Robotics and Computer–Integrated Manufacturing 78 (2022) 102412

Contents lists available at ScienceDirect

Robotics and Computer-Integrated Manufacturing

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

Dynamic job shop scheduling based on deep reinforcement learning for multi-agent manufacturing systems

Yi Zhang, Haihua Zhu *, Dunbing Tang, Tong Zhou, Yong Gui

College of Mechanical and Electrical Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210000, China

ARTICLE INFO

Keywords:
Flexible job-shop scheduling problem
Smart manufacturing
Multi-agent manufacturing system
Reinforcement learning
Proximal policy optimization

ABSTRACT

Personalized orders bring challenges to the production paradigm, and there is an urgent need for the dynamic responsiveness and self-adjustment ability of the workshop. Traditional dispatching rules and heuristic algorithms solve the production planning and control problems by making schedules. However, the previous methods cannot work well in a changeable workshop environment when encountering a large number of stochastic disturbances of orders and resources. Recently, the potential of artificial intelligence (AI) algorithms in solving the dynamic scheduling problem has attracted researchers' attention. Therefore, this paper presents a multi-agent manufacturing system based on deep reinforcement learning (DRL), which integrates the self-organization mechanism and self-learning strategy. Firstly, the manufacturing equipment in the workshop is constructed as an equipment agent with the support of edge computing node, and an improved contract network protocol (CNP) is applied to guide the cooperation and competition among multiple agents, so as to complete personalized orders efficiently. Secondly, a multi-layer perceptron is employed to establish the decision-making module called AI scheduler inside the equipment agent. According to the perceived workshop state information, AI scheduler intelligently generates an optimal production strategy to perform task allocation. Then, based on the collected sample trajectories of scheduling process, AI scheduler is periodically trained and updated through the proximal policy optimization (PPO) algorithm to improve its decision-making performance. Finally, in the multi-agent manufacturing system testbed, dynamic events such as stochastic job insertions and unpredictable machine failures are considered in the verification experiments. The experimental results show that the proposed method is capable of obtaining the scheduling solutions that meet various performance metrics, as well as dealing with resource or task disturbances efficiently and autonomously.

1. Introduction

With the development of social productivity, customer demand for personalized products has become more and more intense. In order to meet the demand of customized products, the mainstream of manufacturing mode has changed from mass production to small-scale customized production [1]. The customer orders show the typical characteristics of multi-variety and variable-batch. It is extremely difficult for manufacturers to realize efficient and energy-saving production under the current situation. There is a good deal of heterogeneous manufacturing equipment with various processing capabilities in a smart workshop. If these manufacturing resources are reasonably organized and configured, the personalized manufacturing tasks can be performed efficiently [2].

In the workshop environment, the customer order types, processing techniques, machine types and production status are diversified and complicated, which makes it difficult to improve production efficiency, balance the workload between machines and reduce energy consumption simultaneously depending on manual management. In this context, dynamic scheduling methods become an important research issues in the field of intelligent manufacturing system, and can reasonably assign production tasks to manufacturing resources, as well as realize some optimization objectives (e.g., minimum completion time, workload balance and maximum profit). This paper focuses on solving the job-shop scheduling problem (JSP) in discrete manufacturing. JSP is a classic NP-hard problem in the field of workshop scheduling [3]. Each job within the production task is composed of several kinds of operations, and the energy consumption of the operation of the job processed on different machines is inconsistent. JSP assumes that each operation of the job merely can be processed on a specific machine tool. While in

* Corresponding author.
E-mail address: h.zhu@nuaa.edu.cn (H. Zhu).

https://doi.org/10.1016/j.rcim.2022.102412
Received 21 January 2022; Received in revised form 5 June 2022; Accepted 1 July 2022
Available online 6 July 2022
0736-5845/© 2022 Elsevier Ltd. All rights reserved.

actual workshop, multiple machine tools are qualified to handle one operation with different processing speed and energy efficiency. In order to make the scheduling model meet the actual needs, the flexible extensions of JSP have been developed. Among them, flexible job-shop scheduling problem (FJSP) model is more in line with the characteristics of actual workshop. In FJSP, multiple machine tools are available for processing one operation.

At present, the scheduling methods used to solve FJSP are mainly divided into two categories. The first type is the static scheduling methods based on meta-heuristic algorithm, such as genetic algorithm (GA), particle swarm optimization (PSO) algorithm, gray wolf optimization (GWO) algorithm, etc. The static scheduling methods firstly model the scheduling problem as a gene or particle vector via coding operation, and then explore the feasible solution space by adopting search strategies including crossover, mutation, particle updating and so on. After a certain number of iterations, the optimal solution to scheduling problem would be obtained. The second type is rule-based scheduling methods, which are composed of shortest processing time (SPT), shortest job first (SJF), etc. These methods perform task allocation through the dispatching rules. For instance, SPT rule prefers to assign production tasks to manufacturing equipment with the shortest processing time. The above static scheduling methods can find the optimal or near optimal solution through iterations and search strategies, but they lack of adaptability and intelligence. When encountering unpredictable exceptions during manufacturing execution, production activities will be forced to stagnate until the scheduling problem is solved again. The rule-based scheduling methods can still work normally when the production disturbance occurs. However, the dispatching rules should be manually selected from the knowledge base, and the quality of the generated scheduling solution is poor, especially in the scenario of frequent disturbance events.

In practical application, neither of the aforementioned scheduling methods meet the robust and efficient production requirements of the manufacturing systems, especially in the rapidly changing and disturbance-prone workshop environment. Therefore, an intelligent and adaptive scheduling method is desperately required to solve the FJSP and realize efficient and energy-saving production in a smart shop floor.

With the help of ubiquitous sensors and the Internet of Things (IoT) [4,5], a great deal of on-site work-in-process information and equipment execution status of the manufacturing plants can be instantly perceived and automatically analyzed, such as machine abnormal alarm, machine workload, and task operation progress. Potentially, the data-rich workshop environment provides a great possibility for AI algorithms to realize the data-driven self-adaptive and self-optimization task scheduling. Reinforcement learning (RL) is a popular and useful AI algorithm based on the trial-and-error method, and is suitable for optimizing the scheduler of workshop system [6]. A classical RL model consists of three basic elements, namely environment, RL agent, policy. The policy represents a function projection from state space to action space, and provides appropriate action to the RL agent according to the perceived environment state. Specifically, RL method fulfills the training of RL agent by feeding the obtained reward value when RL agent explores the environment, and the RL agent interacts with the environment via taking actions and the feedback. Firstly, RL agent observes the real-time environment state and extracts the features of the input environment state. Then, an optimal action is taken by RL agent at a decision point through the policy criteria, and the next environment state would be generated as well as the reward value simultaneously. Furthermore, the reward value fed back is used to evaluate the performance of action the RL agent takes at that moment, as well as to optimize the policy until the maximum number of iterations or the convergence is reached. Recently, the burgeoning deep reinforcement learning (DRL) has aroused great interest of research scholars. DRL is the combination of deep learning and RL. Specifically, DRL combines the structure of deep learning (i.e., neural network) with the idea of RL. With the help of the powerful representation ability of neural network (NN), DRL can fit Q-table or directly fit strategy to solve the sequential decision-making problem in a large-scale or continuous state-action space. The multi-layer NN is utilized as a policy approximator of the DRL to produce an appropriate action, which could greatly improve the ability of RL to solve the large-scale problem and deal with the curse of dimensionality.

In manufacturing system, FJSP can be modeled as a RL problem, and a RL agent is designed for workshop scheduling as shown in Fig. 1. The learning process of RL agent is constructed according to markov decision process (MDP). The state space of workshop environment consists of the job state and machine state. The action space of RL agent is composed of the set of available machine tools for processing various workpieces. Firstly, the RL agent observes the current workshop state S_t, and generates an optimal production behavior a_t through the intelligent decision-making model. Secondly, the workshop system assigns the tasks in job sequence to the manufacturing equipment for processing according to the production instructions. When a production task is completed, the workshop system will generate a reward r_t reflecting the influence of production behavior on the production efficiency and feed it back to RL agent. Finally, the RL agent optimizes the decision model according to the collected rewards.

The main contributions of this paper are listed as follows:

(1) The manufacturing equipment in the workshop is constructed as an equipment agent with the support of edge computing node, and a multi-agent manufacturing system with the ability of online scheduling and scheduling strategy optimization is presented.

(2) A self-organizing negotiation mechanism based on contract network protocol (CNP) is proposed to guide the cooperation and competition among multiple agents, and to provide support for intelligent decision-making.

(3) The decision-making module of equipment agent called AI scheduler is established based on the multi-layer perceptron, and enables the manufacturing equipment to intelligently generate an optimal production strategy according to the observation of workshop environment. An adaptive learning strategy of AI scheduler based on proximal policy optimization (PPO) algorithm is proposed to improve its decision-making performance under the disturbance of orders and resources.

The presentation of this paper is organized as follows: the development, significance and general idea of dynamic job shop scheduling method are presented in section “Introduction”. An overview of relevant research is discussed in section “Literature review” in term of multi-agent manufacturing system, scheduling and reinforcement learning. In section “Research methodology”, the architecture of multi-agent manufacturing system and the formulation of the mathematical scheduling model are given, and self-organizing negotiation mechanism and self-adaptive decision-making method are elaborated in detail. A test case is applied to verify the effectiveness and superiority of the proposed AI scheduler in section “Experimental results”. Finally, the section “Conclusions” summarizes the main contents of this study and puts forward further research ideas.

2. Literature review

2.1. Multi-agent manufacturing system

Thomas Schelling firstly presented the concept and definition of agent in 1971 [7]. In the computer simulation environment, multiple agents successfully interact with each other and handle emergencies autonomously. This laid a technical foundation for the development of multi-agent system. In nature, the agent is an independent software or hardware entity designed to realize specific functions.

In order to integrate multi-agent technology into manufacturing system, Monotori and Suganuma gave the definition of each functional module inside the manufacturing equipment agent [8,9], which mainly consisted of information layer, analysis layer and control layer, and analyzed the working ability of each functional module and the

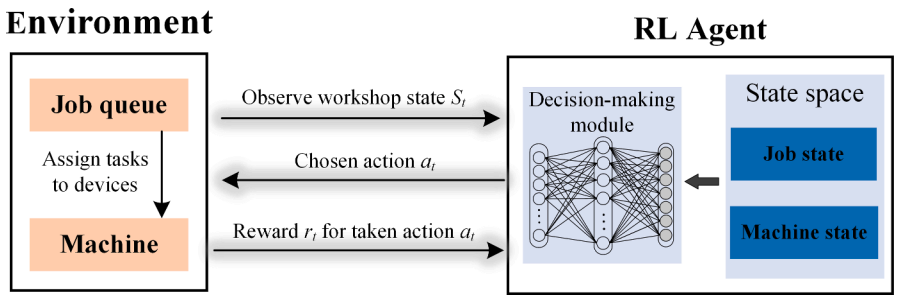


Fig. 1. The design of RL-based scheduling method in a smart workshop.

input-output relationship among functional modules. Krothapalli abstracted the physical entities at the workshop level into two typical agents [10], namely machine agent (MA) and part agent (PA), and realized the dynamic process planning and scheduling of manufacturing system by using multi-agent technology. In order to further improve the overall performance of multi-agent manufacturing system, researchers put forward various functional agents, including supervisory agent [11], database agent [12] and job agent [13]. The functional agents deployed in the workshop server work together with the machine agents to solve the problem of falling into local optimum that caused by the myopic nature of single agent, thus ensuring the global performance at workshop level.

The abnormal events such as emergency orders and machine failures make the production control of the workshop complicated, and the traditional centralized and hierarchical control approaches cannot adapt to the complex and changeable workshop environment. Barbosa and Valckenaers presented the agile and adaptive control architectures called ADACOR [14,15] and PROSA [16,17] respectively, which made a great contribution to improving the performance of distributed and discrete manufacturing system in responding to and handling abnormal event.

In addition, negotiation mechanism and negotiation strategy are key factors in the implementation of multi-agent manufacturing system. They enable workshop entities to carry out effective cooperation and competition, and complete complex manufacturing tasks jointly and handle abnormal events efficiently. Schild proposed a ring-like model combined with contract net protocol (CNP) [18], which could efficiently complete the task allocation of holonic manufacturing system. Pascal designed a rescheduling mechanism by combining CNP with Distributed Constrained Satisfaction Problem (DisCSP) [19], which improves the adaptability of multi-agent manufacturing system when deal with the fluctuation of resources and orders. Zhang developed a negotiation mechanism based on game theory to coordinate the machine agents [20], so as to efficiently handle unpredictable exceptions. Rajabinasab proposed a pheromone-based coordination method among agents to solve the flexible job shop scheduling problem and cope with dynamic events [21]. Xiong established the self-organizing negotiation strategy of immune multi-agent scheduling system through imitating humoral immunity [22]. Wang designed an auction-based manufacturing planning and control approach to realize dynamic task allocation [23].

2.2. Job-shop scheduling

Job-shop scheduling problem is a significant issue in the field of intelligent manufacturing system and combinatorial optimization. Reasonable task allocation not only effectively shorten the delivery cycle of products, but also improve the resource utilization rate of workshop.

From the relative literature review, various heuristic algorithms have been applied to solve the job-shop scheduling problem (JSP). Gao proposed an improved genetic algorithm to acquire the solution to flexible job-shop scheduling problem (FJSP) with the goal of minimizing the maximum completion time [24]. Zhang presented an improved particle swarm optimization algorithm to solve FJSP efficiently, using a novel crossover and mutation strategy [25]. Meng put forward a hybrid artificial bee colony algorithm to solve FJSP with the objective of minimizing the total flowtime [26].

In addition, scholars put forward some rescheduling and dynamic scheduling methods to keep production activities running smoothly when the abnormal events occur. Nouiri proposed a distributed particle swarm optimization algorithm that can be embedded in multi-agent manufacturing system with the superior ability of dealing with unforeseen events and making real-time scheduling decisions [27]. Raviteja adopt two-stage teaching-learning-based optimization method to figure out the problem of machine breakdown in the scheduling process [28]. Artificial bee colony algorithm [29] and discrete jaya algorithm [30] were used to accomplish workshop rescheduling with the constraints of fuzzy processing time and stochastic job arrival. Genetic programming (GP), as a hyper-heuristic method, has been successfully applied to automatically evolve dispatching rules for JSP, FJSP, and dynamic FJSP [31]. A new strategy of subtree selection for crossover was proposed to help GP converge fast and obtain optimal routing and sequencing rules when solving dynamic FJSP [32]. The ensembles of priority rules were designed to improve the performance of priority rules created with GP for resource constrained project scheduling problem [33].

