate the POMDP issues caused by dynamic disturbances such as stochastic machine failures. This design enables the model to capture temporal dependencies in the environment, thereby validating the effectiveness of the historical information modeling mechanism in dynamic scheduling scenarios.

In addition, to comprehensively evaluate the performance of the proposed algorithm, two categories of traditional methods are selected as baselines. The first category consists of metaheuristic algorithms, including the Genetic Algorithm (GA) and Particle Swarm Optimization (PSO), which represent optimization methods with strong global search capability. The second category consists of heuristic rule-based methods, such as the Longest Processing Time (LPT) rule, which can rapidly generate feasible scheduling solutions and is closely aligned with the objective of minimizing makespan considered in this study.

5.2 Experimental Results

5.2.1 Validation of the Effectiveness of Action Space Rules

The action space in this study consists of 16 rules. To verify whether the proposed composite rules can effectively reduce the maximum completion time, training was conducted on MK01(10, 6) from the Brandimarte dataset. A comparison was made between training with only a single rule selected and training with random selection from the 16 rules. The results are presented by the boxplot in Fig. 5, where the horizontal axis represents the 16 individual rules and the random rule, and $C_{\max}$ denotes the maximum completion time.

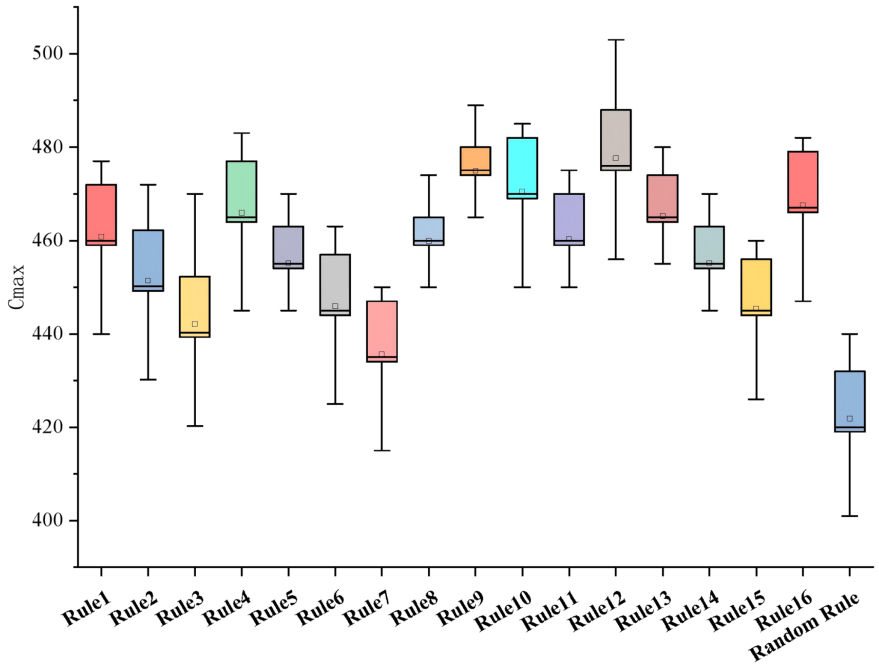

Figure 5: Boxplot of the distribution of maximum completion times under different rules.

As can be seen from the above figure, each rule affects the overall training results. Among the 16 predefined rules, Rule 7 (SPT + ECT) yields the smallest maximum completion time, with both its mean

and median significantly lower than those of the other 15 individual rules. In contrast, Rule 12 (LPT + WINQ) exhibits the largest mean and the greatest fluctuation among the 16 rules. Compared with Rule 7, when all rules are freely selectable, the resulting maximum completion time is the smallest among all rules, and the values of the maximum completion time are more concentrated. This random rule selection is therefore more beneficial to agent training. In addition, this study further recorded the selection frequency of each rule at the 100th, 300th, and 500th training episodes, respectively, and presented the results in percentage form, as shown in Fig. 6.

