Distributed Real-Time Scheduling in Cloud Manufacturing by Deep Reinforcement Learning

Lixiang Zhang, Chen Yang, *Member, IEEE*, Yan Yan, and Yaoguang Hu

***Abstract*—With the extensive application of automated guided vehicles, real-time production scheduling considering logistics services in cloud manufacturing (CM) becomes an urgent problem. Thus, this study focuses on the distributed real-time scheduling (DRTS) of multiple services to respond to dynamic and customized orders. First, a DRTS framework with cloud–edge collaboration is proposed to improve performance and satisfy responsiveness, where distributed actors and one centralized learner are deployed in the edge and cloud layer, respectively. And, the DRTS problem is modeled as a semi-Markov decision process, where the processing services sequencing and logistics services assignment are considered simultaneously. Then, we developed a distributed dueling deep Q network (D3QN) with cloud–edge collaboration to optimize the weighted tardiness of jobs. The experimental results show that the proposed D3QN obtains lower weighted tardiness and shorter flow-time than other state-of-the-art algorithms. It indicates the proposed DRTS method has significant potential to provide efficient real-time decision-making in CM.**

***Index Terms*—Cloud–edge collaboration, cloud manufacturing, deep reinforcement learning, distributed, real-time scheduling.**

I. Introduction

With considerably increasing attention, cloud manufacturing (CM) has been proposed to integrate and collaborate manufacturing resources among enterprises to tackle challenges, such as customized orders and dynamic changes in manufacturing systems [1], [2]. It integrates the advantages of enabling technologies to respond to customized production, such as Internet of Things, cloud computing, and artificial intelligence. However, due to dynamics and uncertainties in CM, services scheduling has not been fully studied yet. Scheduling rules were usually utilized to satisfy the responsiveness to dynamic changes [3], [4]. Owing to the poor performance of the single scheduling rule, researchers show great interest in reinforcement learning (RL) intending to explore better scheduling policies which could make decisions according to real-time information [5]. It obtains better performance than scheduling rules. Based on the representation capacity of the deep neural network, more researchers have focused on deep reinforcement learning (DRL) and achieved excellent performance in solving the scheduling of a single resource, which presented the significant potential to respond to dynamic changes in manufacturing systems [6], [7].

However, two main limitations must be overcome to improve performance and satisfy responsiveness in dynamic environments. First, training processes need stronger computing power although DRL has obtained significant performance. On the shop floor, the computing capacity can hardly satisfy these requirements. Thus, it is difficult to obtain an efficient learning and decision-making model to support production scheduling. On the other hand, the iteration optimization methods just explore optimal or near-optimal solutions for specific problems, which lack the adaptability to tackle dynamic or uncertain events and provide real-time decision-making.

As a result, this study aims to propose a DTRS method with learning capacity in dynamic environments and provide new insights to make full use of special capacities in different layers of CM to respond to customized production. The main contributions of this study are highlighted as follows.

1) Based on distributed DRL with cloud–edge collaboration, a DRTS framework is proposed to realize centralized learning and distributed decision-making to improve performance and satisfy responsiveness simultaneously.
2) The semi-Markov decision process based on global information sharing in the edge layer is constructed to deal with the coupling interaction of multiple services.
3) A D3QN algorithm is developed to improve the learning capacity of the centralized learner in the cloud layer and support distributed actors corresponding to machines on the shop floor to make real-time decisions efficiently.

The rest of this article is organized as follows. Section II reviews relevant articles and provides the improvement direction of this study. Section III mainly describes the DRTS framework and the semi-Markov decision process. Section IV designs the architecture and process of the D3QN algorithm. Section V

Manuscript received 28 December 2021; revised 30 April 2022; accepted 22 May 2022. Date of publication 27 May 2022; date of current version 30 September 2022. This work was supported by the National Key R&D Program of China under Grant 2021YFB1715700 and in part by the National Natural Science Foundation of China under Grant 52175451. Paper no. TII-21-5771. *(Corresponding author: Yaoguang Hu.)*

Lixiang Zhang, Yan Yan, and Yaoguang Hu are with the Laboratory of Industrial and Intelligent Systems, School of Mechanical Engineering, Beijing Institute of Technology, Beijing 100081, China (e-mail: zhanglx@bit.edu.cn; yanyan331@bit.edu.cn; hyg@bit.edu.cn).

Chen Yang is with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100081, China (e-mail: yangchen666@bit.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TII.2022.3178410.

Digital Object Identifier 10.1109/TII.2022.3178410

1551-3203 © 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on February 19,2025 at 06:26:12 UTC from IEEE Xplore. Restrictions apply.

verifies the performance and discusses the main advantages of the proposed method. Finally, Section VI concludes this article.

II. Literature Review

A. Cloud-Based Manufacturing

The first study of CM can date back to the 2010s [1], which explored the potential of cloud computing in manufacturing and provided new perspectives to take advantage of cloud computing and manufacturing as services to enhance productivity. A considerable amount of studies have been conducted to realize real-time monitoring, online smart diagnosis, and process planning and scheduling. For example, Helo *et al.* [8] utilized centralized cloud computing to support production scheduling in distributed and dynamic environments. Mourtzis *et al.* [9] proposed a cloud-based, knowledge-enriched framework, which enabled ubiquitous data access and use-as-service to realize real-time monitoring. Caggiano [10] developed the online smart diagnosis, where knowledge-based algorithms and cognitive pattern recognition were treated as a cloud service. Moreover, cloud computing and edge computing were integrated to improve performance with low latency [11].

To improve the utilization of resources in cloud platforms and the satisfaction of users, DRL methods were explored to solve complex problems in different manufacturing stages. Cheng *et al.* [12] utilized DRL to realize resource provisioning and task scheduling for cloud service providers. As far as quality of service (QoS) was considered, Huang *et al.* [13] combined imitation learning and DRL to map real-time cloud jobs to resources without prior knowledge. A hierarchical and hybrid online DRL was used to provision cloud resources in clusters and servers as well as schedule tasks into appropriate times to satisfy QoS for warehouse-scale data centers [14]. The service composition with logistics involved was optimized by a dueling DRL to improve QoS in CM [15]. These methods treat CM as a service to support distributed production.