With the support of IoT and artificial intelligence technology, the manufacturing system not only is equipped with the ability to acquire real-time status data of workshop environment, but also makes instant decisions to support dynamic scheduling based on the collected data. Zhang introduced the digital twin to closely blend the physical and virtual space of job shop [34], which greatly enabled dynamic scheduling. Zhang put forward a multiagent-based real-time scheduling approach to solving FJSP with the help of bargaining-game-based negotiation mechanism [20]. Lin combined the cloud-edge collaboration framework with multiclass deep Q network (DQN) to fulfill real-time decision-making of JSP in a semiconductor manufacturing factory [35]. Wang presented a dynamic adaptive scheduling system with a goal of decreasing the completion time of task, which used proximal policy optimization (PPO) to deal with fortuitous events and task allocation in actual workshop environment [36]. Zhou proposed an agent-based approach integrating deep reinforcement learning for dynamic scheduling in a smart shop floor [37].

2.3. Reinforcement learning

With the great success of Google's AlphaGo, reinforcement learning (RL) has attracted extensive attention from scholars in various fields [38]. RL is an important branch of machine learning, which consists of three basic elements: environment, RL agent and policy. From the perspective of learning objects of RL agent, RL algorithms can be divided into three categories: value-based, policy-based and Actor-Critic. The value-based methods mainly consist of Q-learning, State–Action–Reward–State–Action (SARSA) and DQN. The typical representative of policy-based method is REINFORCE algorithm. The Actor-Critic methods refer to the combination of value-based and

policy-based methods, mainly including DDPG, A2C, TRPO and PPO. Different from the above two methods, the Actor-Critic algorithm needs to optimize the parameter of both policy network and value network in one training episode, and obtains the best behavior through the interaction between them.

Among all Actor-Critic algorithms, PPO has achieved excellent performance in coping with decision-making problems in both discrete and continuous action space. Due to its superior performance and universal applicability, OpenAI takes PPO as the baseline algorithm to attempt to figure out unprecedented problems. In this paper, PPO is chosen to establish the decision-making module of AI scheduler to realize dynamic and adaptive scheduling in the multi-agent manufacturing system.

The significant advantage of RL algorithm is to optimize sequential decision-making problem. Generally, the sequential decision-making problem can be described by Markov Decision Process (MDP), that is, the state S_{t+1} of the next timepoint $t+1$ is merely determined by the state S_t of current timepoint t and the taken action a_t. In workshop environment, the goal of production planning and control is to assign the workpiece tasks to the idle machines for processing, and the machine selection strategy completely depends on the current workshop state. The change of workshop environment is only related to the state and production action at the current moment, and has nothing to do with the previous state. Therefore, the production planning and control (PPC) problem can be modeled as MDP. Job shop scheduling is not only a PPC problem, but also a discrete problem. At present, the design of action space in JSP mainly consists of two kinds: machine selection and dispatching rule selection. RL algorithms have been applied to solve the PPC problems, mainly including DQN [39], Q-learning [40], SARSA [41], contextual bandit [42], and TRPO [43]. In addition, a series of remarkable achievements have been made by RL in solving other types of discrete decision-making problems such as e Bin Packing Problem [44], Traveling Salesman Problem [45] and Vehicle Routing Problem [46]. To sum up, RL methods shows the great potential in solving discrete and sequential decision-making problems.

2.4. Research gaps

From the above literature review, significant progress has been made in the research fields of multi-agent manufacturing system, workshop scheduling and RL. However, there are still the following deficiencies worthy of further improvement:

(1) The traditional RL-based scheduling method is a centralized architecture, and makes a request to the central workshop server when the scheduling behavior is needed. In this way, the workload of the central server is too large, which is not conducive to solving large-scale problems and frequent disturbances. However, a distributed scheduling architecture is proposed in this paper based on multi-agent technology. The state space of workshop environment is obtained by means of self-organizing negotiation mechanism, and then the equipment agent calls its own scheduling module to complete task allocation.

(2) The decision-making model of the scheduler of multi-agent manufacturing system is invariable, lacking adaptability and intelligence. In this paper, a self-learning strategy of scheduling model based on PPO algorithm is proposed, and the scheduling module of each equipment agent is regularly updated by means of cloud-edge coordination.

(3) Most of the previous RL-based scheduling methods merely focus on a single optimization goal, ignoring other goals such as workload balance. However, the workshop managers pay more attention to the comprehensive metrics of decision results. In this paper, a composite reward function is designed to improve the performance of the scheduling model in terms of time, cost and workload.

3. Research methodology

3.1. The architecture of multi-agent manufacturing system

The overall architecture of a multi-agent manufacturing system is shown in Fig. 2, and is divided into three parts: manufacturing resources, workshop system and cloud server. The manufacturing resources mainly consist of manufacturing equipment, warehouse, robots, workpieces, and ubiquitous sensors. The workshop system is responsible for controlling various types of manufacturing resources to complete the personalized orders, and is composed of the construction of equipment agents, negotiation strategies among agents, and intelligent decision-making methods in unpredictable situations. The cloud server is responsible for decomposing the personalized orders and monitoring the equipment workload, production progress and exception alarms. In addition, the scheduling experience data stored in the database is employed to train and optimize the decision model by the cloud server. The updating of decision model of AI scheduler and the insertion of production tasks are accomplished in a cloud-edge collaboration manner.

In a multi-agent manufacturing system, the machine, workpiece, and functional modules in the workshop are abstracted and constructed as different agents, namely, machine agent (A_M), part agent (A_P), warehouse agent ($A_{AS/RS}$), logistics agent (A_{AGV}). Each agent has its own unique function depending on its characteristics.

$A_{AS/RS}$ represents an abstraction of the automated storage and retrieval system (AS/RS). AS/RS is composed of raw material warehouse and finished goods warehouse. The main function of $A_{AS/RS}$ is to receive orders from the cloud platform, sort these orders according to the priority scores, and subsequently initiate the first-round negotiation process of the selected workpiece tasks with the highest priority. The logistics system of workshop is composed of several automated guided vehicles (AGVs). A_{AGV} represents the functional module of logistics system, and is applied to respond to the transportation request and generate a transportation scheme. The assignment of transportation tasks is autonomously performed by the scheduling module within the logistics system. Once the starting point and destination of transportation task are determined, the most suitable AGV will be automatically selected by the internal decision model of A_{AGV} to carry out the workpiece transportation task. The workpiece and a pallet with a RFID tag are integrated to form a smart workpiece. A_P represents an abstraction of a smart workpiece, and is applied to record the task attribute and the whole lifecycle information of a workpiece through RFID tags.

The manufacturing equipment is the main executor of processing tasks, and is composed of machine tool, buffer, and robot. An industrial personal computer (IPC) is embedded in the manufacturing equipment to establish the smart equipment. A_M represents an abstraction of the smart machine. The operation logic of a machine agent is illustrated as Fig. 3. It is noteworthy that A_M is the most critical component in a multi-agent manufacturing system. For the machining equipment, A_M not only interacts with the equipment entities through the TCP/IP, but also controls the execution of machining tasks, such as controlling the robot to grasp a workpiece from the buffer to the machine workbench, transmitting the specific NC codes to the CNC machine tools, and fetching the progress of machining tasks. For other machine agents, A_M is the primary participant in the cooperation and competition process of production tasks. The CNP is applied to guide various agents to perform the self-organizing negotiation, and its detailed description will be presented in Session 3.3. In addition, A_M requests the customized NC codes from the cloud server according to the operation attributes information on the RFID tag of the workpiece, and periodically download and update the latest decision model parameters of the scheduler from the cloud server.

As shown in Fig. 3, the internal structure of a smart machine agent consists of communication layer, adaptation layer and analysis layer.

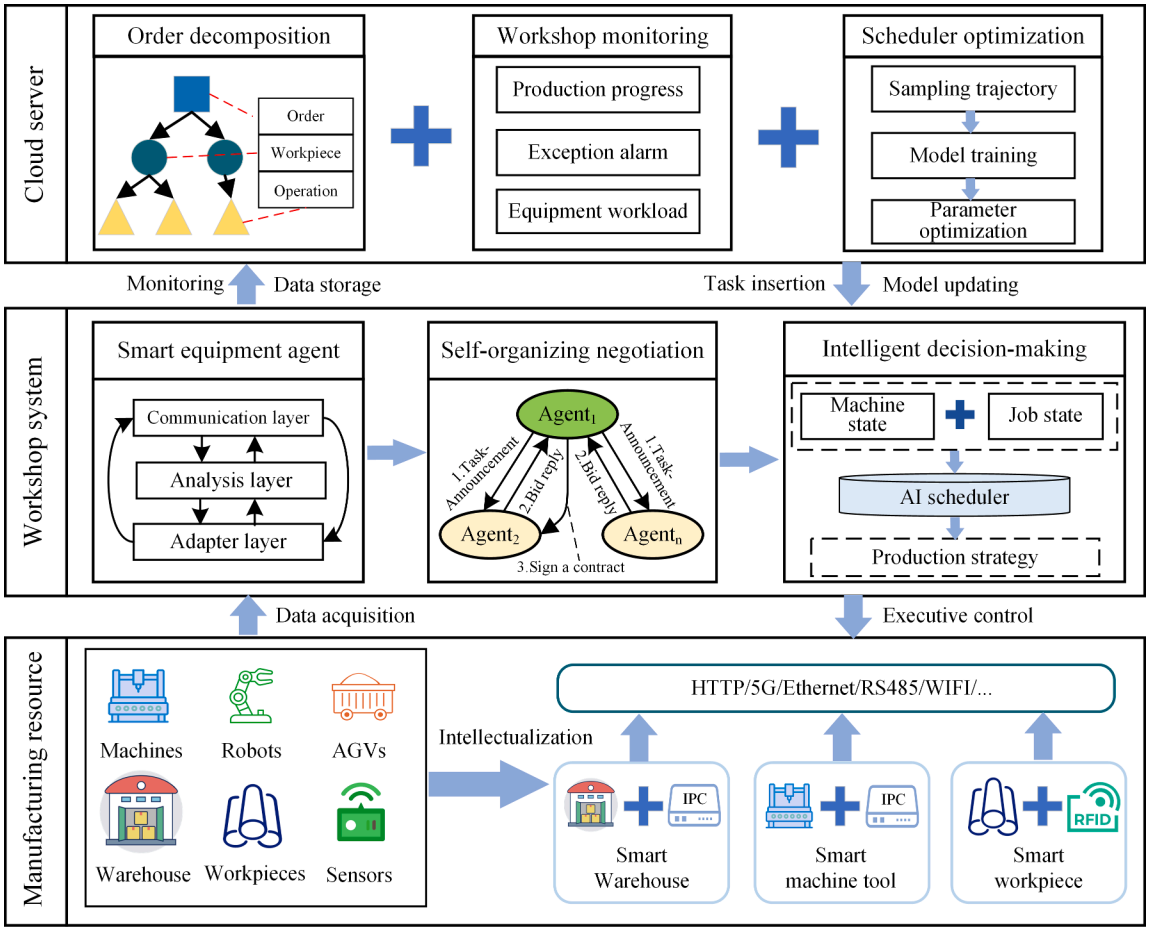


Fig. 2. The architecture of multi-agent manufacturing system.

The communication layer is a bridge for a machine agent to interact and communicate with other agents and the cloud sever through the IoT network (such as Ethernet and WiFi). The adaptation layer is applied to support the machine agent to collect the operational data of corresponding equipment entities equipped with various communicate interfaces and numerical control protocol, and issue the control instructions to the specific robot. The main function of analysis layer is to analyze and evaluate the communication content among various agents in the negotiation process. With the support of edge computing devices and IoT, equipment agents (e.g., $A_{AS/RS}$ and A_M) possess various autonomous capabilities such as environment awareness, information sharing, data analysis and decision-making. Both $A_{AS/RS}$ and A_M are embedded with decision-making modules called AI scheduler in this paper, which are used to produce optimal decision results after the self-organizing negotiation process. AI scheduler is essentially a policy approximator based on deep neural network. Its responsibility is to choose an appropriate idle machine to perform the production tasks according to the input equipment working status and specific task demand. The trigger conditions of AI scheduler mainly include: the completion of a certain operation task of work in progress (WIP), the insertion of new work orders into the manufacturing system, and machine breakdown.

The execution flow of the multi-agent manufacturing system is as follows: First, cloud platform delivers the customer orders to workshop production system. The production system releases tasks from the scheduling list one by one according to the priority, and initializes the RFID information of A_P according to the operation attributes. Then, a task-announcement of the work order to be scheduled is established by $A_{AS/RS}$ and sent to the all available A_M. $A_{AS/RS}$ conducts bidding with other equipment agents through the CNP-based negotiation mechanism, and observes the working status and processing capacity of available manufacturing equipment in the workshop. After the negotiation among agents is finished, the latest state information of workshop environment is obtained naturally, and the AI scheduler of $A_{AS/RS}$ generates a production strategy according to the state information. When an operation task assigned to the manufacturing equipment is completed, A_M will launch a new round of negotiation process with other equipment agents. Subsequently, the production strategy is generated through the AI scheduler of A_M. Repeat the same procedures until all production tasks are completed. Finally, all AI schedulers of the agents are trained and optimized by the self-learning mechanism based on deep reinforcement learning, so that the scheduling system can adapt to changeable workshop environment and efficiently perform personalized production tasks. The self-organizing negotiation mechanism and self-adaptive decision-making method will be described in detail in Sections 3.3 and 3.4.

3.2. Problem formulation

There is a set of heterogeneous machining equipment$M = \{M_1, M_2, ..., M_M\}$ in a workshop, each of which may be equipped with the same or different processing capacity (i.e., turning, milling or grinding). The common tasks of machining equipment are to process N workpieces $J = \{J_1, J_2, \ldots, J_N\}$ with different arrival time and due time, and each workpiece J_i consists of multiple operations $O = \{O_{i,1}, O_{i,2}, \ldots, O_{i,j}, \ldots, O_{i,H_i}\}$.

