Computers & Industrial Engineering 171 (2022) 108406

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

Using real-time manufacturing data to schedule a smart factory via reinforcement learning

Wenbin Gu [a,*], Yuxin Li [a], Dunbing Tang [b], Xianliang Wang [a], Minghai Yuan [a]

[a] Department of Mechanical and Electrical Engineering, Hohai University, Changzhou 213022, China
[b] College of Mechanical and Electrical Engineering, Nanjing University of Aeronautics and Astronautics, 29 Yadao Street, Nanjing 210016, China

ARTICLE INFO

Keywords:
Smart factory
Real-time scheduling
Genetic programming
Production state clustering
Reinforcement learning

ABSTRACT

Under the background of intelligent manufacturing, internet of things and other information technologies have accumulated a large amount of data for manufacturing system. However, the traditional scheduling methods often ignore the production law and knowledge hidden in the manufacturing data. Therefore, this paper proposes a cyber-physical architecture and a communication protocol for smart factory, and a multiagent-system-based dynamic scheduling mechanism is given using contract net protocol. In the dynamic scheduling mechanism, the problem formulation module and scheduling point module are designed first. Then, a genetic programming (GP) method is proposed to form sixteen high-quality rules, which constitute the scheduling rule library. Meanwhile, combining with autoencoder, self-organizing mapping neural network and k-means clustering algorithm, the state clustering module is designed to realize the efficient clustering of production attribute vector. Moreover, an improved Q-learning algorithm is used to train the GP rule selector, so that the decision-making agent can choose the appropriate GP rule according to the production state at each scheduling point. Finally, the experimental results show that the proposed method has feasibility and superiority compared with other methods in real-time scheduling, and can effectively deal with disturbance events in the manufacturing process.

1. Introduction

With the increasingly fierce competition in the global market, manufacturing enterprises are faced with more stringent requirements in improving production efficiency, improving product quality, reducing resource consumption and reducing production costs (Dai et al., 2019). In order to enhance the competitiveness of manufacturing industry and expand the international market, countries have formulated corresponding national strategies, such as Industrial Internet in the US, Industry 4.0 in Germany and Made in China 2025. These strategies can help industry realize smart manufacturing through the application of various advanced technologies in the manufacturing field. Among them, the rapid development of Internet of things (IoT) and other new information technologies provides rich real-time data for manufacturing system (Bueno et al., 2020). According to statistics, in the modern industrial system, the total amount of data generated by mechanical equipment, production management, application services, etc., is more than 1000 Exabytes every year (Yin & Kaynak, 2015), which forms an explosive growth. By analyzing and processing these data, the manufacturing system can obtain valuable knowledge and make wise and forward-looking decisions, so as to achieve a positive impact on the production of enterprises (Cui et al., 2020). In other words, data-driven manufacturing is gradually becoming the main means to realize smart manufacturing.

From the perspective of the whole industry, big data is becoming more and more important in the manufacturing industry. It will play an important role in the fourth industrial revolution, that is, bringing intelligence to manufacturing. The first three industrial revolutions led to the steam era, electrification era and information era respectively, which make production become large-scale and automated. Germany calls the fourth industrial revolution Industry 4.0 (Olsen & Tomlin, 2020). Industry 4.0 uses cyber physical system (CPS) to digitize the supply, manufacturing and sales information in production. It aims to improve the intelligent level of manufacturing industry and build a smart factory with adaptability, resource efficiency and genetic engineering (Nakayama et al., 2020). In the smart factory, various manufacturing resources interact in real time using communication protocols and standards. Meanwhile, sensors and automatic identification technology are used to collect data at each stage of the product lifecycle (Li et al., 2015). These data include user data, supplier data,

* Corresponding author.
E-mail address: 20021592@hhu.edu.cn (W. Gu).

https://doi.org/10.1016/j.cie.2022.108406
Received 23 March 2021; Received in revised form 22 March 2022; Accepted 29 June 2022
Available online 2 July 2022

equipment data, order data, etc. Through the analysis of these data, enterprises can achieve lots of goals (Usuga Cadavid et al., 2020), such as predicting product price trend, improving product design, reducing energy consumption, reducing production cost and providing intelligent maintenance service. To sum up, big data technology has changed the traditional mode of product manufacturing and management, which makes the production process transparent. It overcomes the problem of lack of coordination and information sharing in the traditional supply chain, and avoids the bullwhip effect, thus realizing the objective of providing intelligent application services using smart manufacturing.

In recent years, many scholars have conducted relevant research on the big data-driven manufacturing. Majeed et al. (Majeed et al., 2021) proposed a framework of big data-driven sustainable and smart additive manufacturing, and it can help additive industry leaders to make better decisions at the beginning stage of product lifecycle. Tao et al. (Tao et al., 2018) discussed the role of big data in supporting smart manufacturing and proposed a conceptual framework for data-driven smart manufacturing. Ku et al. (Ku et al., 2020) developed a solution to support traditional industries to adopt smart manufacturing and empower digital transformation. Mörth et al. (Mörth et al., 2020) introduced a design perspective for IoT-driven analytics in intralogistics to optimize the internal logistics systems. Tao and Qi (Tao & Qi, 2019) proposed a new information technology driven service-oriented smart manufacturing framework to facilitate the visions of smart manufacturing. Qu et al. (Qu et al., 2019) summarized the evolution, definition, objectives, three requirements and components of smart manufacturing system, and proposed an autonomous model driven by dynamic demand and key performance indicators. Subramaniyan et al. (Subramaniyan et al., 2020) developed a data-driven approach to diagnosing throughput bottlenecks, using the combined knowledge of the maintenance and machine learning (ML) domains. Zhang et al. (Zhang et al., 2020) proposed an integrated framework to holistically describe the active discovery and optimal allocation of smart manufacturing services. Li et al. (Li et al., 2019) developed an uncertain learning curve and some useful theorems by utilizing uncertainty theory, and applied it to the single-machine optimization problem. Cheng et al. (Cheng et al., 2020) constructed a high-quality training dataset and proposed an adaptive ensemble model to achieve fast and accurate makespan estimation.

As the core of intelligent manufacturing, production scheduling can significantly improve the decision-making level of shop floor system and realize the efficient operation of manufacturing execution process. The traditional scheduling methods mainly include heuristic algorithm and metaheuristic algorithm. Heuristic algorithms assign priority to manufactured objects, thereby implementing scheduling and allocation. It mainly includes shortest processing time, longest processing time, first in first out and other scheduling rules. Meta-heuristic algorithms are intelligent optimization algorithms, which can find high-quality scheduling schemes through a large number of iterations and searches. Wu et al. (Wu et al., 2021) established a mixed integer linear programming model for the scheduling problem in three successive processes in car production, and used several *meta*-heuristic algorithms to find good solutions. Huang et al. (Huang et al., 2021) proposed an improved iterative greedy algorithm based on the groupthink for solving the distributed assembly permutation flowshop scheduling problem. Zhu and Zhou (Zhu & Zhou, 2020) proposed an efficient evolutionary multi-objective grey wolf optimizer for the flexible job shop scheduling problem with job precedence constraints. Yuan et al. (Yuan et al., 2021) proposed an improved nondominated sorting genetic algorithm for the resource scheduling problem in intelligent manufacturing workshop. Chen et al. (Chen et al., 2021) proposed six constructive heuristics and an iterated greedy algorithm for the distributed blocking flowshop scheduling problem.

However, the traditional scheduling methods often ignore the information accumulated by the manufacturing system, resulting in the production law hidden in the manufacturing data not being mined. At the same time, in the actual production workshop, disturbance events often occur, such as machine breakdown and new order insertion. These disturbance events make the original scheduling scheme no longer have superior scheduling performance, and even make the original scheme infeasible. In the current smart factory, the new information technology can quickly perceive, collect and control the manufacturing information of various resources in the production system. Therefore, it is of great significance to study manufacturing data-driven dynamic scheduling under the framework of intelligent manufacturing. At present, more and more scholars try to combine industrial big data with scheduling optimization. Wang et al. (Wang et al., 2019) proposed a new multiagent-based real-time scheduling architecture for an IoT-enabled flexible job shop, which uses real-time data to deal with unpredictable exceptions. Wang and Gombolay (Wang & Gombolay, 2020) developed a novel graph attention network-based scheduler to automatically learn features of scheduling problems towards coordinating human-robot collaboration in real-time environment. Hu et al. (Hu et al., 2020) proposed an adaptive deep reinforcement learning-based automated guided vehicle (AGV) real-time scheduling approach with mixed rule for the flexible shop floor to minimize the makespan and delay ratio. Caldeira et al. (Caldeira et al., 2020) proposed an improved backtracking search algorithm for the flexible job shop scheduling problem considering new job arrivals. Kong et al. (Kong et al., 2021) established a novel energy-efficient rescheduling model with Time-of-use energy cost and applied a variable neighborhood search algorithm to obtain near-optimal solutions.

It can be seen from the above that in the intelligent manufacturing environment, it is of great significance to study the data-driven scheduling method for workshop. A reasonable manufacturing paradigm can provide a technical framework for data-driven manufacturing, and various advanced technologies make data-driven manufacturing possible. However, there are still few literatures on manufacturing data-based production scheduling, and few studies take into account the redundancy of production characteristics. Therefore, combined with the effective manufacturing information processing technology, it is possible and meaningful to develop a real-time data-driven dynamic scheduling method for smart factory on the basis of the advanced data acquisition hardware and artificial intelligence (AI) method. This is the main motivation of this paper.

From the perspective of dynamic scheduling method design, heuristic algorithm (i.e., scheduling rule) has strong real-time performance and is suitable for dynamic scheduling. However, a single scheduling rule cannot maintain high-quality scheduling performance in the face of orders with different sizes. Therefore, the ideal situation is that at each scheduling point, the workshop selects an appropriate rule from the rule library according to the real-time manufacturing data, so as to realize efficient production operation. Based on this idea, this paper takes real-time manufacturing data as state and scheduling rule as action, and then uses reinforcement learning (RL) algorithm to design a dynamic scheduling method for smart factory. This is the method design idea of this paper.

In order to address these challenges, this paper proposes the cyber-physical architecture for smart factory, and design the smart-factory-oriented dynamic scheduling mechanism based on multi-agent system (MAS) and contract net protocol (CNP). The dynamic scheduling mechanism includes five key modules. Firstly, the problem formulation module is proposed for modeling the smart factory scheduling problem. Secondly, the scheduling point module is designed for agent to determine whether a time point meets the requirements of scheduling point. Thirdly, the genetic programming (GP) method is designed to generate sixteen high-quality rules and form the scheduling rule library. Fourthly, combining with autoencoder, self-organizing mapping (SOM) neural network and K-means clustering algorithm, the state clustering module is proposed to realize the efficient clustering of manufacturing data. Fifthly, an improved Q-learning algorithm is used to train the GP rule selector, so that the decision-making agent can select the appropriate GP

rule according to the production state at each scheduling point. Finally, the experimental results show that the proposed method has feasibility and superiority in real-time scheduling, and can effectively deal with disturbance events in the production process.

The rest of this paper is organized as follows. Section 2 generally describes the smart factory for intelligent scheduling. Section 3 presents the five key modules in MAS. In Section 4, the performance of smart factory is evaluated through training experiment and comparison experiments. In Section 5, the conclusion and future work are given.

2. Smart factory for intelligent scheduling

2.1. Cyber-physical architecture

In order to realize the intelligent scheduling, a cyber-physical architecture is established for smart factory to execute the effective management based on real-time manufacturing data, and it is shown in Fig. 1. Smart factory is divided into the physical world and cyber world.

The physical world is composed of lots of manufacturing resources, including the machines, automated guided vehicles, automated storage and retrieval system (ASRS), etc. Through the combination of mechanical equipment, workpieces and intelligent sensors, the physical world can obtain massive, multi-source and heterogeneous data in the product lifecycle, which can be divided into three categories: structured data, semi-structured data and unstructured data. Meanwhile, all manufacturing resources are connected through local area network to realize the production data sharing. In summary, the physical world can achieve the manufacturing information acquisition and accurate process state tracking of mechanical equipment.

On the other hand, based on the data integration, data cleaning, data reduction, data mining and other advanced information technologies, the cyber world can map and influence the physical world. In other word, it is the management center of smart factory, and employees can realize the optimization and control of production process through the cyber world. The cyber world is a MAS, and it includes multiple machine agents, multiple transportation equipment agents, manufacturing task agent, real-time monitoring agent and decision-making agent. The functions of these agents are illustrated below.

Machine agent. Machine agent is the mapping of machine entity. Machine entity obtains its own production data (real-time or non real-time) through sensors and automatic identification technology. Then machine agent synchronizes the real-time state of corresponding physical entity through data storage, data preprocessing and other technologies. At the same time, based on data transmission technology, these machine agents provide real-time monitoring agent and decision-making agent with the state information of corresponding entities. In addition, machine agent can inform the corresponding physical entity to perform operations according to the arrangement of decision-making agent.