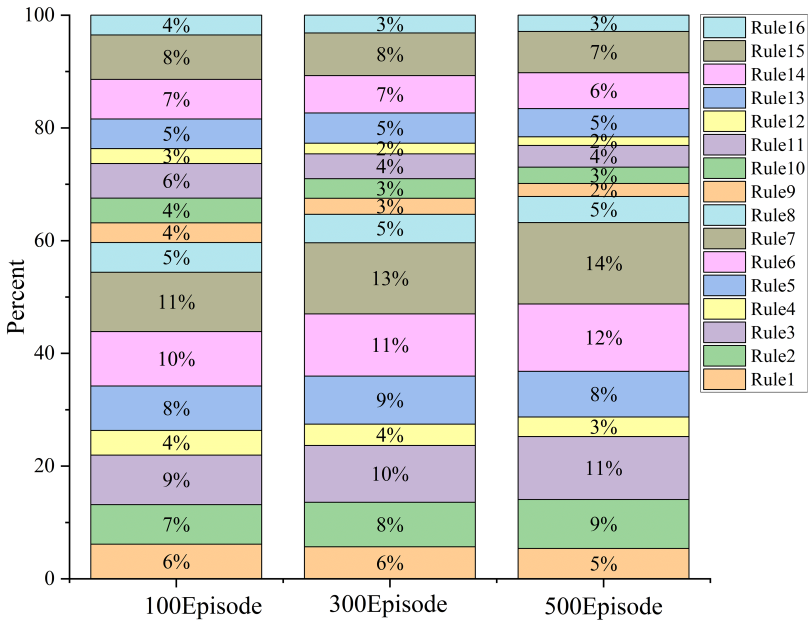

Figure 6: Selection frequency of different rules.

As shown in the figure, with the increase in the number of training episodes, the selection frequencies of Rule 3, Rule 6, and Rule 7 all increase significantly, indicating that these three rules are more suitable for the agent to accomplish the scheduling process. By the end of the 500th training episode, the combined selection probability of these three rules exceeds one-third, and Rule 7 is selected more frequently, suggesting that Rule 7 is particularly critical to the agent's scheduling performance.

5.2.2 Effectiveness of the Action Masking Mechanism

To verify the effectiveness of Action Masking during the training process, this study designed two comparative experiments using the MK02 dataset for training. Specifically, the GRU-PPO algorithm with the Action Masking mechanism and the GRU-PPO algorithm without the Action Masking mechanism were each trained for 500 episodes, and the average reward during the training process was recorded, as shown in Fig. 7.

Action Masking improves both convergence speed and final reward stability by restricting exploration to feasible actions.

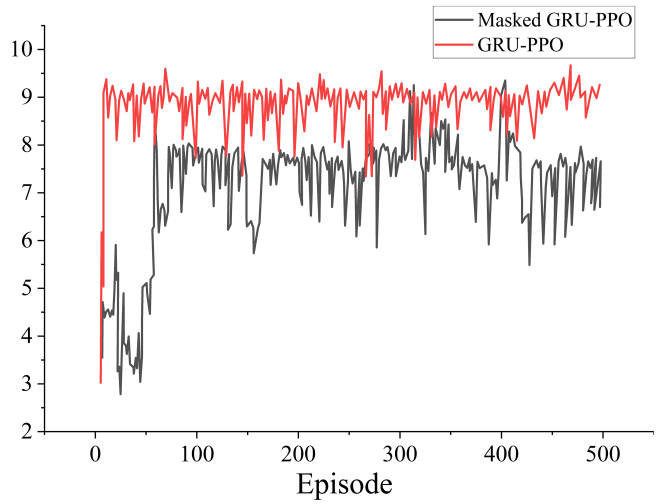

Figure 7: Comparison of rewards with and without the action masking mechanism.

5.2.3 Optimality Verification on Small-Scale Instances

For small-scale instances, an exact mixed-integer linear programming (MILP) model was solved using Gurobi to evaluate the optimality gap of the proposed method. Due to the NP-hard nature of the PMSP with collaborative delivery constraints, exact optimization becomes computationally intractable for large-scale instances. Therefore, optimality validation was conducted only on small-scale benchmark instances. The proposed GRU-PPO method was compared with the optimal solutions obtained by Gurobi within a predefined time limit. The results are shown in Table 4. The optimality gap is calculated as follows:

$$\text{Gap}\,(\%) = \frac{\left(F_{DRL} - F_{opt}\right)}{F_{opt}} \times 100\% \tag{29}$$

where F_{DRL} denotes the objective value obtained by the proposed method and F_{opt} represents the optimal objective value obtained by the MILP solver.

Table 4: Test results of small-scale instances.