A same type of workpiece production task goes through the same process route. The ideal workload time consumed by each operation of workpieces has been estimated by the cloud server, which is denoted by

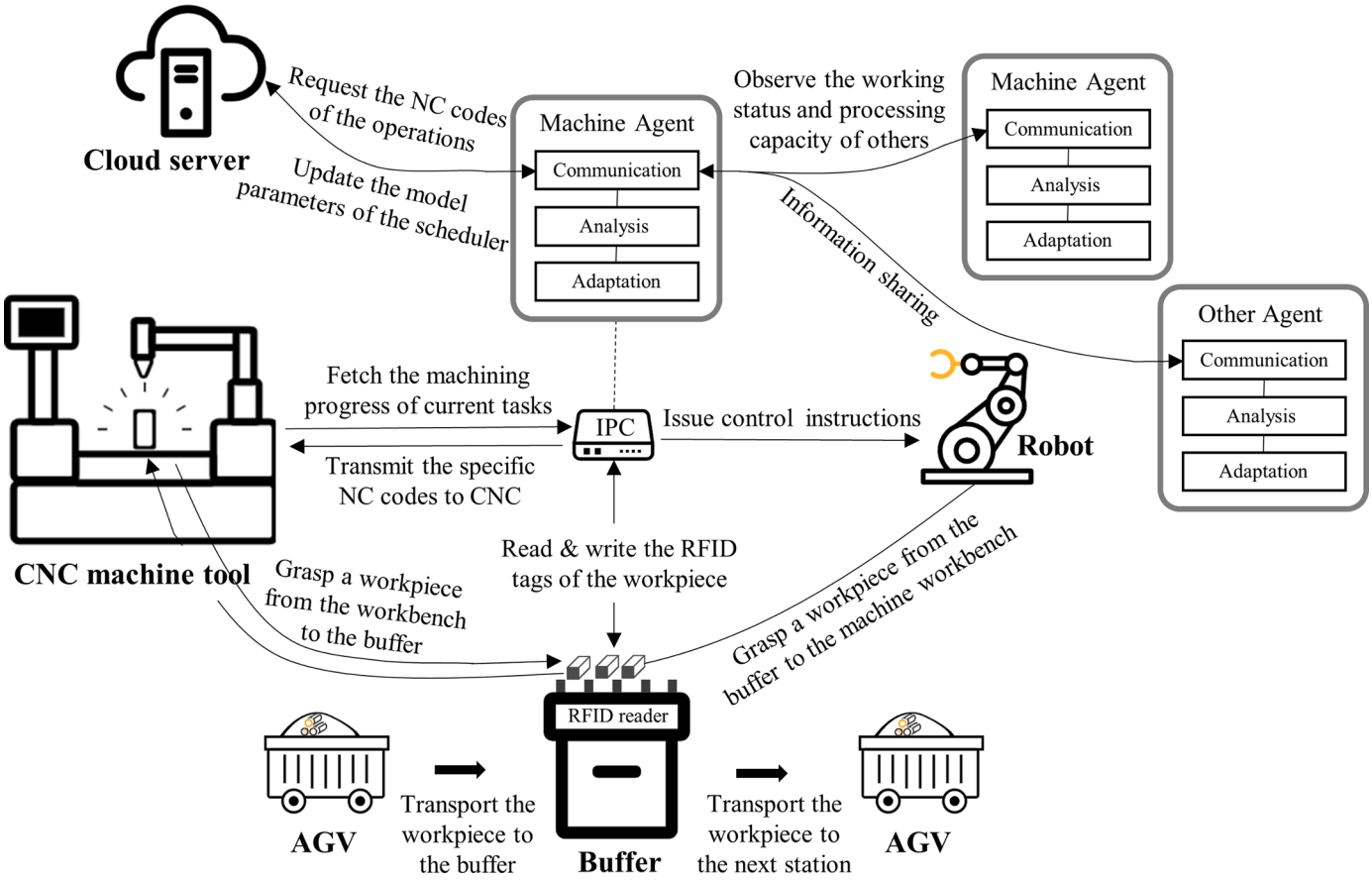


Fig. 3. The operation logic of a machine agent.

$T_{i,j}$. Nevertheless, the occupied machining time $T_{i,j,m}$ and consumed processing cost $P_{i,j,m}$ of the same operation task are distinct when assigned to machining equipment with different speed factor $K_m^{(S)}$ and energy efficiency factor $K_m^{(E)}$. The nominal operating time $\widetilde{T}_{i,j}$ denotes the sum of set-up time of machine, move time between two work stations and processing time of workpiece, and is set to $\widetilde{T}_{i,j} = 3T_{i,j}$ in this paper.

The goal of workshop scheduling is to assign each operation task $O_{i,j}$ of various workpieces to appropriate equipment M_m for processing, and ultimately achieve comprehensive and superior performance in terms of time, cost, and resource utilization once all production tasks are completed. In the dynamic workshop environment, there are unpredictable events such as order insertion and machine failure, which have a bad influence on the development of production activities. Under the disturbance of orders and resources, it is necessary to consider how to generate a reasonable scheduling scheme while meeting the requirements of multi-objective optimization.

The mathematical model of workshop scheduling is as follows:

$$T_{i,j,m} = K_m^{(S)} \times T_{i,j} \tag{1}$$

$$C_{i,j} = S_{i,j} + T_{i,j,m} \tag{2}$$

$$C_{max} = \left\{ \max_{1 \le i \le N} C_i \right\} \tag{3}$$

$$st \begin{matrix} C_{ij} - C_{i,j-1} \ge T_{i,j,m} \times x_{i,j,m} \\ C_{ij} > 0, i = 1,2,..,N \end{matrix} \tag{4}$$

$$\sum_{m \in \Omega_{i,j}} x_{i,j,m} = 1, \forall i,j \tag{5}$$

where $T_{i,j,m}$ indicates the occupied workload time of operation O_{ij} assigned on specific machining equipment M_m. $C_{i,j}$ denotes the completion time of operation O_{ij}, and C_i denotes the completion time of job J_i. C_{max} indicates the maximal completion time of all jobs so far. Eq. (4) guarantees the precedence constraints between two operations. Eq. (5) indicates that each operation can be machined on one of available machines set $\Omega_{i,j}$.

Hypotheses considered in this paper are summarized as follows:

(1) All machines are available at time 0;

(2) The machine can process only one workpiece at a moment;

(3) Job move time between stations is not considered separately and estimated as a part of processing time;

(4) Each type of job has a unique process route;

(5) The buffer of each machine possesses finite capacity.

To formulate the workshop scheduling, the notations used in this paper are defined as Table 1.

3.3. Self-organizing negotiation mechanism

The design of negotiation mechanism is the key factors to promote the successful implementation of multi-agent manufacturing system in an actual factory. The negotiation mechanism is employed to support the cooperation and competition among equipment agents (i.e., $A_{AS/RS}$ and various A_M), and assist multiple machine tools to efficiently complete the production tasks together. CNP introduces the bidding mechanism from the market to form a contractual relationship between the task initiating node and the task executing node within the network, and is suitable for dynamic assignment of tasks among nodes when solving distributed scheduling problem. In addition, CNP is simple and easy to implement, and can be flexibly deployed in a multi-agent system to adapt to various situations. Therefore, CNP is adopted as the negotiation mechanism among equipment agents in this paper.

The detailed description of multi-agent negotiation mechanism based on CNP is shown in Fig. 4:

Step1: When customers submit personalized orders to cloud platform, the orders are decomposed into part-level production tasks by the order management system. Firstly, the $A_{AS/RS}$ sorts the part-level tasks according to the arrival time, order priority, and delivery date. Then, the workpiece task with the highest priority is released from the schedule of $A_{AS/RS}$, and an A_P is generated in the multi-agent manufacturing system subsequently. At this time, the scheduling event is triggered, and the $A_{AS/RS}$ is ready to negotiate with available A_M in the next step.

Step2: Each workpiece task contains multiple operations. The $A_{AS/RS}$ encapsulates the processing technology type and order delivery date of the first operation into a task-announcement, searches the A_M with processing capacity of the operation in the JADE platform, and sends the

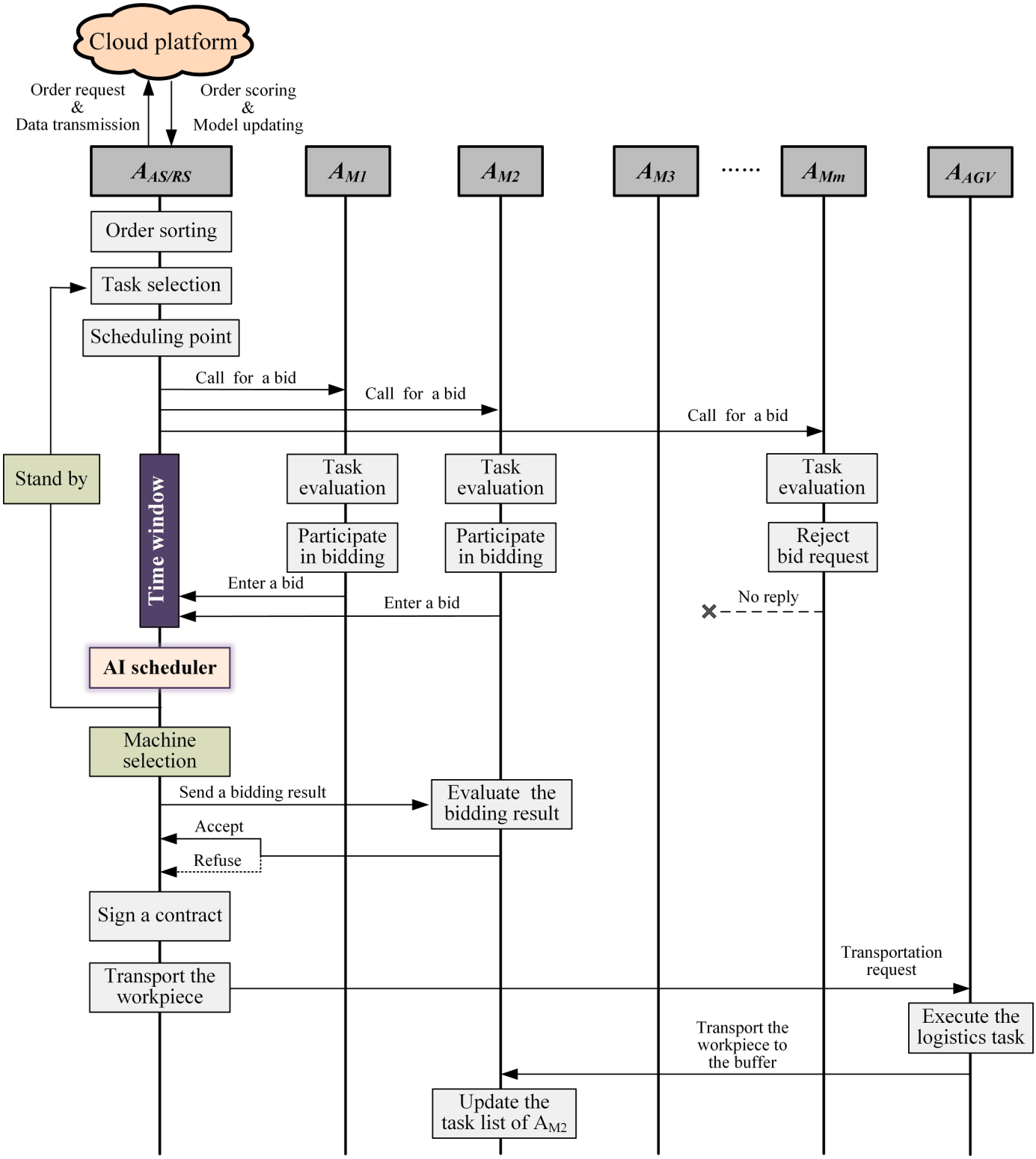


Fig. 4. Self-organizing negotiation process among multiple agents.

task-announcement to all available A_M (such as A_{M1}, A_{M2}, and A_{Mm}).

Step3: After receiving the task-announcement, the A_M extracts and analyzes the production task contained in it, and obtains the operation attributes of the workpiece to be scheduled (i.e., operation type, expected completion time, and order urgency). Then, based on the remaining buffer length and health status of the machine tool, it is evaluated that whether the machine tool is suitable for processing a new production task under the current situation. As shown in Fig. 4, due to the failure of the machine M_m or its buffer being full, the result of task evaluation is that the M_m is not suitable to undertake a new production task, and the A_{Mm} will not reply the tender to the $A_{AS/RS}$. The machine agents participating in the bidding (i.e., A_{M1} and A_{M2}) measure the performance of completing the production task based on the workload and processing capacity of the machine tools and their buffer occupancy, and the evaluation results are encapsulated into bidding documents and fed back to the $A_{AS/RS}$.

Step4: The $A_{AS/RS}$ collects all received bidding documents within a certain time window, and analyzes the contents of these bids. According to analysis results, the occupation time of available machine tools and the estimated completion time of the operation can be obtained to update the state space of workshop environment. In the state space, the occupation time of the machine tools that do not participate in bidding is set to a large value, such as 999. In addition, the amount of their remaining buffers is set to 0. In the case, the unqualified machine tools will not be selected to perform the production task by the AI scheduler.

Step5: The updated workshop state is input to the AI scheduler of $A_{AS/RS}$, and then a decision-making result will be generated, that is, a production action. If the chosen production action is “standby”, it means that the $A_{AS/RS}$ has not received valid bids from the equipment agents or the workshop equipment has no idle capacity. Steps 2 to 5 will be repeated until there are valid bids and idle capacity. If the chosen production action is “machine allocation”, the $A_{AS/RS}$ sends a notice of winning the bid to the successful bidder A_{M2}.

Step6: After receiving the bid-winning notice, the A_{M2} determines whether to sign a contract with the $A_{AS/RS}$ according to its current buffer load and working status. If the cooperation is confirmed, the buffer length of the corresponding equipment M_2 will be reduced by 1, and the information of new task will be stored in the task list of A_{M2}. Otherwise, the negotiation process of this round ends.

Step7: After receiving the confirmation reply from the A_{M2}, the $A_{AS/RS}$ initiates a cooperation request to the A_{AGV} for transporting the workpiece. According to task requirements, the most appropriate logistics equipment is automatically chosen to carry out transportation operation by A_{AGV}.

Step8: The selected logistics equipment transports the workpiece to the buffer of designated machine M_2, and the task list of A_{M2} is updated subsequently. Then, the A_{M2} executes the production tasks in the order of its task list. So far, the first round of the negotiation process has been finished, and the remaining negotiation steps continue until the order list of manufacturing system is empty.

In addition, the process of cooperation and competition between equipment agents will begin when the scheduling events (i.e., operation completion and machine failure) are triggered.