B. Production Scheduling

The scheduling problem on the shop floor is an important issue to meet customized production, which has been fully explored. Previous research had investigated the integrated scheduling of multiple services in static environments [16]–[18]. In contrast, due to the complexity of dynamic environments, recent studies just focused on exploring subproblems, respectively. For the aspect of scheduling rules, Fang *et al.* [19] proposed scheduling resource parameter updating methods and dynamic interactive scheduling strategies to achieve real-time job-shop scheduling by utilizing the real-time mapping characteristic of digital twins. Zhou *et al.* [20] constructed the mathematical model of dynamic scheduling problems in CM and selected better scheduling rules according to real-time information in simulation models. Moreover, Zhang *et al.* [21] proposed a multitask generative hyperheuristic approach based on gene programming to improve the dynamic scheduling performance.

In addition, few studies based on RL were proposed to explore scheduling problems. Wang and Usher [22] presented a Q-learning method to learn how to select dispatching rules by interacting with its environment. Additionally, Shiue *et al.* [23] presented a multiattribute scheduling rule based on RL, which combined offline learning with Q learning to achieve real-time decision-making. The results outperformed single-attribute and heuristic scheduling rules. Because RL methods may be inefficient with the problem scale increases [24], Hu *et al.* [25] designed an adaptive deep Q network (DQN) for real-time automated guided vehicle (AGV) scheduling, which selected the suitable scheduling rules and AGVs simultaneously. Moreover, Tang *et al.* [26] presented a hierarchical actor-critic method to effectively solve the order-picking problem with the collaboration of multiple logistics robots.

In summary, DRL has been applied to resource allocation or production scheduling problems in CM. However, the scheduling of multiple services with coupling interaction in CM has not been fully resolved, because there are uncertain and dynamic events that occurred on the shop floor [27]. In this article, DRL, as a powerful method for solving real-time decision-making problems according to environmental information, is developed to learn the implicit interaction relationship between processing services and logistics services for making short-term decisions in dynamic environments while maximizing long-term rewards to satisfy responsiveness and improve performance simultaneously.

III. Methods

A. Problem Description

The real-time scheduling in this article considers processing services sequencing (PSS) and logistics services assignment (LSA) simultaneously. Some common assumptions about this are described as follows.

1) Each machine or AGV performs only one processing service or logistics service at a time.
2) The processing service order and the processing service time of jobs are not available until the jobs arrive.
3) The operating processes of processing and logistics services are all continuous and must not be disrupted.
4) The logistics service time of AGVs with a certain speed is fixed between two machines.

Three main characteristics of real-time scheduling problems are considered as follows.

1) *Dynamic job arrival*. Jobs arrive at the manufacturing system dynamically as the system runs. As shown in Fig. 1, n jobs are released to the manufacturing system at different times. As the system runs, several jobs are completed after all processing and logistics services are completed. Thus, this study aims at minimizing the tardiness of jobs to respond to the dynamic arrival of jobs, not the makespan.
2) *Multiservice coupling interaction*. To optimize the tardiness, processing and logistics services must be considered simultaneously at each decision point. We need to decide which processing service should be selected to be first processed and which AGV is assigned to transport this job to the machine of the next processing service. That is because the next processing service could not be started

Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on February 19,2025 at 06:26:12 UTC from IEEE Xplore. Restrictions apply.

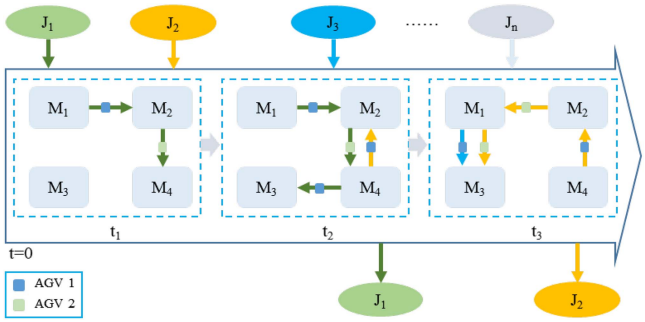

Fig. 1. Real-time scheduling problem of processing services sequencing and logistics services assignment.

until the current processing and logistics services are all completed.

3) *Real-time decision-making*. The state of the manufacturing system changes dynamically. The iteration optimization method takes a long time to explore optimal or near-optimal solutions, generating waiting time as well as reducing the utilization of machines and AGVs. Thus, the decision must be made with high responsiveness while maintaining production performance.

Therefore, considering dynamic job arrival and multiservice coupling interaction, this article mainly focuses on exploring an efficient method to provide real-time decision-making for PSS and LSA in CM.

B. Distributed Real-Time Scheduling Framework

To explore the real-time scheduling of processing services and logistics services in CM, a novel distributed real-time scheduling framework with cloud–edge collaboration is proposed to learn a better policy from historical experience data and provide decision-making according to the real-time information, as shown in Fig. 2, which consists of a cloud layer and an edge layer. In the cloud layer, the cloud platform integrates virtual manufacturing resources, collects customized needs, decomposes entire needs, and allocates orders as well as resources to enterprises. In addition, the cloud platform integrates computing resources to learn a better policy from historical experience collected from distributed actors in the edge machines. In the edge layer, distributed physical equipment and controllers are connected with the Internet of Things based on wireless sensor networks (WSN), which collect data from sensors installed on machines and AGVs, such as the tasks' data obtained by the radio frequency identification reader (RFID), the AGVs' position obtained by the radar and the inertial measurement unit, and the machines' data obtained by the programmable logic controller (PLC), and so on. Based on the sensors and WSN, the distributed actors download the learned decision-making policy to support real-time decision-making in the edge layer and store experiences into the replay buffer in the cloud layer.

As for the distributed real-time scheduling of processing services and logistics services with cloud–edge collaboration in CM, there is only one centralized learner in the cloud layer to learn the decision-making policy from historical experiences.

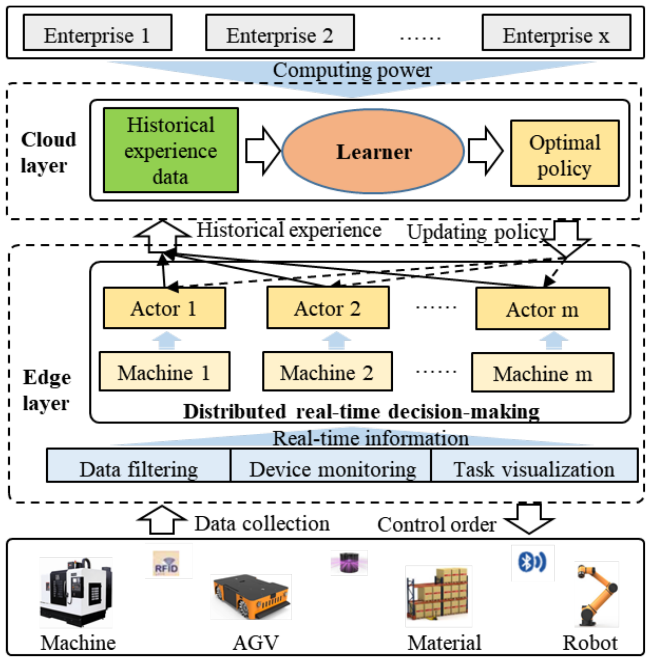

Fig. 2. Distributed real-time scheduling framework.