Size	Optimal	GRU-PPO	Gap (%)	MILP Time (s)	GRU-PPO Time (s)
5 × 2	128	130	1.56	0.18	0.03
6 × 2	146	149	2.05	0.41	0.04
6 × 3	173	177	2.31	0.86	0.05
8 × 3	214	220	2.80	2.47	0.07
10 × 3	267	275	3.00	6.93	0.10
12 × 4	386	398	3.11	15.27	0.16

As shown in Table 4, the average optimality gap is 2.47%, and the maximum gap is only 3.11%, indicating that the proposed GRU-PPO can obtain near-optimal solutions while maintaining significantly higher computational efficiency than exact optimization methods.

5.2.4 Experimental Results under the Static Environment

Under the static environment, machine failures are not considered in this study. The la-series datasets from the edata, rdata, and vdata subsets of the Hurink benchmark are selected as the training data. To present the experimental results in a clear and concise manner, 8 representative datasets are selected from the original 120 datasets. These datasets cover different task scales, numbers of machines, and constraint conditions, and can therefore reflect the requirements of practical scenarios in a relatively comprehensive manner. The GRU-PPO algorithm is compared with heuristic rules (LPT), classical metaheuristic algorithms (GA and PSO), and the deep reinforcement learning algorithm (DQN, DDQN, and PPO). The results are shown in Table 5. In Table 5, column $C_{\max}$ denotes the maximum completion time obtained by a given algorithm or rule, and Imp denotes the gap between the $C_{\max}$ value obtained by this algorithm and those obtained by the other algorithms, as defined in Eq. (30). A positive result indicates that the algorithm reduces the makespan.

$$\text{Imp (a)} = \frac{(\text{makespan\_a} - \text{makespan\_b})}{makespan} \times 100\% \tag{30}$$

Table 5: Test results on the Hurink dataset.

Name	Size	LPT		GA		PSO		DQN		DDQN		PPO		GRU-PPO	
		Cmax	Imp	Cmax	Imp	Cmax	Imp	Cmax	Imp	Cmax	Imp	Cmax	Imp	Cmax	Imp
La05	10 × 8	657	19.2%	531	—	573	7.4%	551	3.6%	534	2.25%	602	13.4%	583	8.9%
La10	15 × 8	1082	18.4%	1126	21.6%	1047	15.7%	1013	12.8%	924	4.6%	931	5.4%	883	—
La14	20 × 5	1324	14.9%	1316	14.4%	1279	11.9%	1252	10.0%	1242	10.2%	1210	7.4%	1127	—
La16	10 × 10	1087	12.9%	1052	10.0%	1059	10.6%	947	—	964	1.8%	1105	16.7%	1042	9.1%
La21	15 × 10	1268	8.4%	1225	5.1%	1263	8.0%	1162	—	1238	6.5%	1271	9.3%	1218	4.6%
La30	20 × 10	1427	8.5%	1363	4.3%	1344	2.9%	1352	3.5%	1330	1.9%	1402	7.4%	1305	—
La35	30 × 10	1983	7.4%	1964	6.5%	1944	5.6%	1874	2.1%	2002	9.0%	2045	11.4%	1836	—
La40	15 × 15	1459	13.6%	1380	8.7%	1427	11.7%	1365	7.7%	1314	4.3%	1421	12.7%	1260	—

As can be seen from Table 5, among the above algorithms, the GRU-PPO algorithm achieves the best overall performance on most instances. However, different algorithms may be paired with different models and decision conditions during training. To further verify the performance of the algorithm at different scales, we classify the La datasets in Hurink according to a × b: if a × b ≤ 100, the dataset is categorized as a small-scale dataset; if 100 < a × b ≤ 200, it is categorized as a medium-scale dataset; and if a × b > 200, it is categorized as a large-scale dataset. According to this classification criterion, the dataset partition is shown in Table 6:

Table 6: Classification of the La datasets.

Category	Dataset
Small size	la01, la02, la03, la04, la05, la06, la07, la08, la09, la10
Medium size	la11, la12, la13, la14, la15, la16, la17, la18, la19, la20, la21, la22, la23, la24, la25
Large size	la26, la27, la28, la29, la30, la31, la32, la33, la34, la35, la36, la37, la38, la39, la40

To comprehensively evaluate the performance of the proposed GRU-PPO algorithm in parallel machine scheduling, one dataset was selected from each of the small-, medium-, and large-scale categories listed in Table 6 and compared with three representative types of algorithms: heuristic scheduling algorithms

(LPT), classical metaheuristic algorithms (GA and PSO), and a deep reinforcement learning algorithm (DQN, DDQN, PPO).

