IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 8, 2020, accepted December 18, 2020, date of publication December 23, 2020,
date of current version January 5, 2021.

Digital Object Identifier 10.1109/ACCESS.2020.3046784

Reinforcement Learning With Composite Rewards for Production Scheduling in a Smart Factory

TONG ZHOU, DUNBING TANG, HAIHUA ZHU, AND LIPING WANG
College of Mechanical and Electrical Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China

Corresponding author: Dunbing Tang (d.tang@nuaa.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2020YFB1710500, in part by the National Natural Science Foundation of China under Grant 52075257, and in part by the Fundamental Research Funds for the Central Universities under Grant NP2020304.

ABSTRACT Rapid advances of sensing and cloud technologies transform the manufacturing system into a data-rich environment and make production scheduling increasingly complex. Traditional offline scheduling methods are limited in the ability to handle low-volume-high-mix workorders with diverse design specifications. Simulation-based methods show the promise for distributed scheduling of manufacturing jobs but are mostly implemented with historical data and empirical rules in a static manner. Recently, artificial intelligence (AI) algorithms fuel increasing interests to solve dynamic scheduling problems in the manufacturing setting. However, it's difficult to utilize high-dimensional data for production scheduling while considering multiple practical objectives for smart manufacturing (e.g., minimize the makespan, reduce production costs, balance workloads). Therefore, this paper presents a new AI scheduler with composite reward functions for data-driven dynamic scheduling of manufacturing jobs under uncertainty in a smart factory. Internet-enabled sensor networks are deployed in the smart factory to track real-time statuses of workorders, machines, and material handling systems. A novel manufacturing value network is developed to take high-dimensional data as the input and then learn the state-action values for real-time decision making. Based on reinforcement learning (RL), composite rewards help the AI scheduler learn efficiently to achieve multiple objectives for production scheduling in real time. The proposed methodology is evaluated and validated with experimental studies in a smart manufacturing setting. Experimental results show that the new AI scheduler not only improves the multi-objective performance metrics in the production scheduling problem but also effectively copes with unexpected events (e.g., urgent workorders, machine failures) in manufacturing systems.

INDEX TERMS Production scheduling, reinforcement learning, composite reward, smart factory, neural network.

NOTATIONS

I	total number of workorders
o_i	ith workorder ($i = 1, \ldots, I$)
J_i	total number of jobs of workorder o_i
$b_{i,j}$	jth job of workorder o_i ($j = 1, \ldots, J_i$)
M	total number of machines
m	machine number ($m = 1, \ldots, M$)
C	total number of machine or job types
c	machine or job type ($c = 1, \ldots, C$)
M_c	total number of type-c machines ($M_1 + \ldots + M_C = M$)
$\boldsymbol{S}_t$	states of a manufacturing shop at time t
a_m	assignment of a current job to machine m ($m = 0, 1, \ldots, M$; a_0 denotes "waiting")
A_t	a chosen action from the action space at time t
$T_{i,j}^{(A)}$	initialization time of job $b_{i,j}$
$T_{i,j}^{(S)}$	starting time of job $b_{i,j}$
$T_{i,j}^{(C)}$	actual completion time of job $b_{i,j}$
$\hat{T}_{i,j}^{(C)}$	target completion time of job $b_{i,j}$
$T_{i,j}$	workload time of job $b_{i,j}$
$\tilde{T}_{i,j}$	nominal operating time of job $b_{i,j}$, $\tilde{T}_{i,j} = 3T_{i,j}$
K_i	urgency factor of workorder o_i
$K_{i,j}^{(P)}$	price factor for job $b_{i,j}$
$K_m^{(E)}$	energy efficiency factor of machine m

The associate editor coordinating the review of this manuscript and approving it for publication was Ting Wang.

$K_{i,j}^{(T)}$	speed factor of machine m
R_D	reward for tardiness level
R_P	reward for profits
R_U	reward for machine utilization rates
R_V	reward for workload balance
R_{t+1}	composite reward for action A_t
u_m	utilization rate of machine m
U_c	standard deviation of utilization rates

I. INTRODUCTION

The new generation of smart factories is highly digitalized and is equipped with a network of internet-enabled sensors and devices to increase information visibility about real-time statuses of ''manufacturing things'' (e.g., workorders, machines, and material handling systems). As a result, large amounts of data are readily available in the manufacturing environment. This provides an unprecedented opportunity to improve the performance of manufacturing systems. However, big data from the smart factories is high-dimensional and updated in real time during the operation, which is subjected to uncertainties and disturbances (e.g., workorder and machine variations). Realizing the full potential of big data depends on the development of analytical algorithms to improve the ''smartness'' of manufacturing operations. For example, production scheduling is a complex combinatorial problem, which is traditionally implemented offline in a centralized manner with static and deterministic assumptions. These offline methods are often ineffective and need to be updated periodically when applied in a dynamic environment (e.g., abrupt changes in market demands, an increasing number of product variants, or smaller order sizes). For example, mass customization entails diverse design specifications and personalized products from customers. Indeed, the new paradigm shift from mass manufacturing to low-volume-high-mix production brings a higher level of disturbances and uncertainties, which may come from internal (e.g., machine failures, operator absence), external (e.g., urgent workorders, limited availability of raw materials), and other extraneous factors (e.g., temporal variations of market demand, resource uncertainty) in real industrial conditions.

To handle disturbances in manufacturing shops, early studies periodically update and reschedule the resources to achieve newly adaptive results. This is, however, not suitable to realize the full potential of data for smart manufacturing. Traditional offline scheduling methods are limited in the ability to handle low-volume-high-mix workorders with diverse design specifications. With the increasing availability of data, simulation-based methods (e.g., multi-agent system) leverage the historical data and empirical rules for distributed scheduling of manufacturing jobs. A variety of manufacturing agents (e.g., workorder agent, machine agent, transport agent) are modeled to interact with others for production scheduling. Nonetheless, simulation methods tend to be limited in the ability to handle real-time data flows and uncertainty factors for dynamic scheduling. Recently, artificial intelligence (AI) fuels increasing interests to solve dynamic scheduling problems in the manufacturing setting. However, it's difficult to utilize high-dimensional data for production scheduling while considering multiple practical objectives for smart manufacturing (e.g., minimize the makespan, reduce production costs, balance workloads). Indeed, the system complexity and rich data environment pose significant challenges to the development of new AI methods and tools for multi-objective dynamic scheduling in a smart factory.

This paper presents a new AI scheduler with composite reward functions for data-driven dynamic scheduling of manufacturing jobs under uncertainty in a smart factory. We design an AI scheduler to learn, adapt, and make scheduling decisions by interacting with the factory. After a workorder is received in the cloud platform, it will be decomposed into a series of manufacturing jobs by process planning. A novel manufacturing value network is designed to take high-dimensional data as the input and then learn the state-action values for real-time decision making. Further, a composite reward function is designed to account for multiple production objectives (i.e., minimize the makespan, reduce production costs, balance workloads) for training the AI scheduler. The proposed methodology is evaluated and validated with experiments in a smart manufacturing setting. We benchmark the performance of the AI scheduler with a variety of traditional scheduling methods. Experimental results show that the new AI scheduler not only improves the multi-objective performance metrics in the production scheduling problem but also effectively copes with unexpected events (e.g., urgent workorders, machine failures) to achieve an optimal balance between efficiency and energy consumption. The proposed AI scheduler shows strong potential for general applicability in a smart factory.

The remaining sections of this paper are organized as follows. Section II presents a review of relevant literature in production scheduling. Section III presents the proposed methodology of an AI scheduler with manufacturing value networks, composite rewards, and online reinforcement learning (RL) methods. Section IV provides the details of the experimental design to evaluate the performance of the AI scheduler. Section V shows experimental results and makes comparison with different scheduling methods. Section VI concludes this paper and discusses potential topics for future research.

II. RESEARCH BACKGROUND

A. PRODUCTION SCHEDULING

Production scheduling optimizes the allocation of manufacturing assets and resources to fulfill workorders received in manufacturing shops. Typically, the scheduling problem involves a set of workorders to be completed. Each workorder comprises a set of jobs, and each job needs to be performed on certain machines that meet with requirements on resources and specifications. There are also other factors to be considered such as workorder timeliness, machine attributes, material availability, and cost effectiveness.

Production scheduling is known to be a complex problem. The computing loads to derive optimal schedules increases exponentially with respect to the job sizes. Conway *et al*. [1] pioneered the study of production scheduling in 1967 and significantly contributed to the modeling and optimization of manufacturing systems. In the past few decades, manufacturing systems have been transformed from large-batch to low-volume-high-mix production, which demands the continuous improvement of scheduling approaches [2]. In the literature, there exist a variety of methods and tools (e.g., optimization models, knowledge-based methods, heuristic algorithms) for production scheduling. Classical optimization models for production scheduling use the mathematical programming approach to achieve optimal solutions [3]. The scheduling objective is formulated in a programming function, whose critical solution represents optimal schedules under constraints. Naive optimization models are limited in solving a scheduling problem with numerous variations and multiple constraints. Thus, the knowledge-based method (i.e., expert system) is proposed to simulate a scheduling system with human expertise that is a primitive form of AI. Kumara *et al*. [4] introduced a framework to develop such an expert system for job-shop scheduling and fault diagnosis. Further, Lawrynowicz [5] proposed the use of scheduling rules to achieve higher efficiency, which integrates an expert system with heuristic algorithms to solve the scheduling problem under the constraints of supply chains. As the manufacturing systems become more complex and stochastic, Bellman *et al*. [6] applied dynamic programming in production scheduling by converting a combinatorial optimization problem into an analytic one. There are also previous studies that introduced the biomimetic method [7] or swarm intelligent algorithms [8] for production scheduling with a large action space.

