Robotics and Computer–Integrated Manufacturing 72 (2021) 102202

Contents lists available at ScienceDirect

Robotics and Computer-Integrated Manufacturing

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

Multi-agent reinforcement learning for online scheduling in smart factories

Tong Zhou, Dunbing Tang *, Haihua Zhu, Zequn Zhang

College of Mechanical and Electrical Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China

ARTICLE INFO

Keywords:
Online scheduling
Smart factory
Composite reward
Multi-agent system
Reinforcement learning

ABSTRACT

Rapid advances in sensing and communication technologies connect isolated manufacturing units, which generates large amounts of data. The new trend of mass customization brings a higher level of disturbances and uncertainties to production planning. Traditional manufacturing systems analyze data and schedule orders in a centralized architecture, which is inefficient and unreliable for the overdependence on central controllers and limited communication channels. Internet of things (IoT) and cloud technologies make it possible to build a distributed manufacturing architecture such as the multi-agent system (MAS). Recently, artificial intelligence (AI) methods are used to solve scheduling problems in the manufacturing setting. However, it is difficult for scheduling algorithms to process high-dimensional data in a distributed system with heterogeneous manufacturing units. Therefore, this paper presents new cyber-physical integration in smart factories for online scheduling of low-volume-high-mix orders. First, manufacturing units are interconnected with each other through the cyber-physical system (CPS) by IoT technologies. Attributes of machining operations are stored and transmitted by radio frequency identification (RFID) tags. Second, we propose an AI scheduler with novel neural networks for each unit (e.g., warehouse, machine) to schedule dynamic operations with real-time sensor data. Each AI scheduler can collaborate with other schedulers by learning from their scheduling experiences. Third, we design new reward functions to improve the decision-making abilities of multiple AI schedulers based on reinforcement learning (RL). The proposed methodology is evaluated and validated in a smart factory by real-world case studies. Experimental results show that the new architecture for smart factories not only improves the learning and scheduling efficiency of multiple AI schedulers but also effectively deals with unexpected events such as rush orders and machine failures.

1. Introduction

The new mass customization generates large numbers of low-volume-high-mix orders, which requires the manufacturing systems to be more flexible to handle the variations in design specifications. Traditional factories are inadequate for fluctuating markets with a lot of uncertainties. Smart factories integrate heterogeneous manufacturing resources (e.g., material, orders, and machines) based on internet of things (IoT) technologies and coordinate these resources by scheduling algorithms. The statuses of resources are derived by sensors and transmitted by the internet in real time. As a result, the smart factory turns into a data-rich environment, which provides an unprecedented opportunity to improve the "smartness" of factories. However, big data from smart factories are high-dimensional and updated dynamically due to resource complexity and unexpected disturbances (e.g., material shortages, rush orders, and machine failures). Traditional offline scheduling algorithms are implemented in fixed procedures with determined attributes, so they are ineffective in handling high-dimensional data from a dynamic environment.

The "smartness" of a factory can be improved from two aspects, including architectural redesign and scheduling optimization. Rapid advances of IoT promote the development of technologies for data sensing, transmission, and analysis. Different architectures for smart factories are proposed to derive sensor data from machines and orders. For example, a factory can be divided into different functional layers such as execution, adaptation, communication, and computation. Big data are accumulated and analyzed by the computation layer to monitor the factory or update scheduling policies made by simulation-based methods. However, the centralized architecture is ineffective in data processing when large numbers of operations are initialized simultaneously. Therefore, the distributed architecture is proposed to decompose the tasks of the central computing layer and assign them to different computing units in a manufacturing workshop. To handle uncertainties in the low-volume-high-mix manufacturing setting, early studies periodically reschedule resources to achieve new optimal

* Corresponding author.
E-mail addresses: jayantzh@outlook.com (T. Zhou), d.tang@nuaa.edu.cn (D. Tang).

https://doi.org/10.1016/j.rcim.2021.102202
Received 13 November 2020; Received in revised form 30 May 2021; Accepted 31 May 2021
Available online 12 June 2021

Notations

I	total number of work orders
o_i	ith work order ($i = 1, \ldots, I$)
J_i	total number of operations of work order o_i
$b_{i,j}$	jth operation of work order o_i ($j = 1, \ldots, J_i$)
M	total number of machines
m	machine number ($m = 1, \ldots, M$)
B_m	buffer length of machine m
N_m	operator number of the AI scheduler of machine m
$T_{i,j}^{(A)}$	initialization time of operation $b_{i,j}$
$T_{i,j}^{(S)}$	starting time for performing operation $b_{i,j}$
$T_{i,j}^{(C)}$	completion time of operation $b_{i,j}$
$\widehat{T}_{i,j}^{(C)}$	target completion time of operation $b_{i,j}$
$T_{i,j}$	workload time of operation $b_{i,j}$
$\widetilde{T}_{i,j}$	nominal operating time of operation $b_{i,j}$
$T_m^{(W)}$	current waiting time of machine m
$T_m^{(V)}$	transportation time to machine m
$u_m(t)$	utilization time of machine m before time t
$\|M\|_m$	total number of optional machines for AI scheduler m
$\boldsymbol{\mu}$	continuous policies for M AI schedulers
$\boldsymbol{\theta}$	parameters for scheduling policy networks
$\boldsymbol{\omega}$	parameters for manufacturing value networks
$\boldsymbol{s}$	states of all AI schedulers
$\boldsymbol{s}_m$	states of the AI scheduler of machine m
$\boldsymbol{a}$	actions of all AI schedulers
$\boldsymbol{a}_m$	actions of the AI scheduler of machine m
$\boldsymbol{r}$	composite reward for a state transition
r_m	composite reward for the AI scheduler of machine m
$r^{(D)}$	reward for operation sequence considering urgency
$r^{(W)}$	reward for waiting-time rates
$r^{(U)}$	reward for machine-utilization rates
w_1	weight for reward $r^{(D)}$
w_2	weight for reward $r^{(W)}$
w_3	weight for reward $r^{(U)}$

schedules in an offline manner, which needs extra computing time. Cloud and sensing technologies connect heterogeneous manufacturing resources (e.g., orders, machines, warehouses, and material handling equipment) and realize online scheduling in a simulation way, e.g., multi-agent system (MAS). However, most simulation-based methods are limited in the ability to use real-time sensor data for online scheduling. Recently, artificial intelligence (AI) arouses an increasing interest in solving dynamic scheduling problems by learning from data and accumulated experiences. However, it is difficult to utilize high-dimensional sensor data for production scheduling, especially considering multiple objectives in a distributed architecture.

This paper presents a new architecture for cyber-physical integration to implement data-driven online scheduling by coordinating multiple AI schedulers in smart factories. We design an AI scheduler for each physical unit in a smart factory to schedule orders based on real-time statuses of operations and machines. Decision-making algorithms run on distributed computing units in the workshop rather than the central server, which reduces unnecessary communication and improves scheduling efficiency. Each AI scheduler has four novel neural networks that take high-dimensional data for online decision making. New composite reward functions are designed to help AI schedulers optimize multiple objectives such as minimizing makespan and balancing workloads. To achieve global optimization in a dynamic environment with multiple decision makers, the proposed AI scheduler learns from not only its own operating data but also the experiences of other AI schedulers.

The proposed methodology is implemented in a smart factory and evaluated with a series of real-world experiments. We benchmark the performance of multiple AI schedulers with a variety of common scheduling methods such as metaheuristic methods (e.g., genetic algorithm (GA)), contract net protocol (CNP), and centralized reinforcement learning (RL). Experimental results show that the new manufacturing architecture with AI schedulers can not only improve online scheduling performances for orders but also effectively handle unexpected events (e.g., rush orders, machine failures) in real time. The novel methodology for cyber-physical integration can be generally implemented in most manufacturing systems to make factories smarter in the low-volume-high-mix manufacturing setting.

The remaining sections of this paper are organized as follows. Section 2 presents a review of relevant literature in building and scheduling a manufacturing workshop. Section 3 presents the proposed methodology of cyber-physical integration in a smart factory to implement the collaboration of multiple AI schedulers based on RL. Section 4 designs a testbed to evaluate the performance of the smart factory with AI schedulers. Section 5 shows experimental results and makes comparisons with different methods for operating a smart factory. Section 6 rounds up the paper and discusses future researches on potential topics.

2. Research background

2.1. Manufacturing system architecture

A manufacturing system includes various equipment (e.g., machines, warehouses, and material handlers) to complete orders according to design requirements. We define the "work order" as a customized order which contains only one part, i.e., a work order represents an individualized part. A work order, containing one or more operations, is stored in the inventory (i.e., warehouse) and then handled among machines by the material handling system (MHS). The architecture design for a manufacturing system includes unit (i.e., machines, MHS) layout sketching, work order tracking, and communication protocols. Typically, an effective architecture helps improve the operating performances of a manufacturing workshop by saving time, reducing costs, and balancing workloads.

In the early years, a machine is independently operated by workers and has few connections with other equipment. The rapid development of sensor and communication technologies enables machines to automatically perform operations and work together to effectively complete work orders. Harriongton [1] proposed computer-integrated manufacturing (CIM) by integrating discrete manufacturing resources to improve the overall performance of a workshop. The traditional centralized architecture classifies a manufacturing system into different functional layers and operates the workshop by hierarchical organizations [2]. These centralized organizations limit the flexibility for expandability and reconfiguration of a manufacturing system. In recent years, the distributed manufacturing system has attracted increasing attention because of its effectiveness in responding to the turbulent manufacturing environment [3]. Distributed manufacturing systems are comprised of a set of autonomous units that make local decisions and communicate with other components to complete tasks without any central controllers [4]. Usually, distributed manufacturing systems require a supervisor to coordinate intelligent units for achieving global objectives [5]. The agent technology provides a natural way to model units in a distributed manufacturing system as distinctive modules such as machining agent, material handling agent, and inventory management agent [6]. An agent-based system is built by the distributed

architecture, where each agent handles the schedule of its own machine, material handlers, or inventory. IoT technologies connect all manufacturing resources and drive a factory to become a new generation of cyber-physical systems towards cloud manufacturing [7]. Wang [8] introduced a new cyber-physical system to remotely monitor and operate a workshop by a data-driven model. Liu et al. [9] built a virtual mirror (i.e., digital twin) model of the real-world workshop, and operated the event-driven model in the cloud to schedule, monitor, and analyze the manufacturing system. Lu et al. [10] proposed a generic system architecture for cloud-based manufacturing equipment towards service offering in the cloud and discussed how the research was implemented in industry. Although the cloud-based architecture can effectively handle big data, the transmission of large amounts of data could be a heavy load for the networks and may expose private details. Edge computing processes data at the edge of system networks to reduce data-transmission time, so most events can be handled immediately in the workshop without waiting for the response from the central scheduler in the cloud [11].

The architecture of a manufacturing system evolves from discrete and isolated operation to centralized integration and then to distributed self-organization. Distributed manufacturing and multi-agent systems are widely studied in recent years, but related technologies are rarely applied in industrial conditions. Notably, the distributed architecture is flexible in configuring units and effectively processing operations by combing agent-based or edge-computing methods. Data in smart factories are high-dimensional and vary dynamically. However, traditional distributed architectures analyze data either in the cloud or at the network edge and are limited in the ability to realize the full potential of big data to improve the scheduling ability of each manufacturing unit.