When the scheduling event is "operation completion", the machine agent (i.e., A_{M2}) needs to assign a new machine tool for processing the next operation of the workpiece. If all the operations of the workpiece have been completed, the A_{M2} sends a transportation request to the A_{AGV}, and the A_{AGV} assigns an optimal AGV to transport the workpiece to the warehouse. Otherwise, the A_{M2} obtains the attribute information of the next operation, encapsulates the information into a task-announcement, and sends it to all available machine agents. After receiving the task-announcement, the machine agents extract and analyze the production task contained in it, and determine whether to participate in bidding based on the buffer length and health status of the corresponding machine tool. The machine agents that participate in bidding measure the performance of completing the production task based on the processing capacity and workload of the corresponding machine tools. According to the evaluation results, the AI scheduler of the A_{M2} selects the most suitable machine agent to perform the processing task. Then, the selected machine agent initiates a cooperation request to the A_{AGV} for transporting the workpiece. When the scheduling event is "machine failure", the machine agent (i.e., A_{Mm}) judges whether there are any unfinished tasks in its task list. If the task list of A_{Mm} is empty, the A_{Mm} enters the standby state and cannot participate in the negotiation process until the machine M_m returns to normal work. Otherwise, the A_{Mm} schedules the unfinished tasks in the order of task list through negotiation mechanism. First, the A_{Mm} encapsulates the attribute information of the next operation into a task-announcement and sends it to available machine agents. After receiving the task-announcement, the machine agents analyze the production task contained in it, and evaluate the task based on the processing capacity and workload of the corresponding machine tools. The AI scheduler of A_{Mm} selects the optimal machine agent to perform the processing task based on the evaluation results. Then, the A_{AGV} is requested by the selected machine agent to transport the workpiece. So far, two kinds of scheduling events are dealt with through self-organizing negotiation mechanism.

3.4. Self-adaptive decision-making mechanism

After finishing the self-organizing negotiation, the machine agent responsible for the evaluation of bids could receive different tenders from other machine agents. It is crucial for a machine agent to make decisions in selecting the most appropriate machine tool to carry out the production task based on these tenders. Traditionally, multiple evaluation metrics (such as completion time, tardiness time, workload and processing cost) are linearly combined to build a multi-objective decision-making model, so as to fulfill task allocation. The parameters of above decision-making model are constant and fixed, and it is easy to fall into local optimum. Moreover, the satisfaction of managers’ preferences depends on modifying the weight coefficients of the decision-making mathematical model.

To solve the above problems, an adaptive decision-making method called AI scheduler based on deep reinforcement learning is proposed in this paper. Fig. 5 illustrates the adaptive decision-making process of AI scheduler. Once the state information of job and machines are input, the decision model of AI scheduler automatically generates an appropriate production action to fulfill the task allocation. During the task execution process, AI scheduler is also capable of continuously optimizing the weight parameters of neural networks according to the feedback reward. In addition, the effectiveness of RL-based scheduling model depends on the design of three key elements: state space, action space and reward function.

(1) State space

State space is an accurate representation of workshop environment characteristics and the key basis for AI scheduler to make decisions. At the scheduling moment t, the state space S_t of workshop environment consists of operation attributes O_t of the workpiece J_i to be scheduled and working status M_t of all available machining tools, and can be described as $S_t = \{O_t, M_t\}$. The operation attributes of the workpiece to be scheduled $O_t = \{O_{1t}, O_{2t}, O_{3t}\}$ are composed of processing technology type O_{1t}, ideal operating time O_{2t} and order urgency O_{3t}, and the dimension of job state is 3. O_{1t} denotes the numerical expression of operation types (i.e., 1: turning, 2: milling). O_{2t} indicates the workload time required to complete the operation. Order urgency O_{3t} indicates the difference between due time and current timepoint t, as shown in Eq. (6).

$$O_{3t} = \frac{\widehat{T}_{i,j}^{(C)} - t}{\sum_{j<N_i} T_{i,j}} \tag{6}$$

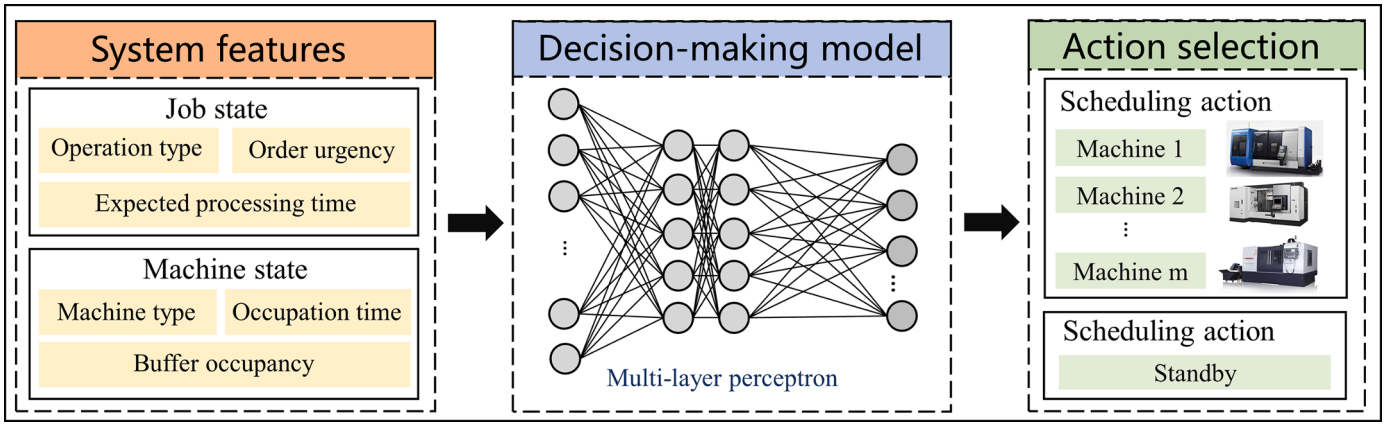


Fig. 5. Adaptive decision-making process of AI scheduler.

where $\widehat{T}_{i,j}^{(C)}$ denotes the due time of operation $O_{i,j}$ and $\sum_{j<N_i} T_{i,j}$ is the sum of workload time of the remaining operation of workpiece J_i.

The state of machine tools $M_t = \{M_{1t}^1, M_{2t}^1, M_{3t}^1, \ldots, M_{1t}^m, M_{2t}^m, M_{3t}^m\}$ consists of machining capacity M_{1t}, buffer occupancy M_{2t} and occupation time M_{3t}. m is the total number of various machine tools in the workshop, and the dimension of their state space is $m \times 3$. M_{1t} denotes the numerical representation of machine types (i.e., 1: lathe, 2: miller). The buffer occupancy indicates the bearing capacity of a machine tool, and is equal to the remaining amount of the buffer. The occupation time M_{3t} of machine tool is equal to the cumulative nominal operating time consumed by processing tasks in the schedule of corresponding machine agent, and indicates the waiting time of a machine tool to perform a new task.

(2) Action space

At a scheduling moment t, policy network $\pi(a_t \mid s_t)$ of AI scheduler immediately generates a decision-making result based on the workshop state S_t, that is, the probability distribution of production action space A_t. The action space is a collection of all optional production actions $A_t = \{a_0, a_1, \ldots, a_m\}$ for AI scheduler in a workshop environment, as shown in Fig. 5. In this paper, there are two kinds of actions within A_t: standby "a_0", and machine allocation "$a_1, \ldots, a_m$". The standby action indicates that the workload of machine tools is heavy (i.e., high buffer occupancy rate and long equipment occupancy time), and it is not suitable to insert new tasks into manufacturing system. If standby action is taken, machining equipment continues to execute the production tasks of its own buffer without accepting new tasks until a certain processing task is completed or the next scheduling moment is triggered. The machine allocation action a_m denotes that the production task to be scheduled is assigned to a designated machine M_m for processing.

(3) Reward function

The reward function is employed to evaluate the decision-making result generated by the AI scheduler, that is, to appraise the influence of selected production action a_t on the workshop environment. It is beneficial to train and optimize the parameters of neural networks in the AI scheduler, and continuously improve the decision-making performance of the AI scheduler in different states. The design of reward function consists of two parts: composite reward and penalty. If the production action chosen by AI scheduler is valid, manufacturing system will feed back a composite reward to the AI scheduler. As shown in Eq. (7), the modeling of a composite reward depends on three key factors: cumulative ahead-of-schedule time r_{tard}, workload balance standard deviation r_{load} and product profit r_{profit}. Otherwise, if the action selected is invalid, e.g., assigning task A to machine B when machine B does not have sufficient capacity to complete task A, a large penalty value will be fed back to AI scheduler. The penalty value is set to -50 in this paper. The goal of reinforcement learning is to maximize the reward value r_t of each training episode.

$$r_t = \begin{cases} -50 & \text{invalid action} \\ w_1 \times r_{tard} - w_2 \times r_{load} + w_3 \times r_{profit} & \text{order to machine or standby} \end{cases} \tag{7}$$

$$r_{tard} = \frac{\widehat{T}_{i,j}^{(c)} - T_{i,j}^{(c)}}{\widetilde{T}_{i,j}} \tag{8}$$

Where $\widehat{T}_{i,j}^{(c)}$ and $T_{i,j}^{(c)}$ are the expected completion time and actual completion time of operation $O_{i,j}$, and $\widetilde{T}_{i,j}$ is the nominal operating time of operation $O_{i,j}$. If the actual completion time of operation is earlier than expected, r_{tard} will be positive. The goal of AI scheduler is to maximize the value of r_{tard}.

$$LD_m = \int_{t_0}^{t} \sum_{i,j} K_m^{(S)} \times T_{i,j} \tag{9}$$

Where LD_m represents the cumulative workload time of machine M_m from timestep t_0 to t.

$$r_{load} = \sqrt{\frac{\sum_{m \in c} (LD_m - \overline{LD})^2}{M_c}} \tag{10}$$

Where r_{load} indicates the workload deviation between isomorphic machine tools, $\overline{LD}$ represents the average workload time of isomorphic machine tool., and M_c denotes the number of types of machine tools.

$$r_{profit} = e - K_m^{(E)} \tag{11}$$

Where r_{profit} indicates the production profit of the work order.

(4) Training method of AI scheduler

The stochastic fluctuation of orders and resources leads to the gradual deterioration of AI scheduler's decision-making performance, and makes it impossible to generate the optimal production scheme in a dynamic workshop environment. Therefore, a self-learning mechanism of AI scheduler based on the PPO algorithm is proposed in this paper. The self-learning process of decision-making model is divided into two stages: pre-training in a simulation workshop environment and training in an actual workshop environment.

At the pre-training stage, AI scheduler interacts with the simulation environment in a trial-and-error manner. According to the obtained decision results and rewards, the decision-making model of the AI scheduler has been continuously trained and optimized. Then, the trained decision-making model is deployed in a multi-agent manufacturing system after reaching the convergence, and is applied to guide multiple machine agents to complete the task allocation. In the simulation environment, AI scheduler randomly takes actions based on the probability distribution of action space. The reason is that it is not

certain which action is good or bad before the end of a training episode. Stochastic policy is beneficial to fully explore the action space of the environment and accelerate the convergence speed of training the decision-making model. At the training stage, AI scheduler gains the probability distribution of action space based on actual workshop state, and selects an action with the highest probability to guide the equipment to perform production tasks. When encountering abnormal events (such as new job insertion and machine failure), AI scheduler observes the characteristics of tasks and resources in the workshop environment, and then runs the policy network $\pi(\theta)$ to generate an optimal production action to guide equipment to solve the disturbance. Over the scheduling process, workshop system records the sampling trajectory of the AI scheduler, and periodically modifies and optimizes the parameters of decision-making model in the cloud server.

Essentially, PPO is a kind of Actor-Critic algorithm, including two different modules named Actor and Critic. Actor network is similar to DQN, and consists of new and old neural networks with the same structure. The parameters of the old neural network are replaced with the new one after a specified number of timesteps. Critic is mainly employed to assess the performance of production actions generated by Actor. The main function of Actor is to generate production strategies and guide equipment to carry out tasks autonomously. As shown in Fig. 6, the output of policy network $\pi(\theta)$ is a SoftMax distribution of the whole action space, that is, the probability value of each production action. Then, a production action is sampled according to the probability distribution, and subsequently executed in the workshop environment. After that, the environment will feed back the next workshop status s_t and corresponding reward r_t to the RL agent. During the model training period, a series of state, action and reward data (s_1, a_1, r_1,…, s_τ, a_τ, r_τ) extracted from sample trajectory are combined into a batch, and the policy network is optimized and updated in a mini-batch manner. In addition, the collected reward set from the trajectory also needs to make a discounted transition.

The self-learning process of the AI scheduler is shown in Fig. 6 and Table 2. The detailed description of key steps is as follows.

Step1: When a scheduling event is triggered, the perceived workshop state s_t is input into the Actor-new network, and Q value table [Q_0, Q_1, Q_2, …, Q_m] of action space is obtained subsequently. These Q values are converted into a probability distribution [P_0, P_1, P_2, …, P_m] of available action set A through the SoftMax operation, and then a production action is sampled according to the obtained probability distribution. The selected production action is executed in workshop environment to guide equipment to perform tasks, and subsequently environment will feed back a reward r_t and latest workshop state s_{t+1}. Meanwhile,

Table. 1
Notations.

