Journal of Manufacturing Systems 57 (2020) 440–450

Contents lists available at ScienceDirect

Journal of Manufacturing Systems

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

Multi-agent system and reinforcement learning approach for distributed intelligence in a flexible smart manufacturing system

Yun Geon Kim [a], Seokgi Lee [b], Jiyeon Son [c], Heechul Bae [c], Byung Do Chung [a,*]

[a] 50 Yonsei-Ro, Seodaemun-gu, Seoul 03722, Department of Industrial Engineering, Yonsei University, Republic of Korea
[b] 1251 Memorial Drive 281, Coral Gables, FL 33146, Department of Industrial Engineering, University of Miami, USA
[c] 218 Gajeong-ro, Yuseong-gu, Daejeon, 34129, Electronics and Telecommunications Research Institute, Republic of Korea

ARTICLE INFO

Keywords:
Distributed decision making
Multi-agent system
Reinforcement learning
Smart manufacturing

ABSTRACT

Personalized production has emerged as a result of the increasing customer demand for more personalized products. Personalized production systems carry a greater amount of uncertainty and variability when compared with traditional manufacturing systems. In this paper, we present a smart manufacturing system using a multi-agent system and reinforcement learning, which is characterized by machines with intelligent agents to enable a system to have autonomy of decision making, sociability to interact with other systems, and intelligence to learn dynamically changing environments. In the proposed system, machines with intelligent agents evaluate the priorities of jobs and distribute them through negotiation. In addition, we propose methods for machines with intelligent agents to learn to make better decisions. The performance of the proposed system and the dispatching rule is demonstrated by comparing the results of the scheduling problem with early completion, productivity, and delay. The obtained results show that the manufacturing system with distributed artificial intelligence is competitive in a dynamic environment.

1. Introduction

The change in the manufacturing paradigm is driven by customer purchasing behavior. As customer demand for personalization of products has increased, flexible and personalized production systems have attracted increasing attention over the past few years. The decision-making problem in a traditional mass production system is repetitive, the current decision is similar to the past decisions, and the decision-making cycle of this system is relatively periodic. A system with these characteristics is more focused on the derivation of an optimal or a near-optimal solution for production planning and scheduling rather than real-time reactivity and agility. Therefore, many studies have been focused on deriving the optimal solution for the system by performing centralized decision-making while reflecting all the information of the system with constructive algorithms [1,2], implicit enumeration [3], meta-heuristic [4,5], heuristic algorithms [6], and approximate algorithms [7].

However, in a personalized production system, a manufacturer should be able to produce a variety of products designed by customers. Therefore, it has an environment wherein orders are received from an unspecified number of customers, with a low-volume high-diversity level of products [8]. In this environment, uncertainty and variability are greater than in traditional manufacturing systems [9]. The characteristics of the flexible and personalized production system differ from those of the traditional manufacturing systems as shown in Table 1. Therefore, centralized decision making with all the information of the factory may be inefficient to respond to the environment. It is important to have a decision-making ability to respond quickly and flexibly with local information and an ability to learn and adapt to a dynamically changing environment as soon as changes are detected in the environment [9–12].

Smart manufacturing systems have been actively studied and there exists research comprising the use of a multi-agent system (MAS) and artificial intelligence (AI), such as reinforcement learning (RL), in areas related to production planning and scheduling in smart manufacturing systems.

The MAS is a system of agents and each distributed agent makes individual decisions based on local information such that the entire system operates efficiently [13]. The use of MAS is useful to respond quickly and efficiently to changing environments in a smart manufacturing system [11,14]. Leung et al. [15] and Wong et al. [16] proposed the use of an algorithm that solves the problem of intelligent

* Corresponding author at: 50 Yonsei-Ro, Seodaemun-gu, Seoul 03722, Department of Industrial Engineering, Yonsei University, Republic of Korea.
E-mail address: bd.chung@yonsei.ac.kr (B.D. Chung).

https://doi.org/10.1016/j.jmsy.2020.11.004
Received 16 August 2019; Received in revised form 6 October 2020; Accepted 3 November 2020
Available online 20 November 2020

Table 1
Characteristics of Traditional Manufacturing Systems and Personalized Production System.

Characteristics	Traditional manufacturing systems	Personalized production system
Product characteristics	Standard	Personalized
Decision-making problem	Repetitive and similar	One-time and new
Decision-making cycle	Periodic	Real time
Manufacturing strategy	Valuable	Flexibility
Manufacturing purpose	Productivity	Sustainability
System attribute	Centralization	Decentralization

process planning and scheduling (IPPS) by combining MAS and ant colony optimization (ACO). The ants are organized as supervisory agents that control the system, and local agents that explore the solution. The supervisory agent makes decisions to derive the solution planning and scheduling solution.

In the MAS-based manufacturing system research, there also exist studies on systems that negotiate with each other when distributed agents make decisions regarding production planning or scheduling. Owliya et al. [17] presented a dynamic scheduling method comprising an algorithm used to allocate tasks using MAS with a bidding method based on a ring structure rather than a contract-net-protocol-based (CNP-based) bidding and negotiation method. Huang and Liao [18] conducted research on a distributed machine scheduling in a parallel machine environment and proposed negotiation and bidding protocol. Wang et al. [19] conducted research on a smart factory framework with MAS and a cyber-physical system, presented a negotiation protocol of agents, and presented a manufacturing system that can efficiently cope with a dynamic environment. Barenji et al. [20] studied the rescheduling of a manufacturing system through MAS in an environment wherein uncertain conditions, such as customer demand fluctuations and machine failures occur. Zhang and Wong [21] used MAS to reschedule flexibly in manufacturing systems under uncertain environments and proposed a method for improving the negotiation mechanism between agents using ACO. Zhang and Wong [22] implemented a flexible manufacturing system using MAS and ACO algorithms in the IPPS problem. Barbosa et al. [23] designed a manufacturing system that took into consideration deterministic and uncertain environments. Recently, Maoudj et al. [24] proposed a distributed multi-agent system (DMAS) composed of supervisory, local, and remote agents. The agents in the DMAS interact and collaborate by allocating and sequencing jobs based on dispatching rules. With illustrative case studies, they were able to show that the proposed DMAS approach outperformed centralized approaches, as well as other type of distributed approaches. There are some studies that compared hierarchical (centralized system) with anarchic structures (distributed system) in relation to the structure of MAS [25–28]. In particular, Ma et al. [25,28] suggested that distributed anarchic system will become a future manufacturing platform rather than a hierarchical structure in an environment of increasing complexity.