(1) Transportation equipment agent. Transportation equipment agent is the mapping of AGV or other transportation resource entity. Its principle is similar to that of machine agent. Each transportation resource entity perceives and obtains its own manufacturing data in real time through IoT technology. Then, the corresponding transportation equipment agent realizes the information synchronization between agent and entity through data storage and other technologies. At the appropriate time

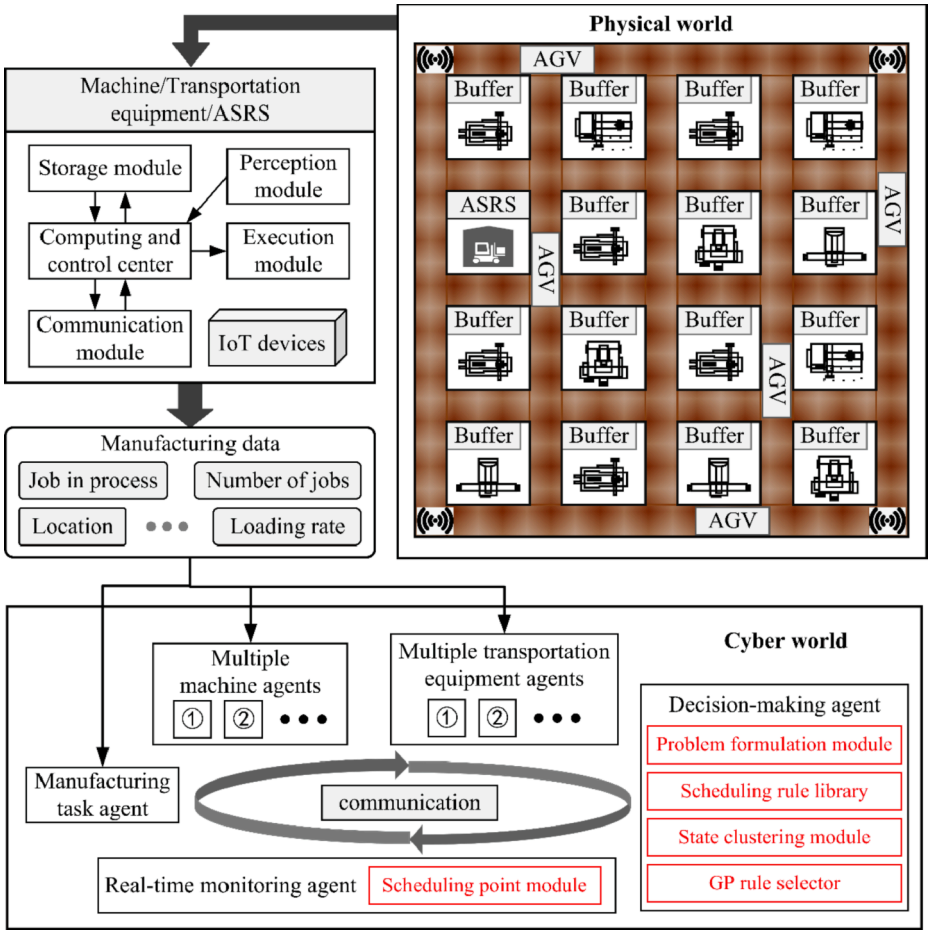

Fig. 1. The cyber-physical architecture of smart factory.

point, transportation equipment agent will send the information of corresponding physical entity to real-time monitoring agent and decision-making agent through the communication technology (IoT devices and communication protocol). Moreover, transportation equipment agent can arrange the physical entity to carry out transportation tasks according to the instructions given by decision-making agent.

(2) Manufacturing task agent. Manufacturing task agent can synchronize the states of all workpieces according to the information of physical entities. It should be noted that the information of workpiece is not directly obtained from the physical entity. For example, when job 1 arrives at machine 1 at time t, the RFID reader configured on machine 1 senses the arrival of job 1 through RFID tag. Therefore, machine 1 immediately informs the system that the position of job 1 at time t is machine 1. Then, when machine 1 starts processing operation 2 of job 1, it will start timing this process. It helps to update the progress information of job 1 in the system. In the production process, manufacturing task agent will send the real-time state information of each workpiece to real-time monitoring agent and decision-making agent in time. In addition, this agent can intelligently judge whether the scheduling can be finished according to the synchronized information.

(3) Real-time monitoring agent. Real-time monitoring agent has no corresponding entity and belongs to pure software service module. This agent is responsible for monitoring the state of smart factory. It internally registers the information of machine agents, transportation equipment agents and manufacturing task agent, which is synchronized with the real-time state of monitored objects. Meanwhile, real-time monitoring agent can turn on the notification function in time according to its monitoring situation. In addition, based on the registered information, real-time monitoring agent can use the internal scheduling point module to determine whether a time point meets the requirements of scheduling point.

(4) Decision-making agent. Decision-making agent is the core of smart factory and plays the role of scheduler for task allocation. It mainly includes four key modules: problem formulation module, scheduling rule library, state clustering module and GP rule selector. 1) The problem formulation module is responsible for modeling the scheduling problem. In the beginning of each production process, the smart factory will model the order scheduling problem using this module, which is the essential preparation work of intelligent dispatching. Meanwhile, it will be used to calculate the scheduling objective and summarize the manufacturing performance by decision-making agent after finishing the order. 2) The scheduling rule library has lots of preset scheduling rules for selection. 3) The state clustering module is used to classify the production state of smart factory. 4) The GP rule selector chooses the appropriate rule according to the result of state clustering module and its own state-rule mapping relationship. In addition to the four key modules, decision-making agent has two functions: 1) It can form a specific scheduling scheme according to the rule selection result and received production information at one decision point. 2) It will reconfigure the manufacturing environment at the appropriate time. To sum up, in the manufacturing execution process, decision-making agent can give a reasonable scheduling scheme according to the real-time manufacturing data sent by other agents, so as to inform the corresponding manufacturing resources to execute the scheme. Besides, the operation principle of decision-making agent is given in the scheduling decision-making stage in the Section 2.3.

2.2. Communication protocol between manufacturing resources

The communication protocol is a key part of smart factory communication. The existing protocols have the problems of complex information frame format and large amount of data in the communication process, which are difficult to meet the requirements of high real-time information for communication between resources in the workshop. Therefore, this paper designs a communication protocol based on TCP/IP, which can improve the communication efficiency between resources and the real-time nature of information. Workshop equipment has the network communication function by connecting to an embedded industrial computer, and the format of information frame for the communication between various resources on the scene must meet the requirements of the application layer protocol. The communication protocol designed in this paper is based on the TCP/IP protocol, which expands the Ethernet application layer. Taking into account the scalability of manufacturing system, the designed protocol adopts the object-oriented idea, and uses the combination of attribute and behavior to describe the characteristics of workshop equipment. The object of this protocol is composed of three concepts: class, behavior and attribute. The basic concepts are as follows. (1) Object: In this protocol, each manufacturing equipment is regarded as an object, and each instance of an equipment class is defined by an object. (2) Attributes and behaviors of objects: Workshop equipment has attributes, including the running time, power consumption, working capacity, etc. The behavior of workshop equipment is used to change the state of object, collaboration between objects, etc. (3) Class: This protocol classifies objects with the same attributes and behaviors into one class for identification, control and mutual communication. On this basis, the communication protocol designed in this paper adopts a unified information frame format, which is convenient for expansion and identification. It mainly includes the frame header (command frame, response frame, handshake frame and abnormal report frame), target device ID, source device ID, length, instruction, parameter and check bit. All resources in the workshop must perform the corresponding handshake after being powered on to ensure normal connection and work. In summary, this communication protocol can ensure the effective information flow and improve the stability and fault-tolerance of workshop operation.

2.3. MAS-based dynamic scheduling mechanism

Based on real-time manufacturing data, the ultimate goal of this paper is to design a dynamic scheduling method and realize the efficient production of smart factory. To this end, a novel scheduling mechanism is designed to optimize the scheduling objectives for smart factory. CNP was proposed by Smith in 1980 and has been widely used in the MAS (Li et al., 2010). It is inspired by the market bidding mechanism and has strong real-time performance. However, CNP is a single-step optimization, and lacks global information and global coherence. Therefore, referring to the CNP, this paper proposes a MAS-based dynamic scheduling mechanism for smart factory. As shown in Fig. 2, based on this mechanism, the smart factory can realize the intelligent interaction between different agents, so as to realize the fast and effective control.

Before introducing the MAS-based dynamic scheduling mechanism, it is necessary to explain the red-marked content in Fig. 2. Each red-marked content represents that an agent uses its internal modules to perform a scheduling step. In other words, the three red-marked contents in Fig. 2 indicate the corresponding implementation positions of the five key modules of MAS in the dynamic scheduling mechanism. Among the five modules, the problem formulation module of decision-making agent and the scheduling point module of real-time monitoring agent need to be set in advance, and then are put into practical use. The other three modules need to be trained in advance and then are put into practical use. Section 3 will describe the setting process and training process of these five key modules in detail. Next, the dynamic scheduling mechanism is introduced.

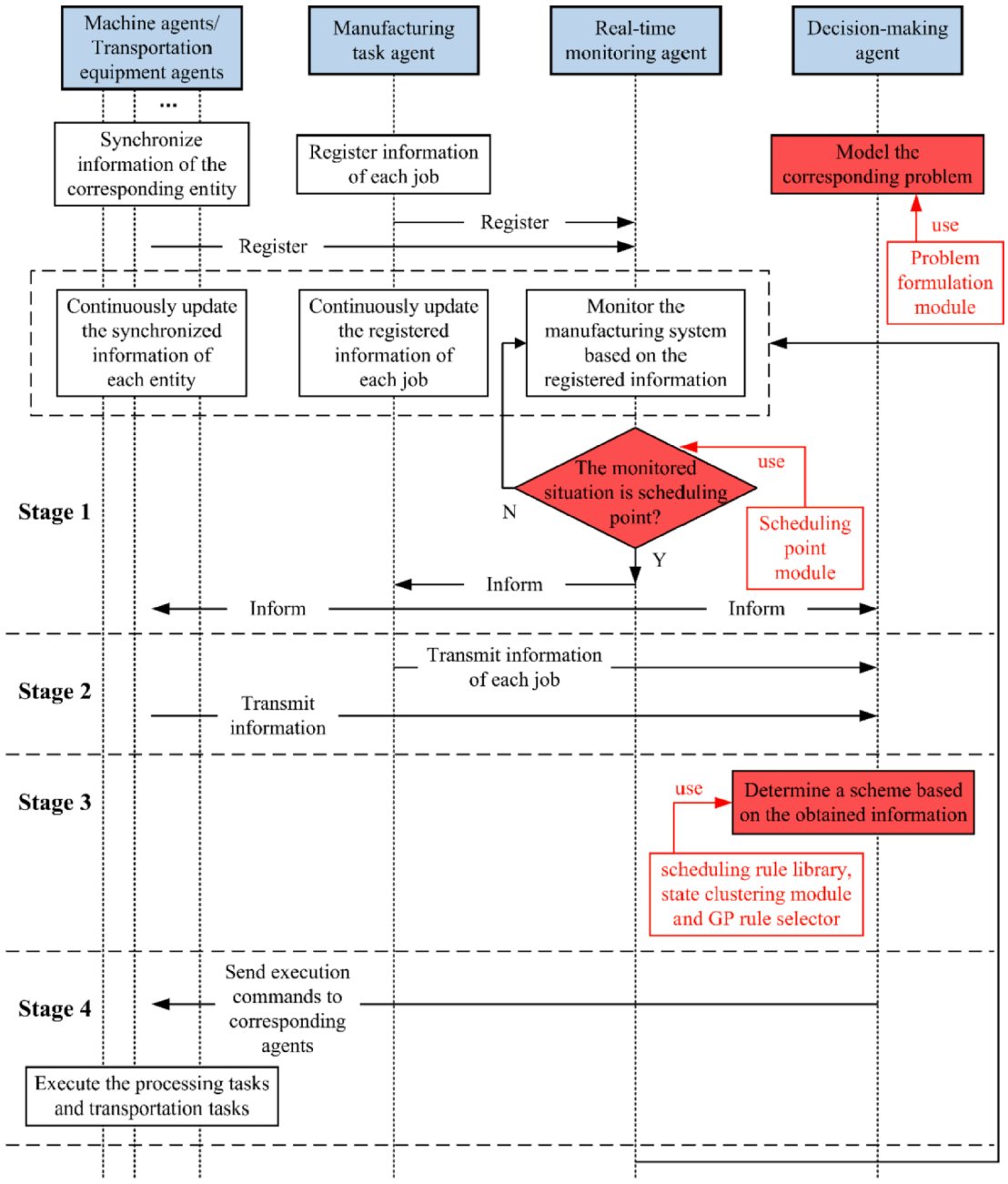

Fig. 2. MAS-based dynamic scheduling mechanism.

The proposed dynamic scheduling mechanism is mainly divided into six steps, and its specific implementation process is given below.

(1) Initial stage (as shown in the Stage 1 in Fig. 2). When the order is published in the shop floor, manufacturing task agent registers all relevant information of each job. Meanwhile, real-time monitoring agent registers each machine agent, each transportation equipment agent and manufacturing task agent. Experts will model the corresponding problem in the problem formulation module of decision-making agent. When real-time monitoring agent judges that the monitored situation meets the requirements of scheduling point through its scheduling point module, it will inform other agents to prepare for scheduling.

(2) Scheduling preparation stage (as shown in the Stage 2 in Fig. 2). All machine agents and transportation equipment agents transmit their real-time state information to decision-making agent. At the same time, manufacturing task agent sends the real-time state information of each job to decision-making agent.

(3) Scheduling decision-making stage (as shown in the Stage 3 in Fig. 2). This stage is mainly undertaken by decision-making agent and falls into five steps. First, according to the information of all operations, decision-making agent finds out all jobs waiting for transportation, namely the decision-making object of scheduling point. Second, decision-making agent selects relevant data from the received information and processes it to form a production attribute vector (PAV), namely the production state of smart factory. Third, the state clustering module classifies the PAV using its trained ability. Fourth, based on the clustering result, the GP rule selector determines the rule according to its state-rule mapping relationship. Finally, decision-making agent fetches the corresponding rule from the scheduling rule library, and

transforms the rule into a specific scheduling scheme based on the shop floor information.

(4) Scheduling execution stage (as shown in the Stage 4 in Fig. 2). According to the scheduling scheme, decision-making agent arranges corresponding agents (transportation equipment agents and machine agents) to transport and process jobs. In this stage, the real-time state information of registered machines, transportation equipment and jobs in real-time monitoring agent is constantly updated.

(5) Reciprocating cycle. When real-time monitoring agent detects the next scheduling point, repeat steps (2), (3) and (4).

(6) When manufacturing task agent determines that all jobs have been finished according to the synchronic information, the scheduling process ends. And decision-making agent utilizes the problem formulation model to evaluate the manufacturing performance.

During the production task execution process, if real-time monitoring agent perceives one disturbance event, it will inform the decision-making agent. Then decision-making agent will reconfigure the manufacturing environment and notify other agents to adjust accordingly.