2.2. Production scheduling

Production scheduling optimizes the allocation of manufacturing units to fulfill work orders received in a manufacturing workshop [12]. Each work order is composed of a set of operations, and each operation should be performed on a suitable machine according to factory statuses, design specifications, and customers' requirements. Scheduling algorithms can be operated in a manufacturing system architecture to achieve optimization objectives such as minimizing makespan, balancing workloads, and saving resources.

The computing complexity of a production scheduling problem increases exponentially with respect to operation and machine sizes. Conway et al. [13] pioneered the study of production scheduling and contributed to the modeling and optimization of manufacturing systems. Mathematical programming approaches achieve global optimal schedules by getting critical solutions of programming functions under constraints [14]. To model and solve an optimization model with large variable space, metaheuristic methods (e.g., GA [15], swarm intelligence [16]) are introduced to simulate manufacturing systems with simple and natural rules. In the past few decades, the manufacturing manner has been transformed from large-batch to low-volume-high-mix production. Traditional offline scheduling methods are limited in handling uncertainties in the real-world environment. Thus, the knowledge-based method (e.g., expert system [17]) is proposed to model a manufacturing system with human expertise, which is a primitive form of AI. In a distributed architecture with multiple agents, each agent is a computer system that can process data and schedule work orders [18]. Manufacturing resources in an MAS can be coordinated by socialized interactive rules, such as game theory [19], CNP [20], and auction mechanism [21]. Computer simulation is increasingly used to improve the decision-making abilities of a production scheduling model. For example, discrete-event simulation (DES) models a manufacturing system with accumulated operating data [22]. Rauf et al. [23] investigated the multi-criteria scheduling problem using a modified simulation method and evaluated each solution by DES. Recently, AI fuels an increasing interest to solve dynamic scheduling problems by continuously improving the scheduling performance of schedulers. Morariu et al. [24] used long short-term memory (LSTM) neural networks and deep learning to reschedule resources and detect anomalies in real time based on predicted energy data. This research provides a systematic approach to collecting data at the edge of workshops and processing data at a centralized cloud platform for dynamic prediction and scheduling.

From rule-based dispatching to intelligent decision-making, production scheduling methods are gradually achieving better performance in handling dynamic work orders. There are lots of uncertainties in dynamically scheduling low-volume-high-mix work orders, which requires the scheduling system to handle unexpected events (e.g., work order variations, machine failures, material shortage) in real time. When disturbances occur in a manufacturing system, offline scheduling methods take extra time to reschedule the unprocessed work orders for achieving a new optimal solution. A simulation-based scheduling model is built according to historical data and empirical expertise, which is limited in handling real-time sensor data from the smart factory. The naive AI method builds a smart scheduler in a centralized architecture to achieve a single objective, but it is still difficult to leverage real-time data in a distributed architecture and continuously improve the performance of each scheduler.

2.3. Smart scheduling for uncertainties

A smart factory is a dynamic environment with uncertainties. Thus, original schedules will become invalid when some unexpected events occur in the manufacturing system [25]. It is common that a real-world manufacturing system experiences uncertainties, including unpredictable external occurrences (e.g., market changes, supply chain fluctuations), planned product model changes (e.g., work order variations), and unexpected intrinsic system events (e.g., machine failures or variations) [26]. Dynamic scheduling methods are aimed at handling unexpected events in real time [27]. For example, if a machine breaks down during operation, the unprocessed work orders assigned to the failed machine will be rescheduled to other machines.

Traditional offline scheduling handles unexpected events in workshops by adjusting the primitive schedules with extra computing time [28]. Simulation-based methods periodically update the scheduling model with accumulated operating data [29]. Ghaleb et al. [30] built a real-time scheduling model to generate new schedules (i.e., rescheduling) by utilizing real-time workshop updates such as unexpected arrivals, the availability of machines (downtime and recovery time), and the completion time of operations. Zhou et al. [31] presented a knowledge graph-driven approach for scheduling optimization, allowing fast decision making for order inserting requests from the knowledge level. The MAS is popular for dynamic scheduling and can be applied in a distributed architecture, but a commonly rule-based MAS has limited decision-making and self-improving abilities due to the lack of AI algorithms. AI methods not only schedule work orders in real time but also provide attractive features such as self-organizing operation and online learning [32]. These AI algorithms equip schedulers with the abilities of online scheduling and learning to deal with uncertainties in a dynamic environment. RL is a machine learning method that helps an agent learn to select optimal actions for achieving long-term objectives through trial-and-error interactions with dynamic environments [33]. Four key elements are defined for dealing with an RL problem: a policy for defining the agent's behavior, a reward function for achieving objectives, a value function for evaluating the state or state-action values, and an environmental model for implementing state transitions. Zhang et al. [34] applied RL methods to schedule a set of tasks to minimize the makespan while satisfying temporal and resource constraints, but these methods are less concerned about other objectives and disturbances. Chen et al. [35] presented a rule-driven method to solve multi-objective dynamic scheduling problems considering composite dispatching rules. Zhang et al. [36] applied an RL method to solve a dynamic unrelated

parallel machine scheduling problem considering a mean weighted tardiness objective. Mannion et al. [37] built an MAS to optimize systems with respect to multiple objectives. In this MAS, each agent is established by Q-learning [38] and all agents cooperate to reach an optimal joint policy by a Nash equilibrium [39], which wastes many computing resources at each action moment. RL scheduling models can be established in the cloud system and interact with the manufacturing workshop via the internet. Rjoub et al. [40] combined deep RL with LSTM in a cloud-computing environment to predict the appropriate virtual machines that should host each incoming task.

Heterogeneous units are common in a manufacturing system, especially in the smart factory with a distributed architecture. Data-rich environments in smart factories provide an opportunity to build dynamic scheduling models for dealing with uncertainties. Traditional RL methods operate scheduling agents on a centralized computer or in the cloud, which tends to be ineffective in data transmission and utilization when scheduling a large number of low-volume-high-mix work orders. Furthermore, it is difficult for a single scheduler to improve the decision-making abilities for scheduling various work orders on different machines in a short time. In addition, most RL algorithms for MAS are implemented with simulation models and driven by events instead of real-time data from smart factories. There is an urgent need to equip each unit with an AI scheduler to schedule corresponding work orders based on its own observations of high-dimensional data. Additionally, a new distributed architecture is indispensable for operating multiple AI schedulers that cooperate to achieve joint objectives by sharing real-time data.

3. Research methodology

3.1. The architecture of a smart factory

3.1.1. Cyber-physical architecture

A cyber-physical system connects all heterogeneous units and transmits data between the cloud and workshops. The proposed architecture is shown in Fig. 1 to illustrate the components and interconnections of the smart factory. IoT connects all units through the local area network (LAN) and shares data with the cloud. The smart factory is composed of machines, material handlers (e.g., auto-guided vehicle (AGV), robot), a warehouse, and a supervisor. Work order generating and process planning are operated in the cloud where some cyber modules (e.g., digital model, database, order preprocessing) are designed to monitor and analyze the manufacturing system.

In the smart factory, each unit is equipped with an industrial computer (IPC) that contains three layers, including an adaptation layer for command sending, a communication layer for data exchanging, and an AI scheduler for online scheduling. The AI scheduler is a smart agent that works with a corresponding unit through the adaptation layer and interacts with other units through the communication layer. Each work order is placed on a pallet with a unique RFID tag where the operation attributes of the current work order are written. RFID readers are fixed on units to track work orders and update the data of operation attributes on RFID tags. The states of each unit are derived from sensors in real time and can be shared among all related units. Hubs are placed on each unit to connect its corresponding devices (e.g., IPC, RFID reader, controller) and communicate with other units or the cloud. All units can directly exchange data with each other in real time through the IoT. A supervisor is designed for deriving and storing real-time data of the whole factory. The supervisor derives the operating data of all manufacturing units to evaluate the scheduling performance of AI schedulers. Only decision-making networks operate on distributed IPCs for online scheduling work orders according to AI schedulers' own observations, which is efficient in scheduling manufacturing resources. All neural networks operate on the supervisor for helping AI schedulers collaborate in improving their decision-making abilities. Simulation models (e.g., digital models), driven by real-time data of the factory, are operated in the cloud to monitor the factory and evaluate the manufacturing performance. The digital models monitor and optimize the manufacturing processes of the smart factory by cloud-edge collaboration. Work orders are generated by customers randomly with various attributes. Process planning and time prediction are operated in the cloud before work orders are scheduled by the inventory that is responsible for deriving work orders from the cloud and scheduling the first operation of each work order.

3.1.2. Product tracking and data flowing

In a smart factory, work orders should be tracked by the scheduling system for making advisable schedules. The operating processes are driven by operation attributes, and large amounts of data are perceived and stored for analysis as they flow in the system. The procedures of product tracking and data flowing are shown in Fig. 2. Three types of work orders, including flanges, shafts, and panels, are chosen as examples to be processed in the smart factory. Operating procedures of a work order include process planning, work order tracking, and data

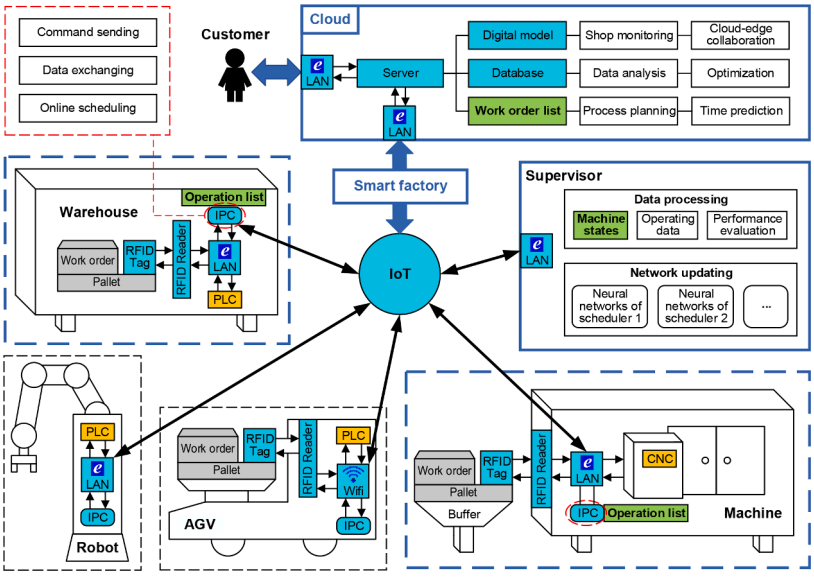

Fig. 1. The proposed architecture for cyber-physical integration in a smart factory.