The problem addressed in this study is a bi-objective dynamic parallel machine scheduling problem. However, to maintain consistency with the training mechanism of reinforcement learning methods, the bi-objective problem is transformed into a single-objective optimization model through a weighted-sum approach. This method is commonly adopted in existing reinforcement learning training models and can effectively improve the stability and reliability of model training [32]. All algorithms were trained for 500 episodes on each dataset, and the results are presented as boxplots in Fig. 8. GRU-PPO shows a smaller makespan distribution and better stability than the competing methods across different dataset scales.

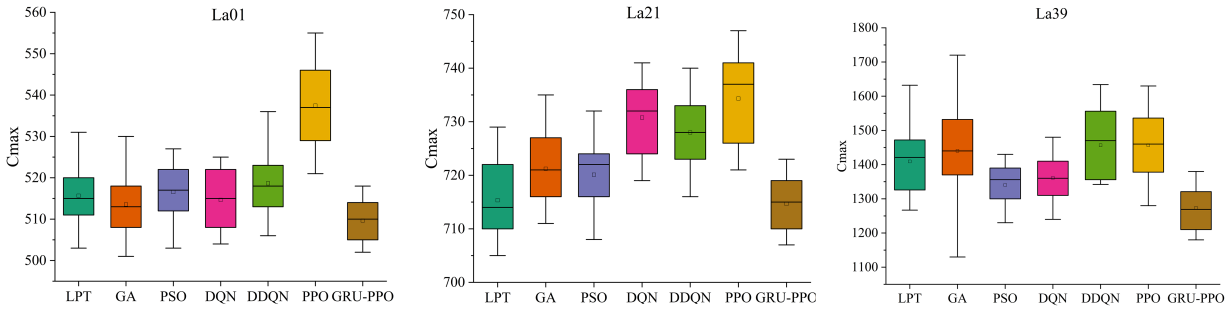

Figure 8: Comparison of boxplots for different algorithms.

As shown in Fig. 9, to validate the effectiveness of the proposed GRU architecture in a POMDP environment, this study compares the performance of the PPO and GRU-PPO algorithms on the Vdata01, Vdata26, and Vdata32 datasets. The average reward curves obtained by training both algorithms for 500 episodes on each dataset are presented. It can be observed from the figure that the GRU-PPO algorithm exhibits good convergence and stability across the training process on all three datasets, and its average reward is higher than that of the PPO algorithm.

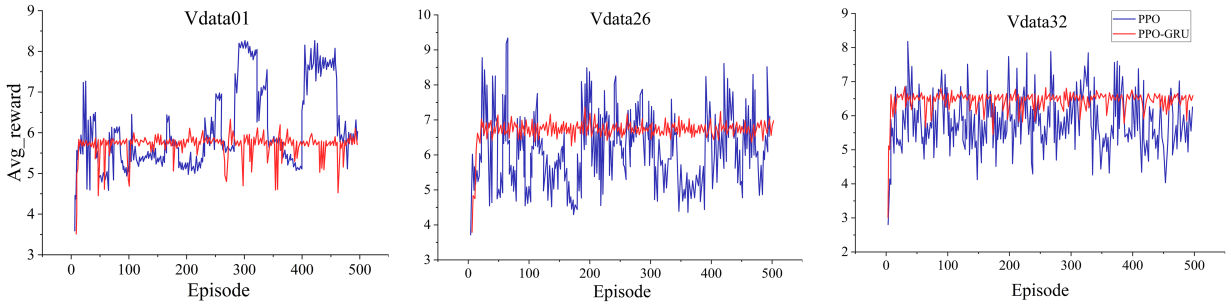

Figure 9: Performance comparison between PPO and GRU-PPO.