3. Five key modules in MAS

3.1. Problem formulation module

Using the problem formulation module, the model of production scheduling problem in smart factory is given below. There exist a set of a machines $M = \{M_x, x = 1, \ldots, a\}$, a set of b transportation equipment $TE = \{TE_y, y = 1, \ldots, b\}$, a set of m jobs $J = \{J_i, i = 1, \ldots, m\}$, and each job J_i has a set of n_i operations $J_i = \{O_{ij}, j = 1, \ldots, n_i\}$. Each operation can be processed on different machines, and the processing time and cutting power of an operation on different machine candidates are different. Each machine has buffers to temporarily store the jobs waiting to be processed and the jobs that have been processed. All jobs and transportation equipment are stored in the ASRS at the beginning. During the manufacturing process, the transportation equipment is responsible for transporting jobs from one location to another. When all operations of a job are finished, this job needs to be transported to ASRS for storage by one transportation equipment. When all jobs return to ASRS in the form of finished products, the manufacturing task ends. It should be noted that if the transportation equipment does not have a transportation task, it will stay in place until a new task arrives. The problem is to determine the appropriate allocation scheme of machines and transportation equipment at each scheduling point to optimize the makespan and total energy consumption.

As for the multi-objective optimization model of smart factory, a mixed integer linear programming (MILP) model is developed to optimize two objectives. The notations used in the MILP model are given in Table 1.

The MILP model includes the objective function and constraints. In order to formulate the objective function, the modeling process of makespan and total energy consumption is as follows.

(1) Makespan.

The first objective is to minimize the makespan, and the makespan is shown in Eq. (1):

$$T = \max_{1 \leqslant i \leqslant m}(CT_i) \tag{1}$$

It should be mentioned that a job is not finished until the corresponding workpiece is transported to the ASRS.

(2) Total energy consumption.

The second objective is to minimize the total energy consumption,

Table 1
Notations used in the MILP model.

Notation	Description
Indexes	
x, z	Index of machines
y	Index of transportation equipment
i	Index of jobs
j	Index of operations
Objectives	
T	Makespan for the schedule
E	Total energy consumption for the schedule
Variables for jobs, machines, transportation equipment and ASRS	
CT_i	Completion time of job J_i
A_{ijx}	Assignment binary variable that is set to 1 if operation O_{ij} is to be processed on machine M_x, and 0 otherwise
t_{ijx}	Processing time of operation O_{ij} on machine M_x
IT_x	Total idle time for machine M_x
PT_x	Total processing time for machine M_x
AT_{ijx}	Time when operation O_{ij} arrives at the processing machine M_x
ST_{ijx}	Time when operation O_{ij} starts processing on machine M_x
B_{yij}	Assignment binary variable that is set to 1 if transportation equipment TE_y is assigned the transportation task of operation O_{ij} after the processing of operation $O_{i(j-1)}$ is finished, and 0 otherwise
LT_{yx}	Time when transportation equipment TE_y leaves machine M_x
$\Delta t\ (c, d)$	Time for transportation equipment TE_y from location c to location d
ML_x	Location of machine M_x
LA	Location of ASRS
Variables for energy consumption	
CP_{ijx}	Cutting power of operation O_{ij} on machine M_x
IP_x	Idle power of machine M_x
TP	Transportation power of transportation equipment
e	Auxiliary average energy requirement per unit time
PEC	Processing energy consumption module
IEC	Idle energy consumption module
TEC	Transportation energy consumption module
AEC	Auxiliary energy consumption module

which is composed of four modules: *PEC*, *IEC*, *TEC* and *AEC*. The detailed description of four energy consumption modules is given below.

PEC is produced on machines in the processing state, which depends on the cutting power and processing time. Therefore, *PEC* is expressed by Eq. (2):

$$PEC = \sum_{i=1}^{m}\sum_{j=1}^{n_i}\sum_{x=1}^{a}\left(CP_{ijx} \cdot A_{ijx} \cdot t_{ijx}\right) \tag{2}$$

IEC is consumed on machines in the idle running state, and it can be calculated by the idle power and idle time. However, as mentioned before, only when all jobs return to the ASRS will all machines stop running. As a result, the total idle time (IT_x) for machine M_x can be converted into the difference between makespan and processing time, and is computed by Eq. (3):

$$IT_x = T - PT_x = \max_{1 \leqslant i \leqslant m}(CT_i) - \sum_{i=1}^{m}\sum_{j=1}^{n_i}\left(A_{ijx} \cdot t_{ijx}\right) \tag{3}$$

Therefore, the equation for calculating *IEC* is shown in Eq. (4):

$$IEC = \sum_{x=1}^{a}\left(IP_x \cdot IT_x\right) = \sum_{x=1}^{a}\left(IP_x \cdot \left(\max_{1 \leqslant i \leqslant m}(CT_i) - \sum_{i=1}^{m}\sum_{j=1}^{n_i}\left(A_{ijx} \cdot t_{ijx}\right)\right)\right) \tag{4}$$

TEC is produced on all transportation equipment in the transportation state, which is decided by the transportation power and transportation time. Among them, the transportation power is equal for all equipment. The transportation time can be divided into three kinds: transportation time from ASRS to a machine, transportation time between different machines and transportation time from a machine to ASRS. Therefore, *TEC* can be obtained by Eq. (5):

$$TEC = \sum_{i=1}^{m}\sum_{x=1}^{a}(TP\cdot A_{i1x}\cdot\Delta t(LA, ML_x)) + \sum_{i=1}^{m}\sum_{j=2}^{n_i}\sum_{x=1}^{a}\sum_{z=1}^{a}(TP\cdot A_{i(j-1)x}\cdot A_{ijz}\cdot\Delta t(ML_x, ML_z)) + \sum_{i=1}^{m}\sum_{x=1}^{a}(TP\cdot A_{in_ix}\cdot\Delta t(ML_x, LA)) \quad (5)$$

AEC is the auxiliary energy required to support the production environment in the entire manufacturing process, which can be calculated by Eq. (6):

$$AEC = e\cdot\max_{1\leqslant i\leqslant m}(CT_i) \quad (6)$$

Hence, the total energy consumption can be expressed by Eq. (7):

$$E = PEC + IEC + TEC + AEC \quad (7)$$

Based on the Eq. (1) and Eq. (7), this paper adopts the weighted sum method and uses two weights (ω_1 and ω_2) to transform multi-objective optimization into single objective optimization. So the MILP model requiring the optimization of two scheduling performance criteria is formulated as follows:

$$\min f = \omega_1\cdot T + \omega_2\cdot E \quad (8)$$

subject to.

$$CT_i \leqslant T, \qquad i = 1, 2, \ldots, m \quad (9)$$

$$\sum_{x=1}^{a} A_{ijx} = 1, \qquad i = 1, 2, \ldots, m;\ j = 1, 2, \ldots, n_i;\ x = 1, 2, \ldots, a \quad (10)$$

Table 2
Three kinds of dispatching rules.

Number	Rule name	Description
1	SPT_J	The job with operation for shortest processing time has the highest priority.
2	LPT_J	The job with operation for longest processing time has the highest priority.
3	SRT_J	The job with shortest remaining time has the highest priority.
4	LRT_J	The job with longest remaining time has the highest priority.
5	NEC_J	The job with operation for minimal energy consumption has the highest priority.
6	XEC_J	The job with operation for maximal energy consumption has the highest priority.
7	SPT_{TE}	The transportation equipment with shortest pickup time is allocated first.
8	NEC_M	The machine with minimal energy consumption is allocated first.
9	XEC_M	The machine with maximal energy consumption is allocated first.
10	NLR_M	The machine with minimal loading rate is allocated first.
11	SPT_M	The machine with shortest processing time is allocated first.
12	LPT_M	The machine with longest processing time is allocated first.
13	NBT_M	The machine with minimal buffer time is allocated first.

$$\sum_{y=1}^{b}\sum_{x=1}^{a}(B_{yij}\cdot A_{i(j-1)x}\cdot LT_{yx}) + \sum_{x=1}^{a}\sum_{z=1}^{a}(A_{i(j-1)x}\cdot A_{ijz}\cdot\Delta t(ML_x, ML_z)) = \sum_{z=1}^{a}(A_{ijz}\cdot AT_{ijz}), \quad i = 1, 2, \ldots, m;\ j = 2, 3, \ldots, n_i \quad (11)$$

$$\sum_{z=1}^{a}(A_{ijz}\cdot AT_{ijz}) \leqslant \sum_{y=1}^{b}\sum_{z=1}^{a}(B_{yij}\cdot A_{ijz}\cdot LT_{yz}), \qquad i = 1, 2, \ldots, m;\ j = 2, 3, \ldots, n_i \quad (12)$$

$$\sum_{x=1}^{a}(A_{ijx}\cdot AT_{ijx}) \leqslant \sum_{x=1}^{a}(A_{ijx}\cdot ST_{ijx}), \qquad i = 1, 2, \ldots, m;\ j = 1, 2, 3, \ldots, n_i \quad (13)$$

$$\sum_{x=1}^{a}(A_{i(j-1)x}\cdot ST_{i(j-1)x}) + \sum_{x=1}^{a}(A_{i(j-1)x}\cdot PT_{i(j-1)x}) + \sum_{x=1}^{a}\sum_{z=1}^{a}(A_{i(j-1)x}\cdot A_{ijz}\cdot\Delta t(ML_x, ML_z)) \leqslant \sum_{z=1}^{m}(A_{ijz}\cdot AT_{ijz}), \quad i = 1, 2, \ldots, m;\ j = 2, 3, \ldots, n_i \quad (14)$$

$$\sum_{j=1}^{n_i}\sum_{x=1}^{a}\left(A_{ijx}\cdot t_{ijx}\right)+\sum_{x=1}^{a}\left(A_{i1x}\cdot\Delta t(LA,ML_x)\right)+\sum_{j=2}^{n_i}\sum_{x=1}^{a}\sum_{z=1}^{a}\left(A_{i(j-1)x}\cdot A_{ijz}\cdot\Delta t(ML_x,ML_z)\right)+\sum_{x=1}^{a}\left(A_{in_ix}\cdot\Delta t(ML_x,LA)\right)\leqslant CT_i,\quad i=1,\ 2,\ \ldots,\ m;\ \ j=1,\ 2,\ \ldots,\ n_i;\ \ x,z=1,\ 2,\ \ldots,\ a \tag{15}$$

Objective function (8) minimizes the two objectives: makespan and total energy consumption. Constraint (9) presents a size constraint between the completion time of each job and makespan. Constraint (10) ensures that one operation only can choose one machine for processing. Constraint (11) shows that transportation equipment will not stop during transportation for any reason. Constraint (12) gives a possibility that transportation equipment may stay at the machine for some time until it accepts a new transportation task. Constraint (13) means that after arriving at the machine, the job may be temporarily placed in the buffer and wait for processing. Constraint (14) shows that there are precedence constraints between different operations of each job. In other words, operation O_{ij} should arrive at the machine M_z after the completion of operation $O_{i(j-1)}$ and the transportation of corresponding workpiece. At the same time, constraint (14) means that job J_i may be temporarily placed in the buffer of machine M_x and wait for transportation after the completion of operation $O_{i(j-1)}$. Constraint (15) guarantees that job J_x is not finished until the corresponding workpiece returns to ASRS.

3.2. Scheduling point module

The judgment result of scheduling point module is the driving force of the whole manufacturing execution process. Hence, the scheduling point module in real-time monitoring agent needs to be explained in detail.

The scheduling point is the time point when the decision-making agent needs to provide the allocation scheme. The object is the jobs waiting for transportation. The allocated manufacturing resources are machines and transportation equipment. Therefore, if there are one or more jobs waiting for transportation and one or more idle transportation equipment, this time point is the scheduling point. The scheduling point in this paper needs to satisfy two conditions. The former can be understood as a mixture of two situations: 1) the job needs to be transported to a machine for processing; 2) the job needs to be transported to ASRS for end product storage. The latter is not usually considered in the literature about real-time production scheduling. However, in order to make the problem closer to reality, this paper considers the transportation constraints in smart factory, and so the number of idle transportation equipment is taken into account in the scheduling point. In addition, the latter serves to delay the scheduling point appropriately. This way can increase the number of optional machines at some time, thus expanding the solution space of scheduling. Theoretically, decision-making agent can have the opportunity to search for a better scheduling scheme.

3.3. Scheduling rule library

In most of the literatures involving heuristics-based scheduling, researchers usually use some common scheduling rules. But the number of these scheduling rules is huge, especially in composite dispatching rules. For example, there are 10 job sequencing rules and 10 machine selection rules, and then 100 composite dispatching rules are formed. So how to select the appropriate rules from these scheduling rules to effectively solve the problem to be studied is a difficult problem. Therefore, aiming at the scheduling problem of smart factory, this paper uses GP to generate high-quality scheduling rules, so as to form the scheduling rule library and improve the performance of proposed method. The contents of GP are given below.

(1) Function set. This paper uses six mathematical operators (+, -, *, /, min, max) to constitute the function set. It should be noted that the division is a protective division. In other words, when the divisor is 0, the division operation returns 1. Otherwise, it returns the normal calculated value.

(2) Terminal set. This paper uses twelve composite dispatching rules Rule_i ($i = 1, \ldots, 12$) to constitute the terminal set. Each composite dispatching rule consists of three kinds of dispatching rules, and it is determined by the nature of smart factory. The smart factory includes multiple jobs, multiple machines and multiple transportation equipment. Therefore, at each scheduling point, the agent needs to consider three problems: 1) how to determine the priorities of multiple jobs waiting for transportation; 2) how to select the transportation carrier from multiple transportation equipment to transport job to the designated location; 3) how to select the processing equipment from multiple machine candidates. Based on these problems, the corresponding rules are job sequencing rule, transportation equipment selection rule and machine selection rule, which are shown in the line 1–6, line 7 and line 8–13 of Table 2. The subscript of each rule is the abbreviation of corresponding object.