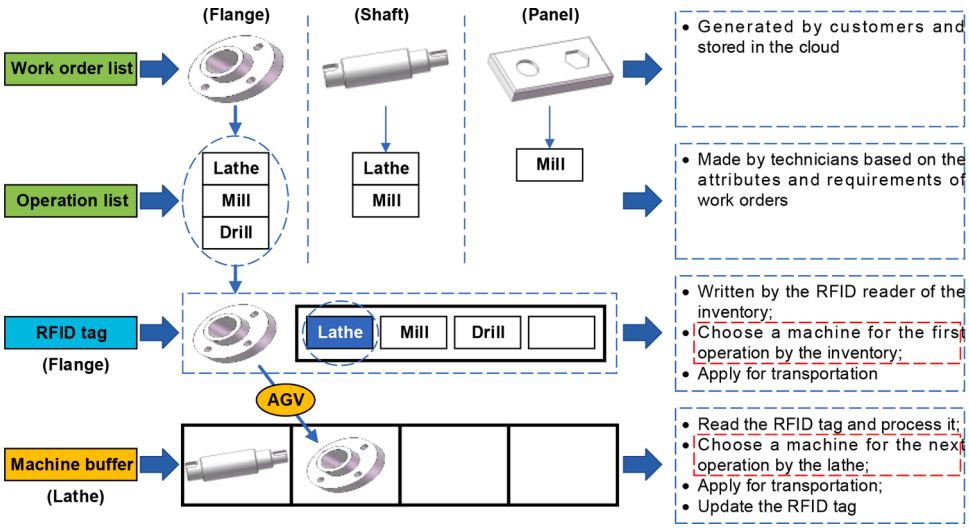

Fig. 2. Work order processing and data transmission procedures of a flange in a smart factory.

transmission.

Work orders are generated by customers and stored in a work order list in the cloud. The processes of work orders are planned according to their attributes with specific requirements. A flange contains three operations of lathing, milling, and drilling; a shaft contains two operations of lathing and milling; a panel contains one operation of milling. The scheduling and machining procedures of a flange are specified to illustrate the operating mechanisms. As shown in Fig. 2, the work order of a flange enters the factory after process planning is completed, and its first operation of the work order is initialized. The AI scheduler of the warehouse will not schedule the work order until there is an available machine. Each scheduled work order is placed on a pallet with an RFID tag. As a work order is scheduled, operation attributes of the work order will be written on the attached RFID tag by the RFID reader of the warehouse. Operation attributes include identification (ID), operation type, nominal machining time, initialization time, and target completion time. A flange contains three operations of lathing, milling, and drilling, so the warehouse scheduler needs to choose a lathe for its first operation and apply for transportation. An AGV transports the flange from the warehouse to the target lathe. The RFID reader of the lathe reads the current operation from the RFID tag of the flange and the lathe performs the operation after former operations are completed. As the current operation of the flange is finished, the AI scheduler of the lathe will update the data on the RFID tag of the flange and choose a miller for the next operation of the work order. Then, similar procedures of transportation, machining, and scheduling continue until all operations of the flange are completed.

3.1.3. Communication among AI schedulers

An AI scheduler is awakened as it has schedulable work orders or other schedulers ask for the states of its corresponding machine. The communications among AI schedulers are illustrated in Fig. 3. Four machines are shown in Fig. 3 to schedule seven work orders marked by o_i, $i = 1, \ldots, I$, where I is the total number of work orders. The jth operation of work order o_i is denoted by $b_{i,j}$, $j = 1, \ldots, J_i$, where J_i is the total number of operations of work order o_i. Work orders o_3 and o_4 are waiting in the buffers of machines 1 and 2 for scheduling at the same time. The AI schedulers of machines 1 and 2 should ask the available machines for their real-time states and choose optimal machines for the next operations of work orders o_3 and o_4 respectively. Two scheduling actions can be conducted independently without interruption, and scheduling policies are made by AI schedulers according to their own observations. The AI schedulers of machines 3 and 4 are partly awakened for providing real-time states of their corresponding machines. AI

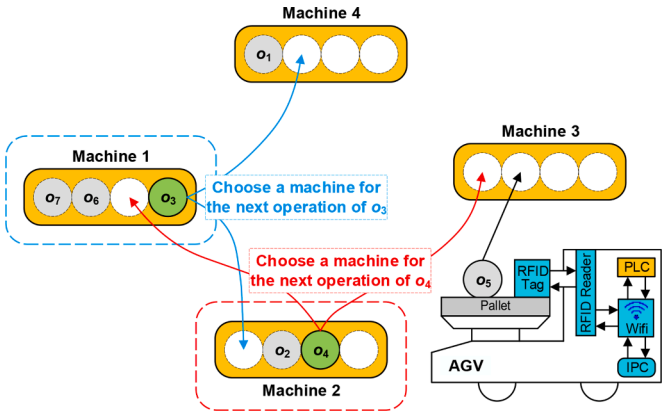

Fig. 3. Collaborations among AI schedulers to make optimal policies.

schedulers will return to the idle mode for saving resources if there is no schedulable work order and no requirement from other AI schedulers.

As shown in Fig. 4, an AI scheduler operates on the IPC of a machine and works with the layers of communication, AI, and adaptation. The communication layer connects the machine with the corresponding RFID reader or other units through the IoT. The adaptation layer enables the AI scheduler to exchange data with the corresponding machine. When a work order enters the buffer of a machine, the RFID reader derives the current operation attributes from the RFID tag, and the communication layer transmits the operation to the AI layer. Then, the AI layer generates numerical control (NC) codes for the current operation based on its attributes as the former operations are completed. NC codes are sent to the computer numerical control (CNC) system through the adaptation layer. The machine starts machining the operation after the work order is placed on the workbench by the robot shown in Fig. 1. The states of the work order and machine are transmitted to the supervisor and other related units in real time. As the current operation of a work order is finished, data on the RFID tag are updated and the next operation of the work order is initialized. The AI scheduler schedules the next operation according to the current states of other machines.

3.2. Online interaction of physical units

3.2.1. Building AI schedulers

Assumptions: This paper makes some assumptions to develop the scheduling model with multiple AI schedulers. First, each machine can

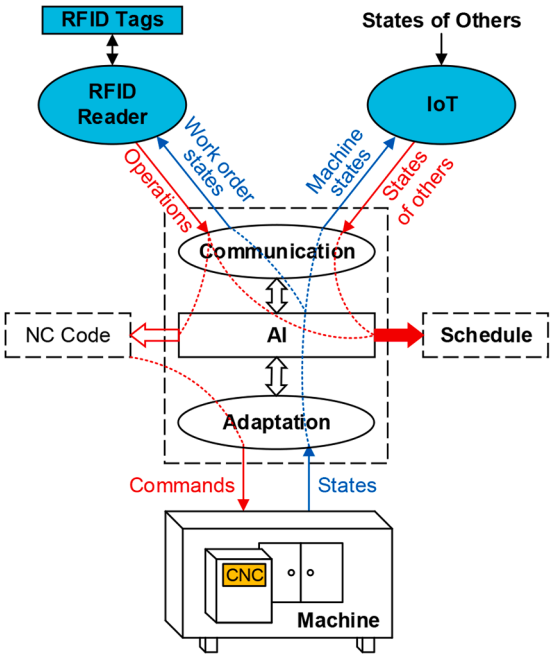

Fig. 4. The implementation of a proposed AI scheduler and data flows on an IPC.

handle only one specific type of operation (e.g., lathing, milling, or drilling), and the chosen machine should be capable of machining the scheduled operation, i.e., machine types match operation types. Second, machines with the same type have similar attributes (e.g., precision, machining speed, energy efficiency, buffer length). Third, a work order has one or more operations whose sequence should satisfy design requirements. Fourth, the total number of work orders scheduled on a machine should be no more than the buffer length.

RL for production scheduling: The analogous relationships between multi-agent reinforcement learning (MARL) and distributed production scheduling are listed in Table 1. In an MARL algorithm, each agent takes actions based on the states of the environment and accumulates rewards for state transitions. Agents interact with others by sharing their own policies to achieve common objectives. In the proposed smart factory, each unit is equipped with an AI scheduler that can independently schedule the work orders in the corresponding buffer. The states of machines are updated dynamically and shared with related schedulers immediately. Scheduling policies are made by AI schedulers according to the attributes of the current operation and real-time states of other machines. Schedulers can get rewards for the scheduling policies by their own reward functions.

Collaborations among AI schedulers: The AI scheduler of each machine can make real-time decisions independently without being influenced by the failures of other machines. As shown in Fig. 5, each AI scheduler has a scheduling policy network and a manufacturing value network. The scheduling policy network outputs schedules based on the sensor data of operation-and-machine attributes in real time. The attributes of operations are derived from the RFID tags of work orders. Sensors, installed in machines, derive the dynamic attributes and

Table 1
The relationships between MARL and production scheduling.

MARL	Production scheduling
Agents	Schedulers
Environment	Factories with machines and work orders
States	Dynamic attributes of machines and work orders
Actions	Scheduling for machines and work orders
Rewards	Optimization objectives

remaining workloads of corresponding machines. For example, the lathing operation of a flange in Fig. 5 has been completed on a lathe and its milling operation is waiting for being scheduled. The AI scheduler of the current lathe is responsible for choosing a suitable miller for the milling operation of the flange, and it can get a reward for the scheduling policy from the smart factory. The manufacturing value network provides state-action values of all related AI schedulers for updating the scheduling policy network.

State and action formulation: There are M machines and a warehouse in a smart factory, and each of them is equipped with an AI scheduler. Sensor data, whose components are specified in Table 2, are denoted by $\boldsymbol{s} = (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_m, \ldots, \boldsymbol{s}_M)$, where $\boldsymbol{s}_m = (d_1, \ldots, d_8)$ denotes the sensor data for the AI scheduler of machine m, $m = 0, 1, \ldots, M$ ("0" represents the warehouse). The AI scheduler of machine m can schedule N_m operations simultaneously. Therefore, the sensor data for an AI scheduler can be formulated by Eq. (1).

$$\boldsymbol{s}_m = \left[(d_1,\ldots,d_4)_1,\ldots,(d_1,\ldots,d_4)_{N_m};\ (d_5,\ldots,d_8)_1,\ldots,(d_5,\ldots,d_8)_{|M|_m}\right]_{4N_m+4|M|_m} \tag{1}$$

where $|M|_m$ denoted the total number of optional machines for the current scheduler of machine m. AI schedulers provide optimal actions at scheduling moments and derive rewards for state transitions from the smart factory. The actions of all schedulers are denoted by $\boldsymbol{a} = (\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m, \ldots, \boldsymbol{a}_M)$, where $\boldsymbol{a}_m = (i_1, \ldots, i_{N_m}; a_1, \ldots, a_{N_m})$ is the action for machine m; $(i_1, \ldots, i_{N_m})$ is the operation sequence; $a_1, \ldots, a_{N_m}$ represent target machines. Rewards for AI schedulers are denoted by $\boldsymbol{r} = (r_1, \ldots, r_m, \ldots, r_M)$. The experience of a state transition from the current state $\boldsymbol{s}$ to the next state $\boldsymbol{s}' = (\boldsymbol{s}'_1, \ldots, \boldsymbol{s}'_m, \ldots, \boldsymbol{s}'_M)$ is denoted by $(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{r}, \boldsymbol{s}')$ and stored on the supervisor.