Research has been conducted to select appropriate dispatching rules for a manufacturing system applying AI. Metan et al. [29] proposed a scheduling system that selects dispatching rules in real time using AI. Hammami et al. [30] proposed the Labeling from the Current Model framework that enables real-time adaptation and dynamic scheduling in scheduling problems using neural network learning model. Wang and Usher [31] studied a scheduling method in which an agent autonomously senses the environment and selects appropriate dispatching rules using RL. Shahrabi [32] improved the performance of the scheduling method in the dynamic job shop scheduling environment by applying the parameter heuristic optimization method, variable neighborhood search, and RL. Shiue [33] studied the real-time scheduling problem and proposed a method for coping with dynamically changing manufacturing environment by applying RL. Recently, RL has been directly applied to the planning and scheduling problems. Ou et al. [34] studied how to respond to disturbances such as machine failure and machine's waiting in real-time gantry scheduling problems through Q-learning of RL. Leng et al. [35] conducted a study to rearrange the automotive production sequence using RL at color-batching problem with minimal color changeover costs for successive automotive production orders. Hu et al. [36] proposed a method of combining a Petri-net convolution network with a deep Q-network for efficient production and deadlock avoidance in a flexible manufacturing system.

Previous studies on smart manufacturing systems have encountered several challenges. Firstly, most studies using MAS consisted of agents who are responsible for specific functions, such as gathering information related to products or resources, and planning or scheduling. Even if they tried to implement a flexible manufacturing system through autonomous decision making or collaboration among agents, the distributed agents played roles to gather data and support an agent that is responsible for scheduling [37–41]. In other words, although functionally divided agents make distributed decision making, this structure is characterized as a centralized decision-making procedure in terms of production planning and scheduling, which has limitations in terms of autonomy, flexibility, and speed in a dynamically changing environment in real time. Secondly, researches on RL applications deals with the selection of appropriate dispatching rules that can respond quickly when disruptions occur, or whether it will help the system to select a machine as a successor machine in the currently working machine [42–46]. However, the ability to learn the environment through RL is not considered in the MAS-based manufacturing system in cooperation with agents through negotiation and agreement. That is, the negotiation protocol between agents is performed with a fixed rule-based, so it remains a limit in the ability to adapt to the environment.

In order to respond flexibly and quickly, the system requires three functions, which (i) allow the machine to make decisions autonomously, (ii) have the ability to interact with other machines, (iii) to make final decisions and have the intelligence to learn the environment. Therefore, in this study, a smart manufacturing system with intelligent agents is proposed with the implementation of MAS and RL.

2. Architecture and functions of the smart manufacturing system

2.1. System architecture

The proposed smart manufacturing system consists of three layers including the enterprise, cloud, and machine layers as in Fig. 1. The enterprise agent (EA) in the enterprise layer interacts directly with the customer, receives the customer orders, and delivers the job schedule information to the customer.

The cloud layer consists of the database agent (DA) and simulation agent (SA). The DA stores customer order information and production plan and schedule regarding the order, and the SA provides a simulation environment for agents responsible for RL in the machine layer.

The machine layer is responsible for the decision making and autonomous negotiation between the intelligent machines for deciding the production quantity and assigning jobs to machines. The machine at this layer consists of six agents: job agent (JA), negotiation agent (NA), job weight learning agent (WLA), job dropout learning agent (DRLA), job dismiss learning agent (DMLA), and execution agent (EXA).

2.2. Procedure of the smart manufacturing system

There are five functions of the smart manufacturing system in this study: information sharing, job index calculation, negotiation, learning, and execution. Furthermore, these functions are implemented as agents

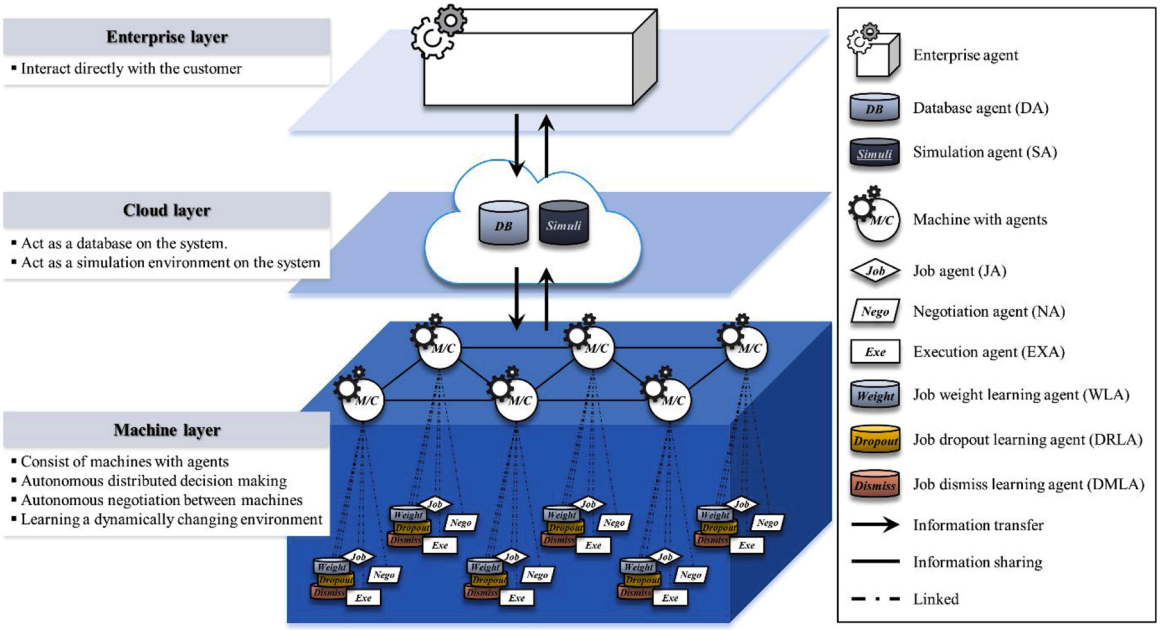


Fig. 1. Smart manufacturing system architecture.

(see Table 2). The process by which agents make decisions when new jobs arrive at the system is presented in Fig. 2.