It should be emphasized that, since the time of one transportation task is usually short, and in order to make it easier for the agent to determine the state-rule mapping relationship, this paper simplifies the composite dispatching rule by fixing the transportation equipment selection rule. At the same time, it is necessary to explain the rule SPT_{TE} in detail. Assume that transportation equipment TE_y is carrying out a transportation task. There are multiple tasks in its buffer, and the target location of the last task is ML_x. All transportation equipment are competing for the transportation task of job J_i, which is located in ML_z. Therefore, the pickup time of transportation equipment TE_y for job J_i includes three parts: 1) the remaining time of the task being performed by transportation equipment TE_y; 2) the completion time of all tasks in the buffer of transportation equipment TE_y; 3)$\Delta t(ML_x,\ ML_z)$. Moreover, in line 13, the buffer time represents the sum of processing times for all operations in the buffer of the machine.

Based on these rules in Table 2, 12 composite dispatching rules Rule_i ($i = 1, \ldots, 12$) are formed in the terminal set, which are as follows: 1) $\text{SRT}_{\text{J}} + \text{SPT}_{\text{TE}} + \text{NEC}_{\text{M}}$; 2) $\text{SRT}_{\text{J}} + \text{SPT}_{\text{TE}} + \text{NLR}_{\text{M}}$; 3) $\text{LRT}_{\text{J}} + \text{SPT}_{\text{TE}} + \text{SPT}_{\text{M}}$; 4) $\text{LRT}_{\text{J}} + \text{SPT}_{\text{TE}} + \text{LPT}_{\text{M}}$; 5) $\text{LRT}_{\text{J}} + \text{SPT}_{\text{TE}} + \text{NBT}_{\text{M}}$; 6) $\text{LPT}_{\text{J}} + \text{SPT}_{\text{TE}} + \text{SPT}_{\text{M}}$; 7) $\text{LPT}_{\text{J}} + \text{SPT}_{\text{TE}} + \text{NBT}_{\text{M}}$; 8) $\text{SPT}_{\text{J}} + \text{SPT}_{\text{TE}} + \text{NEC}_{\text{M}}$; 9) $\text{SPT}_{\text{J}} + \text{SPT}_{\text{TE}} + \text{XEC}_{\text{M}}$; 10) $\text{SPT}_{\text{J}} + \text{SPT}_{\text{TE}} + \text{NLR}_{\text{M}}$; 11) $\text{NEC}_{\text{J}} + \text{SPT}_{\text{TE}} + \text{NBT}_{\text{M}}$; 12) $\text{XEC}_{\text{J}} + \text{SPT}_{\text{TE}} + \text{NEC}_{\text{M}}$.

(3) Generation of initial population. This paper uses the ramped-half-and-half method to generate the initial population. In this method, the half of initial individuals are generated by the complete method and the other individuals are generated by the growth method. Each individual is represented by the structure of binary tree. In the complete method, all terminal operations of an individual are located at the maximum depth of the tree. In the

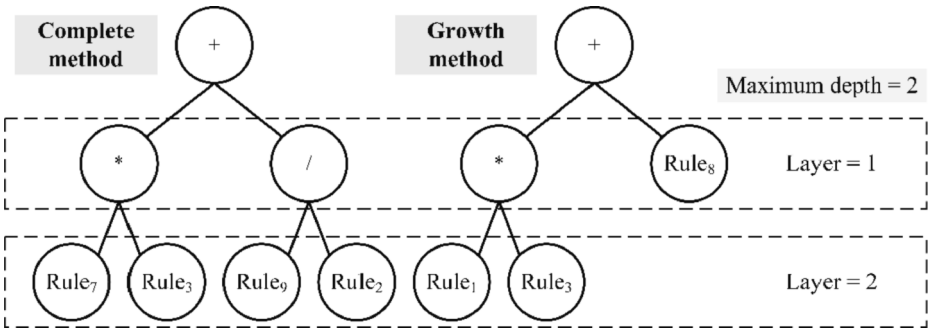

Fig. 3. Example of individuals in two methods.

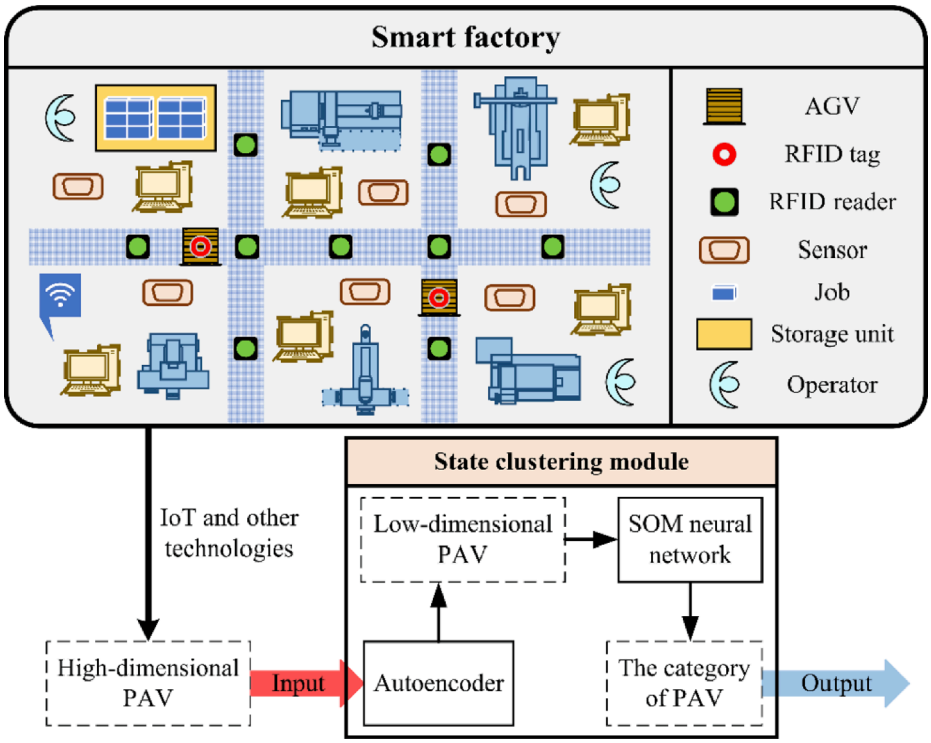

Fig. 4. The specific implementation process of state clustering module.

growth method, all nodes of the tree can be arbitrarily selected from the function set and the terminal set, that is, it is allowed to select the terminal operation at any layer, so the depth of generated tree may be less than the maximum depth allowed. Fig. 3 gives the example of individuals in two methods.

(4) Selection part. This paper adopts the roulette method.

(5) Crossover part. In a pair of individuals, a node is randomly selected as the crossover point, and the subtree with this node as the root node is exchanged to obtain two offspring.

(6) Mutation part. In an individual, a node other than the root node are randomly selected as the mutation point. The subtree with the mutation point as the root node is deleted, and then a new subtree is randomly generated at the mutation point. The depth of the new individual cannot exceed the maximum depth.

(7) Fitness calculation. This part is divided into three parts. Firstly, an individual needs to be converted into a GP rule. For example, the left individual in Fig. 3 corresponds to the GP rule $Rule_7$*$Rule_3$ + $Rule_{10}$/$Rule_2$. Secondly, in the real-time scheduling simulation, the GP rule was used at each scheduling point. It should be mentioned that mathematical operations between different dispatching rules may encounter the problem of different units, such as time(unit: min)/energy consumption (unit: kW). Therefore, each rule involved will perform the priority normalization in the use of each GP rule. In other words, the priority of object d under a rule is calculated by Eq. (16).

Table 3
Production attributes in the PAV.

Number	Description
1–4	The mean/standard deviation/median/range of completion rates for all jobs
5	Number of jobs in the TTP
6–7	The mean/standard deviation of completion rates for all jobs in the TTP
8–10	The mean/standard deviation/range of processing times for all operations in the TTP
11–13	The mean/standard deviation/range of energy consumptions for all operations in the TTP
14–15	The mean/standard deviation of remaining processing times for all jobs in the TTP
16–17	The mean/standard deviation of remaining energy consumptions for all jobs in the TTP
18–19	The mean/standard deviation of utilization rates for all machines
20	The mean of numbers of jobs in the buffer for all machines

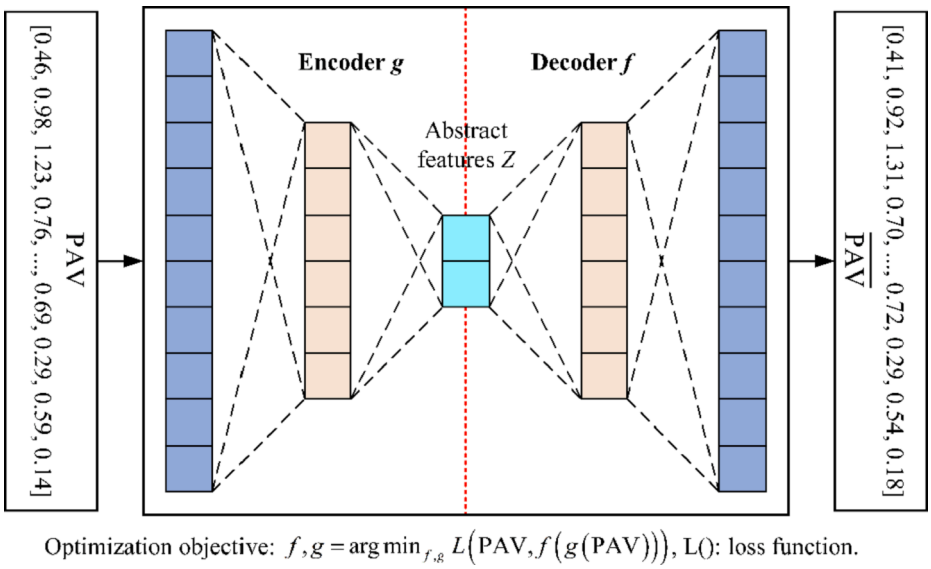

Fig. 5. The architecture of autoencoder.

$$PRI_d = \frac{k_d - k_{\min}}{k_{\max} - k_{\min}} \cdot p + q \tag{16}$$

where k_d represents the attribute value of object d, k_{min} and k_{max} represent the minimum and maximum of attribute values for all sorting objects, and PRI_d represents the final priority of object d that participates in the priority calculation of GP rules. In Eq. (16), when the single rule is SPT_J or other similar rules, $p = -1$ and $q = 1$; when the single rule is LPT_J or other similar rules, $p = 1$ and $q = 0$. Thirdly, when the order is finished, the fitness, which is in inverse proportion to the scheduling objective, is obtained.

Based on the above content, the GP algorithm is as follows.

Algorithm 1 GP rule generation for smart factory

1: Initialize the iteration number, maximum depth, population size, crossover rate, mutation rate
2: Initialize the population using the ramped-half-and-half method
3: Calculate the fitness of each individual.
4: Obtain some best GP rules from the population and form the scheduling rule library
5: **for** $i = 1$ to iteration number **do**
6: Using the roulette method to do the selection
7: Do or do not the crossover part based on the crossover rate
8: Do or do not the mutation part based on the mutation rate
9: Update the fitness of each individual
10: **if** there exist some better GP rules in the population than those of the scheduling rule library **then**
11: Update the scheduling rule library
12: **end if**
13: **next for**

3.4. State clustering module

The state clustering module is responsible for classifying the input PAV, and it includes the autoencoder and SOM neural network. The specific implementation process is shown in Fig. 4. Firstly, the smart factory obtains the PAV by using IoT and other technologies at the scheduling point and do normalization for it. Secondly, input the high-dimensional PAV into the autoencoder to obtain the low-dimensional PAV. Thirdly, input PAV into the SOM neural network to get the corresponding category of PAV. Among them, the PAV need to be defined, and the autoencoder and SOM neural network need to be trained in advance. The relevant details are given below.

3.4.1. Production attribute vector

The PAV is a vector, which represents the production state of smart factory at the scheduling point. It includes a series of representative production attributes, which represent the state features of all jobs, all jobs in the transportation task pool (TTP) and all machines. Here we define the TTP as the set of all jobs that wait for transportation at one time point. It should be emphasized that the PAV does not contain the state features of transportation equipment. In the dispatching rules provided in this paper, the transportation equipment selection rule is fixed, that is, the difference between transportation equipment selection rules is not considered in terms of rule. Therefore, when forming the state-rule mapping relationship, the state does not need to include manufacturing data about transportation equipment. Based on the above description, all production attributes in the PAV are given in Table 3.

The value ranges of state features in Table 3 are different. It can cause that the state features with large values dominate the state features with

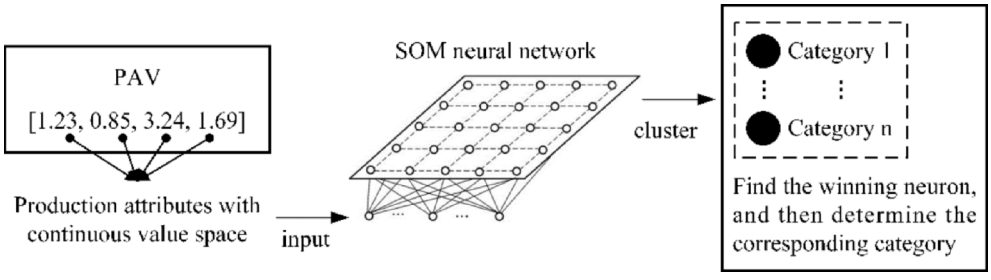

Fig. 6. Transformation of manufacturing data using SOM neural network.

small values in the classification process. To avoid this phenomenon, this paper firstly collects massive PAVs, and then calculates the mean and standard deviation of each production attribute. Finally, if a new PAV at the scheduling point is generated, do normalization for it with the help of the mean and standard deviation.