From simple dispatching rules to basic intelligence, scheduling algorithms become more efficient and effective so as to handle a large number of workorders. However, most traditional methods are operated offline for production scheduling. While optimal policies can be achieved for offline and stationary scheduling problems, they tend to be limited in the ability to handle nonstationarity in the real-world environment. When the order variety and size increase, the computational complexity increases exponentially. In addition, data is not fully utilized in these traditional scheduling methods. When a large number of heterogeneous workorders arrive at a manufacturing shop, traditional offline methods are deemed as one-shot optimization that is limited in the ability to handle newly emerged uncertainties and low-volume-high-mix workorders with diverse design specifications.

B. SIMULATION-BASED PRODUCTION SCHEDULING

Traditional offline scheduling methods are limited in the ability to fully utilize the potentials of data for the model formulation. Hence, computer simulation was increasingly used to increase the decision-making abilities in manufacturing systems [9]. Simulation-based production scheduling is used to model and evaluate the scheduling system for a manufacturing shop. With the rapid advances of new technologies such as internet of things (IoT), big data analytics, cloud computing, the use of ''digital twin'' was proposed for production scheduling [10]. Simulation models describe the factory rules of system operations and run scenario analysis of scheduling policies to derive optimal solutions. Discrete-event simulation (DES) and multi-agent simulation (MAS) are commonly used to build complex scheduling models. DES models the operation of a manufacturing system as a discrete sequence of events in time. Chong and Sivakumar [11] built DES models of a manufacturing system with historical data and evaluated different alternatives to determine the optimal schedule. Bang and Kim [12] proposed a two-level hierarchical method for production planning by combining linear programming with priority-rule-based scheduling to identify scheduling policies, then evaluated the results with DES. Note that the manufacturing system consists of many discrete units and agents (e.g., workorders, machines, and material handling systems). Agent-based methods are proposed to equip discrete manufacturing units with more intelligence. MAS provides a natural and simple way to model and analyze a manufacturing system [13]. An agent is a computer system that is capable of taking autonomous actions in its environment to meet the objectives [14]. The MAS includes frameworks, interactive mechanisms, decision-making algorithms. Lee *et al*. [15] discussed some key issues in the development of agent-based manufacturing systems such as system architectures, distributed production scheduling. Lots of socialized interactive mechanisms, such as contract net protocol (CNP) [16], game theory [17], auction mechanism [18], are used for the coordination of machines in MAS. The rational scheduling policies are generated autonomously based on MAS models [19]. Also, RL methods are used to interact with simulation models, which are built based on historical data for the production scheduling with a single optimization objective [20].

However, most simulation-based methods for production scheduling are established based on empirical rules and historical data, and are less concerned about real-time and high-dimensional data from sensor networks. The real-time statuses of the factory and ''manufacturing things'' are not captured in simulation models. As the new paradigm of mass customization brings lots of uncertainties (e.g., workorder variation, machine variation, machine failures, and urgent workorders), traditional simulation-based methods are limited in the ability to realize the full potentials of real-time data for multi-objective production scheduling.

C. DYNAMIC SCHEDULING UNDER UNCERTAINTY

In a dynamic environment, stationary schedules will become invalid due to an unexpected event in the factory [21]. For example, if a machine breaks down in the manufacturing process, workorders assigned to the failed machine will be in the waiting queue and may be rescheduled to other machines. It is common that real-world manufacturing plants may

experience internal disturbances (e.g., machine failures, operator absence), external uncertainties (e.g., urgent workorders, limited availability of raw materials), and other extraneous factors (e.g., temporal variations of market demand, resource uncertainty). As such, any stationary schedule is difficult to stay optimal in a dynamic manufacturing setting [22].

Dynamic scheduling is aimed at dealing with unexpected events and uncertainty factors in real time. Traditional offline scheduling methods adjust the primitive schedules when some unexpected events occur [23]. Simulation-based methods are offline but are increasingly efficient for periodic updates of schedules due to the availability of high-performance computing [24]. These offline methods rely heavily on central databases and computational efficiency when facing the need to dynamically update production schedules, but are less concerned about the utilization of real-time sensor data. MAS is popular for distributed scheduling [25], but common MAS with simple interactive protocols (e.g., CNP) has limited decision-making abilities due to the lack of AI algorithms. With the recent successes of deep learning [26], AI fuels increasing interests to solve the dynamic scheduling problem in the manufacturing setting. AI methods provide attractive features such as self-organizing operation, dynamic adjustment, online learning, and real-time decision making. This provides an unprecedented opportunity to handle unexpected events in real time, and improves the decision-making abilities of schedulers in the manufacturing processes. Zhang *et al*. [27] developed an RL method for unrelated parallel machine scheduling with the consideration of a single objective of mean weighted-tardiness, but are less concerned about other important factors such as unexpected events, order details, and reward mechanisms. Shahrabi *et al*. [28] proposed to consider random order arrivals and machine failures in the design of RL methods to enhance the performance of dynamic scheduling. Shiue *et al*. [29] investigated the design of an offline learning module and a Q-learning module for RL-based scheduling with multiple dispatching rules. However, it's difficult for RL methods (e.g., Q-learning) to take high-dimensional data as inputs and solve problems with large state-action spaces. Chen *et al*. [30] used an RL-based assigning policy to obtain the non-dominated solution set in the action space that helps yield a better performance than Q-learning. Wang *et al*. [31] used correlated equilibrium to propose a multi-agent RL algorithm for makespan and cost optimization to guide the scheduling of multi-workflows over clouds.

With rapid advances in sensing technology, manufacturing plants become data-rich environments that provide a great opportunity to develop sensor-based dynamic scheduling of manufacturing jobs under uncertainty in a smart factory. The high-dimensional data may come from machine states (e.g., type, machining speed, energy efficiency, buffer level), the states of material handling systems, and workorder states (e.g., type, initialization time, target completion time, process plan). Traditional RL methods either consider a single objective (e.g., makespan or cost) or only interact with historical data for learning instead of real-time data from a smart factory. There is an urgent need to design new neural networks that can take high-dimensional data as the input and then learn the state-action values for real-time decision making. In addition, new composite reward mechanisms are indispensable to consider multiple practical objectives for smart manufacturing (e.g., minimize the makespan, reduce production costs, balance workloads). The system complexity and rich data environment pose significant challenges to the development of new AI methods and tools for multi-objective dynamic scheduling in a smart factory.

III. RESEARCH METHODOLOGY

As shown in Figure 1, a typical manufacturing shop is composed of machines, material handling systems (e.g., automated guided vehicles (AGVs), robots), and inventory zones. The material handling system transports workorders among machines, as well as between machines and the inventory zones. Each machine has a buffer area for workorders. First, customers place workorders via the cloud, which are then transmitted to a database in the manufacturing shop. Process planning decomposes a workorder into a sequence of manufacturing jobs. Second, advanced sensing systems collect the states of machines and workorders in real time, which are then used as inputs of the AI scheduler. Third, a new composite reward function is designed to enable the AI scheduler for multi-objective learning and optimization of production schedules. Finally, a series of experiments are designed to benchmark the performance of the AI scheduler with a variety of traditional scheduling methods.

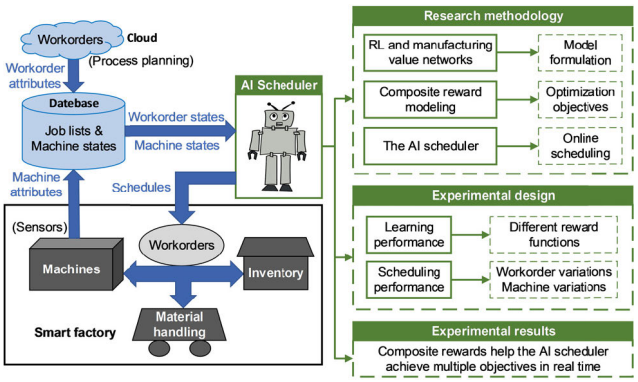

FIGURE 1. Flowchart of the proposed methodology.

A. REINFORCEMENT LEARNING AND MANUFACTURING VALUE NETWORKS

RL is a straightforward framework of learning from interaction to achieve optimal objectives [32]. As shown in Figure 2, a typical RL model of a manufacturing shop is dependent on the sensors for data collection and state estimation. The AI scheduler interacts with the factory to learn and decide what to do according to the rewards received from state transitions of machines and jobs. After an action is taken, sensors will

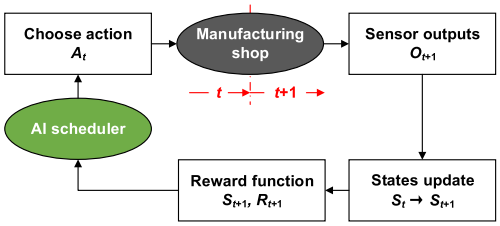

FIGURE 2. The interaction between a scheduler and the manufacturing shop in an RL model.

observe the changes of a factory and feed the data to analytical modules to estimate the new state. The new state will incur a reward to the scheduler. As such, the AI scheduler keeps interacting with the factory to learn and improve the decision-making abilities for production scheduling.

In an online scheduling system, schedulers should make decisions as soon as jobs are created in real time. This is highly dependent on real-time states of manufacturing shops, and therefore turns a conventional production scheduling problem to online learning. The analogous relationships between RL and production scheduling models are listed in Table 1. RL modeling is the logical next step to solve production scheduling problems.

TABLE 1. The relationships between RL and production scheduling.