Notations	Meaning
M	the total number of machines
N	the total number of jobs
H_i	the total number of operations of job J_i
i, j, m	the index of job, operation, and machine
$O_{i,j}$	the j-th operation of job J_i
$\Omega_{i,j}$	the set of available machines of $O_{i,j}$
$x_{i,j,m}$	the decision variable:1,if $O_{i,j}$ is assigned on M_m;0, otherwise
$T_{i,j}$	the workload time of operation $O_{i,j}$
$T_{i,j,m}$	the occupied workload time of operation $O_{i,j}$ on machine M_m
$S_{i,j}$	the start time of operation $O_{i,j}$
$E_{i,j}$	the end time of operation $O_{i,j}$
C_{max}	the maximum completion time of all work orders

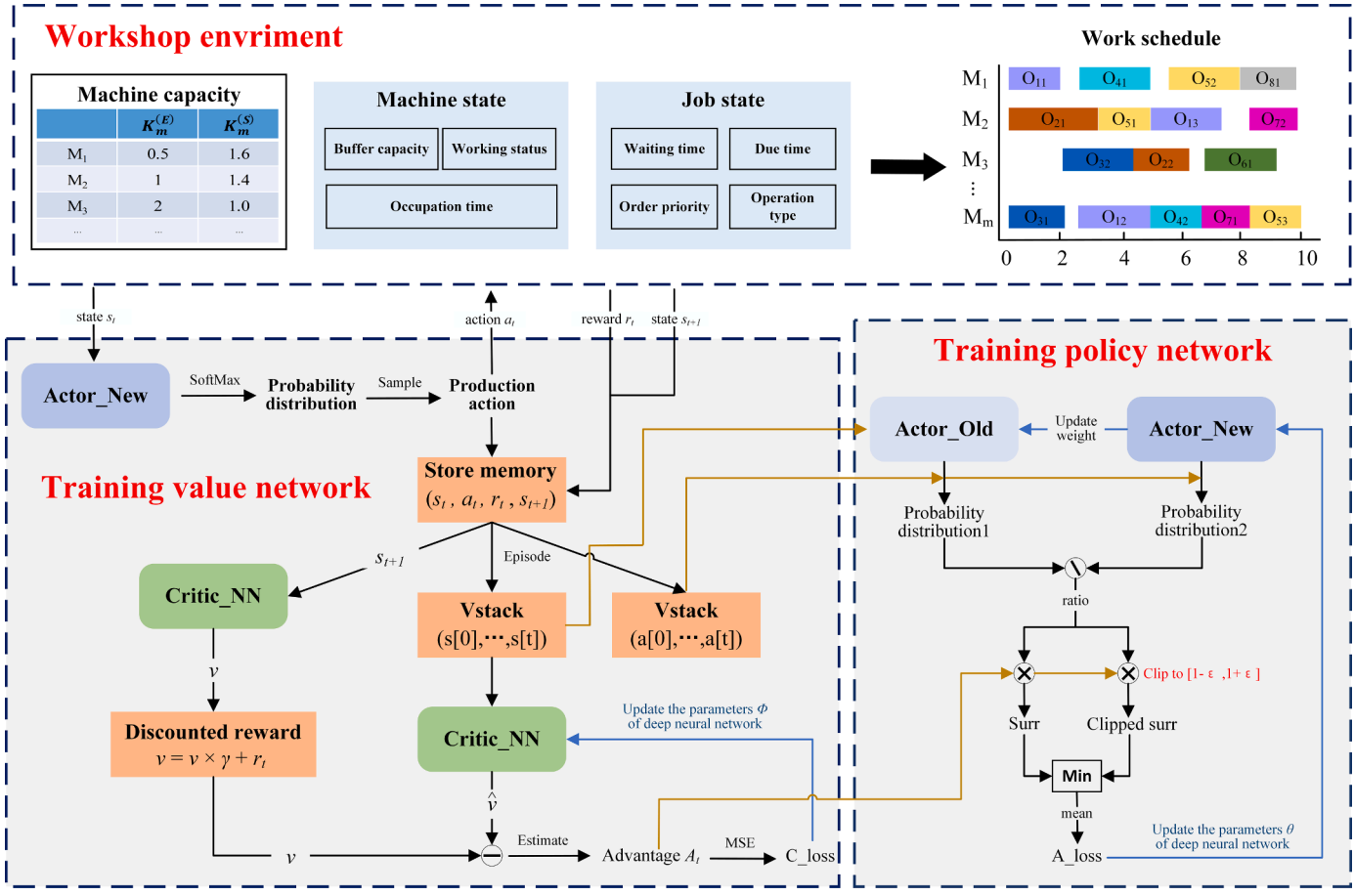


Fig. 6. Self-learning mechanism of decision-making model.

Table. 2
Adaptive optimization method of AI scheduler based on DRL.

Algorithm 1: the pseudo-code of adaptive optimization method of AI scheduler based on DRL

1: Initialization: hyper parameter α_p, α_v, γ, ε and the network parameter of PPO algorithm
2: Generate a simulated job shop environment for training according to predefined criteria
3: For episode number $e = 1, 2, \ldots, N$ do
4: Reset the simulated workshop environment
5: For epoch number $t = 1, 2, \ldots, T$ do
6: AI scheduler observes the state S_t of workshop environment: operation characteristics of the workpiece to be scheduled, running status of machine sets
7: AI scheduler run policy $\pi(\theta)$ to choose a production action a_t based on the workshop state S_t
8: The workshop environment feeds back a composite reward value r_t to AI scheduler after the action a_t executed, and makes a translation to a new state S_{t+1}
9: Store the (s_t, a_t, r_t) in the trajectory buffer $(s_1, a_1, r_1, \ldots, s_\tau, a_\tau, r_\tau)$
10: End for
11: AI scheduler runs critic network and estimates the value of advantage function $\widehat{A}_t$
12: For epoch number $i = 1, 2, \ldots, M$ do
13: Sample mini-batches from the trajectory buffer
14: Update θ by a gradient method to optimize the loss function $J_{ppo}(\theta)$
15: End for
16: For epoch number $j = 1, 2, \ldots, B$ do
17: Sample mini-batches from the trajectory buffer
18: Update $\varnothing$ by a gradient method to optimize the loss function $L_{BL}(\varnothing)$
19: End for
20: Replace the parameters of actor network $\pi(\text{old}) \leftarrow \pi(\theta)$
21: End for

previous scheduling data will be stored in the sample trajectory $[(s_1,a_1,r_1),\ldots,(s_T,a_T,r_T)]$. In addition, the above operations are repeated until a certain amount of operation data has been stored. It is worth noting that the Actor-new model has not been updated so far.

Step2: The workshop state s_t perceived at the last timestep T of the loop in Step1 is input into the Critic network to acquire the state value $V_\varnothing(s_T)$. Then, a reward set $R = \{R_0, R_1, R_{T-1}, R_T\}$ will be obtained through discounted transformation $R_t = r_t + \gamma * r_{t+1} + \gamma^2 * r_{t+2} + \cdots + \gamma^{T-t+1} * r_{T-1} + \gamma^{T-t} * V_\varnothing(s_T)$. The advantage function is computed by $\widehat{A}_t = R_t - V_\varnothing(s_t)$, as shown in Eq. (12). $\widehat{A}_t$ is the estimator of advantage function at the timestep t, and it reflects the relative advantage of selected production action compared with other optional actions. The loss function $L_{BL}(\varnothing)$ of Critic network is computed by Eq. (13), and is equal to the mean square error of advantage function value $\widehat{A}_t$. Repeat the above process for m times. Then, Step2 repeat the above process until the number of loops reaches 8.

$$\widehat{A}_t = \sum_{t'>t} \gamma^{t'-t} \times r_{t'} - V_\varnothing(s_t) \tag{12}$$

$$L_{BL}(\varnothing) = -\sum_{t=1}^{T} (\widehat{A}_t)^2 \tag{13}$$

Step3: By inputting a batch of the stored state data into the neural networks of Actor-old and Actor-new, the probability distribution 1 and 2 of action space are obtained, respectively. After that, by inputting a batch of stored actions into the probability distribution 1 and 2, the probability values $prob_1$ and $prob_2$ corresponding to each action are obtained. Then, a ratio is obtained by dividing $prob_1$ and $prob_2$, that is, important weight $\frac{\pi_\theta(a_t|s_t)}{\pi_{old}(a_t|s_t)}$. Important weight indicates the similarity degree between the parameters θ of policy network $\pi_{old}(a_t|s_t)$ of Actor-old and policy network $\pi_\theta(a_t|s_t)$ of Actor-new. Surrogate objective $\widehat{E}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{old}(a_t|s_t)}\widehat{A}_t\right]$ is adopted to optimize the parameters of policy network.

Step4: Clipped surrogate objective is adopted to limit the update range of Actor network over the process of important sampling, so as to avoid getting the abnormal loss function value due to large difference between the output results of $\pi_\theta(a_t|s_t)$ and $\pi_{old}(a_t|s_t)$. Therefore, the loss function $J_{ppo}(\theta)$ of Actor network is computed by Eq. (14). During a training period, a boundary constrain is added to the loss function $J_{ppo}(\theta)$, and the value of important weight is limited to the range $(1-\varepsilon, 1+\varepsilon)$ by clip function $clip\left(\frac{\pi_\theta(a_t/s_t)}{\pi_{old}(a_t/s_t)}, 1-\varepsilon, 1+\varepsilon\right)$. ε is a hyper-parameter, and is generally set between 0.1 and 0.2.

Then, the obtained loss $J_{ppo}(\theta)$ is applied to update and optimize the weight parameters of $\pi_\theta(a_t|s_t)$ based on backpropagation approach.

$$J_{ppo}(\theta) \approx \sum_{(s_t,a_t)} \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{old}(a_t|s_t)}\widehat{A}_t, clip\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{old}(a_t|s_t)}, 1-\varepsilon, 1+\varepsilon\right)\widehat{A}_t\right) \tag{14}$$

Step5: After repeating Step 3 and 4 for B times, the updated weight parameters θ of Actor-new network is employed to replace the parameters θ of Actor-old network.

Step6: Finally, Steps 1–7 are repeatedly executed until all training episodes are finished.

4. Experimental results

4.1. A multi-agent manufacturing system testbed

In order to verify the effectiveness and efficiency of proposed approaches, a multi-agent manufacturing system testbed is designed as shown in Fig. 7. The digital workshop model is applied to monitor the manufacturing things in the workshop for managers. There are six different machine tools in the workshop including three lathes and three millers. The processing speed and energy consumption factors of machine tools are vary from each other, and their detailed performance parameters are presented in Table 3.

There are also two small-scale warehouses equipped with automated storage and retrieval system (AS/RS) in the workshop, which are applied to store and pick up raw materials and finished products. Besides, each machine tool is equipped with a buffer zone for temporary storage of WIP, and the length of which is four. A workpiece is placed on a pallet with RFID tag, and its operation attributes have been written by the RFID reader of warehouse when the workpiece manufacturing task is initialized. An RFID reader is installed inside the equipment buffer, and is applied to operate and edit the RFID tag information of workpiece (i.e., reading and writing) during the execution of production. WIP is transported between stations by the AGV.

Industrial personal computers (IPCs) are embedded in the machining equipment, AGV and warehouse respectively, and connected with each equipment entity through the LAN ports and WIFI , as shown in Fig. 8. Various software programs and applications are deployed and installed in an IPC, including data acquisition, protocol analysis, action control, autonomous negotiation, and intelligent decision-making.

Through the adapter module, agent programs in IPC are capable of interacting with machining equipment with heterogeneous numerical control protocols, and deriving the real-time operational data of devices (i.e., machining progress, spindle speed and fault alarm). IPC is also connected with the equipment buffer through LAN port, and obtain occupancy rate of buffer and operation attributes of WIP through TCP/IP. In the scheduling process, agent programs deployed in an IPC deliver bidding documents to other agents through the socket protocol, so as to complete the self-organizing negotiation operation based on CNP. When the machining equipment is ready to carry out the next processing task, the corresponding machine agent issues a JSON-formated control command to a robot through TCP/IP. According to the control instructions, the robot grabs a specific workpiece from the designated position of buffer to the machine workbench. Then, the machine agent requests and downloads the customized numerical control (NC) codes from the cloud server based on the operation attributes recorded on RFID tag. These NC codes are transmitted into the NC system of machining equipment

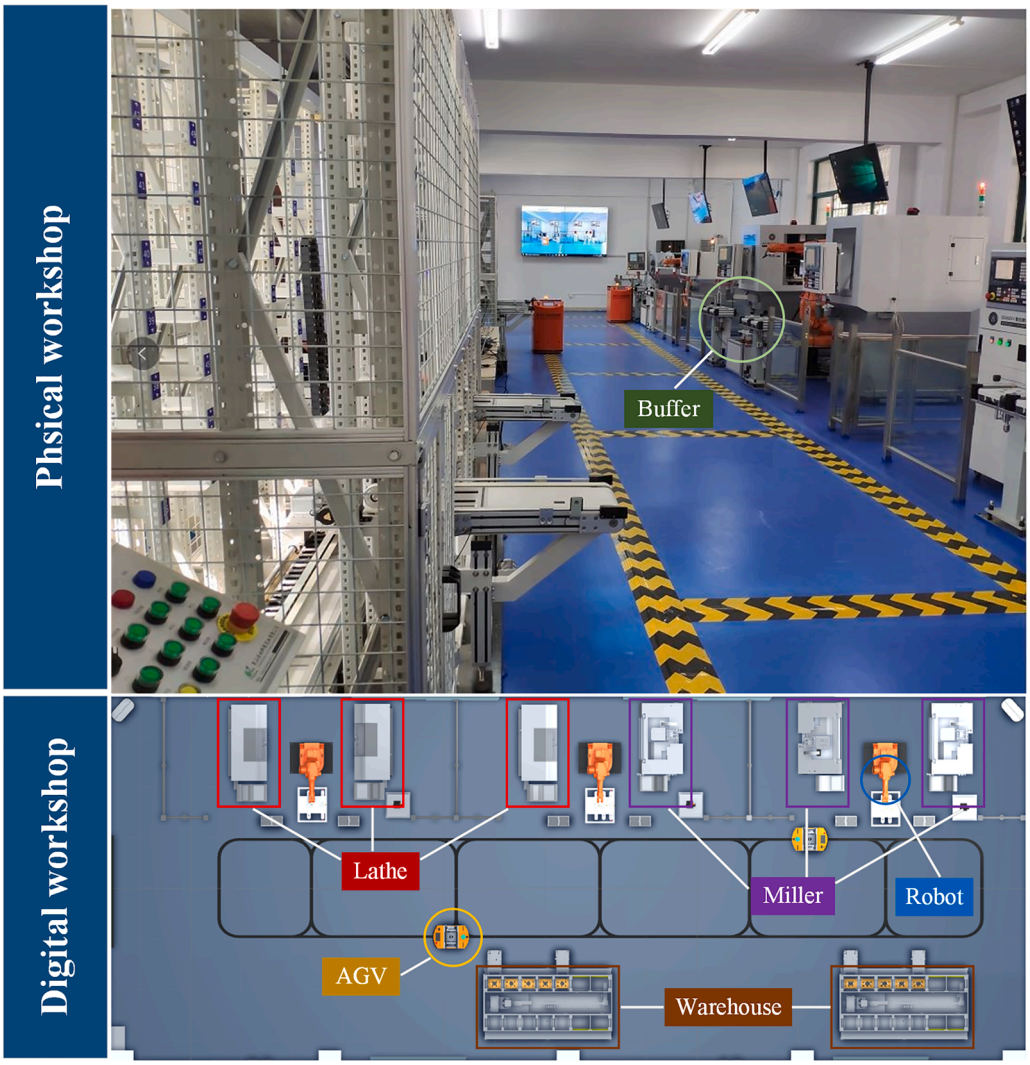


Fig. 7. The overall layout of a smart workshop.

Table. 3
The performance parameters of machine tools in the testbed.

Machine	Type	Speed factor	Energy efficiency factor
M_1	lathe	0.5	1.4
M_2	lathe	1.0	1.2
M_3	lathe	2.0	0.8
M_4	milling	0.5	1.4
M_5	milling	1.0	1.2
M_6	milling	2.0	0.8

through the adapter module, and guide the equipment to carry out machining tasks through a programmable logic controller (PLC) subsequently.

As shown in Fig. 9, there are three types of work orders in the experimental test including shaft, flange and plate. The process route of them are "Turning → Milling", "Turning → Milling" and "Milling". The available machines of turning and milling task are (M_1,M_2,M_3) and (M_4, M_5,M_6), respectively. The product orders submitted by customers within a certain period of time are collected as a test case to evaluate the scheduling efficiency and learning ability of the proposed method. The details of test case are shown in Table 4, and twenty work orders in the case are set with different arrival time and due time.