3.4.2. PAV dimension reduction using autoencoder

The framework of autoencoder includes two modules: encoder and decoder, as shown in Fig. 5. The encoding process is to map the input sample PAV into the abstract features Z through the encoder (g). The decoding process is to map the abstract features Z into the reconstructed sample $\overline{\text{PAV}}$ through the decoder (f). The optimization goal is to optimize the encoder and decoder at the same time by minimizing the reconstruction error, so as to obtain the effective abstract feature representation Z of the input PAV. At the same time, in the autoencoder, the number of neurons in the middle hidden layer is less than that in the input layer. Therefore, the autoencoder can realize the effective dimension reduction of input samples: PAV to Z.

The training and use process of autoencoder is shown in Algorithm 2.

Algorithm 2 The training and use process of autoencoder

1: Collect lots of PAVs and initialize the autoencoder
2: **for** $i = 1$ to iteration number **do**
3: Input a batch of PAVs to the autoencoder and get some reconstructed samples $\overline{\text{PAV}}$
4: Optimize the encoder and decoder with function $f, \quad g = \text{argmin}_{f, \ g} L(\text{PAV}, \ \overline{\text{PAV}})$
5: **next for**
6: Save the parameters of autoencoder
7: **if** a new and high-dimensional PAV is generated **then**
8: Input this new PAV to the trained autoencoder and get the low-dimensional $\text{PAV} = Z = g(\text{PAV})$
9: **end if**

3.4.3. PAV clustering using SOM neural network

After the processing of autoencoder, the low-dimensional PAV is input to SOM neural network for clustering. Compared with the original high-dimensional PAV, SOM neural network can deal with this low-dimensional PAV more effectively. Next, the training and use process of SOM neural network are given below.

The multi-dimensional and continuous characteristics of PAV are not conducive for the agent to use RL method to learn to select the appropriate GP rule at each scheduling point. Therefore, this paper uses a competitive learning method in unsupervised learning, namely SOM (Sinclair et al., 2013). It can realize the clustering of PAV without external help, so as to transform continuous multiple-dimensional data space into discrete data space. The clustering process of SOM neural network for manufacturing data is shown in Fig. 6. This can greatly help the agent learn the state-rule mapping relationship.

In the unsupervised system based on competitive learning, each neuron in the competition layer competes for the opportunity to respond to the input pattern. Only one neuron becomes the winner of competition, that is, winning neuron, and the weight vectors associated with the winning neuron will be adjusted in a more competitive direction. In this paper, a SOM neural network, which is called Kohonen network, is used for PAV clustering. It has two layers: input layer and output layer. The form of input layer is the same as that of back propagation network, and the number of nodes is equal to the dimension of PAV. The output layer is also called the competition layer. The arrangement of its neurons has many forms, and this paper uses two-dimensional plane form. Each neuron connects laterally with the surrounding neurons. Moreover, in the Kohonen network, each neuron in the competition layer corresponds a x-dimensional weight vector, where $\times$ represents the number of neurons in the input layer.

The implementation of SOM neural network is divided into training process and use process. The training process is responsible for training a network that has different responses to different kinds of PAVs. Then in the use process, through inputting PAV into the trained network, the agent can determine the corresponding category according to the category of winning neuron, so as to realize a discrete state space. The explanation of training process is given below. There are many iterations in the training process. At the beginning of each iteration, samples of the training set (namely PAVs) are randomly input into the network. Then the winning neuron corresponding to each sample is found by calculating the dot product of each sample and all weight vectors. Moreover, the weight vectors are adjusted for the neurons in the topological neighborhood of winning neuron. After lots of learning, each neuron in the competition layer gradually becomes sensitive to the specific manufacturing data category. Finally, the k-means clustering algorithm is used to cluster all the trained weight vectors. It should be noted that there are two reasons for using k-means algorithm to cluster the trained SOM neural network. Firstly, there is not a single neuron corresponding to a category in the competition layer, because the adjacent weight vectors are sometimes close, and they can be grouped into a category. Secondly, in the face of a large number of manufacturing data, the scale of competition layer for SOM neural network is often very large. If k-means algorithm is not used to cluster these neurons of competition layer (i.e., the corresponding weight vectors), a neuron will correspond to a category. This leads to the poor clustering effect of SOM neural network on PAV, that is, PAVs are divided into lots of categories. For example, if the scale of competition layer is 15*15, PAVs can be divided into 225 categories. Meanwhile, in this paper, the dimension of discrete state space is equal to the number of categories for neurons in the competition layer. Therefore, the dimension of discrete state space will be too large. In the face of high-dimensional state space, the decision-making agent cannot learn the effective state-rule mapping relationship through training using Q-learning algorithm, and the final objective of optimizing the scheduling objective cannot be achieved. This is another reason to use k-means algorithm to cluster the trained weight vectors.

Based on the above principle, the training and use process of SOM neural network is given in Algorithm 3.

Algorithm 3 Training and use process of SOM neural network

1: Initialize parameters $m, \quad IN, \quad BZ, \quad N(1)$
2: Initialize the weight vectors of output layer $\omega_x(1) \quad (x = 1, \quad 2, \quad ..., \quad m)$
3: **for** $i = 1$ to IN **do**
4: Do normalization for all weight vectors and obtain $\widehat{\omega}_x(1) \quad (x = 1, \quad 2, \quad ..., \quad m)$
5: Take BZ samples from the training set randomly and obtain $S_y \quad (y = 1, \quad 2, \quad ..., \quad BZ)$
6: Do normalization for all samples and obtain $\widehat{S}_y \quad (y = 1, \quad 2, \quad ..., \quad BZ)$
7: **for** $y = 1$ to BZ **do**
8: Calculate the dot products $\widehat{S}_y \cdot \widehat{\omega}_x \quad (x = 1, \quad 2, \quad ..., \quad m)$
9: Find the winning neuron $wn = \underset{x}{\text{argmax}}\left(\widehat{S}_y \cdot \widehat{\omega}_x\right) \quad (x = 1, \quad 2, \quad ..., \quad m)$
10: Determine all the neurons in the topological neighborhood of neuron wn according to $N(i)$
11: **for** each neuron j in the topological neighborhood **do**
12: Determine the topological distance N_j corresponding to neuron j
13: Calculate the learning rate $\eta(i, N_j)$ according to Eq. (17)
14: Update the weight vector by $\widehat{\omega}_j(i+1) = \widehat{\omega}_j(i) + \eta(i, \quad N_j) \cdot \left[\widehat{S}_y - \widehat{\omega}_j(i)\right]$
15: **next for**
16: **next for**
17: Update the Topological neighborhood radius $N(i)$ according to Eq. (18)
18: **next for**
19: Use the k-means clustering algorithm to cluster all weight vectors
20: Save the category of each neuron (or each weight vector) in the competition layer
21: **if** a new and low-dimensional PAV needs to be clustered **then**
22: Input this new PAV to the trained SOM neural network
23: Find the winning neuron and determine the corresponding category
24: **end if**

In Algorithm 3, m represents the number of neurons in the output layer, IN represents the total iteration number, BZ represents the batch size, $N(i)$ represents the topological neighborhood radius at the ith iteration, $\omega_j(i)$ represents the weight vector corresponding to neuron j at the ith

iteration, and N_j represents the topological distance of neuron j and neuron wn. The formula for calculating the learning rate is shown in Eq. (17):

$$\eta(i,\quad N_j) = \frac{e^{-N_j}}{i+2} \tag{17}$$

where learning rate $\eta(i,\quad N_j)$ decreases with the increase of topological distance N_j or iteration times i. The update formula of the topological neighborhood radius is expressed by Eq. (18):

$$N(i) = C\cdot(1 - i/IN) \tag{18}$$

where C represents a constant related to the output layer. Topological neighborhood radius $N(i)$ decreases with the increase of iteration times i.

3.5. GP rule selector

The implementation of GP rule selector includes training process and use process. In the training process, the GP rule selector forms the state-rule mapping relationship through RL. In the use process, the GP rule selector gives the appropriate rule according to the PAV category result given by state clustering module. Next, the explanation of training process is given below.

3.5.1. Background of RL and Q-learning

The general framework of RL problem is Markov decision process (MDP) (Park et al., 2020). In the MDP, agent and environment achieve the goal through the interactive learning. Specifically, the agent observes the environmental state s_t at the time t. Based on the observed state, the agent chooses the action a_t. At the next time $t+1$, due to the implementation of action a_t, the agent receives a reward r_t and enters a new state s_{t+1}. Thus, MDP and agent together give a sequence: $\{s_0, a_0, r_0, s_1, \ldots, s_t, a_t, r_t, s_{t+1}, \ldots\}$. The objective of agent is to find an optimal policyπ^*, which can maximize the expected sum of long-term rewards.

Watkins and Dayan proposed the well-known Q-learning algorithm in 1992 (Watkins and Dayan, 1992). Q-learning algorithm is a model-free and off-policy TD (temporal-difference) control algorithm, which is an important breakthrough in the early stage of RL. The traditional Q-learning algorithm describes the value function in the form of a table, that is, all Q-values are recorded in a table named Q-table. Agent reserves a Q-value $Q(s, a)$ for each state-action pair, which represents the expected long-term reward of taking action a in state s. According to the Q-table, the agent can know the estimated Q-value of each action in the current state, and then decide what action should be taken to achieve the maximum cumulative reward in the long run. Every time when an interaction occurs, $Q(s, a)$ is updated according to the Eq. (19):

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right] \tag{19}$$

where α is the learning rate and$\alpha \in (0, 1]$, γ is the discount factor which differentiates the relative importance of short-term reward and long-term reward, and$\gamma \in (0, 1]$.

3.5.2. RL-based training algorithm

In order to train the GP rule selector, the production scheduling problem needs to be transformed into MDP, and four elements need to be modeled: scheduling point, state space, action space and reward function.

(1) Scheduling point. It is given in Section 3.2.
(2) State space. It is all categories of neurons in the competition layer of SOM neural network.
(3) Action space. It is all GP rules in the scheduling rule library.

Reward function. Reward function should be linked with the scheduling objectives. In this way, maximizing the cumulative reward of the agent is equivalent to optimizing the specified scheduling objectives. Based on this idea, the reward function is divided into two parts. The first part is the negative value of time between two successive scheduling points ($-T_t^{t+1}$). The second part is the negative value of energy consumption between two successive scheduling points ($-E_t^{t+1}$). Therefore, the reward function is given in Eq. (20):

$$r_t = -\omega_1 \cdot T_t^{t+1} - \omega_2 \cdot E_t^{t+1} \tag{20}$$

It can be seen that the cumulative value of r_t is almost equal to the scheduling objective f in Eq. (8).

In the traditional Q-learning, the max operator uses the same values both to select and to evaluate an action, which results in overoptimistic value estimates. To prevent this, double Q-learning (van Hasselt, 2010) decouples the selection from the evaluation. In the original algorithm, two value functions are learned by assigning each experience randomly to update one of the two value functions, such that there are two Q-tables, Q_1 and Q_2. The update formula in double Q-learning is given in Eq. (21):

$$Q_1(s_t,\quad a_t) \leftarrow Q_1(s_t,\quad a_t) + \alpha\left[r_t + \gamma Q_2\left(s_{t+1},\quad \operatorname*{argmax}_a Q_1(s_{t+1},\quad a)\right) - Q_1(s_t,\quad a_t)\right] \tag{21}$$

The probability for the occurrence of Eq. (21) is 50%. In other cases, Q_1 and Q_2 are exchanged and the same update is executed.

Based on the above description, the training algorithm of GP rule selector is provided in Algorithm 4.

Algorithm 4 Training algorithm of GP rule selector

Initialize $TEN,\quad \varepsilon,\quad DC,\quad LB,\quad \alpha,\quad \gamma$
Initialize two Q-tables Q1 and Q2
for episode $i = 1$ **to** TEN **do**
  Initialize the smart factory environment
  for scheduling point $t = 1$ **to** LSP **do**
    Observe the PAC
    Input PAC into state clustering module to get the clustering result, i.e., state s_t
    With probability ε randomly select an action a_t, otherwise $a_t = \operatorname*{argmax}_a(Q_1(s_t,\quad a) + Q_2(s_t,\quad a))$
    Execute the selected action a_t, observe the next state s_{t+1} and receive reward r_t
    With probability 50% choose UPDATE(Q_1), otherwise UPDATE(Q_2)
    if UPDATE(Q_1) **then**
      $Q_1(s_t,\quad a_t) \leftarrow Q_1(s_t,\quad a_t) + \alpha\left[r_t + \gamma Q_2\left(s_{t+1},\quad \operatorname*{argmax}_a Q_1(s_{t+1},\quad a)\right) - Q_1(s_t,\quad a_t)\right]$
    else if UPDATE(Q_2) **then**
      $Q_2(s_t,\quad a_t) \leftarrow Q_2(s_t,\quad a_t) + \alpha\left[r_t + \gamma Q_1\left(s_{t+1},\quad \operatorname*{argmax}_a Q_2(s_{t+1},\quad a)\right) - Q_2(s_t,\quad a_t)\right]$
    end if
    Gradually decrease ε by DC until to LB
  next for
next for

In Algorithm 4, TEN represents the total episode number, ε represents the initial probability and it will gradually decrease by DC to a minimal value LB, DC represents the decreasing coefficient, LB represents the

Table 4
Parameter settings of production scheduling problems.

Parameter	Value
number of machines	15
number of AGVs	5
number of jobs	Unif [30, 80]
processing time of one operation (min)	Unif [10, 40]
Rated power of a machine (kW)	Unif [4, 15]
idle power of a machine (kW)	[1.5, 4.0]
transportation power of each AGV (kW)	3.5
auxiliary power (kW)	2.5

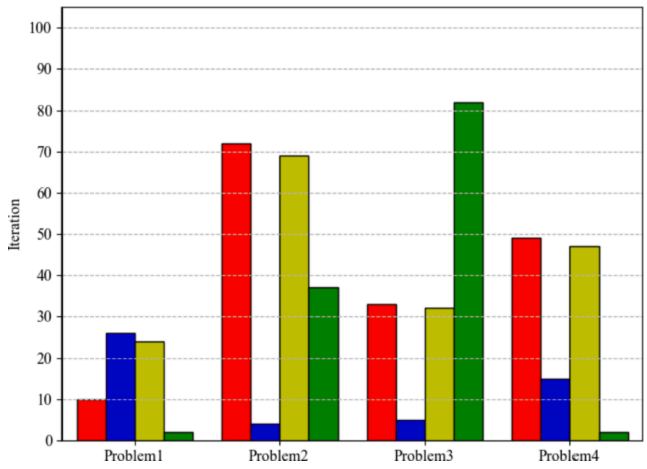

Fig. 7. The number of iterations for obtaining final GP rules.

lower bound, and *LSP* represents the last scheduling point. This paper adopts the ε - greedy policy to balance the exploration and exploitation in the training process. In other words, the agent focuses on exploration in the early stage, while it focuses on exploitation in the later stage. In addition, in the process of executing action a_t, the action needs to be converted into a specific scheduling scheme, which is with the help of decision-making agent. Then the relevant agents perform tasks according to the obtained scheme.