RL	Production scheduling
Agent	Scheduler
Environment	A factory with machines and workorders
State	Dynamic attributes of machines and jobs
Action	Scheduling
Reward	Optimization objectives

1) MODEL FORMULATION

Process planning: As shown in Figure 3, a workorder is denoted by o_i, $i = 1, \ldots, I$, where I is the total number of workorders. The jth job of a workorder o_i is denoted by $b_{i,j}$, $j = 1, \ldots, J_i$, where J_i is the total number of jobs of workorder o_i. If there are a total of I workorders, then the total number of jobs will be $J = J_1 + \ldots + J_I$. Scheduling moments are at the start and end time of each job. When a job is completed, the next job of the same workorder will be initialized and appended to the job list. If there is no available machine for the current job, it will wait for the next scheduling moment. If two or more jobs are initialized at the same time, the scheduler will prioritize the one with more rewards. Production scheduling coordinates jobs and machines to create optimal sequences in the job list.

*Assumptions:*This paper holds the following assumptions during the development of an online scheduling model for smart manufacturing. First, each machine can handle one specific type of manufacturing jobs (e.g., milling, drilling, or lathing); a workorder contains one or more jobs. Each job can be completed in one machine, and the chosen machine should be capable of undertaking the corresponding job,

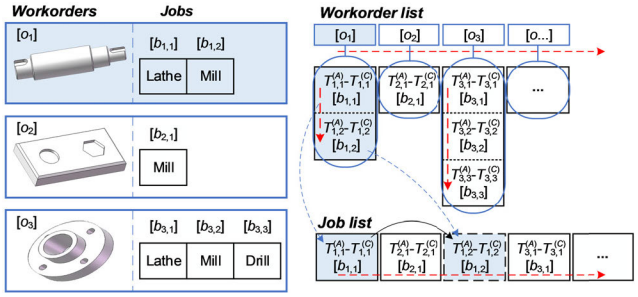

FIGURE 3. The workorder and job lists for production scheduling. E.g., workorder o_1 is composed of two jobs ($b_{1,1}$ and $b_{1,2}$); job $b_{1,1}$ is initialized at time $T^{(A)}_{1,1}$ and completed at time $T^{(C)}_{1,1}$; job $b_{1,2}$ is initialized at time $T^{(A)}_{1,2}$ and completed at time $T^{(C)}_{1,2}$. The workorder list is sequenced by the creating time of workorders, and the job list is sequenced by the initialization time of jobs.

i.e., machine type matches job type. Second, the scheduling moment is at the start and end time of each job and cannot be in the middle of a job. Third, the next scheduled job can be machined instantly when the current job is finished. Also, the job sequence should satisfy design requirements. In other words, a job cannot be processed until prior jobs of the workorder are finished. The total number of jobs assigned to a machine should be no more than the capacity.

2) MANUFACTURING VALUE NETWORKS

To address the high dimensionality of jobs and machines, we propose a manufacturing value network to learn optimal scheduling policies according to the job-machine states that are characterized by real-time sensor data.

Network inputs: The state space consists of attributes of schedulable jobs and machines in the manufacturing shop. When workorders are created, the target completion time will be provided to customers through the order system. This estimate of completion time is computed based on empirical data and the current condition of the manufacturing shop. The attributes for job $b_{i,j}$ can be described as $(s_1, \ldots, s_{d1})_{i,j}$, where $d1$ is the dimension of job attributes. The attributes for machine m ($m = 1, \ldots, M$, where M is the total number of machines) can be described as $(s_1, \ldots, s_{d2})_m$, where $d2$ is the dimension of machine attributes. As shown in Table 2, in the present study, a job includes four attributes, including job type, initialization time, nominal operating time, target completion time, then $d1 = 4$; a machine is assigned with five attributes (i.e., $d2 = 5$), including machine type, machining speed, energy efficiency, remaining workloads, remaining buffer length. There are C types of machines that can take C different types of jobs. The total number of machines in type c is denoted by M_c, where $c = 1, \ldots, C$ and $M_1 + \ldots + M_C = M$. If there are J jobs to be scheduled and M machines are available, the state of the manufacturing shop at time t will be:

$$\tilde{\boldsymbol{S}}_t = [(s_1, \ldots, s_{d1})_{i,j}, \ldots; (s_1, \ldots, s_{d2})_m, \ldots]_{1\times(d1\cdot J + d2\cdot M)} \quad (1)$$

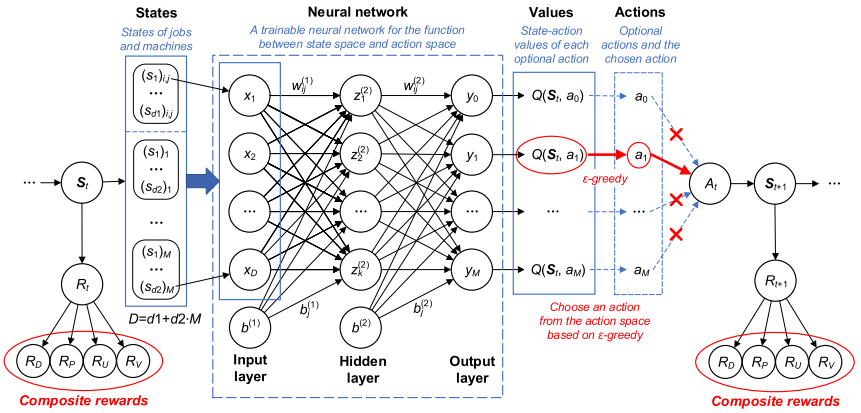

FIGURE 4. The manufacturing value network for selecting an optimal action A_t according to the states S_t.

TABLE 2. The attributes of jobs and machines in a manufacturing shop.

Job attributes	$(s_1)_{i,j}$	Job types
	$(s_2)_{i,j}$	Initialization time
	$(s_3)_{i,j}$	Nominal operating time
	$(s_4)_{i,j}$	Target completion time
Machine attributes	$(s_1)_m$	Machine types
	$(s_2)_m$	Machining speed factor
	$(s_3)_m$	Energy efficiency factor
	$(s_4)_m$	Waiting time for remaining workloads
	$(s_5)_m$	Remaining buffer length

Note that very few jobs will be initialized at the same time. There is an urgent need to standardize the state space for the improvement of learning efficiency and robustness of the RL model. Thus Equation (1) is reduced into Equation (2) with a one-dimensional tuple with $d1 + d2 \cdot M$ elements, which only keeps the current job to reduce the dimensionality of state space. In this way, the dimensionality of network inputs $\boldsymbol{S}_t$ is consistent with the current job and the attributes of all machines in the shop.

$$\boldsymbol{S}_t = [(s_1, \ldots, s_{d1})_{i,j}; (s_1, \ldots, s_{d2})_1, \ldots, (s_1, \ldots, s_{d2})_M]_{1\times(d1+d2\cdot M)} \quad (2)$$

Network architecture: As shown in Figure 4, the manufacturing value network, parameterized by $\boldsymbol{w}$, is composed of an input layer, a hidden layer, and an output layer. The input layer is constituted by $d1 + d2 \cdot M$ neurons. The output layer includes $1+M$ actions, i.e., $a_0, a_1, \ldots, a_M$, where a_0 denotes "waiting" and a_m is the assignment of a current job to the mth machine. The machine states are updated in real time for online scheduling of manufacturing jobs. The input layer takes high-dimensional data $\boldsymbol{S}_t$ from sensors, then regularizes the sensor data by batch normalization. After being processed by interconnected neurons in the neural network, a tuple $(Q(\boldsymbol{S}_t, a_0), Q(\boldsymbol{S}_t, a_1), \ldots, Q(\boldsymbol{S}_t, a_M))$ is obtained. The output layer provides the action to the scheduler. The optimal action A_t can be obtained from the $Q(\boldsymbol{S}_t, \cdot)$ in the output tuple by an ε-greedy policy. The scheduler selects a random action with the probability ε, and an action with the maximum $Q(\boldsymbol{S}_t, \cdot)$ with the probability 1-ε. After an action is chosen for the current job, a new state $\boldsymbol{S}_{t+1}$ and reward R_{t+1} are generated by the manufacturing shop. The performance of an AI scheduler depends on the state transitions and the definition of reward functions. The formulation of composite reward functions is detailed in Section III-B.

Reinforcement learning: An AI scheduler provides real-time scheduling policies via the training of the proposed manufacturing value network. The smartness of an AI scheduler is improved continuously by interacting with the factory and learning from the experiences. The manufacturing value network is trained by optimizing the differential equations of reward functions. Q-learning, developed by Watkins [33], is an off-policy temporal difference (TD) algorithm to get optimal policies of an RL model. Here, Q-learning is utilized to interact with the factory with stochastic state transitions for the derivation of optimal policies as

$$Q'(\boldsymbol{S}_t, A_t) = Q(\boldsymbol{S}_t, A_t) + \alpha[R_{t+1} + \gamma \max_{A_{t+1}} Q(\boldsymbol{S}_{t+1}, A_{t+1}) - Q(\boldsymbol{S}_t, A_t)] \quad (3)$$

where γ is a discount factor for future rewards; α is the learning rate. $Q(\cdot)$ is a state-action value function, which is sequentially updated during the training period; A_t denotes a chosen action from the action space at time t; R_{t+1} is a reward for the state transition from $\boldsymbol{S}_t$ to $\boldsymbol{S}_{t+1}$. A scheduling experience of the AI scheduler is denoted by $(\boldsymbol{S}_t, A_t, R_{t+1}, \boldsymbol{S}_{t+1})$, and all the experiences are stored for training purposes. Such scheduling experiences can be from not only this manufacturing shop but also other shops. When a large number of experiences are accumulated to train the manufacturing value network, the AI scheduler will be more reliable and robust.