Each machine has a decision-maker, Actor. As a result, the number of distributed actors is the same as the number of machines on the shop floor. Based on the DRTS framework with cloud–edge collaboration, the learning capacity and low latency of decision-making can be fully satisfied, respectively, by the sufficient computing power in the cloud layer and distributed actors in the edge layer, which provides new insight to explore efficient methods that need strong computing power to solve complex scheduling problems with low latency in CM.

C. Semi-Markov Decision Process

The real-time scheduling processes of processing services and logistics services can be modeled as a semi-Markov decision process to represent the decision-making and the state transition in CM, which supports optimizing production performance. In this article, decision points occur when a machine is idle and the machine's buffer is not empty. The semi-Markov decision process of the distributed real-time scheduling of processing services and logistics services can be defined as a six-element tuple $(\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{P}, \gamma, \boldsymbol{R}, \pi)$. $\boldsymbol{S}$ represents the real-time state of the dynamic manufacturing system. $\boldsymbol{A}$ represents the combination of actions, including the PSS rule and the LSA rule. $\boldsymbol{P}$ represents the transition probability from state s to state s'. γ represents the discount factor that balances short-term and long-term rewards. $\boldsymbol{R}$ represents the reward function. $\pi(\boldsymbol{s}, \boldsymbol{a})$ means the probability of taking action a in state s. Detailed descriptions are as follows.

1) Notations: The notations are defined as follows.

t The current time.
J Set of jobs that have arrived, $J = \{J_i | J_1, J_2, \ldots, J_n\}$.
O Set of operations, $O_i = \{O_{ij} | O_{i1}, O_{i2}, \ldots, O_{im}\}$.
M Set of machines, $M = \{M_k | M_0, M_1, M_2, \ldots, M_m\}$, where M_0 represents the warehouse.

Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on February 19,2025 at 06:26:12 UTC from IEEE Xplore. Restrictions apply.

M_{ij} The machine that the operation O_{ij} will be processed.
V Set of available AGVs, $V = \{V_h | V_1, V_2, \ldots, V_v\}$.
L_{tk} Set of operations waiting for processing in M_k at time t.
pt_{ij} The processing service time of operation O_{ij}.
r_i The release time of job J_i.
rt_{ij} The release time of operation O_{ij}.
jw_i The weight of job J_i.
df The delay factor of jobs.
dt_{ij} The due date of operation O_{ij}.
DT_i The due date of job J_i.
wt_{th} The remaining working time of V_h at time t.
d_{th} The destination point of V_h at time t.
$T_{kk'}$ The traveling time between M_k and $M_{k'}$.
C_{ij} The completion time of operation O_{ij}.
MS_i The completion time of job J_i.
ETA_i The estimated tardiness of J_i.
TD_i The tardiness of job J_i.
$X_{iji'j'}$ If O_{ij} is processed before $O_{i'j'}$, $X_{iji'j'} = 1$, otherwise, $X_{iji'j'} = 0$.
Y_{ijh} If A_h is assigned to perform the logistics service after O_{ij} is completed, $Y_{ijh} = 1$; otherwise, $Y_{ijh} = 0$.

2) State Representation: The state describes necessary information that affects decision-making in dynamic environments. In this article, six types of features are designed as the environmental state corresponding to machines and AGVs. The features of M_k at a decision point can be defined as follows, where the function mean represents the average value of a set of parameters.

Features 1–3: The state of the processing services time in the buffer is defined in (1).

$$\{\max\{pt_{ij}\}, \min\{pt_{ij}\}, \text{mean}\{pt_{ij}\}\}, \forall O_{ij} \in L_{tk}. \quad (1)$$

Features 4–6: The state of the release time of processing services in the buffer is defined in (2).

$$\{\max\{rt_{ij}\}, \min\{rt_{ij}\}, \text{mean}\{rt_{ij}\}\}, \forall O_{ij} \in L_{tk}. \quad (2)$$

Features 7–9: The state of the due date of processing services in the buffer is defined in (3).

$$\{\max\{dt_{ij}\}, \min\{dt_{ij}\}, \text{mean}\{dt_{ij}\}\}, \forall O_{ij} \in L_{tk}. \quad (3)$$

Features 10–12: The state of the job weight of processing services in the buffer is defined in (4).

$$\{\max\{jw_i\}, \min\{jw_i\}, \text{mean}\{jw_i\}\}, \forall O_{ij} \in L_{tk}. \quad (4)$$

Features 13–15: The state of the remaining working time of all AGVs is defined in (5).

$$\{\max\{wt_{th}\}, \min\{wt_{th}\}, \text{mean}\{wt_{th}\}\}, \forall h \in V. \quad (5)$$

Features 16–18: The state of the traveling time of all AGVs between the destination position and M_k is defined in (6).

$$\{\max\{T_{d_{th}k}\}, \min\{T_{d_{th}k}\}, \text{mean}\{T_{d_{th}k}\}\}, \forall h \in V. \quad (6)$$

Thus, the state set of the DRTS of processing services and logistics services can be represented by 18 features. To improve the robustness of state representation, these features are normalized by max–min normalization.

Algorithm 1: Procedure of Action Execution.