3.2.2. Scheduling policies

Work order handling: The warehouse or a machine has a buffer zone and an operation list. Work orders are stored in buffers and wait to be processed on the current machine or scheduled for the initialized operations. The buffer length of machine m is denoted by a non-negative integer B_m. The first operation of each work order is initialized in the operation list of the warehouse. When the operation of a work order is completed on a machine, its next operation is initialized on the operation list of the current machine. AI schedulers make decisions at scheduling moments when a new operation is initialized or a scheduled operation is completed by a machine. If there is no available machine for the work order with an initialized operation, it will stay in the current buffer and wait for the next scheduling moment. Fig. 6 shows AI schedulers' operating procedures, including online scheduling and offline training, which are detailed in the following paragraphs of this section.

Online scheduling: As shown in Fig. 7, an operation of work order o_{i+2} is initialized at this moment, and will be scheduled with work orders o_i and o_{i+1} that are initialized before. The AI scheduler in Fig. 7 can handle four work orders simultaneously (i.e., $N_m = 4$) by giving scheduling actions, including an operation sequence and target machines. Work orders o_i and o_{i+2} are scheduled successfully, while o_{i+1} stays in the current operation list and waits for the next scheduling moment. If there are more work orders waiting for scheduling in the operation list, the AI scheduler will take another action according to the next observation $\boldsymbol{s}'_m$ until all operations are handled. The schedulable work orders of all units (i.e., the warehouse or machines) are handled at every scheduling moment. Each AI scheduler has a separate operation list which is updated when an operation is initialized or scheduled. AI schedulers continuously derive the statuses of the factory and online schedule work orders whenever operation lists are updated or target machines are available. The trained AI schedulers can make decisions in real time without any duplicated computations as traditional rescheduling methods. The modeling and training processes for multiple AI

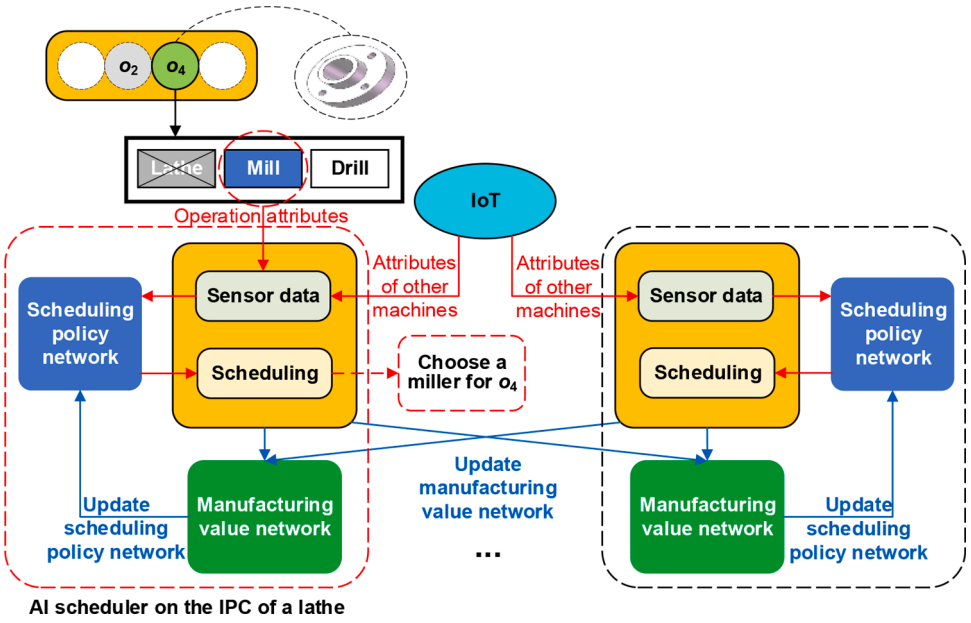

Fig. 5. Collaborations among AI schedulers on scheduling work orders and improving decision-making abilities.

Table 2
Sensor data for the AI scheduler of machine m.

	Param	Description	Value
Operation $b_{i,j}$	d_1	Operation type c	1, 2, …
	d_2	Initialization time $T_{i,j}^{(A)}$	$[0, +\infty)$
	d_3	Nominal operating time $\widetilde{T}_{i,j}$	$[0, +\infty)$
	d_4	Target completion time $\widehat{T}_{i,j}^{(C)}$	$[0, +\infty)$
Machine m	d_5	Machine type c	1, 2, …
	d_6	Remaining buffer length B_m	0, 1, …
	d_7	Waiting time $T_m^{(W)}$	$[0, +\infty)$
	d_8	Transportation time $T_m^{(V)}$	$[0, +\infty)$

schedulers are specified in the following paragraphs of this section.

Neural networks: As shown in Fig. 8, the AI scheduler of a machine has four neural networks: a scheduling policy network, a manufacturing value network, a scheduling target network, and a manufacturing target network. All networks are fully-connected neural networks and each network has an input layer, a hidden layer, and an output layer. Sensor data s_m for operation-and-machine states, shown in Table 2, are continuously input to the scheduling policy network, which has multiple layers with numerous neurons for outputting an advisable scheduling action $\boldsymbol{a}_m$. The manufacturing value network provides a state-action value $Q_m(s, \boldsymbol{a})$ for updating the scheduling policy network by the policy gradient method. The manufacturing environment will become unstable if two or more AI schedulers take actions simultaneously. Thus, we use the manufacturing value network to evaluate $\boldsymbol{a}_m$ based on not only the scheduler's own state-action set (s_m, $\boldsymbol{a}_m$) but also the state-action sets from other AI schedulers, i.e., "$\boldsymbol{s}_{\ldots}$" and "$\boldsymbol{a}_{\ldots}$" in Fig. 8. The scheduling target network gives the next scheduling action $\boldsymbol{a}'_m$ according to sensor data s'_m of the next states. The manufacturing target network provides a state-action value $\widehat{Q}_m(s', \boldsymbol{a}')$ for updating the manufacturing value network by the temporal difference method. The state-action value $\widehat{Q}_m(s', \boldsymbol{a}')$ is obtained according to not only the scheduler's own state-action set $(s'_m, \boldsymbol{a}'_m)$ but also the state-action sets from other AI schedulers, i.e., "$\boldsymbol{s}'_{\ldots}$" and "$\boldsymbol{a}'_{\ldots}$" in Fig. 8. Each manufacturing unit has a unique AI scheduler with four individualized neural networks. The AI scheduler can make decisions based on their own observations and learn

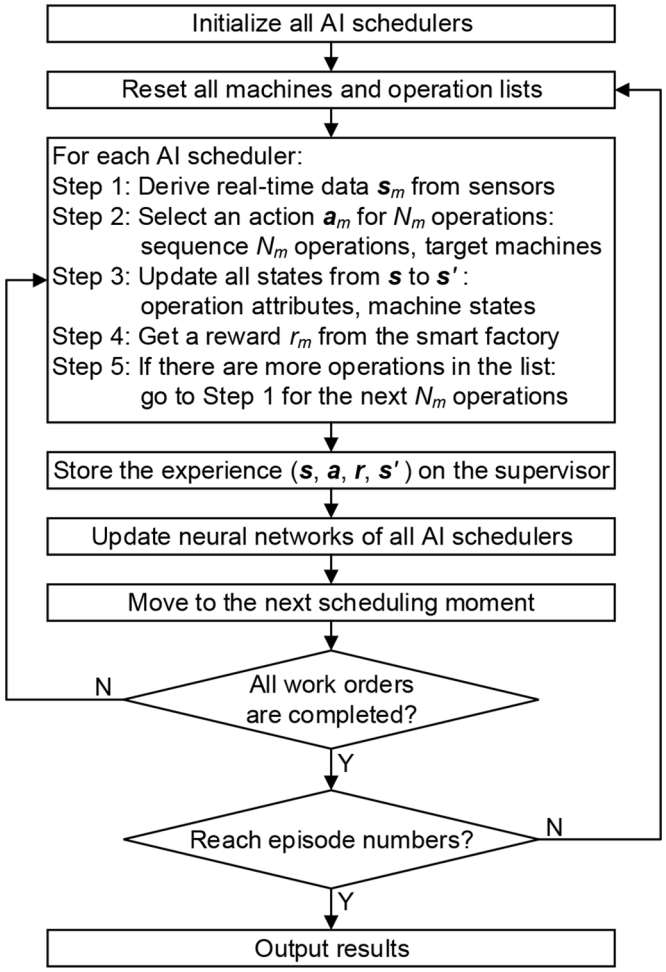

Fig. 6. Operating procedures of multiple AI schedulers in a distributed smart factory.

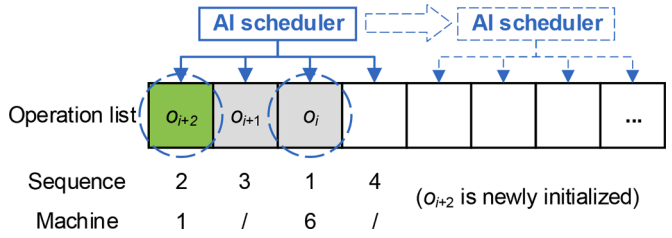

Fig. 7. Scheduling steps of an AI scheduler that can handle four operations simultaneously.

from other schedulers' experiences to achieve global optimization, which significantly improves the efficiency and robustness of the manufacturing system. The next paragraph specifies the procedures of utilizing and updating the neural networks of AI schedulers.

RL for training AI schedulers: Schedules are given by the scheduling policy network of each AI scheduler according to real-time states s of the smart factory, and the continuous policies for M AI schedulers are $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_m, \ldots, \mu_M)$ parameterized by $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_m, \ldots, \theta_M)$. A random noise process $\mathcal{N}_m$ is added to the nominal policy μ_m for realizing policy exploration, i.e., $\boldsymbol{a}_m = \mu_m(s_m) + \mathcal{N}_m$. The gradient of policy μ_m is written in Eq. (2) to show the direction for optimizing the scheduling objectives of an AI scheduler. The main idea is to adjust the parameters $\boldsymbol{\theta}$ of policies $\boldsymbol{\mu}$ for maximizing the objective $J(\mu_m)$ with expected return $E_{\mu_m}[(r_m)_1 + \gamma_m(r_m)_2 + \cdots + \gamma_m^t(r_m)_{t+1}]$, where $\gamma_m \in [0,1]$ is a discount factor. Parameters $\boldsymbol{\theta}_m$ are adjusted by taking iterations in the direction of $\nabla_{\boldsymbol{\theta}_m} J(\mu_m)$ with the gradient-ascend method, i.e., $\boldsymbol{\theta}_m' = \boldsymbol{\theta}_m + \alpha^{\theta} \nabla_{\boldsymbol{\theta}_m} J(\mu_m)$, where $\alpha^{\theta} \in [0,1]$ is a learning rate.

$$\begin{aligned} \nabla_{\boldsymbol{\theta}_m} J(\mu_m) &= \nabla_{\boldsymbol{\theta}_m} \mu_m(s_m) \nabla_{\boldsymbol{a}_m} Q_m^{\mu}(\boldsymbol{s}, \boldsymbol{a}), \\ \boldsymbol{a}_m &= \mu_m(s_m) \end{aligned} \tag{2}$$