4. Comprehensive experiments

4.1. Platform and case introduction

Based on the manufacturing platform in a university laboratory, we built the smart factory system to verify the performance of proposed method. Meanwhile, in order to execute the training and performance comparison, some experiments are carried out in the simulation environment. It is a workstation equipped with Windows 10 64 operating systems, 16G RAM, Intel i9 3.50 GHz CPU.

The parameter settings of production scheduling problems are given in Table 4. In each scheduling problem, new orders are inserted from time to time. The constraints of experiments are as follows: 1) each AGV or machine can only transport or process a job at a time; 2) all jobs, machines and AGVs are available at the beginning; 3) when a job needs to be transported to a machine, any of all AGVs has the ability to become the transportation carrier. In addition, two weights corresponding to T and E used in the experiments are 0.7 and 0.3 respectively.

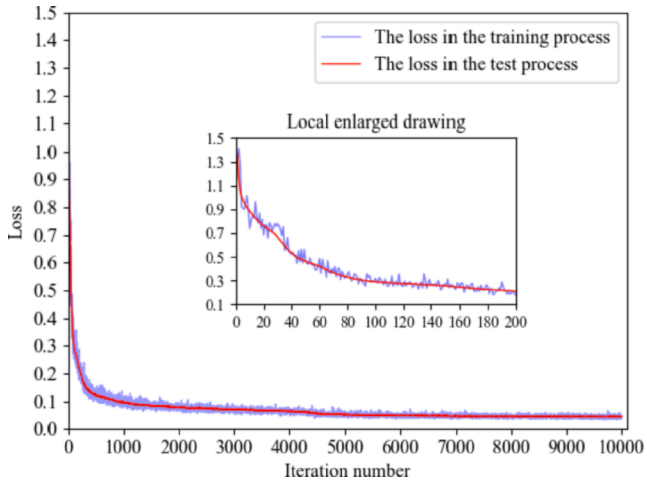

Fig. 8. The training loss and test loss of autoencoder.

4.2. GP rule generation experiment

In order to form the scheduling rule library, the GP rule generation experiment is carried out in this paper. In this experiment, the parameters of GP are given below: (1) population size: 50; (2) maximum depth of an individual: 3; (3) number of iterations: 100; (4) crossover rate: 0.5; (5) mutation rate: 0.05. In order to ensure the generality of the generated GP rules, the GP method iterates and evolves on four instances respectively. Then the four high-quality GP rules are obtained in each instance. The number of iterations for obtaining each final GP rule is shown in Fig. 7. It can be seen that after 100 iterations, the optimal individuals of population have fully evolved.

The obtained final GP rules are as follows.

(1) min(min($Rule_5$, $Rule_6$)*$Rule_6$, $Rule_5$);
(2) min($Rule_1$, $Rule_6$)*($Rule_6$ + $Rule_{10}$);
(3) max(min($Rule_1$, $Rule_6$)*($Rule_6$ + $Rule_{10}$), min($Rule_5$, $Rule_6$)*$Rule_6$);
(4) $Rule_5$ + min($Rule_3$, $Rule_{12}$);
(5) min($Rule_1$/$Rule_5$, $Rule_{11}$);
(6) $Rule_8$*$Rule_7$;
(7) ($Rule_1$-$Rule_9$ + min($Rule_1$, $Rule_{10}$))*($Rule_5$*min($Rule_6$, $Rule_3$));
(8) max($Rule_8$, $Rule_8$-$Rule_3$)*(($Rule_{11}$ + $Rule_6$)*$Rule_7$);
(9) (min($Rule_8$, $Rule_3$) + $Rule_{10}$*$Rule_5$)*(max($Rule_{11}$, $Rule_{10}$)/$Rule_{11}$);
(10) $Rule_8$ + $Rule_3$-$Rule_6$/$Rule_7$;
(11) min($Rule_3$, min($Rule_8$, $Rule_3$) + $Rule_{10}$*$Rule_5$);
(12) min($Rule_1$, min($Rule_3$, $Rule_8$)/($Rule_6$/$Rule_{11}$));
(13) $Rule_{12}$ + min($Rule_{11}$/$Rule_{10}$, $Rule_3$*$Rule_5$);
(14) $Rule_8$ + min($Rule_6$, $Rule_{11}$)*max($Rule_2$, $Rule_1$);
(15) $Rule_8$ + min($Rule_3$, $Rule_7$)-min($Rule_{12}$, $Rule_9$);
(16) min(min($Rule_3$, $Rule_5$), max($Rule_3$, $Rule_5$)) + $Rule_8$.

4.3. Training of state clustering module and GP rule selector

Before training the state clustering module, 50,000 PAVs are collected for preparation. Firstly, these data obtained in order are disturbed to make the training data and test data obey the same distribution as much as possible. Then, 95% of all PAVs are used to form the training data set, and the rest PAVs are used to form the test data set. The parameter settings of autoencoder training experiment are as follows: (1) number of layers: 8; (2) structure of autoencoder: (20, 128), ReLU, (128, 32), ReLU, (32, 16), ReLU, (16, 4), ReLU, (4, 16), ReLU, (16, 32), ReLU, (32, 128), ReLU, (128, 20); (3) weight initialization method: He initialization (He et al., 2015); (4) learning rate: 0.001; (5) batch size: 128; (6) iteration number: 10000; (7) loss function: mean square error. The training loss of each iteration will be recorded in this paper. At the same time, after the back propagation in each iteration, this paper will input all test PAVs into the autoencoder to obtain the corresponding test loss. The situation of two kinds of losses in the iteration process is shown in Fig. 8.

It can be seen in Fig. 8 that the training loss or test loss are close to 0 in the end. In other words, the autoencoder has been fully trained, and has the ability to convert 20-dimensional PAV to 4-dimensional PAV. Therefore, the SOM neural network is trained next. In the SOM neural network, the number of neurons in the competitive layer determines the granularity and scale of final model, which has a great impact on the accuracy and generalization ability of model. There is an empirical formula: Minimum number of neurons in the competition layer $=5\sqrt{N}$, where N represents the number of training samples. This paper has 50,000 PAVs, so the competition layer adopts the scale of 34*34. The number of neurons in the input layer is 4. The iteration number of training for SOM neural network is 500000.

Table 5
The CH index under each K value.

	$K=2$	$K=3$	$K=4$	$K=5$	$K=6$	$K=7$	$K=8$
CH index	889.06	1035.35	1135.95	1288.82	1332.28	1342.57	1408.28
	$K=9$	$K=10$	$K=11$	$K=12$	$K=13$	$K=14$	$K=15$
CH index	1448.48	1481.98	1517.79	1575.59	1605.85	1581.14	1552.23

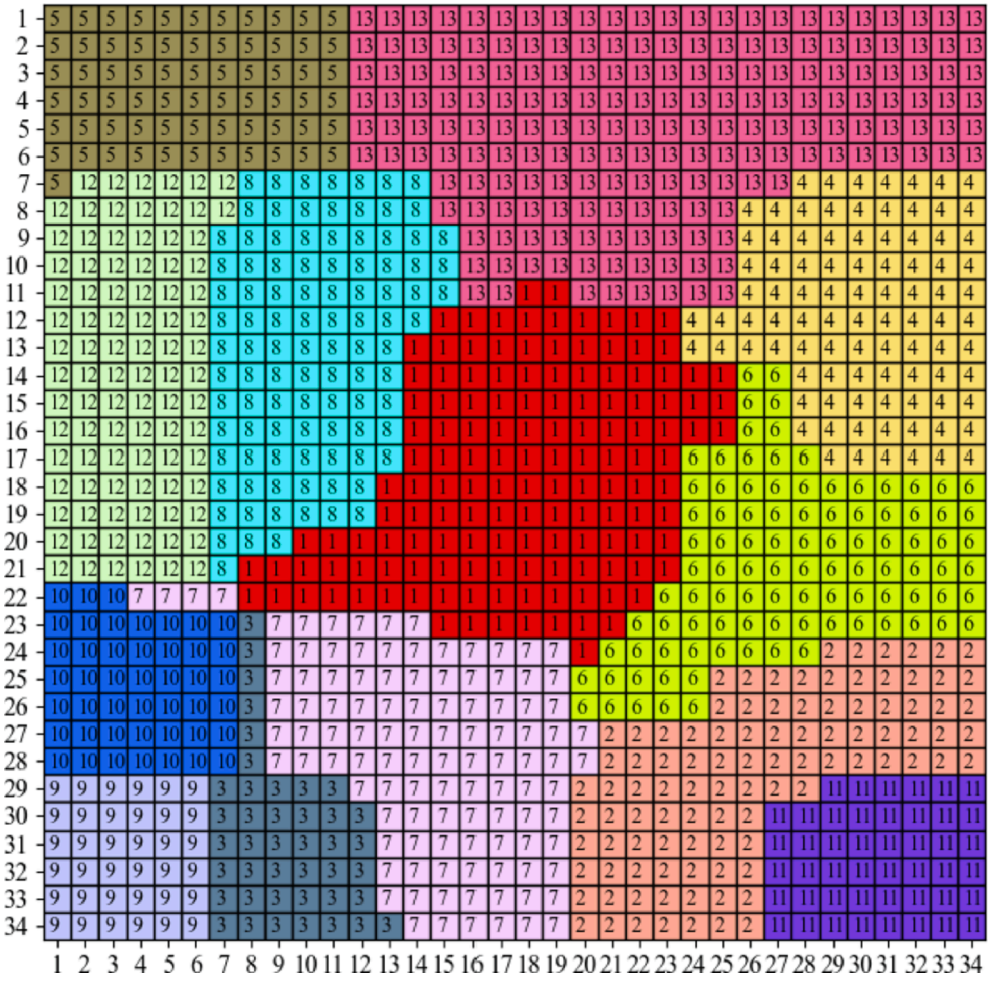

Fig. 9. The clustering result of all neurons in the competition layer.

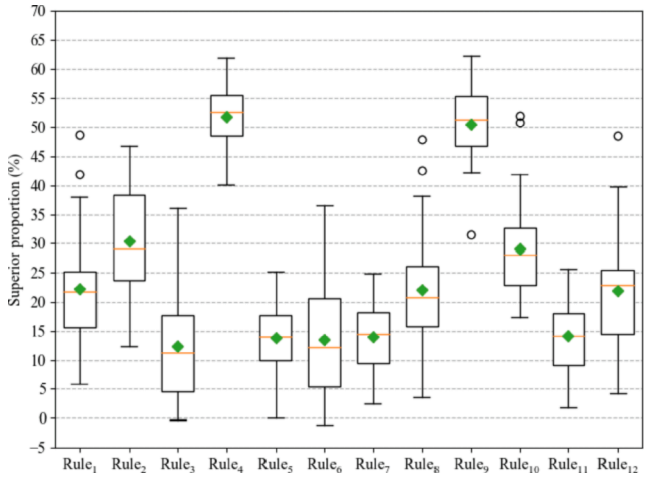

Fig. 10. Box plots of superior proportion of proposed method to $Rule_i$.

After training the SOM neural network, 34*34 trained weight vectors are obtained. This paper uses the k-means algorithm to cluster these weight vectors. The unsupervised clustering method has no direct clustering evaluation method. Meanwhile, the number of clusters (K) corresponding to the best clustering performance is difficult to determine in the k-means algorithm. Here a silhouette coefficient, namely Calinski-Harabasz (CH) index, is adopted in this paper. The CH index evaluates the effect of clustering from the density within clusters and the dispersion between clusters. The larger the CH index, the closer the points in one cluster and the more dispersed different clusters are, that is, the better clustering results. Therefore, this paper uses the k-means clustering algorithm to cluster under different K values, and the corresponding CH indexes are shown in Table 5.

It can be seen from Table 5 that the CH index obtains the maximum value when $K=13$. Hence the k-means clustering algorithm divides the 34*34 wt vectors (34*34 neurons in the competition layer of SOM neural network) into 13 categories. The final clustering result of these neurons is given in Fig. 9.

After training the state clustering module, the GP rule selector needs

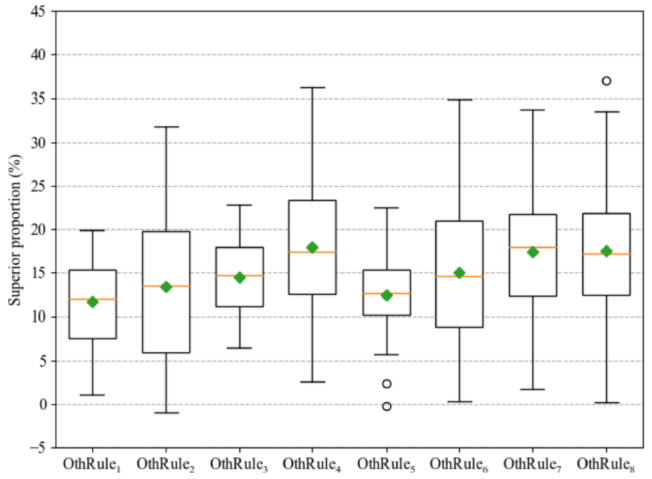

Fig. 11. Box plots of superior proportion of proposed method to OthRule_i.

to be trained next. After the massive iterations, two trained Q-tables are obtained. Add the Q-values in the two tables to obtain the final Q-table. In final Q-table, 13 rows represent 13 production states (13 categories for the competition layer), and 16 columns represent the Q-values obtained by executing the corresponding GP rules in a certain state. The greater the Q-value of GP rule, the greater the reward of agent will be, that is, the better the scheduling performance of GP rule. Therefore, extract the GP rule with the maximum Q-value in each production state, and the results are as follows: state1-GPRule_{14}, state2-GPRule_3, state3-GPRule_{16}, state4-GPRule_5, state5-GPRule_{15}, state6-GPRule_2, state7-GPRule_{15}, state8-GPRule_5, state9-GPRule_4, state10-GPRule_{15}, state11-GPRule_{15}, state12-GPRule_{16}, state13-GPRule_9.