4.2. Comparison of learning performance under different reward settings

In order to verify the learning performance of the proposed PPO-based method with different reward settings, a comparative experiment is carried out in the test case. In the experiment, the reward value r_t in Eq. (7) is set to 0, −50, or −100 when the action selected is invalid.

The PPO-based scheduling algorithm consists of two types of network structures: manufacturing policy network and value network, and the parameters of proposed method are shown in Table 5. The manufacturing policy network of PPO has an input layer with 21 neurons, two hidden layers with 64 and 64 neurons and an output layer with 7 neurons. The manufacturing value network of PPO has an input layer with 21 neurons, two hidden layers with 64 and 64 neurons and an output layer with 1 neuron. The number of training epochs per update of policy network and value network is 80. The weight coefficient of three factors in the composite reward is set as (1,1,1).

Figs. 10 and 11 depict the learning curves of the accumulated reward and step length of one episode conducted with different reward settings (i.e., 0, −50, and −100), and illustrate the influence of reward value on the learning performance of AI scheduler when the selected action is invalid. The step length of one episode indicates the total number of decisions required to assign all jobs of a test case to a set of machines. When the reward value is set to 0, the proposed PPO-based method obtains the larger convergence value of episode reward than where the reward value is set to −50 and −100, as shown in Fig. 10. In that case, as

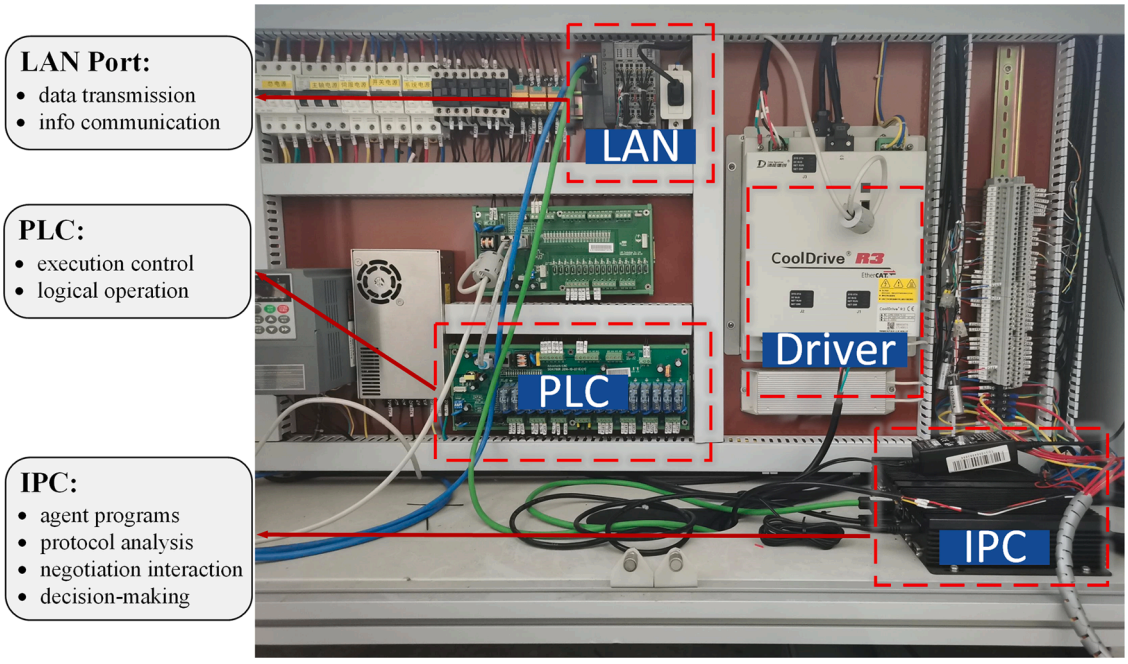
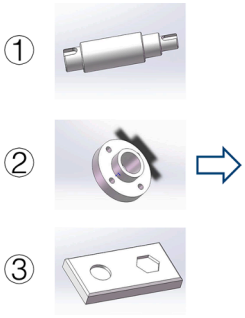


Fig. 8. The internal view of a smart machine.



Job type	Process route	Available machines
Shaft	Turning→ Milling	(M_1,M_2,M_3), (M_4,M_5,M_6)
Flange	Turning→ Milling	(M_1,M_2,M_3), (M_4,M_5,M_6)
Plate	Milling	(M_4,M_5,M_6)

Fig. 9. The process route of jobs and available machines of operations.

Table. 4
Test case.

Index	Job type	Arrival time	Due time	Ideal workload time
1	Shaft	10	151	35,12
2	Flange	45	171	32,10
3	Plate	70	199	43
4	Plate	95	209	38
5	Plate	125	266	47
6	Flange	150	282	30,14
7	Shaft	175	307	33,11
8	Plate	190	340	50
9	Flange	200	368	39,17
10	Plate	210	297	29
11	Flange	235	373	33,13
12	Plate	260	392	44
13	Plate	270	384	38
14	Shaft	295	412	29,10
15	Flange	320	425	26,9
16	Shaft	350	464	28,10
17	Plate	370	475	35
18	Plate	385	496	37
19	Shaft	430	571	31,16
20	Flange	495	651	34,18

Table. 5
The parameter setting of PPO-based method.

Algorithm parameters	Values
Learning rate of policy network α_p	0.0003
Shape of policy network	[21,64,64,7]
Learning rate of value network α_v	0.001
Shape of value network	[21,64,64,1]
Discount factor γ	0.96
Clip range ε	0.2
Number of steps per update	4000
Batch size	50
Optimizer	Adam
Number of training epochs per update	80

shown in Fig. 11, the step length of one episode has been increasing continuously as training proceeds, even reaching the boundary of the maximum length, that is, 4000. The more decisions are made, the more invalid actions are likely to be selected by the AI scheduler. When the total number of decisions in one episode reaches the boundary, it means that the AI scheduler has been unable to generate appropriate production actions to guide equipment agents to complete task allocation. That is to say, the training of AI scheduler cannot really converge when the reward value is set to 0, but falls into the local optimum. When the reward value is set to −50, the proposed PPO-based method obtains the

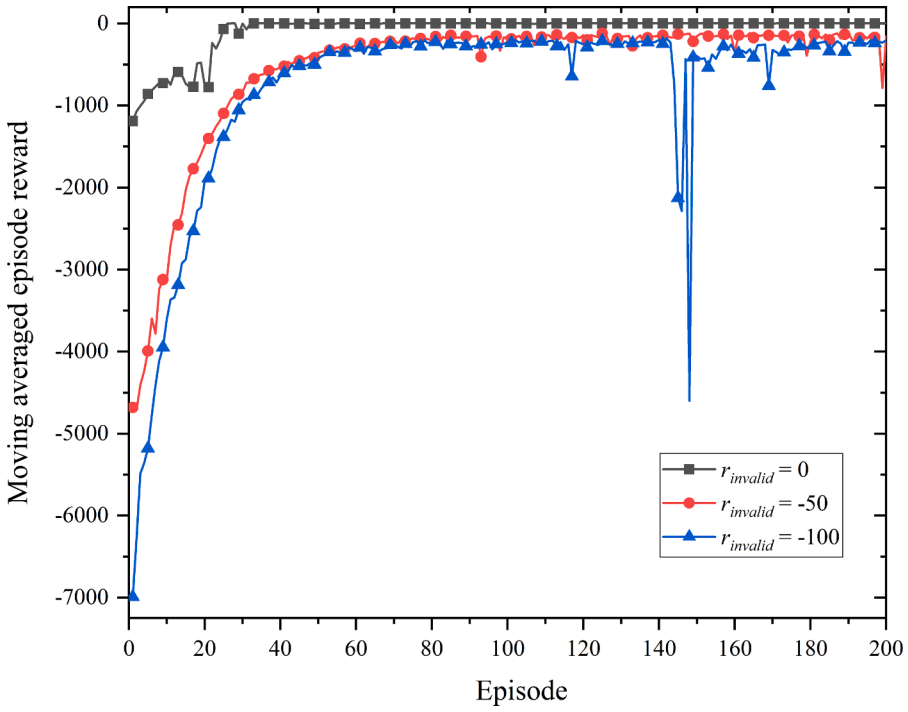


Fig. 10. The learning curves of accumulated reward under three kinds of reward settings.

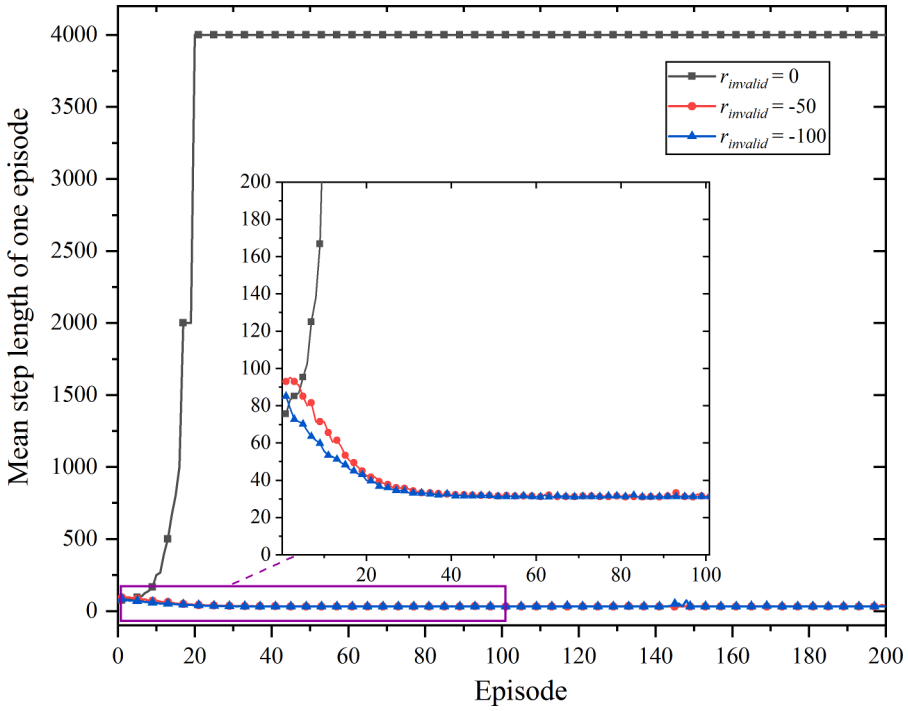


Fig. 11. The learning curves of step length under three kinds of reward settings.

larger convergence value of episode reward than where the reward value is set to −100. The larger the value of accumulated reward, the better the learning performance of the proposed method. As shown in Fig. 11, the step length of one episode obtained by the proposed method is 31 after reaching converges in the 30th episode, which is almost the same as where the reward value is set to −100. In addition, when the reward value is set to −100, the value of episode reward fluctuates greatly during the training process. It is indicated that the decision-making model of AI scheduler is trained and updated in a wrong direction by the proposed method, which results in a poor episode reward. This also means that the learning performance of the proposed method is unstable in that case.

To sum up, when the reward value is set to 0, the proposed PPO-based method falls into local optimum after training, and has lost the ability of accurate decision-making. When the reward value is set to −50, the proposed PPO-based method can obtain the larger value of episode reward and more stable learning performance than where the reward value is set to −100.

4.3. Comparison of learning performance between different scheduling approaches

In order to validate the learning performance of GP-based and RL-based scheduling approaches, a comparative experiment is conducted in the test case. In the experiment, the GP-based, DQN-based, and PPO-based approaches are employed to perform task allocation and make decisions. At a certain interval, the historical scheduling data is applied to train and optimize the decision-making model of AI scheduler, and the total number of episodes is 500 in a training period.

As shown in Fig. 12, the experimental results exhibit that the GP-based, DQN-based, and PPO-based training methods of AI scheduler achieve the convergence of reward value at the 74th, 139th and 65th episode, respectively. Compared with GP-based and DQN-based methods, PPO-based method is 12%, 53% faster in term of convergence speed. The earlier the learning curve converges, the faster the learning speed of the training method is. After reaching convergence, the episode reward of PPO-based method is about −350, while those of GP-based and DQN-based methods are about −368 and −1500. The larger the reward value is, the better the learning effect generated by the training algorithm is. On the whole, the PPO-based method achieves higher episode reward and faster convergence speed than GP-based and DQN-based methods.

Fig. 13 illustrates the learning process of machine load balancing by GP-based, DQN-based, and PPO-based methods. The ordinate indicates the standard deviation of workload time between isomorphic machine tools. The abscissa represents the index of training episode. The result demonstrates that the GP-based, DQN-based, and PPO-based training methods achieves the convergence of standard deviation of isomorphic machine workload time at the 192nd, 121st, and 97th episode, respectively. Compared with GP-based and DQN-based methods, PPO-based method is 49%, 20% faster in term of the learning performance of workload balance. After reaching convergence, the standard deviation of workload time obtained by PPO-based method is about 15, while those of GP-based and DQN-based methods are about 21, 27. The smaller the standard deviation of machine workload is, the better the scheduling performance of AI scheduler is. In summary, PPO-based method achieves better workload balance and faster convergence speed than GP-based and DQN-based methods.

Fig. 14 shows the learning process of order tardiness. The ordinate represents the sum of the differences between the actual completion time and due time of all work orders in the test case. The experimental result reveals that the GP-based, DQN-based, and PPO-based training methods achieves the convergence of order tardiness at the 192nd,126th and 63rd episode, respectively. PPO-based method is 67%, 50% faster than GP-based and DQN-based method in term of the learning performance of order tardiness. After reaching convergence, the value of order tardiness obtained by PPO-based method is about 124, while those of GP-based and DQN-based methods are about 124, 118. The larger the value of order tardiness is, the better the scheduling performance of AI scheduler is. In addition, the convergence curve obtained by DQN-based method fluctuates greatly. It is revealed that the learning effect of DQN-based method is not steady, which leads to unstable scheduling performance of AI scheduler. In conclusion, PPO-based method algorithm achieves better order tardiness and faster convergence speed than GP-based and DQN-based methods.

The learning process of order profit is shown in Fig. 15. The ordinate represents the total profit of all work orders in the test case. The experimental results show that the GP-based, DQN-based, and PPO-based training methods converge at the 192nd, 164th, and 123rd episode, respectively. Compared with the GP-based and DQN-based methods, PPO-based method is 36%, 25% faster in term of the learning performance of order profit. After reaching convergence, the convergence values of order profit obtained by GP-based, DQN-based, and PPO-based are 54.4, 55.5, and 55, respectively. By comparison, PPO-based method can not only obtain the convergence of learning curve with little fluctuation in a shorter time, but also achieve near optimal order profit.