Policy μ_m is given by the scheduling policy network whose parameters are updated according to the Q values from the manufacturing value network for achieving more rewards. Function Q_m^{μ} is formulated by the manufacturing value network with parameters $\boldsymbol{\omega}_m$ from $\boldsymbol{\omega} = (\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_m, \ldots, \boldsymbol{\omega}_M)$, which are updated by minimizing the loss function written in Eq. (3) with the gradient-decent method, i.e., $\boldsymbol{\omega}_m' = \boldsymbol{\omega}_m - \alpha^{\omega} \nabla_{\boldsymbol{\omega}_m} L(\boldsymbol{\omega}_m)$, where $\alpha^{\omega} \in [0,1]$ is a learning rate.

$$\begin{aligned} L(\boldsymbol{\omega}_m) &= \left[r_m + \gamma_m \widehat{Q}_m^{\mu'}(\boldsymbol{s}', \boldsymbol{a}') - Q_m^{\mu}(\boldsymbol{s}, \boldsymbol{a}) \right]^2, \\ \boldsymbol{a}_m' &= \mu_m'(\boldsymbol{s}_m') \end{aligned} \tag{3}$$

where $\boldsymbol{\mu}' = (\mu_1', \ldots, \mu_m', \ldots, \mu_M')$, w. r. t. parameters of $\boldsymbol{\theta}' = (\boldsymbol{\theta}_1', \ldots, \boldsymbol{\theta}_m', \ldots, \boldsymbol{\theta}_M')$, are scheduling policies of scheduling target networks. Function $\widehat{Q}_m^{\mu'}$ is formulated by a manufacturing target network parameterized by $\boldsymbol{\omega}' = (\boldsymbol{\omega}_1', \ldots, \boldsymbol{\omega}_m', \ldots, \boldsymbol{\omega}_M')$. At each training step, a mini-batch of experiences are sampled to update the networks by Eqs. (2) and (3) with their mean values of $J(\mu_m)$ and $L(\boldsymbol{\omega}_m)$. Parameters of the scheduling target network and the manufacturing target network are adjusted respectively by Eqs. (4) and (5) using soft updates for stabilizing the system.

$$\boldsymbol{\theta}_m' = \tau^{\theta} \boldsymbol{\theta}_m + (1 - \tau^{\theta}) \boldsymbol{\theta}_m', \ 0 < \tau^{\theta} \ll 1 \tag{4}$$

$$\boldsymbol{\omega}_m' = \tau^{\omega} \boldsymbol{\omega}_m + (1 - \tau^{\omega}) \boldsymbol{\omega}_m', \ 0 < \tau^{\omega} \ll 1 \tag{5}$$

3.2.3. Collaborative learning

If an AI scheduler is not trained, it cannot give advisable policies due to the lack of experience. AI schedulers can improve their scheduling abilities by learning from interaction with manufacturing systems and collaboration with other schedulers. Learning procedures of schedulers are directed by the accumulated rewards derived from manufacturing systems. The performance of an AI scheduler depends on the reward functions established according to the optimization objectives of production scheduling. Objectives depend on the requirements of customers (e.g., minimizing makespan) and the profits of companies (e.g., saving energy, balancing workloads). We consider minimizing makespan and balancing workloads in modeling the distributed smart factory with multiple AI schedulers.

It is assumed that machines of the same type have similar attributes, i.e., no matter which machine is chosen for an operation, the machining time of the operation keeps the same. If work orders are completed before the target completion time, AI schedulers will get positive re-

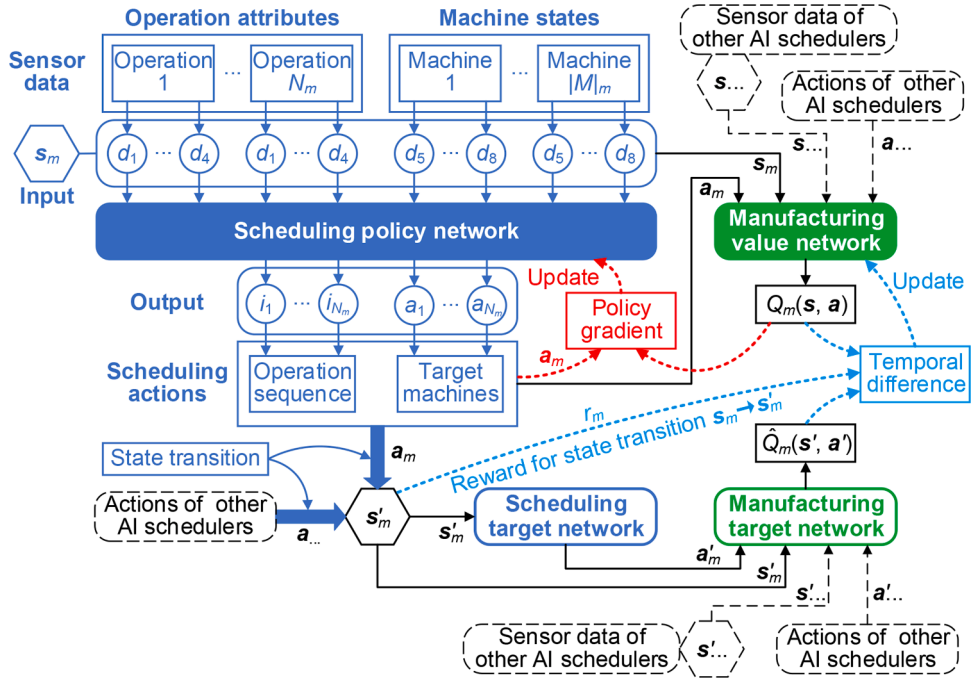

Fig. 8. An AI scheduler's neural networks, including scheduling policy network, scheduling target network, manufacturing value network, and manufacturing target network.

wards, otherwise the rewards are negative. In order to get more rewards, AI schedulers attempt to minimize the makespan of work orders by scheduling operations on available machines as soon as possible. As shown in Fig. 9, the initialization time, starting time, completion time, and target completion time of an operation are respectively denoted by $T_{i,j}^{(A)}$, $T_{i,j}^{(S)}$, $T_{i,j}^{(C)}$, and $\widehat{T}_{i,j}^{(C)}$. Nominal operating time $\widetilde{T}_{i,j}$ of operation $b_{i,j}$ is obtained by multiplying workload time $T_{i,j}$ with a factor assigned with 3 in this paper, i.e., $\widetilde{T}_{i,j} = 3T_{i,j}$. Target completion time $\widehat{T}_{i,j}^{(C)}$ of operation $b_{i,j}$ depends on initialization time $T_{i,j}^{(A)}$ and nominal operating time $\widetilde{T}_{i,j}$, i.e., $\widehat{T}_{i,j}^{(C)} = T_{i,j}^{(A)} + \widetilde{T}_{i,j}$. The first operation of a work order is initialized when the work order is generated (i.e., $T_i^{(A)} = T_{i,1}^{(A)}$), and the work order is completed when its last operation is finished (i.e., $T_i^{(C)} = T_{i,J_i}^{(C)}$). The nominal operating time of a work order is $\widetilde{T}_i = \widetilde{T}_{i,1} + \ldots + \widetilde{T}_{i,J_i}$, and the target completion time of a work order is $\widehat{T}_i^{(C)} = \widehat{T}_{i,J_i}^{(C)} = T_i^{(A)} + \widetilde{T}_i$.

The starting time $T_{i,j}^{(S)}$ of operation $b_{i,j}$ begins when former operations on the same machine are completed. $T_{i,j}^{(S)}$ is influenced by many factors such as transportation time $T_m^{(V)}$ and waiting time $T_m^{(W)}$ that depends on the remaining workloads and preparing time of the target machine. Work order urgency depends on the remaining time before target completion time $\widehat{T}_{i,j}^{(C)}$. When an operation starts to be performed at time $T_{i,j}^{(S)}$, the urgency factor can be formulated by:

$$D_{i,j} = \frac{\widehat{T}_{i,J_i}^{(C)} - T_{i,j}^{(S)}}{T_{i,j} + \ldots + T_{i,J_i}} \tag{6}$$

The urgency factor $D_{i,j}$ will become smaller if a work order o_i reaches the target completion time but has many uncompleted operations. Rush orders also have small urgency factors. Commonly, work orders with smaller urgency factors, whether rush orders or not, will be given priority for scheduling. An AI scheduler can handle N_m work orders simultaneously by providing them an operation sequence and corresponding target machines, which is specified in Section 3.2.2. A smaller standard deviation of the urgency factors of N_m work orders represents that most rush orders are scheduled with higher priority. Thus, a reward $r^{(D)}$ is designed by Eq. (7) to evaluate the operation sequence given by the AI scheduler of machine m. std$(D_{i,j})$ denotes the standard deviation of urgency factors of N_m work orders that are scheduled by the AI scheduler of machine m.

$$r^{(D)} = e^{-\mathrm{std}\left(D_{i,j}\right)},\ 0 < r^{(D)} < 1 \tag{7}$$

An AI scheduler is inclined to schedule an operation on an idle machine that needs less waiting time. At a scheduling moment, machine m needs time $T_m^{(W)}$ to finish the uncompleted operations, and the transportation time to machine m is $T_m^{(V)}$. Thus, a reward for the waiting time rate of machine m is defined in Eq. (8), which increases with the decrease of $\max[T_m^{(W)},\ T_m^{(V)}]/T_{i,j}$.

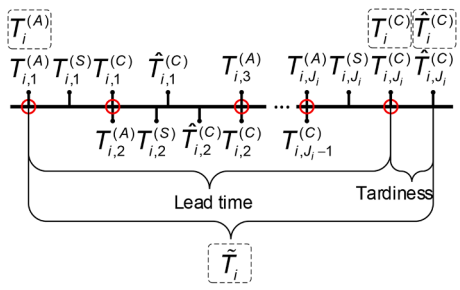

Fig. 9. The timeline of work order o_i and its component operations.

$$r^{(W)} = e^{-\max\left[T_m^{(W)},T_m^{(V)}\right]/T_{i,j}},\ 0 < r^{(W)} < 1 \tag{8}$$

AI schedulers attempt to balance workloads among machines by maximizing the utilization rate of each machine. $u_m(t)$ denotes the utilization time of machine m from start to scheduling moment t. Eq. (9) defines a reward for the utilization rate of a target machine, which increases with the increase of workload time $T_{i,j}$, and the influential level decreases with the increase of utilization time $u_m(t)$.

$$r^{(U)} = \left(1 - e^{-T_{i,j}}\right)\frac{T_{i,j}}{T_{i,j} + u_m(t)},\ 0 < r^{(U)} < 1 \tag{9}$$

A composite reward, derived by the AI scheduler of machine m, is defined in Eq. (10) by integrating the reward for operation priority, waiting time rate, and machine utilization rate. AI schedulers can adapt to different working conditions by applying the composite reward function and adjusting the weights in Eq. (10). For example, we can increase w_1 or w_2 for scheduling rush orders with higher priority and less waiting time. We can also increase the value of w_3 for balancing workloads among machines.

$$r_m = w_1 \cdot r^{(D)} + w_2 \cdot r^{(W)} + w_3 \cdot r^{(U)} \tag{10}$$

4. Experimental design

As shown in Fig. 10, various experiments are designed to evaluate the learning and scheduling performances of multiple AI schedulers in the proposed smart factory. A testbed is designed to realize IoT manufacturing by utilizing cyber-physical technologies. We benchmark the performances of AI schedulers with a variety of traditional scheduling methods such as GA, CNP, and centralized RL.