If *the PSS rule is FIFO:*
  $O_{ij} \leftarrow \arg\min rt_{ij}, \forall O_{ij} \in L_{tk}$
Else if the PSS rule is SPT:
  $O_{ij} \leftarrow \arg\min pt_{ij}, \forall O_{ij} \in L_{tk}$
  Else if *the PSS rule is MDD:*
    $O_{ij} \leftarrow \arg\min dt_{ij}, \forall O_{ij} \in L_{tk}$
  Else:
    $O_{ij} \leftarrow \arg\max jw_i, \forall O_{ij} \in L_{tk}$
$O_{iji'j'} = 1, \forall O_{i'j'} \in L_{tk}$
If *the LSA rule is MTT:*
  $A_h \leftarrow \arg\min_{A_h \in V} T_{d_{th}M_k}, Y_{ijh} = 1$
Else:
  $A_h \leftarrow \arg\min_{A_h \in V} wt_{th}, Y_{ijh} = 1$

3) Action Representation: For the DRTS problem of processing services and logistics services, processing services and logistics services have a coupling influence on the tardiness of jobs. Thus, actions include PSS and LSA simultaneously. In this article, common rules are defined as actions. In other words, the action set is a rule set. First-in-first-out (FIFO), shortest processing time (SPT), minimum due date (MDD), and maximum job weight are included in the PSS rule set. It means an operation is selected according to a rule. The LSA rule set includes minimum waiting time (MWT) and minimum traveling time (MTT), which is used to assign AGVs. The actions are performed as shown in Algorithm 1.

4) State Transition: Due to the coupling interaction of multiple services on the system operation, it is difficult to model the DRTS problem of processing and logistics services. It means the reward and transition probability are not known, which needs the agent to learn by interacting with a real environment in dynamic CM. As a result, the state transition is described in a simulation environment, where the coupling interaction of processing and logistics services is defined. The real-time processing services information in the machine's buffer and the list of AGVs are obtained in the edge layer and then converted to the real-time state representation as an input. Until a new decision point arrives, the real-time information is obtained as the current input again and also represented as the next state of the previous decision-making. Meanwhile, the main constraints of the DRTS problem of processing and logistics services in the simulation environment are defined as follows:

$$rt_{i1} \geq r_i + T_{M_0M_{i1}} + wt_{th} \ \forall i \in J, \exists h \in V \quad (7)$$

$$C_{ij} \geq rt_{ij} + pt_{ij} \ \forall i \in J \ \forall i, j \in O \quad (8)$$

$$MS_i \geq \max\{C_{ij}\} + T_{M_{im}M_0} \ \forall i \in J \ \forall i, j \in O \quad (9)$$

$$rt_{i(j+1)} \geq C_{ij} + T_{M_{ij}M_{i(j+1)}} + wt_{th} \ \forall i \in J \ \forall i, j \in O, \exists h \in V \quad (10)$$

Constraint (7) means jobs are released at the warehouse and then transported to the machine of the first processing service. Constraint (8) means that all processing services of jobs must be processed. Constraint (9) represents the completion time of

Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on February 19,2025 at 06:26:12 UTC from IEEE Xplore. Restrictions apply.

jobs. Constraint (10) means the next operation is ready to be processed after the last operation is completed and transported into this machine.

5) Reward Function: The DRTS problem in CM aims at responding to customized orders in time. It means the objective function can be defined to minimize the weighted tardiness as shown in (11), where the importance of customers is represented by the job weight.

$$\min f = \frac{1}{n}\sum_{i=1}^{n} jw_i \cdot TD_i = \frac{1}{n}\sum_{i=1}^{n} jw_i \cdot \max\{0, MS_i - DT_i\}. \tag{11}$$

Because the objective value is not obtained until all processing services of jobs are completed, the sparse reward issue occurs, which leads to that the objective value cannot be treated as the reward in the current step. Therefore, the current reward is defined according to the estimated tardiness of jobs as shown in (12). When there is no estimated tardiness for this decision-making, the reward is set as the job weight, which is helpful to satisfy the requirements of important customers first. The estimated tardiness is calculated in (13).

$$R_{tk} = \begin{cases} jw_i, & \text{If } ETA_i = 0 \\ 0, & \text{Otherwise} \end{cases} \tag{12}$$

$$\text{ETA}_i = C_{im} + T_{M_{im}M_0}. \tag{13}$$

For long-term decision-making, the future cumulative reward is defined as shown in (14), where ct represents the current decision point and τ represents the future step, which is helpful to improve the satisfaction of all customers in the long-term decision-making.

$$G_{ct} = R_{ct+1} + \gamma R_{ct+2} + \gamma^2 R_{ct+3} + \cdots + \gamma^{n-ct-1}$$
$$R_n = \sum_{\tau=0}^{n-ct-1} \gamma^{\tau} R_{ct+\tau+1}. \tag{14}$$

6) Policy: The policy is how to select a combined action according to the real-time state in dynamic environments, which can be defined by the action-value function $Q_\pi(s, a)$ in (15).

$$q_\pi(s,a) = E_\pi\left[\sum_{ct=0}^{\infty} \gamma^{ct} R(S_{ct}, A_{ct}) | S_{ct} = s, A_{ct} = a\right]. \tag{15}$$

Thus, the distributed real-time scheduling problem in dynamic CM is transformed into a semi-Markov decision process to explore an optimal policy $\pi*$ as shown in (16) to select the PSS rule and LSA rule simultaneously.

$$q_{\pi^*}(s,a) = E\left[R_{ct} + \gamma q_*(S_{ct+1}) | S_{ct} = s, A_{ct} = a\right]$$
$$= \max_{\pi} q_\pi(s,a). \tag{16}$$

IV. Distributed Dueling Deep Q Network

A. D3QN Framework

DRL aims to utilize the neural network to map complex relations between the state and the action. For the DRTS problem, a distributed dueling deep Q network (D3QN) with cloud–edge collaboration is proposed to learn a better policy by interacting with dynamic environments and then provide real-time decision-making. As shown in Fig. 3, D3QN combines and coordinates the computing power in the cloud and edge layer to satisfy the requirements of dynamic manufacturing systems.