1) *Step1. New job arrival and information sharing:* When new jobs from customers arrive at the EA at the enterprise layer, the EA uploads the job information to the DA of the cloud layer and requests the JA to produce new jobs.
2) *Step2. Job evaluation and classification for negotiation:* Job evaluation refers to the process in which the JA of each machine evaluates the priorities of jobs based on the current schedule and the job information such as the production quantity, process time, and due date of the jobs. A first-priority job (FPJ) is then selected using the job information along with the weight of the jobs estimated using the WLA. For each machine, the NA shares the FPJ information with other machines and decides whether the FPJs should be classified as an assignment with negotiation (AWN) or assignment without negotiation (AWON).
3) *Step3. Machine negotiation:* If the *FPJ* of a machine is an AWON, it is assigned to the machine without negotiation with other machines. In contrast, if the *FPJ* is an AWN, one or more machines can take the job, and negotiations must be conducted between the machines. Therefore, for an AWN, each machine negotiates through the NA, DRLA, and DMLA to autonomously allocate jobs. The process of machine negotiation is detailed in Section 3.4.
4) *Step4. Repetition of steps 2 and 3:* If there are unallocated jobs, steps 2 and 3 are repeated until there are no remaining jobs.
5) *Step5. Learning and update of reinforcement learning:* When all the jobs are allocated, the WLA, DRLA, and DMLA of each machine learn and update through RL to make the next decisions. The learning and update methods of this proposed system are described in detail in Section 3.5.
6) *Step6. Execution of the results:* When the final planning and scheduling results are derived, the EXA activates the production plan and schedule. In addition, the JA of each machine uploads the job allocation results at the machine layer to the SA in the cloud layer, and the EA in the enterprise layer delivers the information to the customer.

Table 2
The agents of the smart manufacturing system.

Function	Layer	Agent
Information sharing	Enterprise	Enterprise agent (EA)
	Cloud	Database agent (DA)
		Simulation agent (SA)
Job index calculation	Machine	Job agent (JA)
Negotiation	Machine	Negotiation agent (NA)
Learning	Machine	Job weight learning agent (WLA)
		Job dropout learning agent (DRLA)
		Job dismiss learning agent (DMLA)
Execution	Machine	Execution agent (EXA)

3. Distributed intelligence

Distributed intelligent agents at the machine layer provide the machines with autonomy, sociability, and learning ability. This section deals with how machines with intelligent agents can autonomously distribute new jobs through an awareness of the environment and through mutual communication and how to learn environments for improving decision making. The notations used in this work are as follows:

m	index of machine $m = 1, 2, \cdots, M$
M	total number of machines
c	index of candidate machine $c = 1, 2, \cdots, C$
C	total number of candidate machines
t_s^m	start time of machine m
t_s^{nego}	start time of the negotiator machine
j	index of the job that arrived in real time $j = 1, 2, \cdots, J$
J	total number of jobs
i^m	index of the last job in the schedule of machine m
w_j^m	weight of job j calculated from machine m
p_j	processing time of job j
p_j^m	processing time of job j on machine m
$\overline{p^m}$	average processing time of jobs arriving in real time at machine m
d_j	due date of job j
q_j	order quantity of job j
q_j^m	production quantity of job j of machine m in negotiation
s_{ij}^m	setup time for job j on machine m, where job i is the last job in the schedule
$\overline{s^m}$	average setup time of jobs arriving in real time at machine m
s_{ij}^{nego}	setup time for job j in the negotiator machine, where job i is the last job in the schedule

(*continued on next page*)

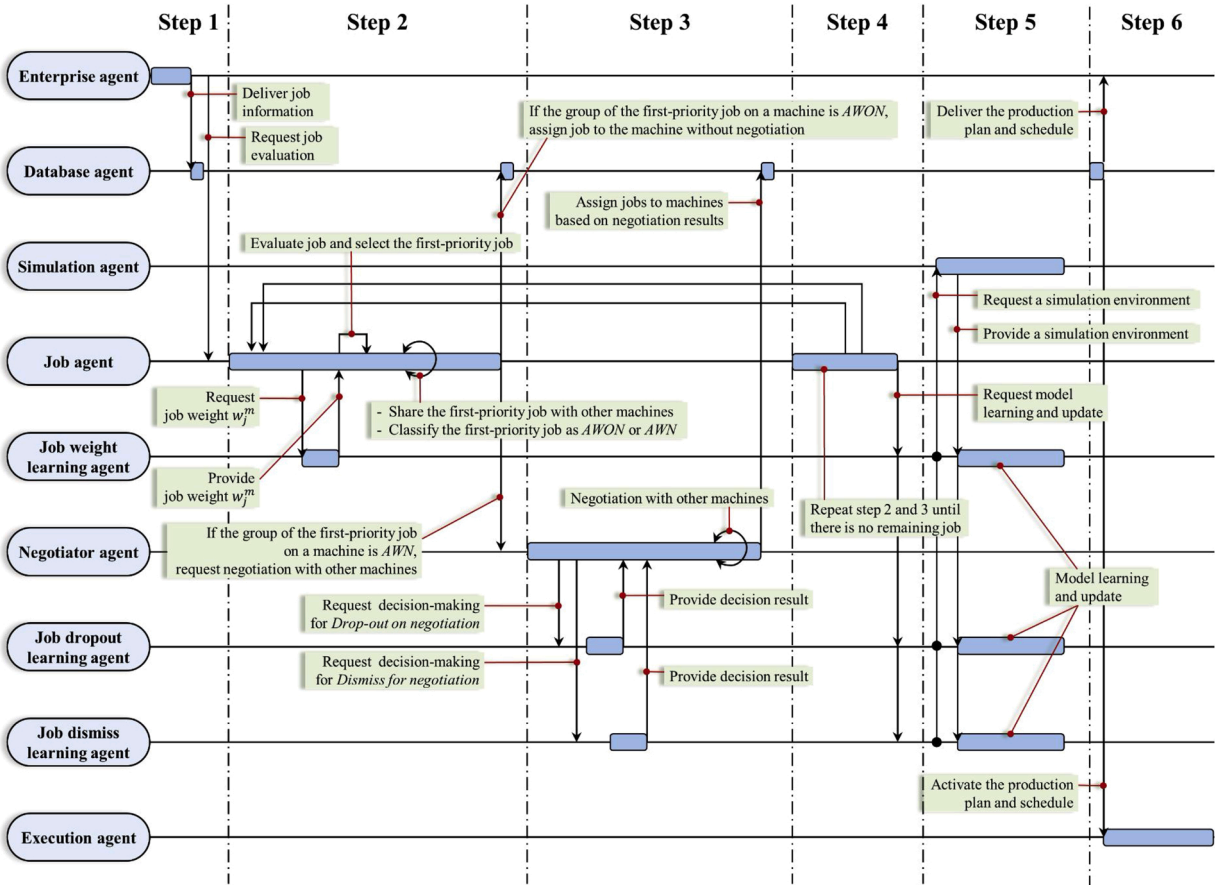


Fig. 2. Smart manufacturing system process.