In Figs. 13-15, it can be observed that the curve of GP-based is quite fluctuant before 192-nd episode, and suddenly keeps a constant number. GP-based method improves the fitness value of the population by updating and eliminating genes, and finally generates an excellent and fixed dispatching rule. After convergence, its fitness value remains unchanged. In the process of the growth of fitness value, three optimization

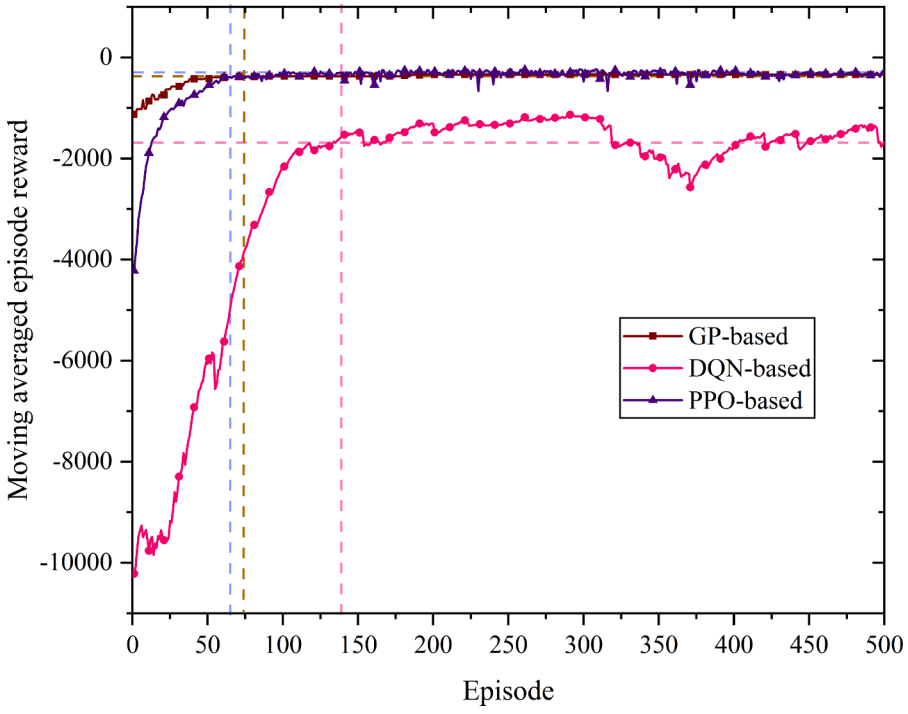


Fig. 12. The learning curve of three different methods in training process.

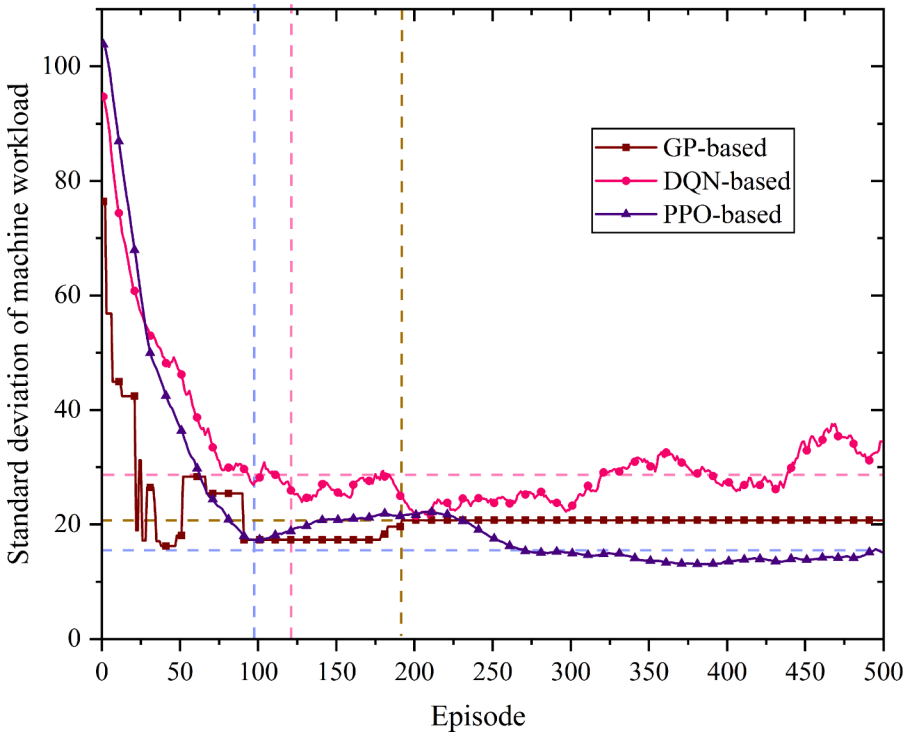


Fig. 13. The learning curve of workload balance between isomorphic machine tools.

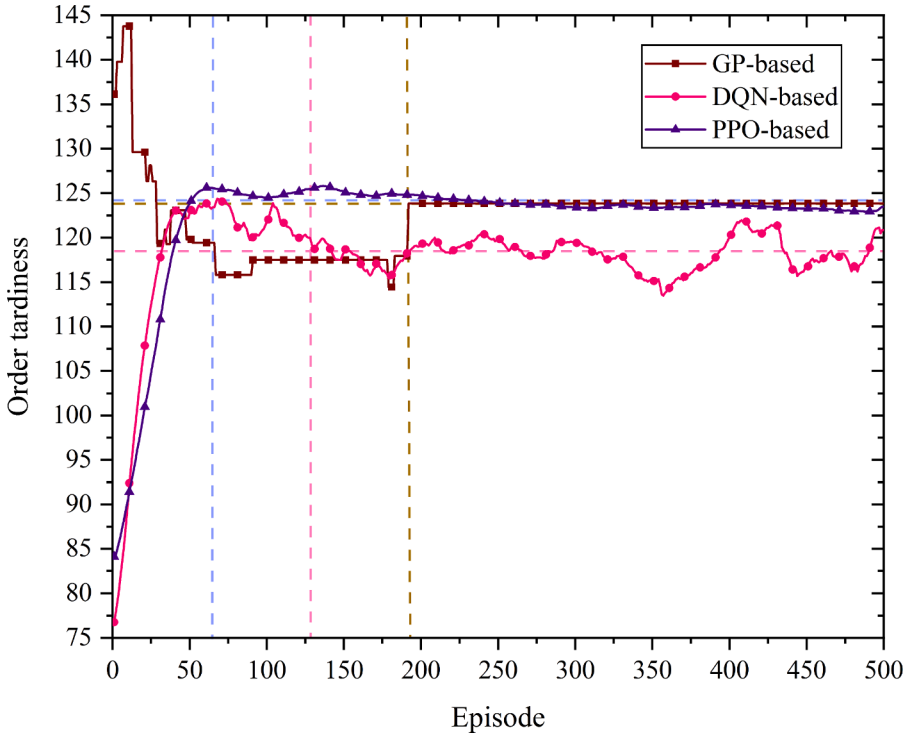


Fig. 14. The learning curve of order tardiness by three different methods.

objectives of the fitness function restrict and influence each other, and eventually reach a balance after the 192-nd episode. In other words, the fitness value of some optimization objectives may increase during the training process, while that of other optimization objectives may decrease simultaneously.

In order to verify the effectiveness of self-organizing negotiation, the performance of scheduling methods with or without self-organizing negotiation mechanism is compared through numerical experiments in terms of execution time. The execution time indicates the accumulated CPU time required to respond to all scheduling events of one episode. The shorter the execution time is, the faster the response speed is.

As shown in Tables 6–8, the symbol "☆" indicates that the scheduler does not adopt self-organizing negotiation mechanism, and is a centralized scheduling architecture. When the scheduling event is

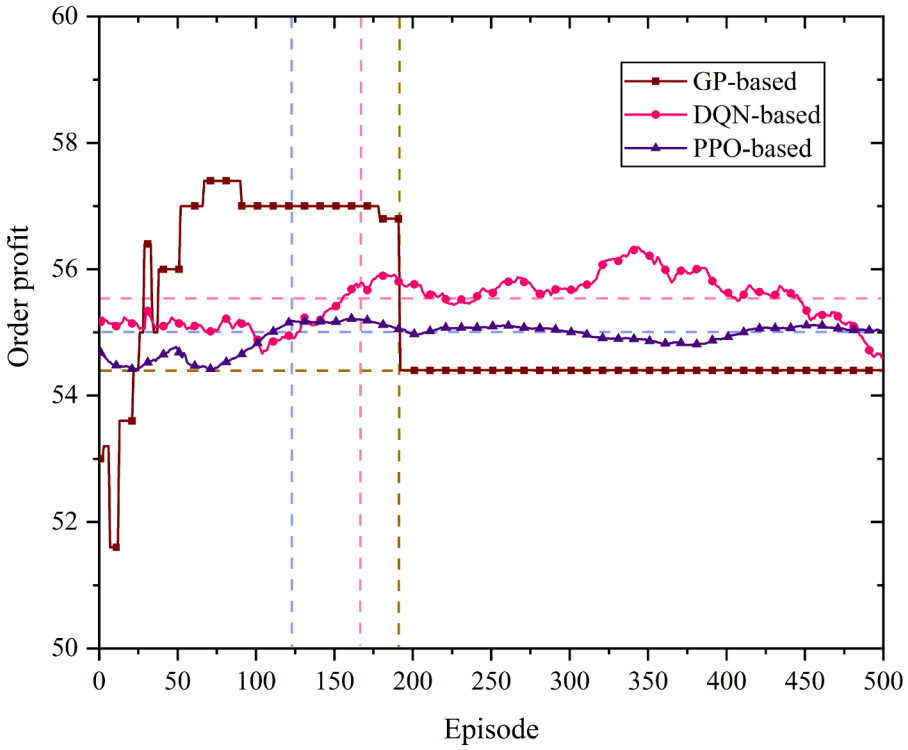


Fig. 15. The learning curve of order profit by three different methods.

Table. 6
Comparison of results on test case by five scheduling methods.

	GA-based	SPT+FIFO		GP-based		DQN-based		PPO-based	
Tardiness	125.3	146.8		124.9		121.9		123.0	
Workload balance	15.6	79.6		20.8		43.9		15.3	
Profit	55.6	50.4		54.4		54.4		55.8	
Evaluation value	165.3	117.6		158.5		132.4		163.5	
Architecture	☆	☆	★	☆	★	☆	★	☆	★
Execution time (s)	28.23	6.33	1.32	6.52	1.52	6.05	1.06	5.99	0.98

Table. 7
Comparison of results on test case under new order insertion.

	GA-based	SPT+FIFO		GP-based		DQN-based		PPO-based	
Tardiness	123.0	145.8		122.9		124.2		117.6	
Workload balance	20.1	71.8		23.1		29.2		17.7	
Profit	54.9	50.8		54.8		55.2		56.6	
Evaluation value	157.8	124.8		154.6		150.2		156.5	
Architecture	☆	☆	★	☆	★	☆	★	☆	★
Execution time (s)	54.22	6.35	1.34	6.56	1.55	6.07	1.07	6.01	0.99

Table. 8
Comparison of results on test case under machine failure.

	GA-based	SPT+FIFO		GP-based		DQN-based		PPO-based	
Tardiness	122.5	143.8		122.4		122.2		121.6	
Workload balance	28.6	60.0		31.7		35.6		31.1	
Profit	55.1	51.6		55.0		55.4		56.0	
Evaluation value	149.0	135.4		145.7		142.0		146.5	
Architecture	☆	☆	★	☆	★	☆	★	☆	★
Execution time (s)	42.53	6.52	1.42	6.67	1.58	6.19	1.08	6.12	1.01

triggered, the working status of the equipment and the operation attributes of the workpieces are collected to the central server of the workshop through SCADA. Then, the scheduler deployed in the central server runs the decision-making model (i.e., SPT+FIFO, GP-rule, DQN-based, or PPO-based) and generates the production strategy based on the collected data. Finally, the scheduler sends the control instructions to manufacturing equipment for production through the IoT network. The symbol "★" indicates that the scheduler is deployed in the embedded IPC

of the equipment, adopts self-organizing negotiation mechanism, and is a distributed scheduling architecture. When the scheduling event is triggered, the scheduler observes the characteristics of tasks and resources through the self-organizing negotiation mechanism. Then, the production strategy is intelligently generated by the decision-making model of the scheduler, and is applied to guide the equipment for production through the self-organizing negotiation mechanism, without requesting commands from the central server.

The experimental results show that four dynamic scheduling methods (i.e., SPT+FIFO, GP-rule, DQN-based, and PPO-based) without self-organizing negotiation mechanism respectively consume (6.33,6.52,6.05,5.99) s, (6.35,6.56,6.07,6.01) s, and (6.52,6.67,6.19,6.12) s to complete the schedule in the cases of normal condition, order insertion and machine failure. However, the above methods with self-organizing negotiation mechanism respectively consume (1.32, 1.52, 1.06, 0.98) s, (1.34, 1.55, 1.07, 0.99) s, and (1.42, 1.58, 1.08, 1.01) s to complete the schedule. With the help of self-organizing negotiation mechanism, the execution time consumed by the four scheduling methods in different cases is reduced by (79%, 77%, 82%, 84%), (79%, 76%, 82%, 84%) and (78%, 76%, 83%, 83%), respectively. Therefore, it can be concluded that the self-organizing negotiation mechanism significantly improves the response speed of the scheduling method.

In order to verify the decision performance of the AI scheduler in solving test case after training, five typical methods are selected to carry out comparative experiments, including GA-based, SPT+FIFO, GP-based, DQN-based and PPO-based.

Among them, GA-based method is a centralized scheduling architecture. It generates the near optimal solution by running GA algorithm in the central server, and then sends control instructions to manufacturing equipment for production. Note that GA-based method assumes that the statuses of jobs and machines are known beforehand, thus achieves the optimal results. The other four methods with the negotiation mechanism are distributed scheduling architecture. Their operation logic is as follows: Firstly, when the scheduling events are triggered, the self-organizing negotiation mechanism is used to obtain the working state of available machines. Then, the four different types of decision-making models (i.e., SPT+FIFO, GP-based, DQN-based, and PPO-based) are applied to generate the optimal production actions based on the obtained state information. SPT+FIFO is a composite dispatching rule. SPT rule is applied to conduct machine selection, while FIFO rule is used to allocate the tasks of machine buffer. There are five performance metrics used to evaluate the quality of scheduling results obtained by different methods, including: tardiness (tard), workload balance (load), profit, evaluation value and execution time. The evaluation value is equal to $tard + profit - load$, and reflects the comprehensive performance of scheduling results. The larger the evaluation value is, the better the quality of the scheduling result is.