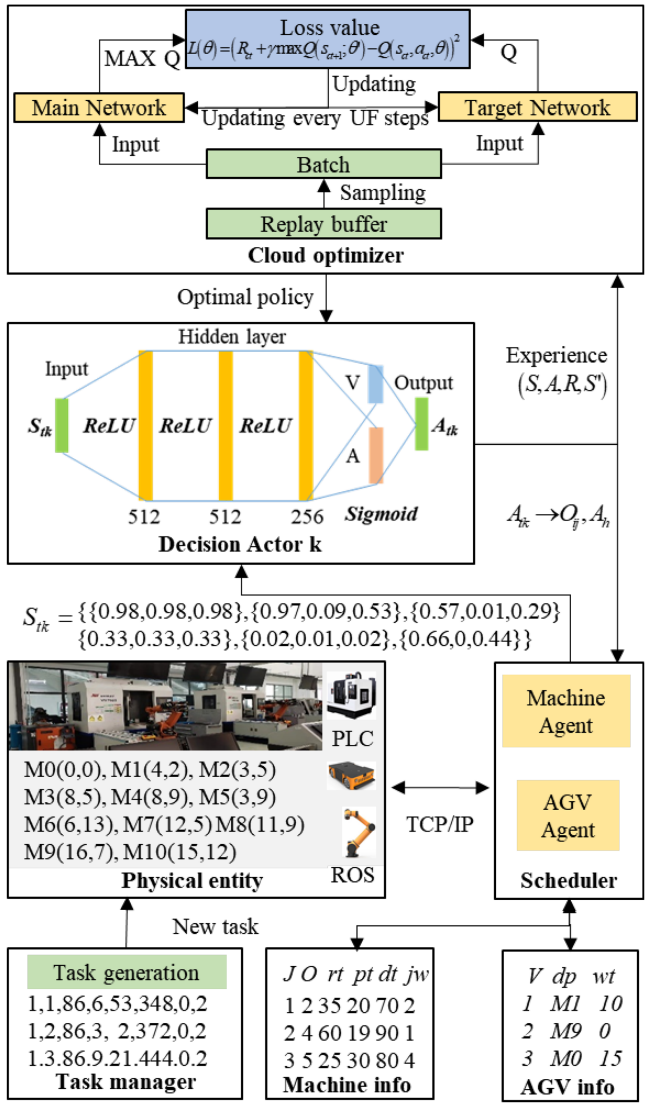

Fig. 3. Distributed dueling deep Q network framework in CM.

In the edge layer, the real-time data of machines, AGVs, and robots is collected from the controller, such as PLC and robot operating system. Then, the data is sent to the machine agent and AGV agent with TCP/IP and transformed into specific information. Machine information includes processing service release time, processing service time, due date, and job weight. AGV information includes the destination position, remaining working time, and relative operation information. The machine agent collects its own information as well as AGV information, sending $\boldsymbol{S}_{tk}$ into the decision actor k and then obtaining $\boldsymbol{A}_{tk}$. According to the action execution in Algorithm 1, the corresponding processing service is selected to be processed and then

Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on February 19,2025 at 06:26:12 UTC from IEEE Xplore. Restrictions apply.

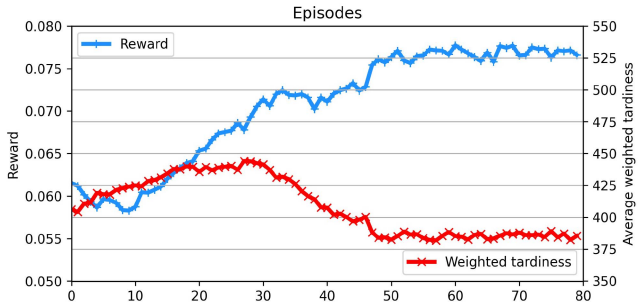

Fig. 4. Convergence of the reward and average weighted tardiness of the proposed D3QN.

transported into the machine of the next processing service by $\boldsymbol{A_h}$. After that, one piece of the decision-making experience is stored in the replay buffer in the cloud optimizer.

For the semi-Markov decision process of the DRTS problem in this article, the value function may be irrelevant with actions. To tackle this challenge, the action-value function ***Q(s, a)*** is represented by the value function ***V(s)*** and the action advantage function ***A(s, a)*** in a dueling network as shown in (17), which is helpful to learn a better value function that is not affected by the action. As a result, the main Q network and target Q network are designed with the same neural networks and parameters, as shown in Fig. 3. The activation functions, ReLU and Sigmoid, are defined in (18), (19).

$$Q(s,a) = V(s) + A(s,a) \tag{17}$$

$$\text{Re}\,Lu(x) = \max(0,x) \tag{18}$$

$$\text{Sigmoid}(x) = \frac{1}{1+e^{-x}}. \tag{19}$$

B. D3QN Training

The training process aims at maximizing the cumulative reward and reducing the loss value between the main Q network and target Q network, which is trained under the simulation environment as shown in Algorithm 2. First, the target Q network and the actors' network are initialized with the same parameters as the main Q target. Also, the simulation environment is set and initialized. Then, the simulation environment is updated continuously. The decision-making is performed by distributed actors. Here, ε-greedy is defined to select actions to balance the exploration and exploitation, as shown in (20), where the ε is decay with the training process, as shown in (21).

$$a = \begin{cases} \text{random} a & \varepsilon \\ \arg\max_a Q(s,a) & 1-\varepsilon \end{cases} \tag{20}$$

$$\varepsilon = \begin{cases} \varepsilon\_\min, & \text{if} \varepsilon \le \varepsilon\_\min \\ \varepsilon \cdot \varepsilon\_decay, & \text{Otherwise} \end{cases}. \tag{21}$$

The processing and logistics services are sent to corresponding machines and AGVs according to Algorithm 1. Meanwhile, the reward is obtained from the simulation environment. Then, the experience is stored in the replay buffer. Finally, some records are sampled from D and sent to these two networks to calculate rewards and the loss function as defined in (22), which is used to update the parameters of the main Q network with the gradient descent algorithm [28].

$$L(\theta) = \left(R_{ct} + \gamma \max Q(s_{ct+1};\theta') - Q(s_{ct},a_{ct},\theta)\right)^2. \tag{22}$$

During training, the target Q network that provides the optimal policy to distributed actors will be updated by the main Q network with a certain frequency. The above steps are repeated until the training steps reach the setting value. The parameters of D3QN in this article are defined in Table I. The batch size (BS) is the number of records sampled from the replay buffer D. The maximum number of records in the replay buffer is defined as replay buffer size (RBS). The training epoch (TE) represents the number of training and backpropagation for each sample. The updating frequency (UF) represents the target Q network that is updated each UF step.

TABLE I
PARAMETERS OF D3QN

Parameters	Description	Value
ε_ini	The initial value of ε	0.9
ε_min	The minimum value of ε	0.01
ε_decay	The delay rate of ε	0.999
BS	The number of samples in each batch size	64
TE	The number of training epochs	5
UF	The updating frequency of target Q network	50
RBS	The maximum records of replay buffer D	1000

Algorithm 2: D3QN Training.