As shown in Fig. 10, the GRU-PPO algorithm was trained and compared with the DQN and DDPG algorithms on the Vdata01, Vdata26, and Vdata32 datasets, respectively, in order to evaluate the performance of the three algorithms. Across all datasets, the GRU-PPO algorithm exhibited superior convergence and stability compared with the other algorithms, and this advantage became even more pronounced as the complexity of the datasets increased. These results indicate that, relative to other reinforcement learning algorithms, the GRU-PPO algorithm possesses stronger stability and better overall performance.

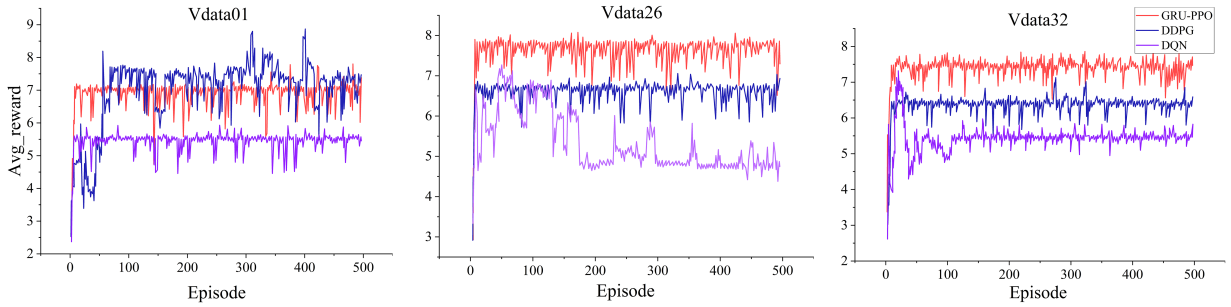

Figure 10: Reward comparison among different algorithms.

5.2.5 Experimental Results under the Dynamic Environment

In real-world scheduling, many unexpected events may occur, such as machine failures and rush order insertions. Once such events arise, they may increase processing time or alter the original scheduling plan. The GRU-PPO algorithm with the masking mechanism proposed in this study has demonstrated high reliability in static-environment experiments. Therefore, under the combined dataset, the datasets were processed according to practical production conditions by introducing machine failures with a probability of 0.03 and a failure duration following a uniform distribution from 1 to 45, so as to obtain rescheduling results. Subsequently, sub-datasets were selected from the Hurink and Brandimarte datasets for experimentation. In selecting these sub-datasets, we classified them according to the number of machines and the job scale in order to evaluate the effectiveness of the model in ensuring the variance of job completion times during scheduling. In each sub-dataset, four jobs were selected as one complete task set, with the objective of minimizing the variance of job completion times within each set. As shown in Table 7, the results of different algorithms with respect to the completion-time variance of job sets under datasets of different scales are presented.

Table 7: Results of the completion-time variance for job sets.

Size of Datasets	Name	LPT	GA	PSO	DQN	DDQN	PPO	GRU-PPO
Machine number is 5	La01(vdata)	1.1	0.8	1.1	0.8	1.0	1.0	0.9
	La05(vdata)	1.0	0.9	1.0	1.0	1.0	1.1	0.7
	La08(vdata)	1.0	0.9	0.8	0.9	0.9	1.0	0.6
	La13(vdata)	1.0	0.8	0.9	1.0	0.7	0.8	0.5
	MK07	1.1	1.0	0.8	1.1	0.6	1.1	0.5
	MK11	1.0	1.1	1.0	0.7	0.8	1.0	0.5
Machine number is 10	La16(vdata)	1.2	1.0	0.8	0.9	0.8	1.1	0.8
	La21(vdata)	1.0	0.9	1.0	0.8	0.9	1.2	0.7
	MK08	1.1	0.8	0.9	0.8	0.8	1.1	0.6
	MK09	0.9	1.0	1.2	1.1	1.0	0.8	0.6
	La13(vdata)	1.0	0.8	0.9	1.0	0.9	1.2	0.8

As can be seen from Table 7, the GRU-PPO algorithm yields smaller completion-time variance across datasets of different scales than algorithms such as LPT, GA, PSO, and DQN. LPT, GA, and PSO exhibit relatively high variance in job completion times, which is particularly evident when multiple machines are available for selection. DQN, DDQN, and PPO show competitive performance in some instances, but overall

GRU-PPO exhibits better stability across datasets. The GRU-PPO algorithm demonstrates good consistency and stability in optimizing both the maximum completion time and the variance of job completion times.