(*continued*)

$s_{c,ij}^{candi}$	setup time for job j in candidate machine c, where job i is the last job in the schedule

3.1. Job evaluation and first-priority job selection method

When new jobs arrive, the JA of a machine evaluates the priorities of jobs based on the current schedule while taking into consideration the production quantity, processing time, and due date of jobs. This process is called job evaluation as shown in Fig. 3. The priorities of jobs are evaluated through $I_{m,j}(t_s^m|i^m)$ which modified the index function of the apparent tardiness cost with setups (ATCS) [47] as in (1) (see *III.* in Fig. 3):

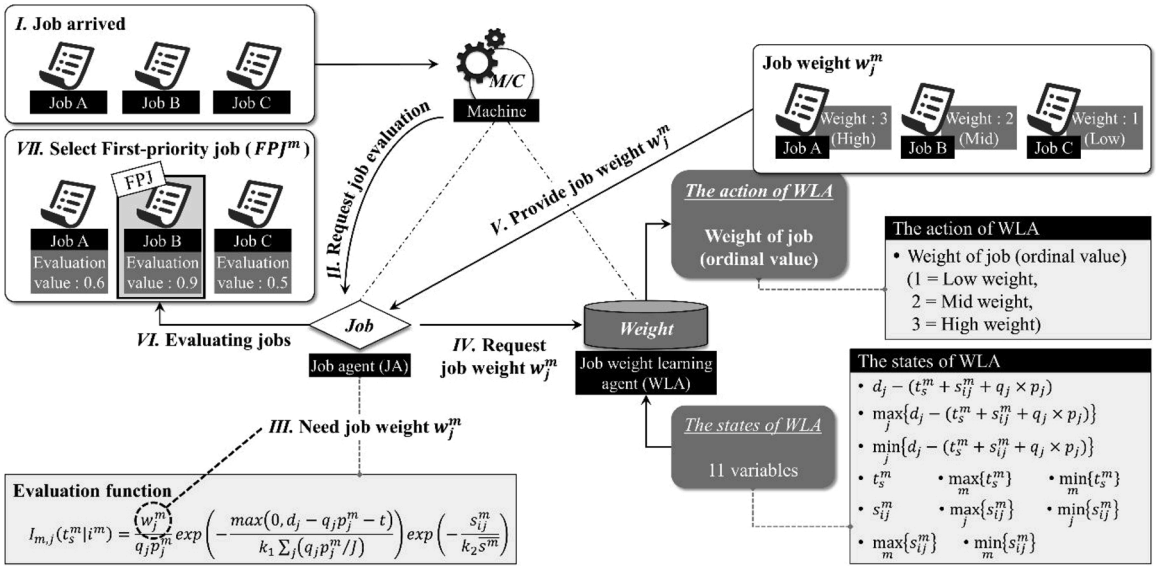


Fig. 3. Job evaluation and reinforcement learning process of WLA.

$$I_{m,j}\left(t_s^m|i^m\right) = \frac{w_j^m}{q_j p_j^m}\exp\left(-\frac{\max\left(0, d_j - q_j p_j^m - t\right)}{k_1 \sum_j \left(q_j p_j^m / J\right)}\right) exp\left(-\frac{s_{ij}^m}{k_2 \overline{s^m}}\right) \quad (1)$$

k_1 and k_2 in (1) represent the due date scaling parameter and setup time scaling parameter and are defined in (2) and (3), respectively [48]:

$$k_1 = \begin{cases} 1.2\ln\mu - R - 0.5, & (\tau < 0.5)\ or\ (\eta\langle 0.5\ and\ \mu\rangle 5\,) \\ 1.2\ln\mu - R, & otherwise \end{cases} \quad (2)$$

$$k_2 = \begin{cases} \dfrac{\tau}{1.8\sqrt{\eta}}, & \tau < 0.8 \\ \dfrac{\tau}{2\sqrt{\eta}}, & \tau \geq 0.8 \end{cases} \quad (3)$$

The index function $I_{m,j}\left(t_s^m|i^m\right)$ consists of five factors: μ, η, β, τ, and R. Let J and M be the number of jobs and machines, respectively. The following five factors are calculated [48].

- Job–machine factor $\mu = J/M$
- Setup time severity factor 1, $\eta = \overline{s^m}/\sum_j \left(q_j p_j^m / J\right)$
- Setup time severity factor 2, $\beta = 0.4 - 10/\mu^2 - \eta/7$
- Due date tightness factor $\tau = 1 - \overline{d}/\left(\beta\overline{s^m} + \sum_j \left(q_j p_j^m / J\right)\right)\mu$
- Due date range factor $R = (d_{max} - d_{min})/\left(\beta\overline{s^m} + \sum_j \left(q_j p_j^m / J\right)\right)\mu$

$I_{m,j}\left(t_s^m|i^m\right)$ combines the properties of the weighted shortest processing time, minimum slack time and shortest setup time first. In (1), w_j^m is the weight of the job and is calculated while taking into consideration the shop-floor and the machine's own environment. The WLA of each machine calculates the weight w_j^m for the jobs by sensing the environment, and the WLA learns dynamically changing environments through RL (see *IV. and V.* in Fig. 3).