Initializing main Q network, target Q network, and actors as $q(.,.;\theta)$
Initializing the replay buffer D;
For episode in Episodes:
  Initializing the simulation environment $t = 0$ and s_{tk} (k ∈ M)
  While all services are not completed:
    If a new job has arrived:
      Using MTT to assign an AGV to transport the job;
    For m in M:
      If the machine is idle and the buffer is not empty:
        Inquiring the current state s_{tk};
        Select an action a_{tk} as shown in (20);
        Executing the action as shown in Algorithm 1;
        Inquiring the reward and s' after performing a_{tk};
        Storing record (s, a, r, s') into D in the cloud layer;
        Sampling BS records from D;
        Calculating the loss function as shown in (22);
        Updating main Q network parameters θ;
        Updating $\theta' = \theta$ every UF steps;
    Updating the simulation environment;
End
Output the parameters of Q network;

Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on February 19,2025 at 06:26:12 UTC from IEEE Xplore. Restrictions apply.

TABLE II
PERFORMANCE OF D3QN WITH DIFFERENT PARAMETERS

Average weighted tardiness / Discount factor	Learning rate		
	0.0001	0.0005	0.001
0.6	424.114	450.559	495.753
0.7	477.842	432.294	419.446
0.8	431.444	467.743	414.075
0.9	442.821	397.096	447.480
1.0	498.439	549.666	570.014

TABLE III
AVERAGE WEIGHTED TARDINESS WITH DIFFERENT METHODS

λ	df	GA	HGP	DQN	D3QN
80	1.5	1056.652	959.870	880.599	846.500
	2.0	513.042	424.982	355.634	358.840
	2.5	161.906	111.396	105.383	63.142
100	1.5	432.878	395.100	405.834	395.080
	2.0	89.092	73.455	69.313	45.648
	2.5	10.798	10.723	8.548	3.573
120	1.5	189.166	177.065	210.993	176.141
	2.0	20.874	17.664	19.168	8.097
	2.5	1.783	2.071	0.948	1.482

V. EXPERIMENTS AND ANALYSIS

A. Experimental Setting

The D3QN for distributed real-time scheduling of processing and logistics services is verified and validated in dynamic job shop scheduling problems. The algorithm of D3QN is programmed by Python 3.8 with TensorFlow 2.2, where the computer is equipped with Inter Core i7-11700KF, 64G RAM, and graphics card RTX 3070(8G).

As for dynamic job shop scheduling problems, experiments are generated according to Nguyen's research [29]. In this article, 10 machines are included in the manufacturing system. The job arrival follows a Poisson distribution, λ. Each job has 10 processing services generated randomly, where processing services of one job are processed on different machines. The processing time follows a discrete uniform distribution [1,99]. Because the logistics services time is considered in this article, the due date is defined by the release time, processing services time, logistics services time, and the delay factor (df), as shown in (23).

$$DT_i = r_i + df\left(\sum_{1}^{m} pt_{ij} + T_{M_0 M_{i1}} + \sum_{2}^{m} T_{M_{i(j-1)} M_{ij}} + T_{M_{im} M_0}\right). \quad (23)$$

Besides, three weights are randomly generated for jobs to simulate the importance of customers, where the weights of 20%, 60%, and 20% jobs are, respectively, set as one, two, and four according to the research of Pinedo and Singer [30].

B. Parameter Sensitivity Analysis

As we all know, the convergence speed and performance of DRL are affected by several parameters, including the learning rate and the discount factor. In this condition, 100 jobs are randomly generated to train the proposed D3QN and explore better parameters, where df, λ, and the number of AGVs are set as 2, 80, and 3, respectively. In this article, the learning rate is investigated with three levels (0.0001, 0.0005, 0.001) and the discount factor, γ, is investigated with five levels (0.6, 0.7, 0.8, 0.9, 1.0). The proposed D3QN is trained with 80 episodes and then tested the performance in 10 instances generated randomly. As shown in Table II, the minimum average weighted tardiness can be obtained when the learning rate is 0.0005 and the discount factor is 0.9, which is set as the experimental parameters in this article.

Furthermore, the training process of D3QN under the above parameters is analyzed. As shown in Fig. 4, as the episode increases, the reward for each episode increases and the average weighted tardiness decreases, which indicate the reward function can represent the real objective.

C. Result Comparison

The proposed D3QN is compared with three baselines from the state-of-the-art to evaluate the performance as follows.

1) Greedy algorithm (GA). GA is usually used as a baseline in dynamic job-shop scheduling. In this article, GA is defined by the action set, where the combined rule with the maximum reward in each decision point is selected as the action. GA is designed to investigate the improvement of D3QN in long-term decision-making problems.
2) Gene programming. This baseline was developed to solve dynamic job-shop scheduling problems [21], which just considered PSS without LSA. In the experiments, the LSA rule, MWT, is mixed with gene programming, named hybrid gene programming (HGP). This baseline helps to evaluate the improvement of D3QN for multiple services scheduling problems with coupling interaction.
3) DQN. This baseline was proposed in [25], using a DQN to select suitable dispatching rules and AGVs according to the real-time state. This baseline is used to evaluate the performance of D3QN in dynamic CM.

In this article, nine scenarios are generated with different distributions of job arrival and dfs to simulate dynamic environments, which are defined as (λ, df). For each scenario, ten instances are generated. The experimental results are shown in Table III. As we can see, the proposed D3QN obtains optimal performance in seven scenarios. Furthermore, the improvement rate in each scenario is analyzed. Compared with the three baselines, GA, HGP, and DQN, the average improvement rate is 35.59%, 28.70%, and 17.33%, respectively. It indicates that the proposed D3QN obtains significant improvement on weighted tardiness and has excellent adaptability for dynamic environments.

Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on February 19,2025 at 06:26:12 UTC from IEEE Xplore. Restrictions apply.

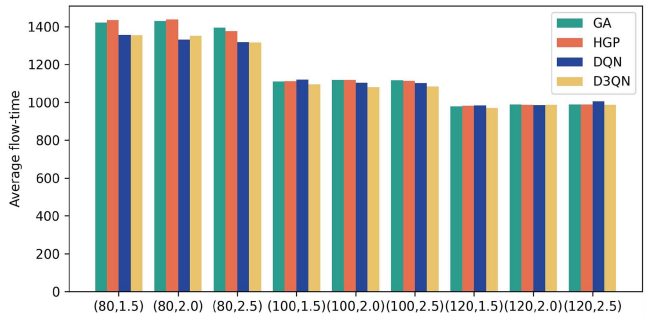

Fig. 5. Average flow-time with different methods.