RL for training the manufacturing value network: If there are only $Q(\boldsymbol{S}_{t+1}, A_{t+1})$ and $Q(\boldsymbol{S}_t, A_t)$ from the same manufacturing value network, the iteration function in Equation (3)

tends to be insufficient for training the network. There is a need to circumvent the correlations between $Q(\boldsymbol{S}_{t+1}, A_{t+1})$ and $Q(\boldsymbol{S}_t, A_t)$ for more effective parameter updating. Therefore, $Q(\boldsymbol{S}_{t+1}, A_{t+1})$ will be derived from a target network denoted by $\hat{Q}(\cdot)$, while $Q(\boldsymbol{S}_t, A_t)$ is still obtained from the manufacturing value network, i.e., evaluation network. The target network has the same architecture and initial parameters as the evaluation network. The evaluation network is trained and updated constantly with stochastic rewards and transitions, while parameters $\boldsymbol{w'}$ of the target networks are replaced with parameters $\boldsymbol{w}$ of the evaluation network in periodic steps. Equation (3) is then divided into two parts: $\hat{y} = R_{t+1} + \gamma \max_{A_{t+1}} \hat{Q}(\boldsymbol{S}_{t+1}, A_{t+1})$ with target-Q value $\hat{Q}(\cdot)$, and evaluation-Q value $Q(\boldsymbol{S}_t, A_t)$. Random samples of experience are drawn to perform minibatch learning. Network parameters are updated by the mean of minibatch samples. The manufacturing value network is trained by tuning parameters $\boldsymbol{w}$ and minimizing the loss function as follows:

$$L(\boldsymbol{w}) = E[\left(\hat{y} - Q\left(\boldsymbol{S}_t, A_t\right)\right)^2] \tag{4}$$

B. COMPOSITE REWARD MODELING

Production scheduling focuses on the performance improvement of a smart factory from various aspects such as efficiency (e.g., makespan, tardiness), cost (e.g., energy consumption, workorder urgency levels), and other metrics (e.g., workload balance, customer satisfaction). A single objective, e.g., minimizing the makespan, is often not enough. Indeed, workload balance and energy consumption should be considered along with shortening the makespan. Multi-objective formulation can not only meet customers' satisfaction, save energy for more profits and cost-reduction, but also balance the workload to reduce the machine failures.

1) MINIMIZING THE MAKESPAN OF WORKORDERS

The initialization time, starting time, completion time, target completion time of a job $b_{i,j}$ are respectively denoted by $T^{(A)}_{i,j}$, $T^{(S)}_{i,j}$, $T^{(C)}_{i,j}$, $\hat{T}^{(C)}_{i,j}$. As shown in Figure 5, the first job of a workorder is initialized at time $T^{(A)}_{i,1}$. The second job is initialized when the first job is completed at $T^{(C)}_{i,1}$. A workorder is completed when its last job b_{i,J_i} is completed at $T^{(C)}_{i,J_i}$. Each job has a target completion time $\hat{T}^{(C)}_{i,j}$. The target completion time of a workorder is determined by its last job with $\hat{T}^{(C)}_{i,J_i}$, which could be before or after the actual completion time $T^{(C)}_{i,J_i}$.

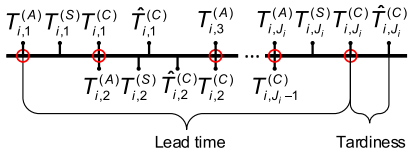

FIGURE 5. The timeline of workorder o_i and its component jobs: scheduling moments are marked by red circles.

The target completion time $\hat{T}^{(C)}_{i,j}$ is estimated by multiplying nominal operating time $\tilde{T}_{i,j}$ with the urgency factor of a workorder K_i, as shown in Equation (5). The nominal operating time $\tilde{T}_{i,j}$ of a job $b_{i,j}$ is obtained by multiplying workload time $T_{i,j}$ with a factor whose value is assigned with 3 in this paper (i.e., $\tilde{T}_{i,j} = 3T_{i,j}$). The urgency factor K_i of a workorder o_i is specified by customers. If a workorder is urgent, then K_i decreases so that $\hat{T}^{(C)}_{i,j}$ is smaller. In this investigation, K_i ranges from 2/3 to 4/3. The workorders (urgent, normal, and delayed) are assigned with the K_i's of 2/3, 1, 4/3 respectively.

$$\hat{T}^{(C)}_{i,j} = T^{(A)}_{i,j} + K_i \cdot \tilde{T}_{i,j}, 2/3 \leq K_i \leq 4/3 \tag{5}$$

If a workorder is completed before the target completion time, the AI scheduler will get a positive reward. Otherwise, the reward is negative. Therefore, we define the first reward function R_D based on the tardiness level of workorders as follows:

$$R_D(o_i) = \frac{1}{e^{1+T^{(\text{tard})}_i}} \tag{6}$$

where $T^{(\text{tard})}_i$ is the tardiness level of a workorder o_i, formulated by

$$T^{(\text{tard})}_i = \frac{T^{(C)}_{i,J_i} - \hat{T}^{(C)}_{i,J_i}}{\hat{T}^{(C)}_{i,J_i} - T^{(A)}_{i,1}}, 0 < T^{(A)}_{i,1} < T^{(C)}_{i,J_i}, \hat{T}^{(C)}_{i,J_i} \tag{7}$$

If a workorder is completed before the target completion time, $T^{(\text{tard})}_i$ is negative. Otherwise, $T^{(\text{tard})}_i$ is positive. The target completion time of workorder o_i is

$$\hat{T}^{(C)}_{i,J_i} = T^{(A)}_{i,1} + K_i \sum_{j=1}^{J_i} \tilde{T}_{i,j}, K_i > 0 \tag{8}$$

2) REDUCING PRODUCTION COSTS

The profit margin comes from the difference between prices and costs. To increase profits, it is critical for manufacturers to reduce production costs, which are influenced by workloads, machining speed, and energy efficiency. Therefore, we formulate the profit reward as

$$R_P(b_{i,j}) = 1 - e^{K^{(E)}_m K^{(T)}_m - K^{(P)}_{i,j}} \tag{9}$$

where $K^{(P)}_{i,j}$ is a price factor for the job $b_{i,j}$, $K^{(E)}_m$ is the energy efficiency factor of a machine, and $K^{(T)}_m$ is the speed factor of the machine. The profits of a job $b_{i,j}$ is then formulated by

$$\begin{aligned} P_{i,j} &= f\left(T_{i,j}, K^{(P)}_{i,j}, K^{(E)}_m, K^{(T)}_m\right) \\ &= \hat{P}_{i,j} - E_{i,j} \\ &= K^{(P)}_{i,j} T_{i,j} - K^{(E)}_m K^{(T)}_m T_{i,j} \end{aligned} \tag{10}$$

where the price $\hat{P}_{i,j}$ is determined by the multiplication of $K^{(P)}_{i,j}$ and workload time $T_{i,j}$. The cost $E_{i,j}$ is defined as the multiplication of energy efficiency factor $K^{(E)}_m$, speed factor

$K_m^{(T)}$, and the workload time $T_{i,j}$. Note that the price factor $K_{i,j}^{(P)}$ increases if a workorder is urgent. Hence, the price factor $K_{i,j}^{(P)}$ is formulated with respect to the urgency factor K_i as

$$K_{i,j}^{(P)} = 3(e^{-\frac{3}{2}K_i}+1),\ 2/3 \leq K_i \leq 4/3 \tag{11}$$

A high-performance machine can finish jobs with less machining time (i.e., smaller $K_m^{(T)}$) and consume more energy per second (i.e., larger $K_m^{(E)}$). Thus, $K_m^{(E)}$ is formulated as a function of $K_m^{(T)}$ by Equation (12). The speed factor $K_m^{(T)}$ varies from 0.4 to 2.

$$K_m^{(E)} = 2e^{-\frac{1}{2}K_m^{(T)}},\ 0.4 \leq K_m^{(T)} \leq 2 \tag{12}$$

3) BALANCING WORKLOADS IN THE MANUFACTURING SYSTEM

A manufacturing shop includes different kinds of machines with varying performance levels. If the workload is imbalanced, there may be more workorders assigned to a specific machine. This will not only cause system congestion but also increase the probability of machine failures. There is an urgent need to improve the average utilization rate and balance workloads among machines. Therefore, we formulate the reward functions of machine utilization and workload balance as

$$R_U = \frac{1}{M_c}\sum_m u_m,\ \text{machine } m \in \text{type } c \tag{13}$$

$$R_V = e^{-U_c} \tag{14}$$

where M_c is the total number of machines of type c; u_m is the utilization rate of machine m in type c that is calculated as

$$u_m = \frac{1}{T}\sum_{i,j} K_m^{(T)} T_{i,j},\ \text{job } b_{i,j} \in \text{machine } m \tag{15}$$

where T denotes the duration of production, $K_m^{(T)}T_{i,j}$ is the machining time of job $b_{i,j}$ on machine m. The standard deviation of utilization rates U_c, formulated by Equation (16), is used as the metric of workload balance among machines.

$$U_c = \sqrt{\frac{1}{M_c}\sum_m (u_m - \bar{u})^2},\ \text{machine } m \in \text{type } c \tag{16}$$

where $\bar{u}$ is the mean of u_m's in the type c machine.