After evaluating jobs using (1), the JA of machine m selects the job with the highest value of FPJ^m.

$$FPJ^m = argmax_j\left\{I_{m,j}\left(t_s^m|i^m\right)\right\}, \quad j = 1, 2, \cdots, J \quad (4)$$

3.2. First-priority job classification method

The FPJ^m of each machine is classified as an AWON or AWN. G^m denotes the group to which FPJ^m belongs; it is given in (5).

$$G^m = \begin{cases} AWON,\ if\ FPJ^m\ is\ unique \\ AWN,\ if\ FPJ^m\ is\ not\ unique \end{cases} \quad (5)$$

3.3. Negotiation process among agents

The following procedures are performed for machines with the same first-priority job: 1) negotiator election, 2) drop-out on negotiation, 3) dismiss for negotiation, and 4) job negotiation. Fig. 4 shows the negotiation process.

1) *Negotiator election:* The process selects a machine that leads the negotiation between machines for an AWN job allocation. The earliest available machine will be selected as the negotiator at the beginning of the planning horizon, and other machines will become the candidates. (see *I.* in Fig. 4).
2) *Drop-out on negotiation:* In an identical parallel machine scheduling problem, when the setup time s_{ij}^m exists, the decision of whether to produce a job on a single machine or not is related to the capacity of the plant. When the production quantity is divided over several machines, the completion time of the job can be reduced. In contrast, as the number of machines participating in the job increases, the setup time increases. Therefore, if the shop-floor has sufficient capacity and the time remaining till the job due date is affordable, the production of the job using a single machine will have the effect of eliminating unnecessary setup time in the long term. If there is insufficient capacity, and the time remaining until the due date is short, it would be more efficient to reduce the completion time of the job by dividing the production quantity of the job between machines. Therefore, it is necessary to have the ability to autonomously make decisions regarding whether it is better for machines to divide the production quantity of the job while taking into consideration the capacity of the shop-floor and due date of jobs. In the proposed system, the aforementioned process is implemented through *drop-out on negotiation* and *dismiss for negotiation* among the machines. *Drop-out on negotiation* is a procedure in which the candidate machines drop out of the AWN-job negotiations based on the information of the

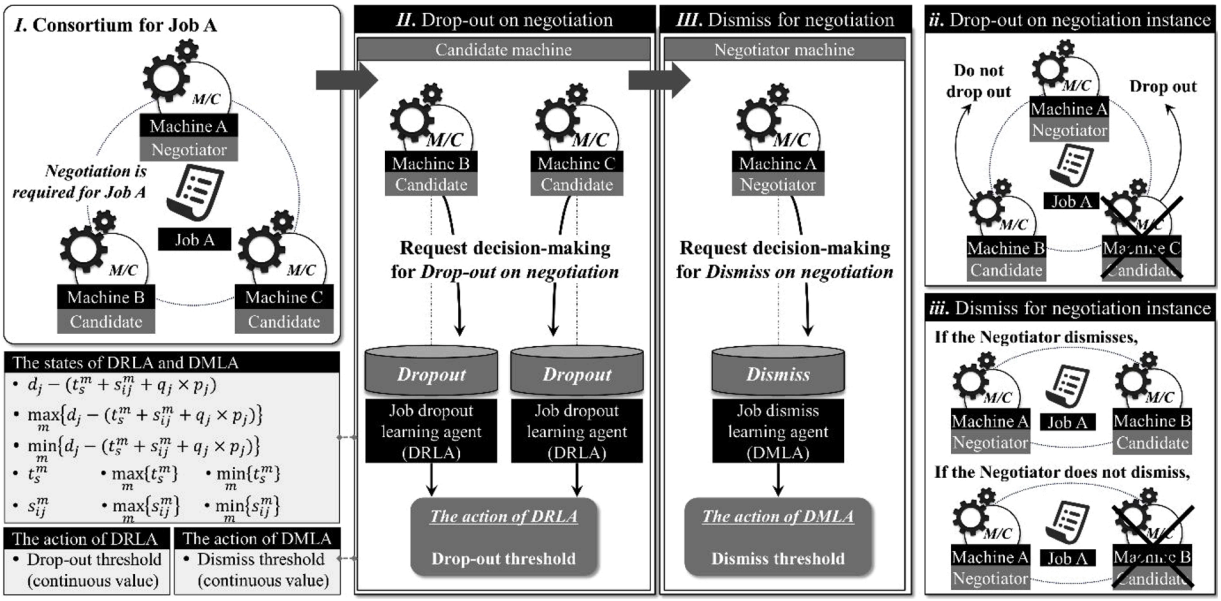


Fig. 4. Negotiation and reinforcement learning process of DRLA and DMLA.

AWN job and shop-floor environment. This is to consider whether it is beneficial to the entire system to leave the production quantity of a job undivided by dropping out of the negotiation. The criteria for the candidate machines to be dropped-out are determined based on the impact of the setup time when completing a job. Let SI_j^{nego} be the impact of the setup time when the negotiator machine produces job j alone and SI_c^{candi} be the impact of the setup time when candidate machine c produces job j with the negotiator. SI_j^{nego} and SI_c^{candi} are evaluated using (6) and (7):

$$SI_j^{nego} = \frac{s_{ij}^{nego}}{s_{ij}^{nego} + q_j \times p_j}, \quad \forall j \tag{6}$$

$$SI_c^{candi} = \frac{s_{ij}^{nego} + s_{c,ij}^{candi}}{s_{ij}^{nego} + s_{c,ij}^{candi} + q_j \times p_j}, \quad c = 1, 2, \cdots, C \tag{7}$$

The decision criteria of the drop-out of negotiation of the candidate machine c is calculated using SP_c^{candi} as given by (8). SP_c^{candi} is the rate of increase in the setup time impact owing to the division of the production quantity by the production quantity as compared with the production of a single machine, which can be interpreted as the increase in inefficiency. The candidate machine c adjusts the *drop-out threshold* with respect to SP_c^{candi} for determining how much of an inefficiency increase will be allowed for determining whether to drop-out on the negotiation, as shown in (9):

$$SP_c^{candi} = \frac{SI_c^{candi} - SI_j^{nego}}{SI_j^{nego}} \tag{8}$$

$$Drop_c^{candi} = \begin{cases} True, \ if \ SP_c^{candi} > drop - out \ threshold \\ False, \ if \ SP_c^{candi} \le drop - out \ threshold \end{cases} \tag{9}$$

The *drop-out threshold* in (9) is adjusted by comparing $d_j - \left(t_s^m + s_{ij}^m + q_j \times p_j\right)$, t_s^m, and s_{ij}^m of the candidate machine c with other machines, and the DRLA adjusts the *drop-out threshold* more precisely using RL. If the value of (9) is *True*, the candidate machine will drop out on the negotiation and fall out of the AWN job negotiation. Otherwise, the candidate machine remains a participant in the negotiation (see *II* and *ii* in Fig. 4).