Moreover, the average flow-time of jobs is calculated to analyze the performance of D3QN on production processes. As shown in Fig. 5, DQN and D3QN obtain better performance than GA and HGP. Compared with GA and HGP, the improvement rate of D3QN is 5.29%, 2.66%, and 0.39% when $\lambda = 80$, $\lambda = 100$, and $\lambda = 100$ respectively. This indicates the methods which make decisions based on real-time information can efficiently improve production performance. Especially, the improvement is significant when the job arrival is more frequent. That indicates DRL algorithms have significant potential to tackle complex and busy manufacturing environments in CM.

D. Discussion

To further analyze the proposed DRTS method enabled by distributed DRL with cloud–edge collaboration, the advantages are discussed by comparing it with cloud-based and edge-based scheduling methods.

Compared with only cloud-based methods, the proposed method has better anti-interference ability and security. The distributed agents utilize the policy learned in the cloud layer to make decisions according to real-time information. In this condition, the scheduling rules are treated as the action in the edge layer, which could provide feasible solutions even if the communication or the computational model is disrupted. Besides, the learning model in the cloud layer just utilizes decision-making experiences that have been normalized, which ensures data security on the shop floor.

On the other hand, the proposed method guarantees both responsiveness and performance. The distributed agents in the edge layer provide real-time decision-making to assure responsiveness. According to the experiments, the average decision-making time is about 0.017 s and the maximum decision-making time is less than 0.134 s, which satisfies the real-time scheduling requirements. As we all know, edge-based scheduling methods usually utilize scheduling rules or a simple optimization model to provide feasible solutions limited by the computing power on the shop floor. In contrast, the proposed DRTS method with cloud–edge collaboration has a strong optimization ability to provide an optimal or near-optimal solution in time due to the powerful computing power in the cloud platform. It means the proposed D3QN with cloud–edge collaboration is valid and efficient for real-time scheduling in CM.

VI. Conclusion

The distributed real-time scheduling problem of processing services with logistics constraints in CM was studied and well resolved in this article by cloud–edge collaboration and distributed DRL. The main conclusions are summarized as follows. First, the real-time scheduling of multiple services with coupling interaction in dynamic environments can be achieved by sharing information among multiple services based on the Internet of Things, which helps to improve responsiveness in CM. Second, distributed DRL is suitable for real-time scheduling in CM, where the responsiveness of decision-making and the efficiency of learning can be satisfied simultaneously by cloud–edge collaboration. Third, the well-designed D3QN is more efficient than other state-of-the-art baselines. It has excellent robustness and generalization ability in various scenarios, indicating significant potential against scheduling problems in CM.

In the future, the distributed real-time scheduling in CM with digital twins will be attempted against more uncertainties in CM.

References

[1] B. Li, L. Zhang, S. Wang, F. Tao, and X. Chai, "Cloud manufacturing: A new service-oriented networked manufacturing model," *Comput. Integr. Manuf. Syst.*, vol. 16, no. 1, pp. 1–7, 2010.

[2] A. Villalonga, G. Beruvides, F. Castano, and R. E. Haber, "Cloud-based industrial cyber-physical system for data-driven reasoning: A review and use case on an industry 4.0 pilot line," *IEEE Trans. Ind. Informat.*, vol. 16, no. 9, pp. 5975–5984, Sep. 2020.

[3] Y. Fan *et al.*, "A digital-twin visualized architecture for flexible manufacturing system," *J. Manuf. Syst.*, vol. 60, pp. 176–201, 2021.

[4] L. Zhang, Y. Yan, Y. Hu, and W. Ren, "A dynamic scheduling method for self-organized AGVs in production logistics systems," *Procedia CIRP*, vol. 104, pp. 381–386, 2021.

[5] A. Khan, F. Shahid, C. Maple, A. Ahmad, and G. Jeon, "Towards smart manufacturing using spiral digital twin framework and twinchain," *IEEE Trans. Ind. Informat.*, vol. 18, no. 2, pp. 1359–1366, Feb. 2022.

[6] S. Mayer, T. Classen, and C. Endisch, "Modular production control using deep reinforcement learning: Proximal policy optimization," *J. Intell. Manuf.*, vol. 32, pp. 2335–2351, 2021.

[7] G. Rjoub, J. Bentahar, O. Abdel Wahab, and A. Saleh Bataineh, "Deep and reinforcement learning for automated task scheduling in large-scale cloud computing systems," *Concurr. Comput.*, vol. 33, 2020, Art. no. e5919.

[8] P. Helo, D. Phuong, and Y. Hao, "Cloud manufacturing – Scheduling as a service for sheet metal manufacturing," *Comput. Operations Res.*, vol. 110, pp. 208–219, 2019.

[9] D. Mourtzis, E. Vlachou, N. Milas, N. Tapoglou, and J. Mehnen, "A cloud-based, knowledge-enriched framework for increasing machining efficiency based on machine tool monitoring," *Proc. Inst. Mech. Engineers, Part B: J. Eng. Manufacture*, vol. 233, no. 1, pp. 278–292, 2019.

[10] A. Caggiano, "Cloud-based manufacturing process monitoring for smart diagnosis services," *Int. J. Comput. Integr. Manuf.*, vol. 31, no. 7, pp. 612–623, 2018.

[11] C. Yang, S. Lan, L. Wang, W. Shen, and G. G. Q. Huang, "Big data driven edge-cloud collaboration architecture for cloud manufacturing: A software defined perspective," *IEEE Access*, vol. 8, pp. 45938–45950, 2020.

[12] M. Cheng, J. Li, and S. Nazarian, "DRL-cloud: Deep reinforcement learning-based resource provisioning and task scheduling for cloud service providers," in *Proc. Asia South Pacific Des. Automat. Conf.*, 2018, pp. 129–134.

[13] Y. Huang *et al.*, "Deep adversarial imitation reinforcement learning for QoS-aware cloud job scheduling," *IEEE Syst. J.*, to be published, doi: 10.1109/JSYST.2021.3122126.

[14] M. Cheng, J. Li, P. Bogdan, and S. Nazarian, "HO-cloud: A resource and quality of service-aware task scheduling framework for warehouse-scale data centers," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 39, no. 10, pp. 2925–2937, 2020.

Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on February 19,2025 at 06:26:12 UTC from IEEE Xplore. Restrictions apply.