4.1. A smart factory testbed

As shown in Fig. 11, a testbed is established to simulate the proposed smart factory and evaluate the performance of multiple AI schedulers. Process planning is operated in the cloud after work orders are generated by customers on a website. A warehouse derives work orders from the cloud and chooses a machine for performing the first operation of each work order. Each work order is placed on a pallet with an RFID tag where operation attributes are written by the RFID reader of the warehouse when the work order is scheduled. There are six machines in the smart factory, including two millers, two drillers, and two lathes. An AGV transports pallets with work orders among the warehouse and machine buffers. The buffer length of each machine is four. Robots handle work orders between machine workbenches and corresponding buffers. Each machine is equipped with an RFID reader, whose antennas are fixed on machine buffers, to derive and update the data on RFID tags of work orders.

All units are interconnected by IoT and scheduled by multiple AI schedulers that are operated on IPCs. As shown in Fig. 12, the control cabinet of a unit is composed of an IPC, a programmable logic controller (PLC), and a LAN hub. The LAN hub connects the IPC with an RFID reader, a manufacturing unit (i.e., warehouse, machine, AGV, or robot), and other units (i.e., cloud, supervisor, or equipment). The AI scheduler of a machine can interact with the CNC system by sending NC codes and monitoring machine statuses.

4.2. Performance evaluation of schedulers

Learning performance: The warehouse and each machine are respectively equipped with an AI scheduler that schedules work orders on available machines to minimize makespan and balance workloads. AI schedulers continuously learn from interacting with the smart factory and collaborating with other schedulers. We benchmark the learning performance of the proposed methodology with a traditional RL-based

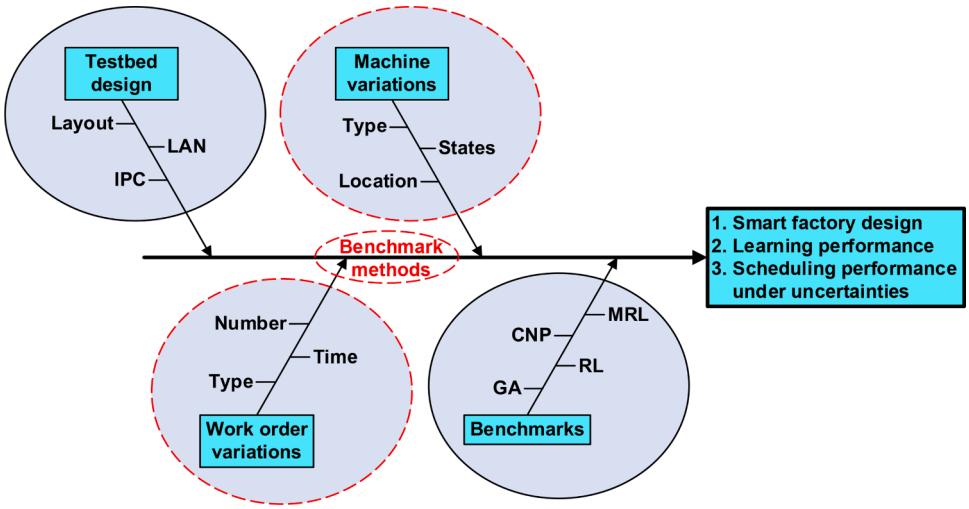

Fig. 10. Testbed and experimental design for performance evaluation of different scheduling methods.

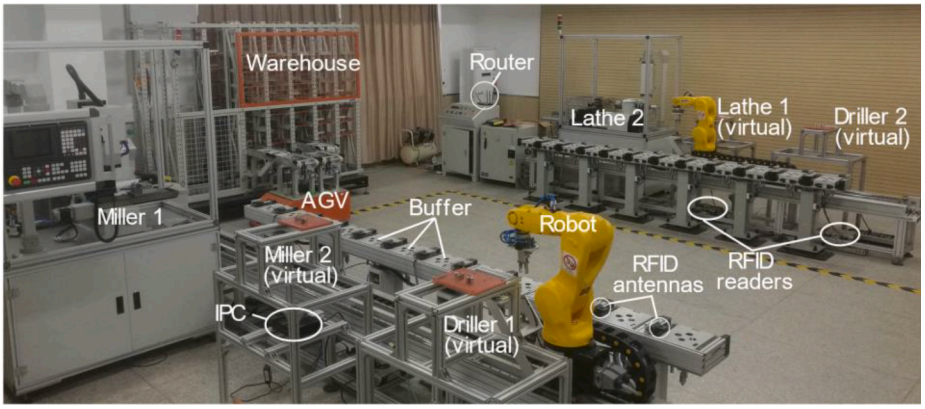

Fig. 11. A bird-eye view of the smart factory testbed for performance evaluation of AI schedulers with real-world case studies.

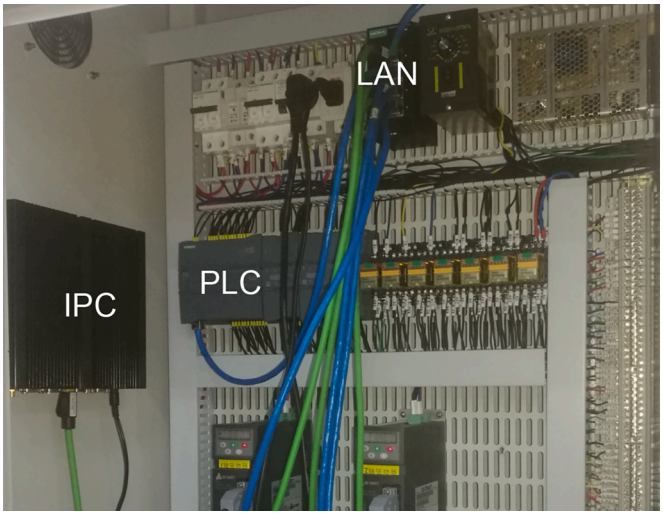

Fig. 12. The control cabinet of a manufacturing unit in the smart factory.

method, i.e., deep Q-networks (DQN). The DQN method models the scheduler on a computer where a virtual environment is established to interact with the scheduler. All AI schedulers and the DQN scheduler are trained with 1,000 random work orders and their learning curves are recorded for comparison. In addition, the performance of the proposed reward function in Eq. (10) is compared with traditional methods that simply minimize the idle time (i.e., maximize the utilization time) of machines.

Scheduling performance: Three types of work orders, including flanges, shafts, and panels, are created with different attributes, e.g., operation type, initialization time $T_{i,j}^{(A)}$, and nominal operating time $\widetilde{T}_{i,j}$. Machines are also varied by machine type, failure level, and location. Work order variations and machine failures cause uncertainties in machining processes. We benchmark the scheduling performance (e.g., makespan, workload balance) of AI schedulers and different scheduling methods (e.g., GA, CNP, DQN). GA algorithms schedule work orders offline and require that all work orders are generated before scheduling. The CNP method is considered with scheduling rules such as first come first serve (FCFS) and shortest waiting time first (SWTF). The DQN method models a centralized RL scheduler to handle all types of operations and machines online.

Parameter specifications: The parameters of an AI scheduler are set as: learning rates for updating four networks $\alpha^{\theta} = \alpha^{\omega} = 0.01$, $\tau^{\theta} = \tau^{\omega} = 0.01$, discount rate for rewards $\gamma_m = 0.9$. There are six machines (i.e., $M = 6$) denoted by 1, …, 6, including two lathes (machine type $c = 1$), two millers (machine type $c = 2$), and two drillers (machine type $c = 3$). The buffer length of the warehouse is 40, i.e., $B_0 = 40$. Each machine has a buffer with a capacity of four work orders, i.e., $B_m = 4$, $m = 1, \ldots, 6$. An AI scheduler can handle four work orders simultaneously, i.e., $N_m = 4$, $m = 0, \ldots, 6$. According to Eq. (1), the dimension of s_m for a machine scheduler is 32 (i.e., $|M|_1 = \ldots = |M|_6 = 4$), while it is 40 (i.e., $|M|_0 = 6$)

for the warehouse scheduler. Thus, a scheduling policy network has an input layer with 32 or 40 neurons, two hidden layers respectively with 1,000 neurons, and an output layer with 8 neurons. A manufacturing value network has an input layer with 288 neurons (i.e., states and actions of 7 AI schedulers), two hidden layers respectively with 1,000 neurons, and an output layer with 1 neuron. A scheduling target network has the same parameters as its corresponding scheduling policy network, and a manufacturing target network has the same parameters as its corresponding manufacturing value network. Work orders can be flanges, shafts, and panels in the experiments. A flange has three operations of lathing (operation type $c = 1$), milling (operation type $c = 2$), and drilling (operation type $c = 3$); a shaft has two operations of lathing and milling; a panel has only one operation of milling. Work orders are generated and initialized at random time $T_{i,1}^{(A)}$. The nominal operating time of a lathing operation, a milling operation, and a drilling operation are respectively in the ranges of [60, 180], [15, 90], and [30, 60]. All time-related variables are taken as positive integers.

Parameters of baseline methods: Reference [15] shows the scheduling model of the baseline method GA. The parameters of GA are set as: the population size is 100; the maximum number of generations is 500; the crossover probability is 0.8; the mutation rate is 0.1. Reference [41] shows the scheduling model of the baseline method DQN. The parameters of DQN are set as: the learning rate is 0.01; the discount rate of rewards is 0.9; the greedy rate for policy choosing is 0.9. The neural network of DQN inputs the states of 4 work orders and 6 machines, and outputs the Q values of work order sequences (4 neurons) and target actions of 4 work orders (7 neurons for each work order, i.e., wait or choose 1 from 6 machines). Hence the neural network of DQN has an input layer with 40 neurons, a hidden layer with 1000 neurons, and an output layer with 32 neurons.

Online scheduling aims at handling discrete machining operations in real time. The experimental work orders (i.e., flanges, shafts, and panels) are typical parts with basic operations including lathing, milling, and drilling. The combination of different basic operations can generate more complex work orders. The proposed AI scheduler of each machine can work independently and handle unexpected events such as machine breakdown or adding new machines. In Section 5.3, we evaluate the scheduling performance by turning a machine off and then restart it to test the AI schedulers' adaptability for a system with more machines. Therefore, the experiments are sufficient to represent generic production scheduling problems even in complex manufacturing scenarios with more work orders and machines.