4) COMPOSITE REWARD MODELING

A composite reward function integrates multiple optimization objectives to make the AI scheduler more efficient, energy-saving, and robust. When a scheduler takes an action at time t, the composite reward R_{t+1} is a weighted function formulated as shown in Equation (17). The weighted average of N reward functions helps regulate the behaviors of an AI scheduler and account for multiple optimization objectives.

$$R_{t+1} = \sum_{i=1}^{N} w_i \cdot R_{t+1}^{(i)},\ \sum_{i=1}^{N} w_i = 1 \tag{17}$$

where $w_1, w_2, \ldots, w_N$ are the weights, and $R_{t+1}^{(1)}, R_{t+1}^{(2)}, \ldots, R_{t+1}^{(N)}$ are different reward functions. Specifically, this paper considers a composite reward function that includes R_D for tardiness, R_P for profits, R_U for machine utilization, and R_V for workload balance as shown in Equation (18) and Figure 6. At each scheduling moment, the reward R is derived from the statuses of jobs, workorders and machines between t and $t+1$.

$$R = w_1 \cdot R_D + w_2 \cdot R_P + w_3 \cdot R_U + w_4 \cdot R_V \tag{18}$$

The weights w_1, w_2, w_3, w_4 can be either uniform or adjusted for specific requirements. For example, if the management is more concerned about tardiness and utilization, then the weights of w_1 and w_3 can be increased for more contributions to the composite reward. As a result, the AI scheduler is more adaptive to new requirements and manufacturing conditions.

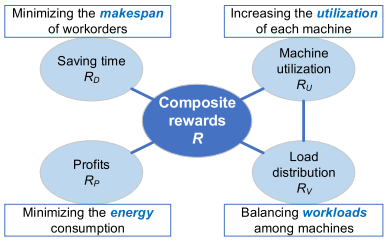

FIGURE 6. The components of composite rewards for multi-objective optimization.

C. THE AI SCHEDULER

The proposed AI scheduler leverages RL to formulate production scheduling into a multi-step decision problem. It is common that manufacturing shops face uncertainties and unexpected events (e.g., urgent workorders, machine failures). As such, schedules by traditional methods need to be updated and adjusted. For some time, rescheduling may cause disruptions to a manufacturing process. In the era of low-volume-high-mix manufacturing, online and dynamic scheduling becomes indispensable to gain competitive advantages in the market. In this investigation, transportation time and preparation time are assumed to be much smaller than machining time. However, this does not preclude others from relaxing the assumption and considering small time intervals for job transportation and preparation in the workshop. As shown in Figure 7, the AI scheduler is embodied with the following components:

Process planning: After a workorder is received from a customer, it will be decomposed into a series of jobs. The first job will be initialized after process planning. Job attributes are stored in the radio frequency identification (RFID) tag of the workorder. When a job is completed on a machine, the next job of the same workorder is initialized and waits to be scheduled. If there is no machine available for this job, the action a_0 (i.e., "waiting") will be chosen and wait for the next scheduling moment. As illustrated in Figure 5,

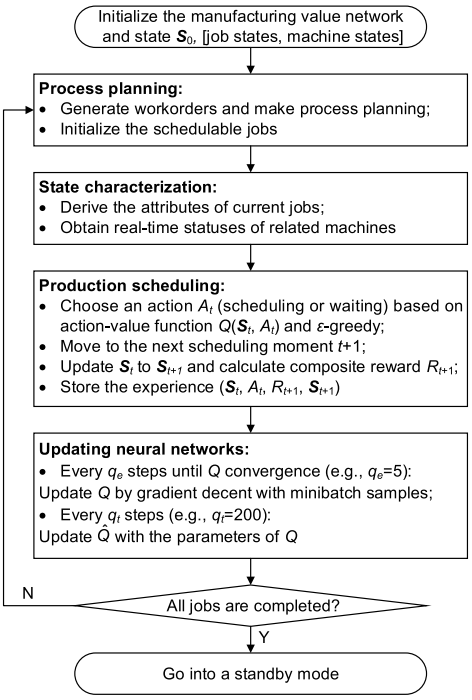

FIGURE 7. Online scheduling and learning procedures of the AI scheduler.

scheduling moments are marked by red circles when new jobs are initialized at $T_{i,j}^{(A)}$ or the workorder is completed at $T_{i,J_i}^{(C)}$. A completed workorder will be transported to the inventory without further scheduling.

State characterization: The job list for the AI scheduler contains all the initialized jobs, including the first job of each workorder in the inventory and the current unscheduled job of a workorder in the machine buffer. At each scheduling moment, the AI scheduler derives the attributes of the current schedulable job from its corresponding RFID tag and obtain the real-time statuses of related machines by sensors. As specified in Equation (2), the state dimension of the manufacturing shop is $d1 + d2 \cdot M$, which determines the input layer of the manufacturing value network.

Production scheduling: At scheduling moments, the AI scheduler makes scheduling policies for the current job based on real-time attributes of the job and related machines. After an action A_t is selected from the action space, the workorder will be transported to the target machine or wait for the next scheduling moment. States of machines are updated to S_{t+1} and a reward R_{t+1} for the state transition is given. Scheduling experiences (S_t, A_t, R_{t+1}, S_{t+1}) are stored for periodically training the manufacturing value network specified in Section III-A-2). Between two scheduling moments, the AI scheduler does nothing but monitors the system as machines process jobs.

Updating neural networks: The manufacturing value network is updated in smaller steps with higher frequency. For example, we can choose thirty samples from the experiences and update the parameters of the network every five steps until the convergence of Q function. Meanwhile, the manufacturing target network is trained in more steps with lower frequency. For example, we can update the target network with the parameters of manufacturing value network every two hundred steps. The training procedures based on RL are specified in Section III-A-2). If there is no job to do or all the workorders are completed, the AI scheduler and machines will be in a standby mode and take no action. If all machine buffers are occupied, schedulable jobs have to wait until a buffer is available (i.e., when a job of a workorder in the buffer is completed).

Figure 8 shows a real-time Gantt chart to illustrate the AI scheduler's actions. Each job is scheduled separately. If there are six machines in a manufacturing shop, i.e., three lathes and three millers. Each machine has a buffer with a capacity of four jobs. In the Gantt chart, the horizontal axis is a timeline and the vertical axis represents machines. Jobs that have been scheduled are marked by rectangles with black boundaries. Jobs to be scheduled are marked by red boundaries. The job list is below the timeline in Figure 8. Job $b_{1,1}$ is initialized at moment $T_{1,1}^{(A)}$. Job $b_{1,1}$ will be completed at moment $T_{1,1}^{(C)}$ on *Lathe* 1. The next job $b_{1,2}$ is created at moment $T_{1,2}^{(A)} = T_{1,1}^{(C)}$ after the completion of job $b_{1,1}$, but cannot be processed until previous jobs in the waiting buffer of *Miller* 2 are completed at moment $T_{1,2}^{(S)}$.

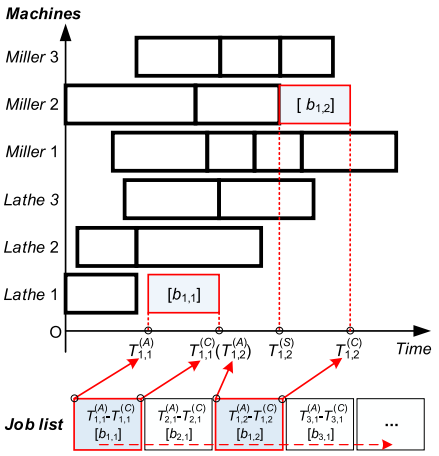

FIGURE 8. A Gantt chart for illustrating the relationships among scheduling moments, timeline, jobs, and machines.

IV. EXPERIMENTAL DESIGN

As shown in Figure 9, a smart factory testbed is established to verify the performances of different scheduling methods. There are three millers ($m = 1, 2, 3$), three lathes ($m = 4, 5, 6$), and a warehouse ($m = 0$) in the manufacturing shop. Each machine has a buffer area which can hold four workorders. An AGV transports workorders among the buffer areas, while two robots handle workorders between machines and their corresponding buffer areas. Workorders are generated in the cloud system and their attributes are written on corresponding RFID tags and updated by machines. The

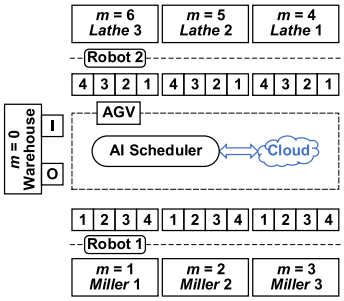

FIGURE 9. The layout of a smart factory testbed. "O" is the exit for the unprocessed workorders, and "I" is the entrance for the completed workorders.

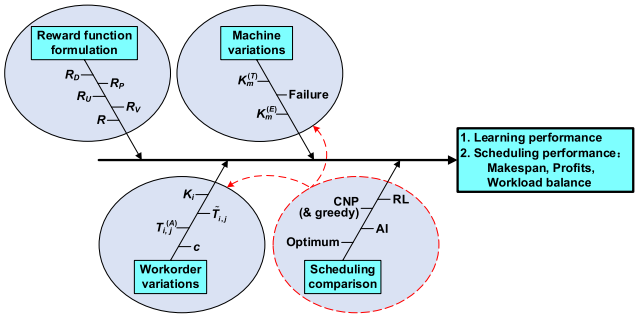

FIGURE 10. Design of experiments for performance evaluation of different AI scheduling models.

AI scheduler processes production scheduling based on real-time attributes of workorders and machines, and shares the statuses of the manufacturing shop with the cloud system.

As shown in Figure 10, four-factor layout experiments are designed to evaluate the performance of AI schedulers. The four-factor groups are described as follows:

Reward functions: We evaluate not only the composite reward R but also each component R_D, R_P, R_U, R_V to compare the learning performance and convergence speed of AI schedulers. In the experiments, we generate 1,000 random workorders to train the proposed AI scheduler.

Workorder variations: The workorders are also created with different attributes, e.g., job type c, initialization time $T_{i,j}^{(A)}$, nominal operating time $\tilde{T}_{i,j}$, urgency factor $K_i \in [2/3, 4/3]$. The workorders with various urgent levels are created to benchmark the performance of different scheduling methods.