Subsequently, the trained GRU-PPO algorithm was applied to the 07a and 12a subsets of the Dauzere dataset, and the corresponding Gantt charts are shown in Fig. 11.

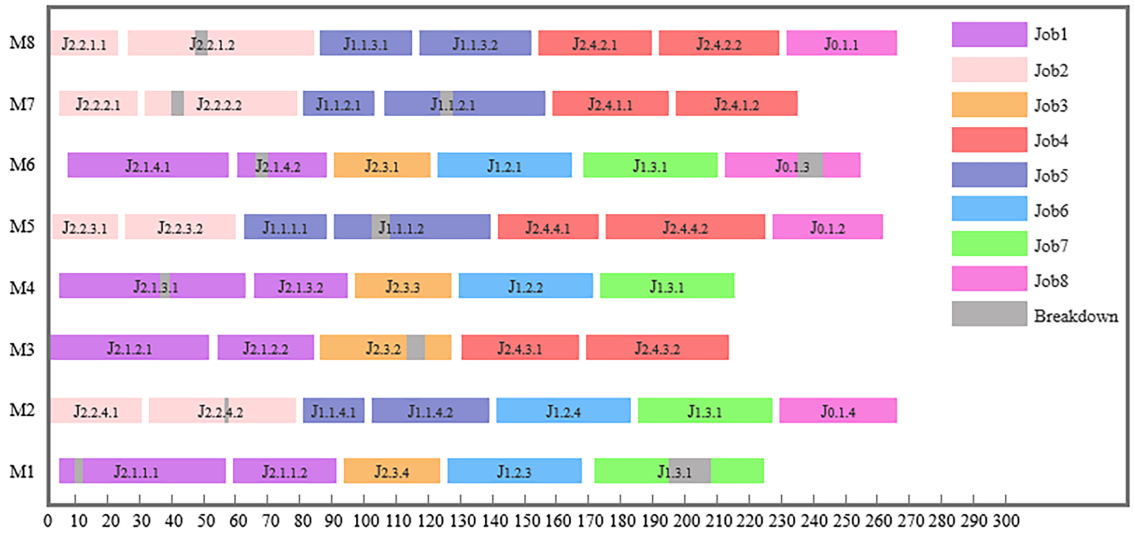

Figure 11: Gantt charts of the scheduling results.

The experimental results show that the jobs assigned to the machines are relatively compact, which can effectively improve machine utilization and reduce idle time, while the completion times of jobs within the same batch remain relatively synchronized. Moreover, under disturbances such as machine failures, the GRU-PPO optimization algorithm is able to satisfy the job shop requirements of minimizing both the maximum completion time and the variance of job completion times. The practical validation further demonstrates the effectiveness and superiority of the decision-making method proposed in this paper.

6 Conclusion

This paper addresses the parallel machine scheduling problem in job shop environments. Subject to constraints such as order priority, it takes the coordinated delivery of multiple tasks within an order as the central objective, and adopts the joint optimization goals of minimizing the maximum completion time of orders and minimizing the variance of completion times among multiple tasks, while considering random machine failure disturbances. On this basis, a parallel machine scheduling optimization model considering multi-task coordination within orders is constructed, and an improved GRU-PPO hybrid intelligent algorithm integrating temporal feature extraction and Action Masking is developed. This algorithm significantly enhances the model's temporal perception capability in partially observable environments and, by effectively reducing the action space, substantially improves both the efficiency of policy updates and the physical feasibility of decisions.

The experimental results demonstrate that, based on extensive tests conducted on the Hurink, Brandimarte, and Dauzere benchmark datasets, the proposed algorithm not only achieves competitive makespan

and lower completion-time variance in most test scenarios, but also exhibits good autonomous decision-making capability and robustness against disturbances, particularly under dynamic disruptions, thereby effectively ensuring the coordinated and on-time delivery of orders.

The principal innovation of this study lies in incorporating completion-time variance into a deep reinforcement learning framework as a quantitative measure for coordinated delivery of multiple tasks within an order, together with the improved design of a GRU-PPO algorithm embedded with an Action Masking mechanism. This work provides important theoretical support and practical engineering value for addressing the challenges of high uncertainty and making-to-order production scheduling in the context of Industry 4.0.

In addition, the proposed algorithm can be further improved, and future work will consider multi-agent scheduling and digital-twin-enabled deployment in large-scale industrial environments.