As shown in Table 6, compared with other four methods, the quality of the solution obtained by GA-based method is the best in terms of evaluation value. However, its response speed is too slow, and the execution time consumed is 28.23 s, which is 21.4, 18.6, 26.6 and 28.8 times longer than that of SPT+FIFO, GP-based, DQN-based, and PPO-based method, respectively. That is to say, the GA-based method obtains high-quality solutions at the cost of longer execution time. SPT+FIFO method attains the optimal solution in terms of tardiness time, while PPO-based method obtains the optimal solution in terms of workload, profit and evaluation value. Compared with SPT+FIFO, GP-based, and DQN-based methods, the PPO-based method achieves better performance in workload balance, order profit and evaluation value. In addition, the PPO-based method achieves great performance close to that of GA-based method in workload balance, order profit and evaluation value. It is revealed from the side that the scheduling performance of AI scheduler has been significantly improved after training in terms of multi-objective optimization.

To sum up, the proposed PPO-based method makes AI scheduler obtain better scheduling results after training compared with the dispatching rules, GP-based, and DQN-based method.

4.4. Adaptive scheduling under abnormal events

The disturbance of resource and order frequently occurs in the workshop environment, resulting in inefficiency and stagnation of the production process. In order to test AI scheduler's performance to handle disturbance events after training, two experiments are designed, which take both order insertion and machine failure into account. First, the arrival time of work order No.8 in the test case is set to 200 s, which is consistent with that of work order No.9. Secondly, the arrival time of work order No.17 is set to 385 s, which is consistent with that of work order No.18. In the case, there are two disturbance events of order insertion at 200 s and 385 s. The experimental results are shown in Table 7. In addition, it is assumed that the machine M_4 in the testbed fails and has no ability to perform the processing task from 300 s to 400 s. During this period, the machine M_4 cannot be assigned tasks until the machine returns to normal operation. The experimental results are shown in Table 8.

When encountering unpredictable events, the GA-based method has to make a new schedule by rescheduling, resulting in longer execution time, and its execution time is equal to the sum of the solution time of the initial scheduling scheme and the re-solution time after disturbance. As shown in Table 7, the execution time of GA-based method is 54.22 s in the case of order insertion, which is 40.5, 35.0, 50.7 and 54.8 times longer than that of SPT+FIFO, GP-based, DQN-based and PPO-based methods. As shown in Table 8, the execution time of GA-based method is 42.53 s in case of machine failure, which is 30.0, 26.9, 39.4, and 42.1 times longer than that of SPT+FIFO, GP-based, DQN-based and PPO-based methods. To sum up, the proposed self-organizing negotiation mechanism greatly improves the response speed of the scheduling system to dynamic events.

Compared with SPT+FIFO, GP-based, and DQN-based methods, the proposed PPO-based method obtains the better quality of the solution in terms of workload balance, order profit and evaluation value under order insertion and machine failure. This shows that PPO-based method has great adaptability in solving dynamic problems, and can reach the accuracy close to that of GA-based method simultaneously.

5. Conclusions

In order to improve the dynamic responsiveness and production efficiency of manufacturing workshop to personalized orders, a multi-agent manufacturing system with the ability of online scheduling and scheduling strategy optimization is constructed in this paper. Firstly, various machine tools and manufacturing equipment are intelligentized by embedding IPC into it, and are modeled as equipment agents, so that they have the ability of environmental perception, information sharing and autonomous decision-making. Then, a CNP-based negotiation mechanism is designed to support the cooperation and competition among equipment agents. In addition, an adaptive and dynamic scheduling model based on deep reinforcement learning is proposed to realize production planning, and is called AI scheduler in this paper. The AI scheduler is an end-to-end decision-making model, and is deployed in each equipment agent. When the scheduling event is triggered, the AI scheduler intelligently generates an optimal production strategy according to machine working state and operation attributes. With the progress of production activities, the parameters of scheduling model are periodically optimized and updated through historical sampling trajectory data, which not only improves the decision-making performance of AI scheduler, but also makes AI scheduler efficiently deal with various unpredictable exceptions (i.e., machine failure and new order insertion) in the workshop environment. Compared with GP, DQN, and dispatching rules, experimental results show that the proposed method achieves better performance in terms of production strategy learning

and disturbance handling.

In the future research, the scale of machine tools and the type of work orders can be further increased and enriched in the testbed. In addition, the design of state space will try to add more operation attributes (such as order priority), and various dispatching rules may be considered into the design of action space.

Authorship contribution statement

Yi Zhang: Conceptualization, Methodology, Software, Investigation, Writing - original draft. Haihua Zhu: Project administration, Funding acquisition, Writing - review & editing. Dunbing Tang: Supervision, Funding acquisition. Tong Zhou: Conceptualization, Resources. Yong Gui: Conceptualization.

Declaration of Competing Interest

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

Data availability

No data was used for the research described in the article.

Acknowledgments

This work was supported by National Key Research and Development Program of China [grant number 2018YFE0177000], National Natural Science Foundation of China [grant number 52075257] and the Fundamental Research Funds for the Central Universities [grant number NT2021021].

References

[1] B. Zhou, J. Bao, J. Li, et al., A novel knowledge graph-based optimization approach for resource allocation in discrete manufacturing workshops, Robot. Comput. Integr. Manuf. 71 (3) (2021), 102160.
[2] A. Kusiak, Smart manufacturing, Int. J. Prod. Res. 56 (1–2) (2018) 508–517.
[3] J. Zhang, G. Ding, Y. Zou, et al., Review of job shop scheduling research and its new perspectives under Industry 4.0, J. Intell. Manuf. 30 (4) (2019) 1809–1830.
[4] T. Fei, Q. Qi, A. Liu, et al., Data-driven smart manufacturing, J. Manuf. Syst. 48 (2018) 157–169.
[5] C. Liu, P. Zheng, X. Xu, Digitalisation and servitisation of machine tools in the era of Industry 4.0: a review, Int. J. Prod. Res. (2021) 1–33, https://doi.org/10.1080/00207543.2021.1969462.
[6] S. Jamal, A.A. Mohammad, M. Masoud, A reinforcement learning approach to parameter estimation in dynamic job shop scheduling, Comput. Ind. Eng. 110 (2017) 75–82.
[7] T.C. Schelling, Dynamic models of segregation, J. Math. Sociol. 1 (2) (1971) 143–186.
[8] L. Monostori, J. Váncza, S.R.T. Kumara, Agent-based systems for manufacturing, CIRP Ann. 55 (2) (2006) 697–720.
[9] T. Suganuma, T. Oide, S. Kitagami, et al., Multiagent-based flexible edge computing architecture for iot, IEEE Netw. 32 (1) (2018) 16–23.
[10] N.K.C. Krothapalli, A.V. Deshmukh, Design of negotiation protocols for multi-agent manufacturing systems, Int. J. Prod. Res. 37 (7) (1999) 1601–1624.
[11] S. Cavalieri, M. Garetti, M. Macchi, et al., An experimental benchmarking of two multi-agent architectures for production scheduling and control, Comput. Ind. 43 (2) (2000) 139–152.
[12] G.K. Yun, S. Lee, J. Son, et al., Multi-agent system and reinforcement learning approach for distributed intelligence in a flexible smart manufacturing system, J. Manuf. Syst. 57 (2020) 440–450.
[13] K. Li, T. Zhou, B.H. Liu, et al., A multi-agent system for sharing distributed manufacturing resources, Expert. Syst. Appl. 99 (2018) 32–43.
[14] P. Leitao, F. Restivo, ADACOR: a holonic architecture for agile and adaptive manufacturing control, Comput. Ind. 57 (2) (2006) 121–130.
[15] J. Barbosa, P. Leitao, E. Adam, D Trentesaux, Dynamic self- organization in holonic multi-agent manufacturing systems: the ADACOR evolution, Comput. Ind. 66 (2015) 99–111.
[16] H.V. Brussel, J. Wyns, P. Valckenaers, et al., Reference architecture for holonic manufacturing systems: PROSA, Comput. Industry 37 (3) (1998) 255–274.
[17] P. Valckenaers, Perspective on holonic manufacturing systems: PROSA becomes ARTI, Comput. Ind. 120 (2020), 103226, https://doi.org/10.1016/j.compind.2020.103226.
[18] M. Owliya, A new Agents-based model for dynamic job allocation in manufacturing shopfloors, IEEE Syst. J. 6 (2) (2012) 353–361, https://doi.org/10.1109/JSYST.2012.2188435.
[19] C. Pascal, D. Panescu, On rescheduling in holonic manufacturing systems, Comput. Ind. 104 (2019) 34–46.
[20] J. Wang, Y. Zhang, Y. Liu, et al., Multiagent and bargaining-game-based real-time scheduling for internet of things-enabled flexible job shop, IEEE Int. Things J. 6 (2) (2019) 2518–2531.
[21] A. Rajabinasab, S. Mansour, Dynamic flexible job shop scheduling with alternative process plans: an agent-based approach, Int. J. Adv. Manuf. Technol. 54 (9–12) (2011) 1091–1107.
[22] W. Xiong, D. Fu, A new immune multi-agent system for the flexible job shop scheduling problem, J. Intell. Manuf. 29 (4) (2018) 857–873.
[23] L.C. Wang, C.Y. Cheng, S.Y. Lin, Distributed feedback control algorithm in an auction-based manufacturing planning and control system, Int. J. Prod. Res. 51 (9) (2013) 2667–2679.
[24] G.H. Zhang, L. Gao, Y. Shi, An effective genetic algorithm for the flexible job-shop scheduling problem, Expert Syst. Appl. 38 (4) (2011) 3563–3573.
[25] Y. Zhang, H.H. Zhu, D.B. Tang, An improved hybrid particle swarm optimization for multi-objective flexible job-shop scheduling problem, Kybernetes 49 (12) (2020) 2873–2892.
[26] T. Meng, Q.K. Pan, H.Y. Sang, A hybrid artificial bee colony algorithm for a flexible job shop scheduling problem with overlapping in operations, Int. J. Prod. Res. 56 (16) (2018) 5278–5292.
[27] M. Nouiri, A. Bekrar, A. Jemai, S. Niar, A.C. Ammari, An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem, J. Intell. Manuf. 29 (3) (2018) 603–615.
[28] R. Buddala, S.S. Mahapatra, Two-stage teaching-learning-based optimization method for flexible job-shop scheduling under machine breakdown, Int. J. Adv. Manuf. Technol. 100 (5–8) (2019) 1419–1432.
[29] K.Z. Gao, P.N. Suganthan, Q.K. Pan, et al., Artificial bee colony algorithm for scheduling and rescheduling fuzzy flexible job shop problem with new job insertion, Knowl. Based Syst. 109 (2016) 1–16.
[30] K.Z. Gao, F.J. Yang, M.C. Zhou, et al., Flexible job-shop rescheduling for new job insertion by using discrete jaya algorithm, IEEE Trans. Cybern. 49 (5) (2019) 1944–1955.
[31] F.F. Zhang, S. Nguyen, Y. Mei, M.J. Zhang, Genetic Programming For Production Scheduling, Springer, Singapore, 2021, https://doi.org/10.1007/978-981-16-4859-5 first ed.
[32] F.F. Zhang, Y. Mei, S. Nguyen, M.J. Zhang, Guided subtree selection for genetic operators in genetic programming for dynamic flexible job shop scheduling. Proceedings of the European Conference on Genetic Programming (Part of EvoStar), Springer, 2020, pp. 262–278.
[33] M. Dumic, D. Jakobovic, Ensembles of priority rules for resource constrained project scheduling problem, Appl. Soft Comput. 110 (2021), 107606, https://doi.org/10.1016/j.asoc.2021.107606.
[34] M. Zhang, F. Tao, A.Y.C. Nee, Digital twin enhanced dynamic job-shop scheduling, J. Manuf. Syst. 58 (2021) 146–156.
[35] C.C. Lin, D.J. Deng, Y.L. Chih, et al., Smart manufacturing scheduling with edge computing using multi-class deep Q network, IEEE Trans. Ind. Inf. 15 (7) (2019) 4276–4284.
[36] L.B. Wang, X. Hu, Y. Wang, et al., Dynamic job-shop scheduling in smart manufacturing using deep reinforcement learning, Comput. Netw. 190 (2021), 107969, https://doi.org/10.1016/j.comnet.2021.107969.
[37] T. Zhou, D.B. Tang, H.H. Zhu, et al., Multi-agent reinforcement learning for online scheduling in smart factories, Robot. Comput. Integr. Manuf. 72 (2021), 102202, https://doi.org/10.1016/j.rcim.2021.102202.
[38] T.T. Nguyen, N.D. Nguyen, S. Nahavandi, Deep reinforcement learning for multiagent systems: a review of challenges, solutions, and applications, IEEE Trans. Cybern. 50 (9) (2020) 3826–3839.
[39] Y.X. Li, W.B. Gu, M.H. Yuan, et al., Real-time data-driven dynamic scheduling for flexible job shop with insufficient transportation resources using hybrid deep Q network, Robot. Comput. Integr. Manuf. 74 (2022), 102283, https://doi.org/10.1016/j.rcim.2021.102283.
[40] Y.R. Shiue, K.C. Lee, C.T. Su, Real-time scheduling for a smart factory using a reinforcement learning approach, Comput. Ind. Eng. 125 (2018) 604–614.
[41] A.I. Orhean, F. Pop, I. Raicu, New scheduling approach using reinforcement learning for heterogeneous distributed systems, J. Parallel Distrib. Comput. 117 (2017) 292–302.
[42] H.H. Zhu, M. Chen, Z.Q. Zhang, et al., An adaptive real-time scheduling method for flexible job shop scheduling problem with combined processing constraint, IEEE Access 7 (2019) 125113–125121.
[43] A. Kuhnle, J.P. Kaiser, F. Thei, et al., Designing an adaptive production control system using reinforcement learning, J. Intell. Manuf. 32 (3) (2021) 855–876.
[44] H. Zhao, Q. She, C. Zhu, et al., Online 3D bin packing with constrained deep reinforcement learning, in: Proceedings of the 35th AAAI Conference on Artificial Intelligence, the 33rd Conference on Innovative Applications of Artificial

Intelligence, the 11th Symposium on Educational Advances in Artificial Intelligence, 2021, pp. 741–749. https://arxiv.org/abs/2006.14978.
[45] Z.Z. Zhang, H. Liu, M.C. Zhou, et al., Solving dynamic traveling salesman problems with deep reinforcement learning, IEEE Trans. Neural Netw. Learn. Syst. (2021), https://doi.org/10.1109/TNNLS.2021.3105905.
[46] J.J.Q. Yu, W. Yu, J Gu, Online vehicle routing with neural combinatorial optimization and deep reinforcement learning, IEEE Trans. Intell. Transp. Syst. 20 (10) (2019) 3806–3817.