4.4. Comparison with composite dispatching rules

After obtaining the trained decision-making agent, we compare the proposed method with composite dispatching rules on some instances. Firstly, the comparison methods are twelve composite dispatching rules Rule_i ($i = 1, \ldots, 12$). These methods are tested on 30 scheduling problems to get the corresponding objectives. In order to do the performance comparison, here we define the superiority proportion of method $\times$ to method y as $(y\text{-}x)/y$. Therefore, the box plots of superior proportion of proposed method to composite dispatching rules under 30 problems in terms of objective f are provided in Fig. 10.

In Fig. 10, each green diamond point and yellow line represent the mean and median of 30 corresponding superior proportions. It can be found that there exist few negative superior proportions in terms of Rule_3 and Rule_6, which are close to 0. In other words, the proposed method achieves the bigger objectives than that of Rule_3 or Rule_6 in only few problems, and the range of exceeding is very small. Meanwhile, the means of 30 superior proportions corresponding to each composite dispatching rule are as follows: 22.20%, 30.33%, 12.43%, 51.90%, 13.89%, 13.55%, 13.94%, 22.14%, 50.56%, 29.16%, 14.22%, 21.90%. Therefore, it can be concluded that the proposed method can achieve smaller objective and better scheduling performance compared with the twelve composite dispatching rules. Moreover, the new job insertions exist in some instances and the proposed method still obtains small objectives. It shows that the proposed method can select the appropriate GP rule based on the production state and effectively deal with the disturbance events when the new job insertions occur.

In order to further demonstrate the superiority of smart factory, we compare it with some other composite dispatching rules OthRule_i ($i = 1, \ldots, 8$). These rules are as follows: (1) $\text{NCR}_\text{J} + \text{SPT}_\text{TE} + \text{SCT}_\text{M}$; (2) $\text{XCR}_\text{J} + \text{SPT}_\text{TE} + \text{SPT}_\text{M}$; (3) $\text{LTP}_\text{J} + \text{SPT}_\text{TE} + \text{NBT}_\text{M}$; (4) $\text{STP}_\text{J} + \text{SPT}_\text{TE} + \text{NEC}_\text{M}$; (5) $\text{NOJ}_\text{J} + \text{SPT}_\text{TE} + \text{SCT}_\text{M}$; (6) $\text{XOJ}_\text{J} + \text{SPT}_\text{TE} + \text{SPT}_\text{M}$; (7) $\text{LRT}_\text{J} + \text{SPT}_\text{TE} + \text{NEC}_\text{M}$; (8) $\text{LPT}_\text{J} + \text{SPT}_\text{TE} + \text{NEC}_\text{M}$. The dispatching rule NCR_J or XCR_J

Table 6
Objectives of proposed method and GP rules.

Instance	Method	GP1	GP2	GP3	GP4	GP5	GP6	GP7	GP8	GP9	GP10	GP11	GP12	GP13	GP14	GP15	GP16
1	701.68	753.22	752.01	782.72	739.4	780.01	792.63	757.83	768.42	747.63	772.2	700.56	781.51	784.94	762.5	808.8	782.54
2	552.11	609.4	706.98	685.0	569.67	581.77	585.23	593.12	600.66	581.27	620.96	648.04	639.16	625.74	617.59	622.45	597.62
3	541.62	556.69	659.85	542.88	552.47	582.42	576.24	590.96	584.99	561.24	593.69	572.53	600.1	550.47	613.47	608.07	593.52
4	742.76	770.92	749.2	751.52	783.08	813.57	797.68	752.28	778.24	772.69	943.75	857.49	915.43	744.62	734.55	754.91	741.12
5	590.16	593.62	683.39	595.66	582.75	592.78	608.06	638.61	584.32	641.37	586.17	635.47	625.35	586.61	582.6	603.12	600.13
6	766.66	765.82	867.02	835.45	756.22	786.6	781.84	795.18	806.68	859.25	934.13	860.4	842.11	846.18	875.75	794.94	853.83
7	526.49	618.13	561.72	561.44	584.42	628.14	604.57	622.28	579.22	594.11	558.14	582.86	542.34	603.03	631.9	582.42	583.53
8	645.93	768.15	984.13	839.48	704.22	728.17	745.44	711.29	739.44	705.98	931.21	984.55	908.78	698.97	735.37	777.03	681.61
9	661.36	755.22	778.1	763.37	729.72	758.71	738.64	729.76	742.89	704.82	770.11	740.29	826.06	717.22	708.95	672.53	666.47
10	607.5	635.14	644.87	610.47	631.54	651.65	627.96	643.67	634.37	613.86	725.4	649.39	674.5	669.8	679.02	841.25	683.45
11	602.42	641.14	615.34	614.88	615.71	648.33	645.78	637.79	642.97	668.76	618.45	651.04	658.46	601.66	659.97	627.94	593.39
12	647.25	663.57	783.63	671.76	659.71	708.0	711.78	725.93	701.22	726.16	801.55	694.79	792.59	763.25	739.08	761.72	740.87
13	604.0	641.7	722.35	647.47	624.73	660.74	678.52	636.99	659.14	696.84	721.91	708.12	705.52	684.98	699.07	660.92	687.51
14	717.28	744.3	780.1	737.76	706.46	802.34	772.1	781.61	768.62	795.26	778.56	792.5	826.31	821.75	872.71	810.89	832.11
15	641.5	699.13	836.79	814.76	723.49	681.52	675.03	792.71	667.38	769.61	633.96	818.54	710.45	661.35	639.86	729.81	675.9
16	551.49	571.35	802.18	665.97	544.89	545.49	610.09	631.32	588.26	768.55	656.64	762.27	622.05	621.12	609.42	692.32	594.02
17	613.33	646.12	806.47	602.89	632.83	646.34	674.34	655.37	661.67	640.21	663.44	695.43	655.08	616.27	637.07	617.0	638.08
18	569.82	590.46	611.18	595.02	570.12	619.45	634.21	604.24	619.23	595.69	657.45	615.16	651.12	562.63	589.75	579.56	567.4
19	562.46	561.55	702.7	567.18	561.81	564.5	561.55	621.0	557.55	568.33	654.52	562.03	708.2	566.38	591.42	632.71	582.14
20	512.17	573.07	643.91	669.11	586.84	599.85	579.57	573.12	575.63	649.5	561.15	666.93	581.47	563.93	592.68	676.75	551.63

Note: GPi is GPRule_i, $i = 1, \ldots, 16$.

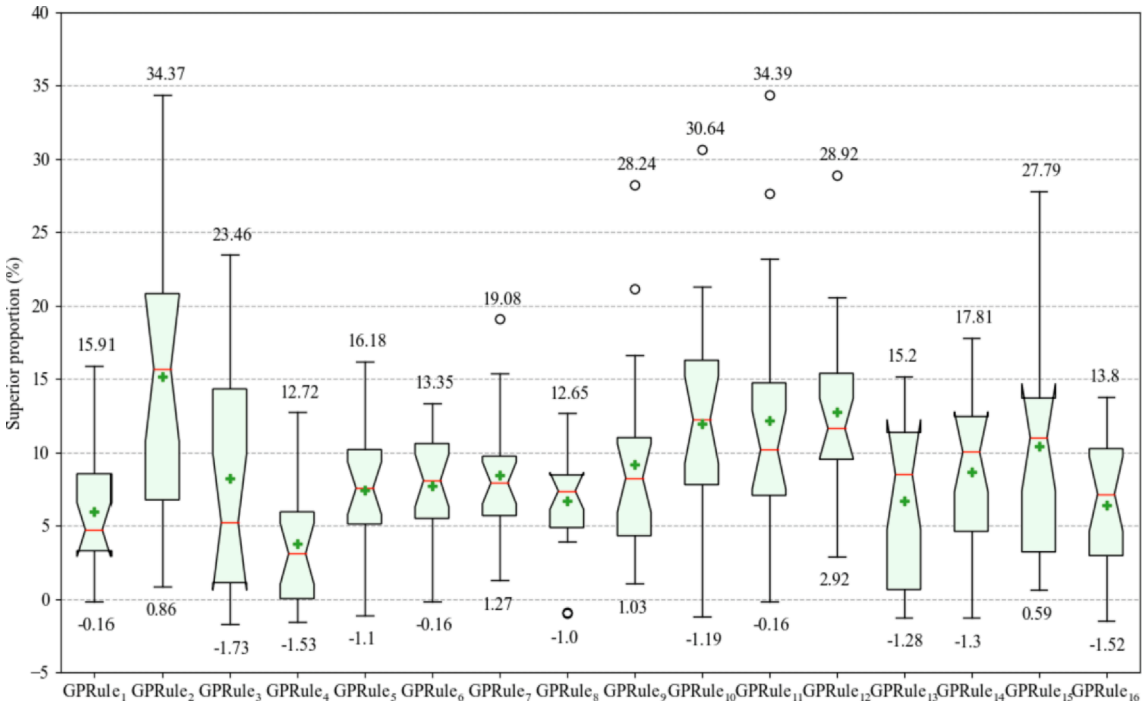

Fig. 12. Box plots of superior proportion of proposed method to $GPRule_i$.

means that the job with minimal or maximal completion rate has the highest priority. The rule LTP_J or STP_J means that the job with longest or shortest total processing time has the highest priority. Here we define ROJ_i as the ratio of the processing time of operation O_{ij} to the total processing time of job J_i. So the rule NOJ_J or XOJ_J means that the job with minimal or maximal ROJ_i has the highest priority. The rule SCT_M means that the machine with shortest completion time is allocated first, and the completion time represents the time for finishing the task that is being executed and all tasks in the buffer. Therefore, these methods are tested on 30 scheduling problems, and the box plots of superior proportion of proposed method to some other composite dispatching rules under 30 problems in terms of objective f are provided in Fig. 11.

In Fig. 11, there exist few negative superior proportions in terms of $OthRule_2$ and $OthRule_5$, and they are in [-1%, 0]. It shows that the objectives obtained by proposed method are close to that of the composite dispatching rule $OthRule_2$ or $OthRule_5$ in few test problems. At the same time, most of obtained superior proportions are much greater than 0, which represents that the proposed method often achieves lower objectives than that of these other composite dispatching rules in test problems. The means of 30 superior proportions corresponding to each $OthRule_i$ are as follows: 11.67%, 13.45%, 14.52%, 17.94%, 12.44%, 15.06%, 17.42%, 17.47%. It can be concluded that the proposed method

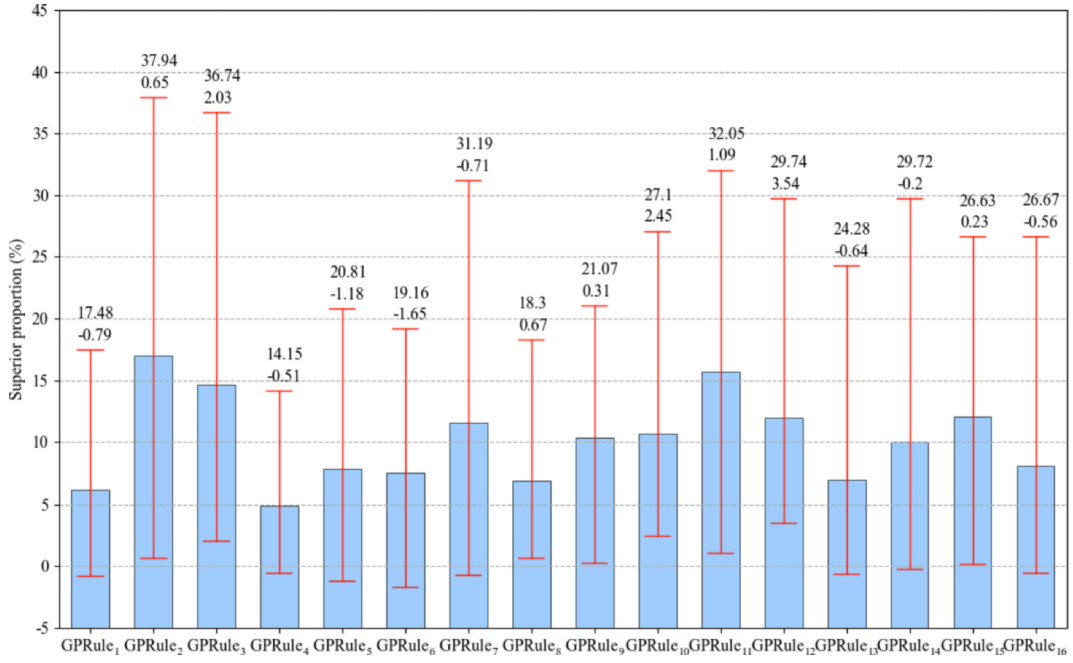

Fig. 13. Result of superior proportion under stochastic time.

has superior scheduling performance compared with the eight other composite dispatching rules through advanced information technologies.

4.5. Comparison with GP method

As a classical real-time scheduling method, GP has been widely studied. Based on the Darwinian natural biological evolution, GP can get a series of high-quality scheduling rules through selection, crossover and mutation. These rules can show good scheduling performance in the face of test problems that are similar to training problems. Therefore, we compare the proposed method with GP to prove the superior scheduling performance of smart factory in this section. The proposed method and sixteen GP rules are tested on 20 instances, and the results are given in Table 6 and Fig. 12.