Acknowledgement: Not applicable.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Pei Xie; data analysis and interpretation: Pei Xie; experimental data collection: Zhijie Pei; draft manuscript preparation: Bo Li; manuscript revision and proofreading: Xiaoying Yang; literature collection and preparation: Fenghai Yang. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The datasets used in this study are benchmark and publicly available datasets that can be readily downloaded from the Internet.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Zhang Q, Zhang Q, Duan J, Qin J, Zhou Y. Energy-efficient scheduling for distributed hybrid flowshop of offshore wind blade manufacturing considering limited buffers. J Mar Sci Eng. 2025;13(11):2176. doi:10.3390/jmse13112176.
2. Zhu X, Xu J, Ge J, Wang Y, Xie Z. Multi-objective parallel machine scheduling with eligibility constraints for the kitting of metal structural parts. Machines. 2022;10(10):836. doi:10.3390/machines10100836.
3. Ouelhadj D, Petrovic S. A survey of dynamic scheduling in manufacturing systems. J Sched. 2009;12(4):417–31. doi:10.1007/s10951-008-0090-8.
4. Lu L, Deng Q, Luo Q, Hu Y, Zhang J, Deng K. Integrated optimization of dynamic flexible job shop scheduling with AGV breakdowns and multi-site equipment maintenance on the demand side. Comput Ind Eng. 2025;208(3):111361. doi:10.1016/j.cie.2025.111361.
5. Hu K, Che Y, Ng TS, Deng J. Unrelated parallel batch processing machine scheduling with time requirements and two-dimensional packing constraints. Comput Oper Res. 2024;162(3):106474. doi:10.1016/j.cor.2023.106474.
6. Fan J, Shen W, Gao L, Zhang C, Zhang Z. A hybrid Jaya algorithm for solving flexible job shop scheduling problem considering multiple critical paths. J Manuf Syst. 2021;60(1):298–311. doi:10.1016/j.jmsy.2021.05.018.
7. Hu C, Zheng R, Lu S, Liu X. Parallel machine scheduling with position-dependent processing times and deteriorating maintenance activities. J Glob Optim. 2026;94(2):377–97. doi:10.1007/s10898-024-01411-2.
8. Xie F, Li K, Chen J, Xiao W, Zhou T. An adaptive large neighborhood search for unrelated parallel machine scheduling with setup times and delivery times. Comput Oper Res. 2025;177(2):106976. doi:10.1016/j.cor.2025.106976.
9. Wang L, Qi Y. Scheduling an energy-aware parallel machine system with deteriorating and learning effects considering multiple optimization objectives and stochastic processing time. Comput Model Eng Sci. 2023;135(1):325–39. doi:10.32604/cmes.2022.019730.