3) *Dismiss for negotiation:* In this process, the negotiator machine recommends that the participant machines be dismissed in the negotiations on a job. The negotiator machine calculates $Slack^{nego}$, which is the difference between the job's due date and the total remaining process time ($t_s^{nego} + s_{ij}^{nego} + q_j \times p_j$) divided by the difference between the due date and start time, as shown in (10). On comparing $Slack^{nego}$ with a specific threshold—*dismiss threshold*—the negotiator machine makes a decision, which is expressed as follows (11):

$$Slack^{nego} = \frac{d_j - \left(t_s^{nego} + s_{ij}^{nego} + q_j \times p_j\right)}{d_j - t_s^{nego}} \tag{10}$$

$$Dismiss^{nego} = \begin{cases} True, \ if \ Slack^{nego} < dismiss \ threshold \\ False, \ if \ Slack^{nego} \ge dismiss \ threshold \end{cases} \tag{11}$$

The *dismiss threshold* is output from the DMLA of the negotiator machine and is adjusted using RL. If $Dismiss^{nego}$ is *True* according to (11), the participant machines accept the negotiator's dismiss recommendation and give up negotiations on the job. If $Dismiss^{nego}$ is *False*, the participant machines negotiate with the negotiator (see *III* and *iii.* in Fig. 4).

4) *Job negotiation:* The negotiator and participants play the following role during negotiation and use (12) to arrive at a consensus. SI^m in (12) represents the impact of the setup time on the work when the machine production is based on the production quantity suggested by the negotiator to the machines participating in the negotiations using (13). The algorithm for the negotiation is presented in Table 3.

$$Consensus_m = \begin{cases} True, \ if \ SI^m < 0.95 \\ False, \ if \ SI^m \ge 0.95 \end{cases} \tag{12}$$

$$SI^m = \frac{s_{ij}^m}{s_{ij}^m + q_j^m \times p_j} \tag{13}$$

3.4. Learning and updating using reinforcement learning

WLA, DRLA, and DMLA, which are the RL agents of this system, employed a deep Q-network (DQN), an approximation algorithm comprising the use of neural networks in Q-learning [49]. $Q(s, a)$ of the Q-learning is updated using the learning rate α and discount rate γ as in (14) and selects the action a_t^* that maximizes the Q function by sensing the state as in (15):

$$Q^{new}(s_t, a_t) \longleftarrow Q(s_t, a_t) + \alpha\left(r_{t+1} + \gamma \max_{a_{t+1}^{'}} Q\left(s_{t+1}, a_{t+1}^{'}\right) - Q(s_t, a_t)\right) \tag{14}$$

$$a_t^* = argmax_{a_t} Q(s_t, a_t) \tag{15}$$

In this study, the action of the WLA is an ordinal variable and can have three ordinal values: 1, 2, and 3 as a low weight, mid weight, and high weight, respectively. Therefore, the reinforcement model of the WLA deals with the classification problem. In contrast, the actions of the DRLA and DMLA are defined as continuous variables having values between 0 and 1.

The states of the WLA, DRLA, and DMLA are associated with the term $d_j - \left(t_s^m + s_{ij}^m + q_j \times p_j\right)$, t_s^m, and s_{ij}^m. $d_j - \left(t_s^m + s_{ij}^m + q_j \times p_j\right)$ is the difference between the due date of job j and the total remaining process time of machine m, which indicates how much time is remaining based on the due date of job j when job j is completed. Furthermore, t_s^m is the start time of machine m and indicates how quickly the machine m can start its work. s_{ij}^m is the setup time for job j of machine m and indicates how short the setup time of job j of machine m is. The states of the WLA, DRLA and DMLA are described as follows.

As the action of the WLA indicates the weight of the job, it is

Table 3
Algorithm for negotiator and participant's negotiation.

1:		Let K be the number of machines participating in the negotiation.
2:		There are 1 negotiator and $K-1$ participants.
3:		Let c be the index of the participants, and $c = 1, 2, \cdots, K-1$
4:		$Consensus^{nego}$ denotes $Consensus_m$ of the negotiator.
5:		$Consensus_c^{part}$ denotes $Consensus_m$ of a participant c.
6:		Q^{nego} denotes the proportion of production quantity of the negotiator.
7:		Q_c^{part} denotes the proportion the production quantity of a participant c.
8:		Δ denotes the production quantity reduction rate
9:		
10:	Step 1.	Set the Q^{nego} to 100%
11:	Step 2.	IF $Consensus^{nego}$ is True as per (12)
12:		For all c, set Q_c^{part} to $\frac{100 - Q^{nego}}{K-1}$%
13:		ELSE
14:		All machines in negotiation produce $\frac{100}{K}$% of the production quantity
15:		END (Negotiation ends)
16:	Step 3.	IF $Consensus_c^{part}$ is True by (12) for all c
17:		Negotiator produce Q^{nego}
18:		All participants produce Q_c^{part}
19:		END (Negotiation ends)
20:		ELSE
21:		Set the production quantity of the negotiator, Q^{nego} to $Q^{nego} - \Delta$%
22:	Step 4.	Go to Step 2 until END (Negotiation ends)

configured such that job j on machine m is learned relative to other jobs, and machine m is learned through the position relative to the other machines. Therefore, for a relative comparison of job j and other jobs on machine m, the states include the term $d_j - \left(t_s^m + s_{ij}^m + q_j \times p_j\right)$, $\max_j\left\{d_j - \left(t_s^m + s_{ij}^m + q_j \times p_j\right)\right\}$, $\min_j\left\{d_j - \left(t_s^m + s_{ij}^m + q_j \times p_j\right)\right\}$, and s_{ij}^m, $\max_j\left\{s_{ij}^m\right\}$, $\min_j\left\{s_{ij}^m\right\}$. Furthermore, for a relative comparison of the environment of machines m and other machines, the states include t_s^m, $\max_m\{t_s^m\}$, $\min_m\{t_s^m\}$, and s_{ij}^m, $\max_m\left\{s_{ij}^m\right\}$, $\min_m\left\{s_{ij}^m\right\}$. Fig. 3 shows the reinforcement learning process of WLA. The DRLA and DMLA comprise the same states, and the actions of the DRLA and DMLA are related to the negotiations between machines for certain jobs, such that the situation of machine m is configured to be learned and calculated relative to other machines. The states include the term $d_j - \left(t_s^m + s_{ij}^m + q_j \times p_j\right)$, $\max_m\left\{d_j - \left(t_s^m + s_{ij}^m + q_j \times p_j\right)\right\}$, $\min_m\left\{d_j - \left(t_s^m + s_{ij}^m + q_j \times p_j\right)\right\}$, and t_s^m, $\max_m\{t_s^m\}$, $\min_m\{t_s^m\}$, and s_{ij}^m, $\max_m\left\{s_{ij}^m\right\}$, $\min_m\left\{s_{ij}^m\right\}$. Fig. 4 shows the reinforcement learning process of DRLA and DMLA.