5. Experimental results

5.1. Leaning performance of RL-based scheduling methods

The first set of experiments aims to evaluate the learning performance of RL scheduling methods. Schedulers attempt to improve their decision-making abilities with the help of different reward functions. The weights in Eq. (10) are assigned with 1/3 (i.e., $w_1 = w_2 = w_3 = 1/3$) for the proposed methodology. Both $r^{(W)}$ in Eq. (8) and $r^{(U)}$ in Eq. (9) are related to makespan that is regarded as the only objective for most RL scheduling methods. However, if most work orders need not wait in a not busy factory, the RL schedulers with only $r^{(W)}$ in Eq. (8) always get the same rewards no matter what decisions they make, which is ineffective in training the schedulers. Therefore, by contrast, only reward $r^{(U)}$ in Eq. (9) is considered (i.e., $w_1 = w_2 = 0$ and $w_3 = 1$) in traditional RL scheduling methods to minimize the makespan of all work orders.

Fig. 13 shows the learning curves of RL scheduling methods with two different reward functions, including composite reward (i.e., "AI" and "DQN") and single reward (i.e., "AI-S" and "DQN-S"). At the very beginning, multiple AI schedulers earn fewer rewards than centralized DQN schedulers, because distributed schedulers need more attempts to establish relationships with other schedulers. The curves in Fig. 13 respectively converge after scheduling 240, 300, 400, and 530 work orders. Each machine is equipped with a unique AI scheduler, which adapts to the corresponding machine and operations effectively. As a result, the mean reward curves of multiple AI schedulers converge faster than DQN schedulers. After reaching the convergence points, four scheduling algorithms respectively earn 0.89, 0.83, 0.69, and 0.58 rewards for scheduling a work order. AI schedulers with composite reward functions earn 7.2%, 29.0%, and 53.4% more rewards than "AI-S", "DQN", and "DQN-S" methods. In summary, multiple AI schedulers not only learn faster than centralized RL methods but also earn more rewards during training and scheduling processes. Meanwhile, composite reward functions improve the training efficiency of schedulers and effectively improve their scheduling performance.

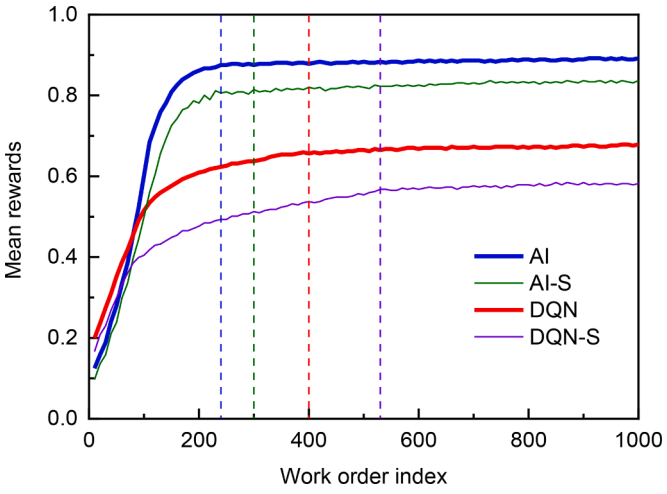

Fig. 13. Mean rewards for AI and DQN schedulers from learning and scheduling 1,000 random work orders: suffix "-S" represents a traditional reward function with a single optimization objective.

5.2. Scheduling performance w. r. t. work order variations

In this section, 50 new work orders are randomly generated with various attributes to evaluate the scheduling performance of different scheduling methods. 10 work orders indexed from 20 to 30 are generated simultaneously. AI schedulers and the DQN scheduler with composite reward functions have been trained by scheduling 1,000 random work orders in the former experiments. The distributed AI, centralized DQN, and common CNP methods can online schedule work orders based on real-time states of workshops. The GA method schedules all work orders offline by searching the whole action space for achieving optimal policies. Schedulers try to minimize the makespan of all work orders by shortening the lead time $(T_i^{(C)} - T_i^{(A)})$ of each work order. The lead time rate of a work order is defined as $(T_i^{(C)} - T_i^{(A)})/\widetilde{T}_i$.

Fig. 14 shows the lead time rates of 50 work orders scheduled by four scheduling methods respectively. Work orders scheduled by the GA scheduler have the shortest lead time. However, the GA scheduler can only schedule a batch of work orders offline, which is limited in handling online scheduling problems. The CNP method achieves good performance before simultaneous work orders are generated, but most work orders approach target completion time $\widehat{T}_i^{(C)}$ or exceed $\widehat{T}_i^{(C)}$ (i.e., $(T_i^{(C)} - T_i^{(A)})/\widetilde{T}_i > 1$) when many work orders are generated simultaneously. Multiple AI schedulers reduce 11.9% and 27.0% mean lead time in comparison with DQN and CNP schedulers. Therefore, the proposed methodology can not only schedule work orders online but also reduce the lead time of work orders.

Workload balance depends on the operating time of machines and influences the robustness of manufacturing systems. Table 3 shows the

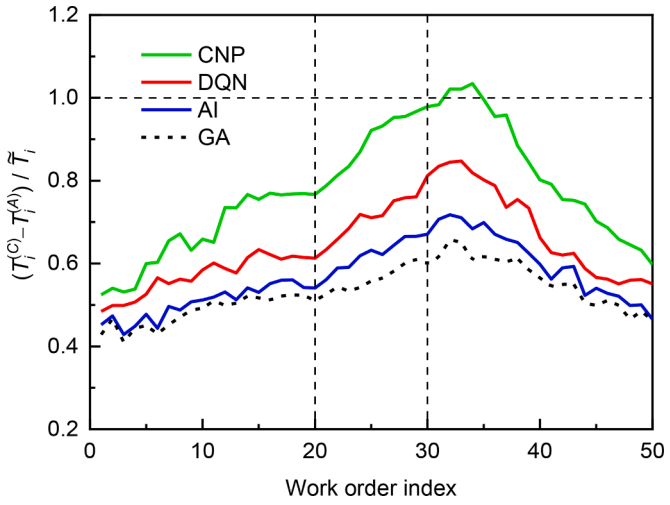

Fig. 14. Lead time rates of 50 work orders with 10 simultaneously generated work orders indexed from 20 to 30.

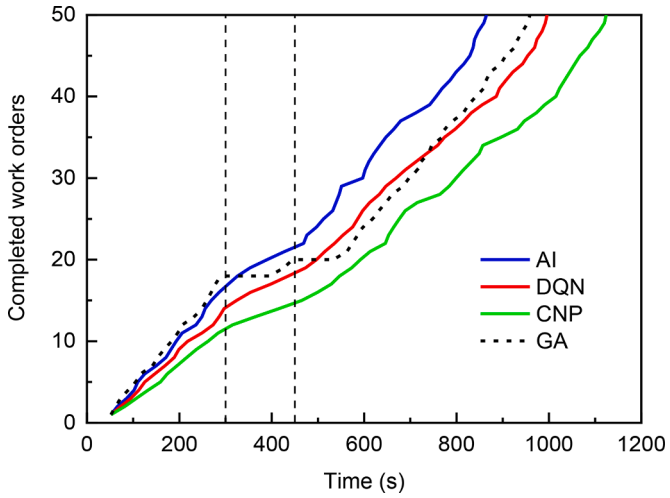

Fig. 15. Completion efficiency of 50 random work orders by four scheduling methods under machine failures from 300s to 450s.

Table 3
Coefficients of variation (σ/μ) for utilization rates among three types of machines by four scheduling methods.

Machines	AI	DQN	CNP	GA
Lathes	0.109	0.154	0.214	0.099
Millers	0.095	0.130	0.155	0.081
Drillers	0.083	0.113	0.122	0.064

coefficients of variation (i.e., standard deviation σ divided by mean μ) for utilization rates of machines of different types. Smaller coefficients of variation denote that workloads are well balanced among machines. All work orders have lathing operations, so it is difficult for schedulers to balance workloads among lathes. Thus, lathes have the largest coefficients of variation among three types of machines. The GA method shows the best performance among four scheduling methods but requires knowing the attributes of all work orders before scheduling and handles them offline. The proposed method with multiple AI schedulers achieves the best performance among three online methods on workload balance.

5.3. Scheduling performance w. r. t. machine variations

This section aims to validate the self-adaptive ability of the proposed AI schedulers for the interference of machine failures. Machine failures influence the operation of manufacturing systems and even break down the whole system. Therefore, we compare the performance of four scheduling methods under the condition of machine failures. A miller breaks down during operations at 300s and the repair takes 150s to restore the operation at 450s. Four scheduling methods face the same machine failure and are used to schedule 50 random work orders. The work order completion is a performance metric that counts how many work orders have been completed at a particular time.

As shown in Fig. 15, the manufacturing system handles 50 work orders that arrive at random and are completed within 1,200s. A miller breaks down at 300s, which demands scheduling algorithms to adjust for the new condition. The completion speed of work orders slows down from 300s to 450s, because there is only one miller available during this time. The GA method obtains better schedules than other online methods before machine failures, but at the expense of time delay for rescheduling work orders under the disturbance. It is delayed nearly 100s to adjust the schedules when a miller fails at 300s and takes another 100s to run the optimization algorithms as the miller is restored at 450s. Three online methods respond to the disturbance of machine failures and restoration in real time with minimal disruptions. The CNP method yields worse performance than two RL scheduling methods, because its scheduling abilities depend on complex rules rather than real-time data and accumulated experiences. The machine failure of a certain type has more impacts on the DQN scheduler than multiple AI schedulers, because an AI scheduler can not only adapt to its corresponding machine but also collaborate with other schedulers. Therefore, multiple AI schedulers show robust performance in handling unexpected machine failures and restorations.

Table 4 shows the makespan of 50 work orders that are scheduled by four methods. A smaller growth rate of makespan means that the interference of machine failure has less impact on the system. "Normal" denotes a normal condition that all machines work well, while "Failure" represents that a miller fails from 300s to 450s as shown in Fig. 15. The GA method obtains the least makespan for work orders if there is no disturbance, but the makespan increases 52.2% because of the time delay for offline rescheduling. The rescheduling time of GA will greatly increase if there are a large number of disturbances in a complex manufacturing system. The makespan obtained by multiple AI schedulers owns the least increasing rate of 31.7% when a miller fails during operation. Therefore, the proposed method with multiple AI schedulers can achieve optimal schedules online and effectively deal with unexpected events (e.g., machine failures) in real time.

6. Conclusions and future work

This paper presents a new distributed architecture with multiple AI schedulers for online scheduling of work orders in a smart factory. Each machine is equipped with a customized IPC, which interacts with the corresponding machine and links the machine to the manufacturing system. AI schedulers are operated on distributed IPCs, which improves their learning and scheduling efficiency. The MARL method organizes multiple schedulers to improve their decision-making abilities under uncertainties. A new formulation of composite reward functions enables AI schedulers to achieve multiple objectives, including minimizing makespan and balancing workloads. Real-world experiments are

Table 4
Makespan comparison of 50 random work orders considering machine failures from 300s to 450s.

Events	AI	DQN	CNP	GA
Normal	657	732	793	630
Failure	865	996	1124	959
Increase	31.7%	36.1%	41.7%	52.2%

conducted to evaluate the learning and scheduling performance of four scheduling methods.