Machine variations: The machines are also varied by machine type c, machining speed $K_m^{(T)}$, energy efficiency $K_m^{(E)}$ and failures. Machine failures cause uncertainty in the manufacturing processes. The machine variations are used to benchmark the performance (e.g., makespan, profits, workload balance) of different scheduling methods.

Scheduling methods: We have also benchmarked the performance of AI schedulers with a variety of traditional scheduling methods. The global optimal results are obtained by analytical methods assuming that the statuses of jobs and machines are known beforehand. A traditional RL scheduling method is also used to consider only a single objective, i.e., the energy consumption in Section V-B and the tardiness level R_D of machines in Section V-C. The CNP method is considered with greedy algorithms such as shortest waiting time first (SWTF), first come first serve (FCFS). At scheduling moments, CNP managers schedule workorders on available contractors with greedy algorithms.

The parameters of an AI scheduler are set as: learning rate α =0.01, discount rate of rewards γ =0.9, greedy rate ε =0.9. As shown in Table 3, there are six machines denoted by 1, 2, …, 6, three lathes and three millers. Each machine has a buffer with a capacity of four jobs. Machining speed (i.e., $(s_2)_m$) and energy efficiency (i.e., $(s_3)_m$) are static attributes for machines. The variations of machining speed factor $K_m^{(T)}$, and the energy efficiency factor $K_m^{(E)}$ are shown in Table 3. The state dimension for dynamic attributes of jobs and machines is 34, which is derived from Equation (2), where $d1$ =4, $d2$ =5, M = 6. The manufacturing value network has an input layer with 34 neurons, a hidden layer with 200 neurons and an output layer with 7 neurons. The workorder can be either shafts or panels in the experiments. A shaft is decomposed into two jobs of lathing and milling, and a panel has only one job of milling. Workorders arrive at the manufacturing shop at random time $T_{i,1}^{(A)}$. The nominal operating time of a milling job ranges from 15s to 90s and a lathing job is in the range from 60s to 180s. The AI scheduler monitors the statuses of jobs, workorders, and machines in real time. Each machine is equipped with an industrial computer for data collection and communication with the AI scheduler.

TABLE 3. The variations of machine types, speed factor, and energy efficiency factor.

m	c	$K_m^{(T)}$	$K_m^{(E)}$
1	1, lathe	0.5	1.6
2	1, lathe	1.0	1.2
3	1, lathe	2.0	0.8
4	2, milling	0.5	1.6
5	2, milling	1.0	1.2
6	2, milling	2.0	0.8

V. EXPERIMENTAL RESULTS

A. LEARNING PERFORMANCE OF AI SCHEDULERS WITH DIFFERENT REWARD FUNCTIONS

In this study, we evaluate the learning performance of the composite reward $R = w_1R_D + w_2R_P + w_3R_U + w_4R_V$ with different weight distributions (w_1, w_2, w_3, w_4), including (1, 0, 0, 0), (0, 1, 0, 0), (0, 0, 1, 0), (0, 0, 0, 1), and (0.25, 0.25, 0.25, 0.25). The weights (w_1, w_2, w_3, w_4) can be adjusted to meet different production requirements. For example, companies can increase the weights of w_2 and w_4 to save production costs, whereas they can increase the weights of w_1 and w_3 to complete workorders as soon as possible. In this current paper, we aim at proposing a general AI scheduler that can online achieve a good performance among minimizing makespan, maximizing profits, and balancing workloads.

Thus, we assign the weights with (0.25, 0.25, 0.25, 0.25) for the following experiments w.r.t. unexpected events (i.e., workorder and machine variations). The AI schedulers with five different types of reward functions are trained by interacting with the manufacturing shop. A total of 1,000 workorders, including 500 shafts and 500 panels, are generated at random with different nominal operating time and urgency factors. The ranges of job attributes are specified in Section IV. In the manufacturing shop, there are six machines, three lathes and three millers, and their attributes are detailed in Table 3. Figure 11 shows the training of AI schedulers with five different reward functions (i.e., time savings R_D, energy profits R_P, machine utilization R_U, workload distribution R_V, and composite reward R). As the AI schedulers interact with more and more workorders, the rewards per workorder increase with the training progresses. It is worth noting that the convergence rate of the composite reward is located in the middle. Although AI schedulers converge at different speeds for five reward functions, all of them converge to approximately 0.75 after the training with 900 workorders. The composite reward function, however, considers four different objectives simultaneously and does not show slower convergence than the schedulers with a single objective.

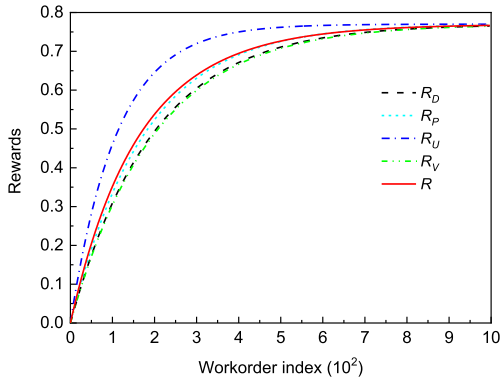

FIGURE 11. The performance comparison of AI schedulers with five different reward functions.

Figure 12 shows the comparison of learning curves, i.e., the differences between $\hat{y}$ and Q, derived from Equation (4) for AI schedulers with five different reward functions. Before 2,000 steps, the discrepancy between $\hat{y}$ and Q is large because AI schedulers are learning to handle random workorders. With more and more experiences accumulated, the decision-making capabilities improve significantly after 10,000 steps. As shown in Figure 12, AI schedulers with a single reward R_D, R_P, R_U, or R_V converge after 30,000 steps, and their convergence points are near step 22,500, step 27,500, step 26,000, step 17,000, respectively. Notably, the composite reward R converges after approximately 10,000 steps, which is much faster than other component reward functions. Also, when the reward functions are different, scheduling outcomes will vary. The next sections will show the composite reward yields better results than a single reward. If an AI scheduler with composite rewards needs fewer steps in the training process, it will learn faster and the time to deployment can be reduced significantly.

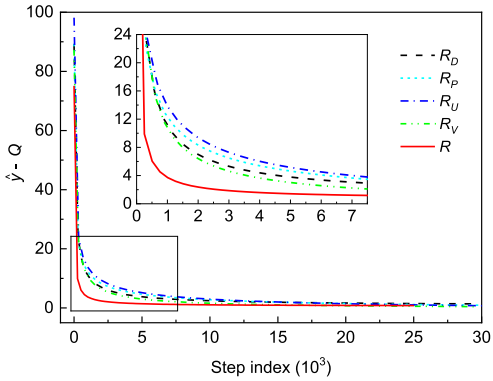

FIGURE 12. The comparison of learning curves for AI schedulers with five different reward functions.

B. SCHEDULING PERFORMANCE W.R.T. WORKORDER VARIATIONS

Although the AI scheduler is trained with a large number of workorders, there is a need to evaluate the scheduling performance with new workorders. As shown in Figure 13, four scheduling methods (i.e., optimum, AI, RL, and CNP, described in Section IV) are evaluated to handle 50 new workorders (i.e., urgency factor K_i =1) after the training. All workorders are generated before 600s. Although the profit curves fluctuate with the variation of workorders, four scheduling methods (i.e., optimum, AI, RL, and CNP) yield the profit mean and standard deviation as 110.8±5.06, 104.6±6.61, 98.6±11.94, 62.2±12.05, respectively. Note that "optimum" assumes that the statuses of jobs and machines are known beforehand, thus achieves the global optimal results. This is however unlikely to happen in the real-world manufacturing shop. The schedulers of AI, RL, CNP achieve 94.4%, 89.0%, 56.1% performance of the "optimum" scheduler. The AI scheduler is closer to the global optimal results than ordinary RL and CNP, due to the data utilization and experiences accumulated during the training. The AI scheduler and ordinary RL yield comparable performances in terms of profits, but ordinary RL has a relatively lower mean and a much bigger standard deviation (i.e., almost doubled from 6.61 to 11.94). On the contrary, the AI scheduler can not only earn more profits but also balance the profits among workorders. Also, the CNP scheduler is based on greedy algorithms and does not use the data, and thereby yields the inferior performance (i.e., lowest mean and highest standard deviation). The AI scheduler shows a stable profit curve to handle new workorders.

Then, the total number of workorders that are generated during the same period as the 50 workorders (i.e., 600s) varies from 10 to 100 at an interval of 10. The profit means of 10, 20, …, 100 workorders are shown in Figure 14. If only ten workorders are generated within 600s, the profit means are 119.4, 117.3, 104.6, and 86.7 for four scheduling methods of

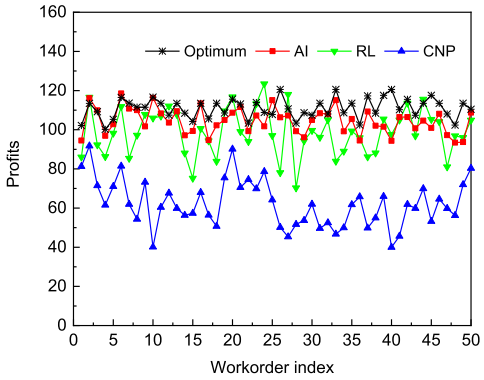

FIGURE 13. Profits from 50 new workorders by four scheduling methods (i.e., optimum, AI, RL, and CNP).