[15] H. Liang, X. Wen, Y. Liu, H. Zhang, L. Zhang, and L. Wang, "Logistics-involved QoS-aware service composition in cloud manufacturing with deep reinforcement learning," *Robot. Comput.-Integr. Manuf.*, vol. 67, 2020, Art. no. 101991.
[16] G. El Khayat, A. Langevin, and D. Riopel, "Integrated production and material handling scheduling using mathematical programming and constraint programming," *Eur. J. Oper. Res.*, vol. 175, no. 3, pp. 1818–1832, 2006.
[17] P. Lacomme, M. Larabi, and N. Tchernev, "Job-shop based framework for simultaneous scheduling of machines and automated guided vehicles," *Int. J. Prod. Econ.*, vol. 143, no. 1, pp. 24–34, 2013.
[18] S. Reddy N., V. Ramamurthy D, P. Rao K., and P. Lalitha M., "Integrated scheduling of machines, AGVs and tools in multi-machine FMS using crow search algorithm," *Int. J. Comput. Integr. Manuf.*, vol. 32, no. 11, pp. 1117–1133, 2019.
[19] Y. Fang, C. Peng, P. Lou, Z. Zhou, J. Hu, and J. Yan, "Digital-twin-based job shop scheduling toward smart manufacturing," *IEEE Trans. Ind. Informat.*, vol. 15, no. 12, pp. 6425–6435, Dec. 2019.
[20] L. Zhou, L. Zhang, L. Ren, and J. Wang, "Real-time scheduling of cloud manufacturing services based on dynamic data-driven simulation," *IEEE Trans. Ind. Informat.*, vol. 15, no. 9, pp. 5042–5051, Sep. 2019.
[21] F. Zhang, Y. Mei, S. Nguyen, K. C. Tan, and M. Zhang, "Multitask genetic programming-based generative hyperheuristics: A case study in dynamic scheduling," *IEEE Trans. Cybern.*, to be published, doi: 10.1109/TCYB.2021.3065340.
[22] Y. C. Wang and J. M. Usher, "Application of reinforcement learning for agent-based production scheduling," *Eng. Appl. Artif. Intell.*, vol. 18, no. 1, pp. 73–82, 2005.
[23] Y. R. Shiue, K. C. Lee, and C. T. Su, "Real-time scheduling for a smart factory using a reinforcement learning approach," *Comput. Ind. Eng.*, vol. 125, no. 101, pp. 604–614, 2018.
[24] C. Qiu, F. R. Yu, H. Yao, C. Jiang, F. Xu, and C. Zhao, "Blockchain-based software-defined industrial internet of things: A dueling deep q-learning approach," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4627–4639, Jun. 2019.
[25] H. Hu, X. Jia, Q. He, S. Fu, and K. Liu, "Deep reinforcement learning based AGVs real-time scheduling with mixed rule for flexible shop floor in industry 4.0," *Comput. Ind. Eng.*, vol. 149, 2020, Art. no. 106749.
[26] H. Tang, A. Wang, F. Xue, J. Yang, and Y. Cao, "A novel hierarchical soft actor-critic algorithm for multi-logistics robots task allocation," *IEEE Access*, vol. 9, pp. 42568–42582, 2021.
[27] Y. Yadekar, E. Shehab, and J. Mehnen, "Taxonomy and uncertainties of cloud manufacturing," *Int. J. Agil. Syst. Manag.*, vol. 9, no. 1, pp. 48–66, 2016.
[28] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[29] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "Automatic programming via iterated local search for dynamic job shop scheduling," *IEEE Trans. Cybern.*, vol. 45, no. 1, pp. 1–14, Jan. 2015.
[30] M. Pinedo and M. Singer, "A shifting bottleneck heuristic for minimizing the total weighted tardiness in a job shop," *Nav. Res. Logistics*, vol. 46, no. 1, pp. 1–17, 1999.

Lixiang Zhang received the B.Eng. degree in industrial engineering in 2019 from the Beijing Institute of Technology, Beijing, China, where he is currently working toward the Ph.D. degree in mechanical engineering with the Laboratory of Industrial and Intelligent System Engineering, School of Mechanical Engineering.

His research interests include artificial intelligence in manufacturing, optimization, and simulation of manufacturing systems, cloud manufacturing, and digital twin.

Chen Yang (Member, IEEE) received the B.Eng. degree in automatic and information technology and the Ph.D. degree in control science and engineering from the School of Automation Science and Electrical Engineering and the Honors College (an elite program), Beihang University (BUAA), Beijing, China, in 2008 and 2014, respectively.

He was a Postdoctoral Fellow and Associate Research Officer with HKU-ZIRI Laboratory for Physical Internet, University of Hong Kong, and a Senior Engineer on R&D tools with Huawei Technologies. He is currently an Associate Professor with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing. His research interests include Internet of Things, Industry 4.0, cloud manufacturing, modeling and simulation of complex systems, artificial intelligence, and Big Data analytics.

Yan Yan received the B.Eng. and Ph.D. degrees in mechanical engineering from the Beijing Institute of Technology, Beijing, China, in 1989 and 2001, respectively.

She is currently a Professor with the School of Mechanical Engineering, Beijing Institute of Technology. She has completed more than ten programs in the digital manufacturing and advanced design field, including National Natural Science Foundation of China and National Key Research and Development Programs. She has authored or coauthored more than 60 journal or conference papers. Her research works have been widely cited in the relevant field. Her research interests include intelligent manufacturing systems, intelligent design, and knowledge engineering.

Yaoguang Hu received the B.Eng. degree in mechanical design and automation and the Ph.D. degree in mechanical engineering from Beihang University (BUAA), Beijing, China, in 1996 and 2003, respectively.

He was a Postdoctoral Fellow with National CIMS Engineering Research Center, Tsinghua University, and a Visiting Scholar with the Department of Mechanical and Aerospace Engineering, UCLA. He is currently a Professor with the School of Mechanical Engineering, Beijing Institute of Technology, Beijing. He is also one of the experts on "Network Collaborative Manufacturing and Smart Factory" and "Industrial Software" in the National Key Research and Development Programs. His research interests include production operation and management, intelligent manufacturing, optimization and simulation of manufacturing system, and digital twin.

Authorized licensed use limited to: BEIJING UNIVERSITY OF TECHNOLOGY. Downloaded on February 19,2025 at 06:26:12 UTC from IEEE Xplore. Restrictions apply.