10. Lu S, Zhang X, Kong M, Fathollahi-Fard AM. Two-stage hybrid flow shop scheduling with sequence-dependent setup times in semiconductor manufacturing: a customized variable neighborhood search. Ann Oper Res. 2025;1159(3):651. doi:10.1007/s10479-025-06541-8.
11. Chen J, Li K, Xie F, Chu C. An effective metaheuristic algorithm for uniform parallel machine scheduling with resource consumption restriction to minimize the maximum lateness. Int J Ind Eng. 2024;31(6):1237. doi:10.23055/ijietap.2024.31.6.10345.
12. Fan X, Sang H, Tian M, Yu Y, Chen S. Integrated scheduling problem of multi-load AGVs and parallel machines considering the recovery process. Swarm Evol Comput. 2025;94(6):101861. doi:10.1016/j.swevo.2025.101861.
13. Qiu J, Liu J, Li Z, Lai X. A multi-level action coupling reinforcement learning approach for online two-stage flexible assembly flow shop scheduling. J Manuf Syst. 2024;76(2):351–70. doi:10.1016/j.jmsy.2024.08.006.
14. Fu T, Zhu H. Digital-twin-based reinforcement learning method for flexible assembly job-shop scheduling with dynamic material kitting. Comput Ind Eng. 2026;214(3):111903. doi:10.1016/j.cie.2026.111903.
15. Bengio Y, Lodi A, Prouvost A. Machine learning for combinatorial optimization: a methodological tour d'horizon. Eur J Oper Res. 2021;290(2):405–21. doi:10.1016/j.ejor.2020.07.063.
16. Paeng B, Park I-B, Park J. Deep reinforcement learning for minimizing tardiness in parallel machine scheduling with sequence dependent family setups. IEEE Access. 2021;9:101390–401. doi:10.1109/ACCESS.2021.3097254.
17. Liu R, Piplani R, Toro C. Deep reinforcement learning for dynamic scheduling of a flexible job shop. Int J Prod Res. 2022;60(13):4049–69. doi:10.1080/00207543.2022.2058432.
18. Luo S. Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning. Appl Soft Comput. 2020;91(21):106208. doi:10.1016/j.asoc.2020.106208.
19. Wang JJ, Wang L. A cooperative memetic algorithm with learning-based agent for energy-aware distributed hybrid flow-shop scheduling. IEEE Trans Evol Comput. 2022;26(3):461–75. doi:10.1109/TEVC.2021.3106168.
20. Song W, Mi N, Li Q, Zhuang J, Cao Z. Stochastic economic lot scheduling via self-attention based deep reinforcement learning. IEEE Trans Autom Sci Eng. 2024;21(2):1457–68. doi:10.1109/TASE.2023.3248229.
21. Chen R, Li W, Yang H. A deep reinforcement learning framework based on an attention mechanism and disjunctive graph embedding for the job-shop scheduling problem. IEEE Trans Ind Inform. 2023;19(2):1322–31. doi:10.1109/TII.2022.3167380.
22. Sha S, Liu Y, Huo B. Dynamic proximal policy optimization: enhancing PPO with adaptive entropy and smooth clipping. Neurocomputing. 2026;674:132861. doi:10.1016/j.neucom.2026.132861.
23. Huang D, Zhao H, Cao J, Chen K, Zhang L. Optimizing the flexible job shop scheduling problem via deep reinforcement learning with mean multichannel graph attention. Appl Soft Comput. 2025;177(2):113128. doi:10.1016/j.asoc.2025.113128.
24. Luo S, Zhang L, Fan Y. Real-time scheduling for dynamic partial-no-wait multiobjective flexible job shop by deep reinforcement learning. IEEE Trans Autom Sci Eng. 2022;19(4):3020–38. doi:10.1109/TASE.2021.3104716.
25. Zheng L, Chen X, Liu J, Zhuang C. Real-time dynamic integrated process planning and scheduling with reconfigurable manufacturing cells via multi-agent reinforcement learning. J Manuf Syst. 2026;85:127–54. doi:10.1016/j.jmsy.2026.01.004.
26. Wang M, Zhang J, Zhang P, Xiang W, Jin M, Li H. Generative deep reinforcement learning method for dynamic parallel machines scheduling with adaptive maintenance activities. J Manuf Syst. 2024;77(3):946–61. doi:10.1016/j.jmsy.2024.11.004.
27. Arulkumaran K, Deisenroth MP, Brundage M, Bharath AA. Deep reinforcement learning: a brief survey. IEEE Signal Process Mag. 2017;34(6):26–38. doi:10.1109/MSP.2017.2743240.
28. Liu C-L, Tseng C-J, Huang T-H, Wang J-W. Dynamic parallel machine scheduling with deep Q-network. IEEE Trans Syst Man Cybern Syst. 2023;53(11):6792–804. doi:10.1109/TSMC.2023.3289322.
29. Gao J, Gen M, Sun L, Zhao X. A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems. Comput Ind Eng. 2007;53(1):149–62. doi:10.1016/j.cie.2007.04.010.
30. Zheng T, Zhou Y, Hu M, Zhang J. Dynamic scheduling for large-scale flexible job shop based on noisy DDQN. Int J Netw Dyn Intell. 2023;2(4):100015. doi:10.53941/ijndi.2023.100015.

31. Zhang L, Feng Y, Xiao Q, Xu Y, Li D, Yang D, et al. Deep reinforcement learning for dynamic flexible job shop scheduling problem considering variable processing times. J Manuf Syst. 2023;71(21):257–73. doi:10.1016/j.jmsy.2023.09.009.
32. Cota LP, Coelho VN, Guimarães FG, Souza MJF. Bi-criteria formulation for green scheduling with unrelated parallel machines with sequence-dependent setup times. Int Trans Oper Res. 2021;28(2):996–1017. doi:10.1111/itor.12566.