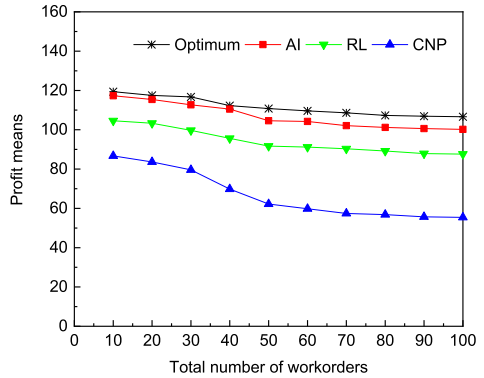

FIGURE 14. Profit means of different numbers of workorders that are generated during the same period of time.

optimum, AI, RL, and CNP. If 100 workorders are generated within 600s, the profit means decrease by 10.7%, 14.6%, 16.3%, and 36.1% respectively, and the AI scheduler earns 14.4% and 80.9% more profits than other online scheduling methods (i.e., RL and CNP). Therefore, the AI scheduler shows good performance in handling the variations in workorder numbers.

Further, Table 4 shows the comparisons of standard deviations of utilization rates U_c for lathes and millers after scheduling 50 random workorders. The AI scheduler leverages the composite reward function and thus yields the smallest standard deviations of utilization rates among machines. In other words, workloads are well balanced among machines by the AI scheduler. However, the "optimum" formulation focuses more on the minimization of makespan and the maximization of profits, but is less concerned about the balance of workloads among machines. Similarly, ordinary RL and

TABLE 4. The comparison of standard deviations of utilization rates among machines by four scheduling methods (i.e., optimum, AI, RL, and CNP).

Machine	Optimum	AI	RL	CNP
Lathe	0.366	0.312	0.474	0.576
Miller	0.258	0.223	0.348	0.445

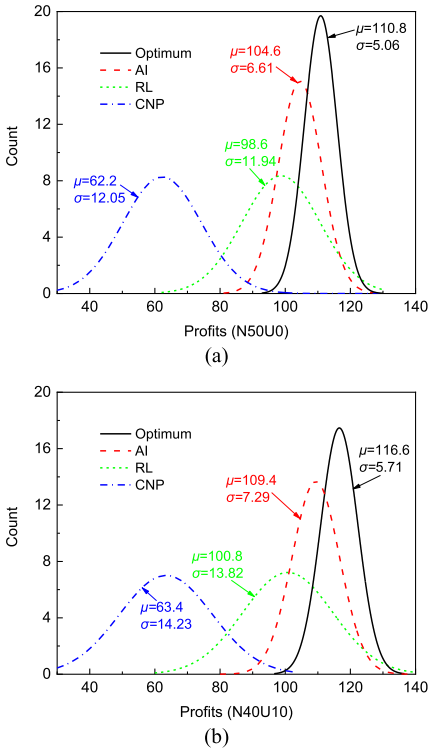

FIGURE 15. Profit distributions for four scheduling methods (i.e., optimum, AI, RL, and CNP) for two cases: (a) 50 normal workorders (N50U0); (b) 40 normal workorders and 10 urgent workorders (N40U10).

CNP schedulers do not specifically consider the workorder balancing, thereby yielding higher standard deviations.

In addition, it is not uncommon that the manufacturing shop receives urgent workorders. Therefore, we evaluate and compare the performance of these four schedulers with a mix of 40 normal workorder and 10 urgent workorders (N40U10). The normal workorder is with the urgency factor $K_i = 1$, and the urgent workorder is with the urgency factor $K_i = 2/3$. Figure 15 shows the comparison of scheduling results for N40U10, which are benchmarked with 50 normal workorders (N50U0). The profit means (μ) of the N40U10 case are 116.6, 109.4, 100.8, 63.4 for optimum, AI, RL, CNP, which are increased by 5.2%, 4.6%, 2.2%, 1.9% in comparison with the N50U0 case. The standard deviations of profits (σ) for the N40U10 case are 5.71, 7.29, 13.82, 14.23, which are increased by 12.8%, 10.3%, 15.7%, 18.1% in comparison with the N50U0 case. Note that the AI scheduler is much closer to the "optimum" than RL and CNP in terms of profit mean and standard deviation. The "optimum" is assumed to know the urgent workorders beforehand. Therefore, the AI scheduler yields higher profits and smaller deviations in the handling of urgent workorders.

C. SCHEDULING PERFORMANCE W.R.T. MACHINE VARIATIONS

For some time, machines fail and bring disruptions to the manufacturing process. Therefore, we compare the

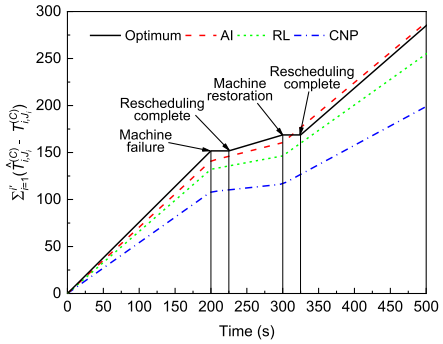

FIGURE 16. The comparison of cumulative ahead-of-schedule times by four scheduling methods (i.e., optimum, AI, RL, and CNP) under machine failures.

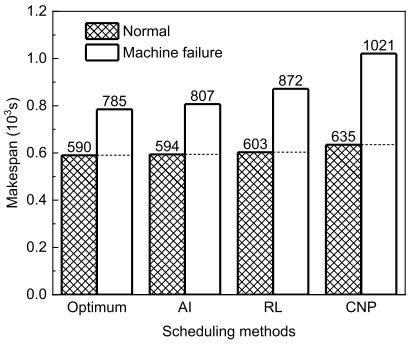

FIGURE 17. The comparison of makespans by four scheduling methods (i.e., optimum, AI, RL, and CNP) under both normal conditions and machine failures.

performance of the proposed AI scheduler with other scheduling methods under the condition of machine failures. Among six machines specified in Table 3, a miller breaks down during operations at 200s and the repair takes 100s to restore the operation. In other words, a miller fails between 200s and 300s in the manufacturing process. Four scheduling methods (i.e., optimum, AI, RL, and CNP) face the same machine failure and are used to schedule 50 workorders with the urgency factor $K_i = 1$. The performance metric is the cumulative ahead-of-schedule times, i.e., $\sum_{i=1}^{i'} \left(\hat{T}_{i,J_i}^{(C)} - T_{i,J_i}{}^{(C)}\right)$, $i = 1, \ldots, i'$ when $T_{i,J_i}^{(C)} < t$. In other words, at a particular time t, the performance metric counts how many workorders (i.e., i') have been completed and then computes the cumulative summation of the differences between estimated completion time and actual completion time of a workorder o_i.

As shown in Figure 16, the manufacturing system handles workorders that arrive at random. A miller fails at 200s, which demands scheduling methods to be adjusted for this new condition. The milling jobs should be rescheduled to functional millers. Because there are only two millers available, the cumulative ahead-of-schedule time does not increase as fast as before. Notably, the ''optimum'' assumes that the failure is known when it happens, rather than beforehand. Therefore, the ''optimum'' is delayed 25s to reschedule workorders based on the newly available conditions. When the miller restores the operation at 300s, it will then take another 25s to run the optimization algorithms for the global optimal solution. The AI, RL, and CNP schedulers respond to the failure and restoration in real time with minimal disruptions. The ''optimum'' can obtain optimal schedules for workorders, but at the expense of time delay for adjustment and rescheduling under machine failures. In this small-scale case study, ordinary RL and CNP yield worse performance than the AI scheduler because of the limited ability to utilize the data and perform multi-objective optimization. The AI scheduler achieves as good performance as the global optimal solutions and slows the ability to handle machine failures in real time.

Figure 17 shows the comparison of makespans for four scheduling methods (i.e., optimum, AI, RL, and CNP) under both normal conditions and machine failures. If a miller breaks down from 200s to 300s, the makespan is 785s, 807s, 872s, and 1,021s for optimum, AI, RL, and CNP, respectively. If all six machines are in good condition when processing 50 workorders, the makespan will be 590s, 594s, 603s, and 635s, respectively. The makespan is increased by 33.1%, 35.9%, 44.6%, 60.8% respectively when machine failures occur. Note that the AI scheduler yields a comparable performance with the optimum solutions. Note that there is only one miller failed in this experiment. The time delay is only 25s for the ''optimum'' to make adjustment and rescheduling. In a real-world workshop, there may be more failures and a bigger time delay for computation and update the schedule with offline methods. However, the AI scheduler can perform online learning in the process and deal with unexpected events (e.g., machine failures) in real time. Also, most traditional scheduling methods are designed with the primary concerns about workorder sequence, lead time, and makespan. Nonetheless, there is a need to consider not only makespan, time constraints, but also production costs, energy efficiency, and workload balancing. The AI scheduler realizes the full potentials of real-time data feeds, training experiences, and online learning for the generalization of new workorders and the effective handling of unexpected events (e.g., machine failures).

VI. CONCLUSION AND FUTURE WORK

This paper presents an AI scheduler for online and dynamic scheduling of manufacturing jobs in a smart factory. The RL method equips the proposed system with self-organizing and self-learning capabilities under uncertainty. A new formulation of composite reward function is developed to enable the AI scheduler for multi-objective learning and optimization of production schedules. A series of experiments are conducted to evaluate how different reward functions influence the scheduling results, including rewards for time savings R_D, energy profits R_P, machine utilization R_U, workload distribution R_V.

The proposed AI scheduler utilizes a manufacturing value network to estimate state-action values from high-dimensional sensor data of manufacturing things and then learns real-time policies according to the states of available machines and pending jobs. The ''smartness'' of AI schedulers is improved from streamed data feeds, training experiences, and online learning, which has shown strong potentials for the generalization of new workorders. It may be noted that the proposed method is capable of handling simultaneously created workorders and uncertainty factors such as machine failures. The proposed composite reward function effectively tackles the urgent workorders, while maintaining a balance between efficiency and profits. The proposed methodology is evaluated and validated with experimental studies in a smart manufacturing setting. Experimental results show that the new AI scheduler not only improves the multi-objective performance metrics in the production scheduling problem but also effectively copes with unexpected events (e.g., urgent workorders, machine failures) in manufacturing systems.