In Fig. 12, each green plus point and yellow line represent the mean and median of 20 corresponding superior proportions. The two numbers above and below each box plot represent the maximum and minimum value of 20 corresponding superior proportions. In most cases, the superior proportion is positive, and it shows that the proposed method gets a smaller objective that that of GP method. Compared with all GP rules, the numbers of times the proposed method achieves smaller objective are as follows: 18, 20, 19, 15, 19, 19, 20, 18, 20, 18, 18, 20, 17, 17, 20, 17. The means of 20 superior proportions corresponding to each $GPRule_i$ are as follows: 5.97%, 15.18%, 8.19%, 3.79%, 7.43%, 7.70%, 8.42%, 6.71%, 9.20%, 11.94%, 12.14%, 12.78%, 6.66%, 8.69%, 10.40%, 6.43%. Therefore, it can be found that the proposed method can effectively balance the makespan and total energy consumption, and achieves better performance compared with GP method. In other words, through selecting the appropriate GP rule based on the real-time manufacturing data, the smart factory can help enterprises achieve a series of goals based on IoT and big data, such as improving production efficiency and reducing energy consumption.

4.6. Performance clarification under disturbance events

In the actual manufacturing process, various disturbance events are inevitable to occur, which affect the execution of original planning scheme and even cause the workshop to stop production. These events have severely reduced the production efficiency of enterprises, and are one of the key problems to be solved in real-time scheduling methods. This paper considers a classic disturbance event in the previous experimental case: the new order insertion. In addition, in the production workshop, the actual processing time of workpiece will undergo small changes at the predefined time due to various manufacturing activities, such as loading, unloading, AGV delay and tool change. Therefore, this section studies the scheduling performance of proposed method under stochastic time.

Here we assume that the processing time follows the normal distribution$N(PT_{xy}, \ \sigma^2)$, where PT_{xy} is the predefined processing time and σ is the standard deviation. σ can be 1, 2 or 3. Based on this setup, this paper obtains 30 instances to verify the adaptability and robustness of proposed method. In this experiment, we first use the trained Q-tables to continue training and learning on one problem with stochastic time. When the Q-tables finish the evolution, they will be saved and then tested on 30 instances. Meanwhile, all GP rules are also tested on these instances to make comparison. The results of superior proportions of proposed method to GP in 30 instances are provided in Fig. 13.

As shown in Fig. 13, each blue rectangle represents the mean of superior proportions, and the sixteen means are as follows: 6.23%, 16.99%, 14.61%, 4.92%, 7.86%, 7.54%, 11.52%, 6.97%, 10.34%, 10.7%, 15.67%, 11.98%, 6.99%, 9.99%, 12.05% and 8.08%. The two numbers above each rectangle represent the minimum and maximum of superior proportions of proposed method to each GP rule. Hence, the proposed method has better performance than GP rules in the problems with stochastic time. It can be concluded that RL gives the agent ability to learn how to effectively deal with unseen situations through exploration. In summary, the proposed method can continuously learn and adjust its scheduling strategy through training experience, so as to own the good performance under sudden changes in the workshop.

5. Conclusion and future work

In order to realize the data-driven manufacturing, this paper proposes the cyber-physical architecture for smart factory, and uses CNP to design the MAS-based dynamic scheduling mechanism, which includes the five key modules: problem formulation module, scheduling point module, scheduling rule library, state clustering module and GP rule selector. Comprehensive experiments are conducted, and the results show that the decision-making agent can select the appropriate GP rule according to the PAV at each scheduling point.

The contributions of this paper are as follows: (1) A cyber-physical architecture is proposed for smart factory, and a MAS-based dynamic scheduling mechanism is given using CNP. (2) The dynamic scheduling mechanism includes the five key modules. Firstly, the problem formulation module is designed to give a MILP model. Secondly, the scheduling point module is designed to give the specific definition of scheduling point. Thirdly, the scheduling rule library is designed using GP method and provides sixteen high-quality rules for selection. Fourthly, the state clustering module is designed to realize the efficient clustering of PAV based on the autoencoder, SOM neural network and k-means clustering algorithm. Fifthly, using an improved Q-learning algorithm, the GP rule selector is designed to obtain the efficient state-rule mapping relationship. (3) Experimental results show that the proposed method has feasibility and superiority compared with other methods in real-time scheduling, and can effectively deal with disturbance events in the manufacturing process.

Future research will focus on the application of other AI method in the smart factory. Meanwhile, it is worth exploring how to design better state features and actions for a production system. Moreover, more kinds of disturbance events and optimization objectives are worth being considered. The effective combination of the proposed method with different types of workshops is also a valuable research direction.

CRediT authorship contribution statement

Wenbin Gu: Methodology, Conceptualization, Supervision. **Yuxin Li:** Writing – original draft, Methodology, Software. **Dunbing Tang:** Conceptualization, Investigation. **Xianliang Wang:** Visualization, Software. **Minghai Yuan:** Visualization, Investigation.

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.

Acknowledgements

This work was supported by the National Science Foundation of China (No. 51875171), the General program of Natural Science Foundation of Jiangsu Province (BK20201162) and the Postgraduate Research & Practice Innovation Program of Jiangsu Province (KYCX21_0465).

References

Dai, M., Tang, D., Giret, A., & Salido, M. A. (2019). Multi-objective optimization for energy-efficient flexible job shop scheduling problem with transportation constraints. *Robotics and Computer-Integrated Manufacturing*, *59*(October 2018), 143–157. 10.1016/j.rcim.2019.04.006.

Bueno, A., Godinho Filho, M., & Frank, A. G. (2020). Smart production planning and control in the Industry 4.0 context: A systematic literature review. *Computers and*

Industrial Engineering, 149(August), Article 106774. https://doi.org/10.1016/j.cie.2020.106774
Yin, S., & Kaynak, O. (2015). Big Data for Modern Industry: Challenges and Trends. *Proceedings of the IEEE, 103*(2), 143–146. https://doi.org/10.1109/JPROC.2015.2388958
Cui, Y., Kara, S., & Chan, K. C. (2020). Manufacturing big data ecosystem: A systematic literature review. *Robotics and Computer-Integrated Manufacturing, 62*(January 2019), 101861. 10.1016/j.rcim.2019.101861.
Olsen, T. L., & Tomlin, B. (2020). Industry 4.0: Opportunities and challenges for operations management. *Manufacturing and Service Operations Management, 22*(1), 113–122. https://doi.org/10.1287/msom.2019.0796
Nakayama, R. S., de Mesquita Spínola, M., & Silva, J. R. (2020). Towards I4.0: A comprehensive analysis of evolution from I3.0. *Computers and Industrial Engineering, 144*(February 2019), 106453. 10.1016/j.cie.2020.106453.
Li, J., Tao, F., Cheng, Y., & Zhao, L. (2015). Big Data in product lifecycle management. *International Journal of Advanced Manufacturing Technology, 81*(1–4), 667–684. https://doi.org/10.1007/s00170-015-7151-x
Usuga Cadavid, J. P., Lamouri, S., Grabot, B., Pellerin, R., & Fortin, A. (2020). Machine learning applied in production planning and control: A state-of-the-art in the era of industry 4.0. *Journal of Intelligent Manufacturing, 31*(6), 1531–1558. https://doi.org/10.1007/s10845-019-01531-7
Majeed, A., Zhang, Y., Ren, S., Lv, J., Peng, T., Waqar, S., & Yin, E. (2021). A big data-driven framework for sustainable and smart additive manufacturing. *Robotics and Computer-Integrated Manufacturing, 67*(June 2020), 102026. 10.1016/j.rcim.2020.102026.
Tao, F., Qi, Q., Liu, A., & Kusiak, A. (2018). Data-driven smart manufacturing. *Journal of Manufacturing Systems, 48*, 157–169. https://doi.org/10.1016/j.jmsy.2018.01.006
Ku, C. C., Chien, C. F., & Ma, K. T. (2020). Digital transformation to empower smart production for Industry 3.5 and an empirical study for textile dyeing. *Computers and Industrial Engineering, 142*(January), Article 106297. https://doi.org/10.1016/j.cie.2020.106297
Mörth, O., Emmanouilidis, C., Hafner, N., & Schadler, M. (2020). Cyber-physical systems for performance monitoring in production intralogistics. *Computers and Industrial Engineering, 142*(February), Article 106333. https://doi.org/10.1016/j.cie.2020.106333
Tao, F., & Qi, Q. (2019). New IT driven service-oriented smart manufacturing: Framework and characteristics. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 49*(1), 81–91. https://doi.org/10.1109/TSMC.2017.2723764
Qu, Y. J., Ming, X. G., Liu, Z. W., Zhang, X. Y., & Hou, Z. T. (2019). Smart manufacturing systems: State of the art and future trends. *International Journal of Advanced Manufacturing Technology, 103*(9–12), 3751–3768. https://doi.org/10.1007/s00170-019-03754-7
Subramaniyan, M., Skoogh, A., Muhammad, A. S., Bokrantz, J., Johansson, B., & Roser, C. (2020). A data-driven approach to diagnosing throughput bottlenecks from a maintenance perspective. *Computers and Industrial Engineering, 150*(September). https://doi.org/10.1016/j.cie.2020.106851
Zhang, G., Chen, C. H., Zheng, P., & Zhong, R. Y. (2020). An integrated framework for active discovery and optimal allocation of smart manufacturing services. *Journal of Cleaner Production, 273*, Article 123144. https://doi.org/10.1016/j.jclepro.2020.123144
Li, Y., Yang, X., & Yang, Z. (2019). Uncertain learning curve and its application in scheduling. *Computers and Industrial Engineering, 131*(January 2018), 534–541. 10.1016/j.cie.2018.11.055.
Cheng, L., Tang, Q., Zhang, Z., & Wu, S. (2020). Data mining for fast and accurate makespan estimation in machining workshops. *Journal of Intelligent Manufacturing, 32*(2), 483–500. https://doi.org/10.1007/s10845-020-01585-y
Wu, J., Ding, Y., & Shi, L. (2021). Mathematical modeling and heuristic approaches for a multi-stage car sequencing problem. Computers and Industrial Engineering, 152 (August 2020), 107008. 10.1016/j.cie.2020.107008.
Huang, Y. Y., Pan, Q. K., Huang, J. P., Suganthan, P. N., & Gao, L. (2021). An improved iterated greedy algorithm for the distributed assembly permutation flowshop scheduling problem. Computers and Industrial Engineering, 152(December 2020), 107021. 10.1016/j.cie.2020.107021.
Zhu, Z., & Zhou, X. (2020). An efficient evolutionary grey wolf optimizer for multi-objective flexible job shop scheduling problem with hierarchical job precedence constraints. *Computers and Industrial Engineering, 140*(January), Article 106280. https://doi.org/10.1016/j.cie.2020.106280
Yuan, M., Li, Y., Zhang, L., & Pei, F. (2021). Research on intelligent workshop resource scheduling method based on improved NSGA-II algorithm. *Robotics and Computer-Integrated Manufacturing, 71*(February), Article 102141. https://doi.org/10.1016/j.rcim.2021.102141
Chen, S., Pan, Q. K., & Gao, L. (2021). Production scheduling for blocking flowshop in distributed environment using effective heuristics and iterated greedy algorithm. Robotics and Computer-Integrated Manufacturing, 71(September 2020), 102155. 10.1016/j.rcim.2021.102155.
Wang, J., Zhang, Y., Liu, Y., & Wu, N. (2019). Multiagent and bargaining-game-based real-time scheduling for internet of things-enabled flexible job shop. *IEEE Internet of Things Journal, 6*(2), 2518–2531. https://doi.org/10.1109/JIOT.2018.2871346
Wang, Z., & Gombolay, M. (2020). Learning Scheduling Policies for Multi-Robot Coordination With Graph Attention Networks. *IEEE Robotics and Automation Letters, 5*(3), 4509–4516. https://doi.org/10.1109/lra.2020.3002198
Hu, H., Jia, X., He, Q., Fu, S., & Liu, K. (2020). Deep reinforcement learning based AGVs real-time scheduling with mixed rule for flexible shop floor in industry 4.0. *Computers and Industrial Engineering, 149*(August), Article 106749. https://doi.org/10.1016/j.cie.2020.106749
Caldeira, R. H., Gnanavelbabu, A., & Vaidyanathan, T. (2020). An effective backtracking search algorithm for multi-objective flexible job shop scheduling considering new job arrivals and energy consumption. *Computers and Industrial Engineering, 149* (February), Article 106863. https://doi.org/10.1016/j.cie.2020.106863
Kong, M., Xu, J., Zhang, T., Lu, S., Fang, C., & Mladenovic, N. (2021). Energy-efficient rescheduling with time-of-use energy cost: Application of variable neighborhood search algorithm. *Computers and Industrial Engineering, 156*(April), Article 107286. https://doi.org/10.1016/j.cie.2021.107286
Li, Y., Zheng, J. Q., & Yang, S. L. (2010). Multi-agent-based fuzzy scheduling for shop floor. *International Journal of Advanced Manufacturing Technology, 49*(5–8), 689–695. https://doi.org/10.1007/s00170-009-2410-3
Sinclair, N., Harle, D., Glover, I. A., Irvine, J., & Atkinson, R. C. (2013). An advanced SOM algorithm applied to handover management within lte. *IEEE Transactions on Vehicular Technology, 62*(5), 1883–1894. https://doi.org/10.1109/TVT.2013.2251922
Park, I. B., Huh, J., Kim, J., & Park, J. (2020). A Reinforcement Learning Approach to Robust Scheduling of Semiconductor Manufacturing Facilities. *IEEE Transactions on Automation Science and Engineering, 17*(3), 1420–1431. https://doi.org/10.1109/TASE.2019.2956762
Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. *Machine Learning, 8*, 279–292.
van Hasselt, H. (2010). Double Q-learning. *Advances in neural information processing systems, 23*, 2613–2621.
He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: surpassing human-level performance on imagenet classification, *CVPR IEEE Computer Society*.