The proposed architecture integrates heterogeneous manufacturing resources (e.g., machines and work orders) in a manufacturing system and enables manufacturing units to make advisable decisions on their own. We design four deep neural networks for each AI scheduler to make online schedules based on high-dimensional sensor data and learn real-time policies from the smart factory. AI schedulers learn from real-world operating data and the policies shared by other schedulers, which significantly improves the learning efficiency in comparison to centralized schedulers. The proposed composite reward functions effectively evaluate schedules in real time, while organizing all AI schedulers to optimize the performance of the manufacturing system. We design a testbed to evaluate and validate the proposed methodology with experimental studies. Experimental results show that multiple AI schedulers in the new architecture not only efficiently learn from high-dimensional sensor data but also effectively cope with unexpected events (e.g., simultaneous work orders, machine failures).

Further research will focus on co-learning of multiple material handlers, operation of assembled work orders, and online optimization of AI algorithms. This present paper makes an attempt to realize the distributed AI architecture in smart factories, which shows great potential in improving the smartness of various manufacturing resources. We hope this work will offer valuable insights to implement smart manufacturing in multiple dimensions such as the cloud, the factory edge, and distributed objects.

CRediT authorship contribution statement

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

Declaration of Competing Interest

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

Acknowledgements

This work is supported by the National Key Research and Development Program of China (No. 2020YFB1710500), the National Natural Science Foundation of China (No. 52075257), and the Fundamental Research Funds for the Central Universities (No. NP2020304). The authors express great thanks and appreciation to the editors and anonymous reviewers whose constructive comments significantly improved the paper.

References

[1] J. Harrington, Computer integrated manufacturing, Industrial Press, New York, USA, 1974.
[2] P. Leitão, F. Restivo, ADACOR: a holonic architecture for agile and adaptive manufacturing control, Comput. Ind. 57 (2) (2006) 121–130, https://doi.org/10.1016/j.compind.2005.05.005.
[3] T. Hao, L. Di, S. Wang, Z. Dong, CASOA: an architecture for agent-based manufacturing system in the context of Industry 4.0, IEEE Access 6 (99) (2017) 12746–12754, https://doi.org/10.1109/ACCESS.2017.2758160.
[4] P. Leitão, Agent-based distributed manufacturing control: a state-of-the-art survey, Eng. Appl. Artif. Intell. 22 (7) (2009) 979–991, https://doi.org/10.1016/j.engappai.2008.09.005.
[5] K. Li, T. Zhou, B.-h. Liu, H. Li, A multi-agent system for sharing distributed manufacturing resources, Expert Syst. Appl. 99 (2018) 32–43, https://doi.org/10.1016/j.eswa.2018.01.027.
[6] W. Shen, D.H. Norrie, Agent-based systems for intelligent manufacturing: a state-of-the-art survey, Knowl. Inf. Syst. 1 (2) (1999) 129–156, https://doi.org/10.1007/BF03325096.
[7] H. Yang, S. Kumara, S.T.S. Bukkapatnam, F. Tsung, The internet of things for smart manufacturing: a review, IISE Trans. 51 (11) (2019) 1190–1216, https://doi.org/10.1080/24725854.2018.1555383.
[8] L. Wang, An overview of internet-enabled cloud-based cyber manufacturing, Trans. Inst. Meas. Control 39 (4) (2017) 388–397, https://doi.org/10.1177/0142331216687817.
[9] C. Liu, P. Jiang, W. Jiang, Web-based digital twin modeling and remote control of cyber-physical production systems, Rob. Comput. Integr. Manuf. 64 (2020), 101956, https://doi.org/10.1016/j.rcim.2020.101956.
[10] Y. Lu, X. Xu, Cloud-based manufacturing equipment and big data analytics to enable on-demand manufacturing services, Rob. Comput. Integr. Manuf. 57 (2019) 92–102, https://doi.org/10.1016/j.rcim.2018.11.006.
[11] W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, Edge computing: vision and challenges, IEEE Internet of Things J. 3 (5) (2016) 637–646, https://doi.org/10.1109/jiot.2016.2579198.
[12] M. Pinedo, Scheduling: theory, algorithms, and systems, Springer, New York, USA, 2012, https://doi.org/10.1007/978-1-4614-2361-4 fourth ed.
[13] R.W. Conway, W.L. Maxwell, L.W. Miller, Theory of Scheduling, Addison-Wesley, Reading, MA, USA, 1967.
[14] J. Blazewicz, M. Dror, J. Weglarz, Mathematical programming formulations for machine scheduling: a survey, Eur. J. Oper. Res. 51 (3) (1991) 283–300, https://doi.org/10.1016/0377-2217(91)90304-E.
[15] M. Dai, D. Tang, A. Giret, M.A. Salido, Multi-objective optimization for energy-efficient flexible job shop scheduling problem with transportation constraints, Rob. Comput. Integr. Manuf. 59 (2019) 143–157, https://doi.org/10.1016/j.rcim.2019.04.006.
[16] H.S. Park, N.H. Tran, An autonomous manufacturing system based on swarm of cognitive agents, J. Manuf. Syst. 31 (3) (2012) 337–348, https://doi.org/10.1016/j.jmsy.2012.05.002.
[17] A. Kusiak, M. Chen, Expert systems for planning and scheduling manufacturing systems, Eur. J. Oper. Res. 34 (2) (1988) 113–130, https://doi.org/10.1016/0377-2217(88)90346-3.
[18] N.R. Jennings, M. Wooldridge, Applications of intelligent agents, in: N.R. Jennings, M.J. Wooldridge (Eds.), Agent Technology: Foundations, Applications, and Markets, Springer, Berlin, Heidelberg, Germany, 1998, pp. 3–28, https://doi.org/10.1007/978-3-662-03678-5_1.
[19] Y. Zhang, J. Wang, Y. Liu, Game theory based real-time multi-objective flexible job shop scheduling considering environmental impact, J. Clean. Prod. 167 (2017) 665–679, https://doi.org/10.1016/j.jclepro.2017.08.068.
[20] W.L. Yeung, Agent-based manufacturing control based on distributed bid selection and publish-subscribe messaging: a simulation case study, Int. J. Prod. Res. 50 (22) (2012) 6339–6356, https://doi.org/10.1080/00207543.2011.636763.
[21] L.C. Wang, C.Y. Cheng, S.K. Lin, Distributed feedback control algorithm in an auction-based manufacturing planning and control system, Int. J. Prod. Res. 51 (9) (2013) 2667–2679, https://doi.org/10.1080/00207543.2012.738944.
[22] C.S. Chong, A.I. Sivakumar, Simulation-based scheduling for dynamic discrete manufacturing, in: Proceedings of the 2003 Winter Simulation Conference, IEEE, 2003, pp. 1465–1473, https://doi.org/10.1109/WSC.2003.1261590.
[23] M. Rauf, Z. Guan, S. Sarfraz, J. Mumtaz, E. Shehab, M. Jahanzaib, M. Hanif, A smart algorithm for multi-criteria optimization of model sequencing problem in assembly lines, Rob. Comput. Integr. Manuf. 61 (2020), 101844, https://doi.org/10.1016/j.rcim.2019.101844.
[24] C. Morariu, O. Morariu, S. Răileanu, T. Borangiu, Machine learning for predictive scheduling and resource allocation in large scale manufacturing systems, Comput. Ind. 120 (2020), 103244, https://doi.org/10.1016/j.compind.2020.103244.
[25] M.A. Salido, J. Escamilla, F. Barber, A. Giret, Rescheduling in job-shop problems for sustainable manufacturing systems, J. Clean. Prod. 162 (2017) S121–S132, https://doi.org/10.1016/j.jclepro.2016.11.002.
[26] Y. Koren, M. Shpitalni, Design of reconfigurable manufacturing systems, J. Manuf. Syst. 29 (4) (2010) 130–141, https://doi.org/10.1016/j.jmsy.2011.01.001.
[27] D. Ouelhadj, S. Petrovic, A survey of dynamic scheduling in manufacturing systems, J. Sched. 12 (4) (2008) 417–431, https://doi.org/10.1007/s10951-008-0090-8.
[28] R.J. Abumaizar, J.A. Svestka, Rescheduling job shops under random disruptions, Int. J. Prod. Res. 35 (7) (1997) 2065–2082, https://doi.org/10.1080/002075497195074.
[29] A.S. Kunnathur, P. Sundararaghavan, S. Sampath, Dynamic rescheduling using a simulation-based expert system, J. Manuf. Technol. Manag. (2004), https://doi.org/10.1108/09576060410513779.
[30] M. Ghaleb, H. Zolfagharinia, S. Taghipour, Real-time production scheduling in the Industry-4.0 context: Addressing uncertainties in job arrivals and machine breakdowns, Comput. Oper. Res. 123 (2020), 105031, https://doi.org/10.1016/j.cor.2020.105031.
[31] B. Zhou, J. Bao, J. Li, Y. Lu, T. Liu, Q. Zhang, A novel knowledge graph-based optimization approach for resource allocation in discrete manufacturing workshops, Rob. Comput. Integr. Manuf. 71 (2021), 102160, https://doi.org/10.1016/j.rcim.2021.102160.
[32] D. Li, H. Tang, A semantic-level component-based scheduling method for customized manufacturing, Rob. Comput. Integr. Manuf. 71 (2021), 102144, https://doi.org/10.1016/j.rcim.2021.102144.
[33] R.S. Sutton, A.G. Barto, Reinforcement learning: an introduction, MIT Press, Cambridge, MA, USA, 2018.
[34] W. Zhang, T.G. Dietterich, A reinforcement learning approach to job-shop scheduling, in: International Joint Conference on Artificial Intelligence (IJCAI), Citeseer, 1995, pp. 1114–1120.

[35] X. Chen, X. Hao, H.W. Lin, T. Murata, Rule driven multi objective dynamic scheduling by data envelopment analysis and reinforcement learning, in: 2010 IEEE International Conference on Automation and Logistics, IEEE, 2010, pp. 396–401, https://doi.org/10.1109/ICAL.2010.5585316.

[36] Z. Zhang, L. Zheng, N. Li, W. Wang, S. Zhong, K. Hu, Minimizing mean weighted tardiness in unrelated parallel machine scheduling with reinforcement learning, Comput. Oper. Res. 39 (7) (2012) 1315–1324, https://doi.org/10.1016/j.cor.2011.07.019.

[37] P. Mannion, S. Devlin, J. Duggan, E. Howley, Reward shaping for knowledge-based multi-objective multi-agent reinforcement learning, Knowl. Eng. Rev. (2018) 33, https://doi.org/10.1017/s0269888918000292.

[38] C.J.C.H. Watkins, Learning from delayed rewards, King's College, London, UK, 1989. Ph.D. Thesis.

[39] J. Nash, Non-cooperative games, Annal. Math. 54 (2) (1951) 286–295.

[40] G. Rjoub, J. Bentahar, O.A. Wahab, A. Bataineh, Deep smart scheduling: a deep learning approach for automated big data scheduling over the cloud, in: 2019 7th International Conference on Future Internet of Things and Cloud (FiCloud), IEEE, 2019, pp. 189–196, https://doi.org/10.1109/FiCloud.2019.00034.

[41] T. Zhou, D. Tang, H. Zhu, L. Wang, Reinforcement learning with composite rewards for production scheduling in a smart factory, IEEE Access 9 (2020) 752–766, https://doi.org/10.1109/ACCESS.2020.3046784.