Further, there are more challenges to be considered in future research, such as AGV route planning and manufacturing process planning in the smart factory. Future investigations will focus on the reward-scheduling mechanisms, computational efficiency of AI schedulers, distributed learning with AI agents resides in each manufacturing thing. This paper makes an attempt to improve the AI for smart manufacturing, but realizing a smart factory depends on the significant improvements of ubiquitous ''smartness'' in manufacturing things that span in every corner of the manufacturing factory. We hope this work will help catalyze more in-depth investigations and multi-disciplinary research efforts to advance the AI for smart manufacturing.

REFERENCES

[1] R. W. Conway, W. L. Maxwell, and L. W. Miller, *Theory of Scheduling*. Reading, MA, USA: Addison-Wesley, 1967.

[2] M. Pinedo, ''Scheduling: Theory, algorithms and systems development,'' in *Operations Research Proceedings*. Berlin, Germany: Springer, 2012.

[3] J. Blazewicz, M. Dror, and J. Weglarz, ''Mathematical programming formulations for machine scheduling: A survey,'' *Eur. J. Oper. Res.*, vol. 51, no. 3, pp. 283–300, Apr. 1991.

[4] S. R. Kumara, S. Joshi, R. Kashyap, C. Moodie, and T. Chang, ''Expert systems in industrial engineering,'' *Int. J. Prod. Res.*, vol. 24, no. 5, pp. 1107–1125, 1986.

[5] A. Ławrynowicz, ''Integration of production planning and scheduling using an expert system and a genetic algorithm,'' *J. Oper. Res. Soc.*, vol. 59, no. 4, pp. 455–463, Apr. 2008.

[6] R. E. Bellman, A. O. Esogbue, and I. Nabeshima, *Mathematical Aspects of Scheduling and Applications*. Oxford, U.K.: Pergamon Press, 1982.

[7] K. Reisen, U. Teschemacher, M. Niehues, and G. Reinhart, ''Biomimetics in production organization—A literature study and framework,'' *J. Bionic Eng.*, vol. 13, no. 2, pp. 200–212, Jun. 2016.

[8] H.-S. Park and N.-H. Tran, ''An autonomous manufacturing system based on swarm of cognitive agents,'' *J. Manuf. Syst.*, vol. 31, no. 3, pp. 337–348, Jul. 2012.

[9] A. Negahban and J. S. Smith, ''Simulation for manufacturing system design and operation: Literature review and analysis,'' *J. Manuf. Syst.*, vol. 33, no. 2, pp. 241–261, Apr. 2014.

[10] H. Yang, S. Kumara, S. T. S. Bukkapatnam, and F. Tsung, ''The Internet of Things for smart manufacturing: A review,'' *IISE Trans.*, vol. 51, no. 11, pp. 1190–1216, 2019.

[11] C. S. Chong, A. Iyer Sivakumar, and R. Gay, ''Simulation-based scheduling for dynamic discrete manufacturing,'' in *Proc. Int. Conf. Mach. Learn. Cybern.*, vol. 2, 2003, pp. 1465–1473.

[12] J.-Y. Bang and Y.-D. Kim, ''Hierarchical production planning for semiconductor wafer fabrication based on linear programming and discrete-event simulation,'' *IEEE Trans. Autom. Sci. Eng.*, vol. 7, no. 2, pp. 326–336, Apr. 2010.

[13] W. Shen and D. H. Norrie, ''Agent-based systems for intelligent manufacturing: A state-of-the-art survey,'' *Knowl. Inf. Syst.*, vol. 1, no. 2, pp. 129–156, May 1999.

[14] N. R. Jennings and M. Wooldridge, ''Applications of intelligent agents,'' in *Agent Technology*. Berlin, Germany: Springer, 1998, pp. 3–28.

[15] Y. H. Lee, S. R. T. Kumara, and K. Chatterjee, ''Multiagent based dynamic resource scheduling for distributed multiple projects using a market mechanism,'' *J. Intell. Manuf.*, vol. 14, no. 5, pp. 471–484, Oct. 2003.

[16] W. L. Yeung, ''Agent-based manufacturing control based on distributed bid selection and publish-subscribe messaging: A simulation case study,'' *Int. J. Prod. Res.*, vol. 50, no. 22, pp. 6339–6356, Nov. 2012.

[17] Y. Zhang, J. Wang, and Y. Liu, ''Game theory based real-time multi-objective flexible job shop scheduling considering environmental impact,'' *J. Cleaner Prod.*, vol. 167, pp. 665–679, Nov. 2017.

[18] L.-C. Wang, C.-Y. Cheng, and S.-K. Lin, ''Distributed feedback control algorithm in an auction-based manufacturing planning and control system,'' *Int. J. Prod. Res.*, vol. 51, no. 9, pp. 2667–2679, May 2013.

[19] C.-Y. Hsu, B.-R. Kao, V. L. Ho, and K. R. Lai, ''Agent-based fuzzy constraint-directed negotiation mechanism for distributed job shop scheduling,'' *Eng. Appl. Artif. Intell.*, vol. 53, pp. 140–154, Aug. 2016.

[20] 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, Feb. 2005.

[21] M. A. Salido, J. Escamilla, F. Barber, and A. Giret, ''Rescheduling in job-shop problems for sustainable manufacturing systems,'' *J. Cleaner Prod.*, vol. 162, pp. S121–S132, Sep. 2017.

[22] F. Yalaoui and C. Chu, ''An efficient heuristic approach for parallel machine scheduling with job splitting and sequence-dependent setup times,'' *IIE Trans.*, vol. 35, no. 2, pp. 183–190, Feb. 2003.

[23] M. Masin, M. O. Pasaogullari, and S. Joshi, ''Dynamic scheduling of production-assembly networks in a distributed environment,'' *IIE Trans.*, vol. 39, no. 4, pp. 395–409, May 2007.

[24] L. Zhang, X. Li, L. Gao, and G. Zhang, ''Dynamic rescheduling in FMS that is simultaneously considering energy consumption and schedule efficiency,'' *Int. J. Adv. Manuf. Technol.*, vol. 87, nos. 5–8, pp. 1387–1399, Nov. 2016.

[25] D. Ouelhadj and S. Petrovic, ''A survey of dynamic scheduling in manufacturing systems,'' *J. Scheduling*, vol. 12, no. 4, pp. 417–431, Aug. 2009.

[26] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, ''Mastering the game of go with deep neural networks and tree search,'' *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[27] Z. Zhang, L. Zheng, N. Li, W. Wang, S. Zhong, and K. Hu, ''Minimizing mean weighted tardiness in unrelated parallel machine scheduling with reinforcement learning,'' *Comput. Oper. Res.*, vol. 39, no. 7, pp. 1315–1324, Jul. 2012.

[28] J. Shahrabi, M. A. Adibi, and M. Mahootchi, ''A reinforcement learning approach to parameter estimation in dynamic job shop scheduling,'' *Comput. Ind. Eng.*, vol. 110, pp. 75–82, Aug. 2017.

[29] 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, pp. 604–614, Nov. 2018.

[30] S. Chen, S. Fang, and R. Tang, ''A reinforcement learning based approach for multi-projects scheduling in cloud manufacturing,'' *Int. J. Prod. Res.*, vol. 57, no. 10, pp. 3080–3098, May 2019.

[31] Y. Wang, H. Liu, W. Zheng, Y. Xia, Y. Li, P. Chen, K. Guo, and H. Xie, ''Multi-objective workflow scheduling with deep-Q-network-based multi-agent reinforcement learning,'' *IEEE Access*, vol. 7, pp. 39974–39982, 2019.

[32] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[33] C. J. C. H. Watkins, ''Learning from delayed rewards,'' Ph.D. dissertation, King's College London, London, U.K., 1989.

TONG ZHOU was born in Xuzhou, Jiangsu, China, in 1991. He received the B.S. and M.S. degrees in mechanical engineering and automation from the Jiangsu University of Science and Technology, Zhenjiang, Jiangsu, in 2014 and 2017, respectively. He is currently pursuing the Ph.D. degree in mechanical and electrical engineering with the Nanjing University of Aeronautics and Astronautics, Nanjing, Jiangsu. His research interests include machine learning, the Internet of Things, and scheduling of manufacturing systems.

DUNBING TANG was born in Xiantao, Hubei, China, in 1972. He received the Ph.D. degree in mechanical engineering and automation from the Nanjing University of Science and Technology, Nanjing, Jiangsu, China, in 2000.

From 2000 to 2002, he did his Postdoctoral Research at Tsinghua University, Beijing, China. From 2002 to 2004, he was a Humboldt Research Fellow with RWTH Aachen University, Aachen, Germany. He was a Research Fellow with Cranfield University, Bedford, U.K., in 2005. Since 2005, he has been a Professor with the College of Mechanical and Electrical Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing. His research interests include smart manufacturing systems, complex system modeling, and open design.

HAIHUA ZHU was born in Ningbo, Zhejiang, China, in 1985. He received the Ph.D. degree in mechanical engineering and automation from the Nanjing University of Science and Technology, Nanjing, Jiangsu, China, in 2013.

From 2009 to 2011, he was a Research Scholar with the University of Greenwich, London, U.K. He is currently an Associate Professor with the College of Mechanical and Electrical Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing. His research interests include smart manufacturing systems, digital twin, and the Internet of Things.

LIPING WANG was born in Rizhao, Shandong, China, in 1990. He received the M.S. degree in mechanical engineering from the Nanjing University of Aeronautics and Astronautics, Nanjing, Jiangsu, China, in 2018, where he is currently pursuing the Ph.D. degree with the College of Mechanical and Electrical Engineering. His research interest includes multi-agent collaboration in a smart factory.