4. Computational experiments

The purpose of computational experiments is to illustrate the advantage of the proposed system and compare the performance of the system with benchmarking approaches. This section explains the production environment, test data, and experimental setup. Then, numerical results are discussed in term of effectiveness of the three important functions in the proposed system and performance comparison with dispatching rules and DMAS.

4.1. Production environment and test data

To verify the proposed system performance, a serial production line, consisting of three operations with identical parallel machines, is considered. Data from the data science competition held by Bosch at Kaggle, is used for the experiments. The Bosch data provides information regarding moving parts of an anonymous product manufactured using Bosch's production lines [50]. This data is the production information for approximately 12,000 anonymous products at 52 stations and four production lines. For the convenience of experimentation, all the products of the Bosch data are preprocessed such that three operations are required with five job types. The setup time for each pair of job types is shown in Table 4. We use a random due date and production quantity for each product and processing time for each operation as the experimental data. Uniform distribution is used to generate the data. Through this process, the experimental data is configured to have 677 product orders at 100 different times. For the purpose of performance measure, order arrival time is grouped into 10 clusters and it is defined as a time window. Table 5 shows the number of jobs per batch in the experimental data. Also, the value of Δ, the production quantity reduction in the negotiation algorithm, is set to 5 (line 21 in Table 3). It is assumed that the raw materials are sufficient for manufacturing the products and the transportation time between machines is ignored

Table 4
Setup time s_{ij}^m by job type.

	Job i type				
Job i type	1	2	3	4	5
1	0.0125	0.03	0.035	0.04	0.045
2	0.04	0.0125	0.035	0.045	0.03
3	0.03	0.045	0.0125	0.04	0.035
4	0.03	0.04	0.035	0.0125	0.045
5	0.03	0.04	0.045	0.035	0.0125

4.2. Experimental setup

The performance of the proposed system is evaluated through the following experimental scenarios. First, the effectiveness of the three functions proposed in this paper is tested. The proposed system has three functions of (i) adjusting weight for the job, (ii) dropping the negotiation, and (iii) requesting a machine to be dismissed from the negotiation. The effectiveness of individual or integrated functions is evaluated by selectively embedding these functions as shown in Table 6. SMS is a complete system that includes all three functions and is ultimately proposed in this study. NonDrop is a system that excludes dropping the negotiation among the three functions. NonDismiss is a system that excludes Dismiss for negotiation. OnlyWt is a system that only exists adjusting weight for the job function among the three functions. NonWt is a system that excludes adjusting weight for the job function. Second, the performance of the proposed system is compared with the several dispatching rules, which include the shortest processing time (SPT), earliest due date (EDD), longest processing time (LPT), and list scheduling (LIS). Third, the prposed system is compared with the distributed multi-agent system (DMAS) algorithm of Maoudj et al. [24]. The experiments are implemented using Python 3.0 and executed on a personal computer (Intel® Core™ i7-7700HQ CPU (2.80 GHz) with 16-GB RAM).

4.3. Effectiveness of the three functions

This section presents a comparison of the performance indicators including maximum lateness (L_{max}), the number of tardy jobs (n_T), and makespan (C_{max}) of the systems to validate the effectiveness of the three functions of the proposed model. Firstly, the effects of *Dismiss for negotiation* can be verified by comparing the performance of systems 3 and 4 versus the performance of other systems. In Table 7, as the time window increases, L_{max} of systems 3 and 4 shows a positive value, thus indicating that a tardy job occurs at an earlier time than in other systems. In addition, even if the tardy jobs occur in other systems, it can be confirmed that the time window becomes negative again in time window 10 and satisfies the due date of the jobs. Similarly, Table 8 shows that n_T of systems 3 and 4 increases with the increase in the time window as compared to other systems. In Table 9, for all the time windows, C_{max} in systems 3 and 4 has a larger C_{max} value than other systems. This shows that the system with the *Dismiss for negotiation function* has the ability to reduce the workload as compared to the system without the function in the production system environment wherein the workload of the system increases as the time window increases, and thus the *Dismiss for negotiation* function can be considered to be effective. Secondly, the effectiveness of the *Drop-out on negotiation* function can be observed by comparing systems 1 and 2 or systems 3 and 4. In all the time windows, the values of L_{max} and C_{max} in system 1 are smaller than those in system 2, even if the gap is not significant. In particular, n_T in Table 8 can be found to be the same for both systems 1 and 2. However, when comparing systems 3 and 4, system 3 with the *Drop-out on negotiation* function exhibits a lower value of L_{max}, n_T, and C_{max} than system 4 without the *Drop-out on negotiation* function, and as time window increases, the difference in the performance indicators of the two systems becomes greater. Therefore, the *Drop-out on negotiation* can also be regarded as a function that is sufficiently effective as a function of the proposed system in this study. Thirdly, systems 1 and 5 are compared based on *Adjusting weight* w_j^m to determine their effectiveness. The three performance indicators L_{max}, n_T, and C_{max} of these two systems are approximately equal. However, in time window 7, when both the systems began to have a tardy job, the difference between the two systems tended to increase slightly as compared to the previous time windows. This may imply that the function *Adjusting weight* w_j^m has a slight effect when the production system has a heavy workload.

Table 5
Number of jobs per batch in the experimental data.

Time window	1	2	3	4	5	6	7	8	9	10	Total
Number of jobs	34	54	70	83	99	46	104	99	36	52	677

Table 6
Systems for evaluating the effectiveness of three functions.

Systems	Functions		
	Adjusting weight w_j^m	Drop-out on negotiation	Dismiss for negotiation
1. SMS	✓	✓	✓
2. NonDrop	✓		✓
3. NonDismiss	✓	✓	
4. OnlyWt	✓		
5. NonWt		✓	✓

Table 7
Maximum lateness (L_{max}) of systems by time window.

Time window	Systems				
	1. SMS	2. NonDrop	3. NonDismiss	4. OnlyWt	5. NonWt
1	−7.06	−7.05	−6.47	−6.27	−7.06
2	−4.89	−4.72	−4.24	−3.94	−4.89
3	−3.85	−3.71	−2.56	−1.68	−3.85
4	−4.47	−4.33	−2.56	−0.89	−4.43
5	−2.09	−1.92	0.02	1.89	−2.09
6	−0.61	−0.50	1.90	4.16	−0.62
7	0.94	1.09	4.33	7.18	0.96
8	0.00	0.23	3.97	7.17	0.06
9	0.28	0.51	4.42	7.96	0.33
10	−1.01	−0.67	3.40	7.24	−0.87

Table 8
Number of tardy jobs (n_T) of systems by time window.

Time window	Systems				
	1. SMS	2. NonDrop	3. NonDismiss	4. OnlyWt	5. NonWt
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	1	1	0
6	0	0	1	4	0
7	1	1	4	21	1
8	1	1	5	37	1
9	2	2	6	24	2
10	0	0	11	34	0

Table 9
Makespan (C_{max}) of systems by time window.

Time window	Systems				
	1. SMS	2. NonDrop	3. NonDismiss	4. OnlyWt	5. NonWt
1	1.16	1.18	1.61	1.82	1.16
2	1.67	1.82	2.63	3.17	1.68
3	2.29	2.48	3.71	4.66	2.29
4	3.68	3.84	5.59	7.38	3.67
5	5.06	5.15	7.38	9.4	5.05
6	5.53	5.65	8.19	10.52	5.56
7	6.96	7.09	10.33	13.21	7.04
8	8	8.23	11.97	15.23	8.06
9	8.57	8.85	12.63	16.22	8.72
10	9.49	9.65	13.68	17.57	9.51

4.4. Performance comparison with dispatching rules

The performance of the proposed system is compared with some dispatching rules depending on whether setup time exists or not, as shown in Tables 10 and 11. Bold values represent the best performance among the dispaching rules and proposed approach. In the experiments with setup time, results show that EDD produces a lower L_{max} than the other approaches in most time windows. SMS performs better than others only in time windows 2 and 9. However, SMS always has minimum C_{max} values for all time windows. When there is no setup time, both SMS and EDD have lower L_{max} values than the other systems. Also, SMS and LPT have good performances in terms of C_{max} in the experiments without setup time. Although the production systems relative performance depends on setup time, in general, the proposed system performance has lower L_{max} and C_{max} values than the other dispatching rules; therefore, the proposed system is effective.

4.5. Performance comparison with DMAS

In this section, SMS is compared with the DMAS algorithm proposed in [24]. DMAS is an agent based scheduling and control system for distributed decision making. DMAS is composed of three autonomous agents: 1) Supervisory Agent (SA) for managing and interacting with the other agents, 2) Local Agents (LAs) for making cooperative decisions related to planning and scheduling, and 3) Remote Agent (RA) for carrying out the scheduled operations. The SA and LAs are responsible for the decision side, while the RA is responsible for the physical side. Of these three, we implemented SA and LAs to compare the performance of scheduling results. It should be noted that some experimental environments are different from the original DMAS. First, we consider identical parallel machines and that processing time is not machine-dependent. Second, in this study, the preemptive scheduling problem is considered, but DMAS assumed non-preemptive operations. Therefore, we allow DMAS to make preemptive scheduling by equally dividing execution time of each job. Third, both experiments are considered with and without setup time.

Table 12 and Fig. 5 show L_{max} for the proposed approach and DMAS. As it can be seen in Table 12, SMS outperforms DMAS in the experiments with setup time. Fig. 5-(a) presents a graphical comparison between SMS and DMAS with setup time, in which lateness occurs from time window 5–10 on the DMAS results. In the experiments without setup time, regardless of algorithms, orders are completed within the due date, as shown in Fig. 5-(b).

Table 12 and Fig. 6 show C_{max} results for each approach with and without setup time. In this case, the performance gap between SMS and DMAS is also calculated. The C_{max} performance gap (%) is defined as DMAS over SMS; SMS outperforms DMAS when the gap is greater than 100 %. When setup time is considered, SMS performs better than DMAS,

Table 10
Maximum lateness (L_{max}) of systems according to setup time.

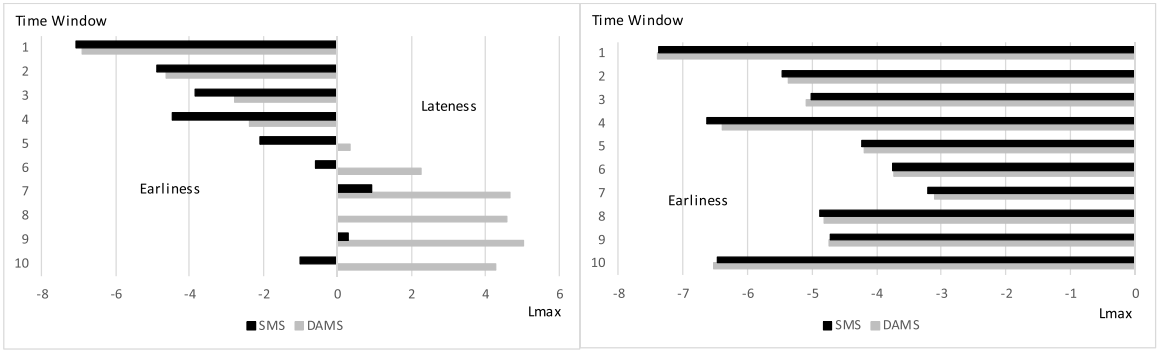
Time window	Experiments with setup time					Experiments without setup time				
	SMS	SPT	EDD	LPT	LIS	SMS	SPT	EDD	LPT	LIS
1	−7.06	−6.41	**−7.26**	−6.65	−6.05	−7.38	−6.95	**−7.42**	−7.26	−6.88
2	**−4.89**	−4.35	−4.87	−4.05	−3.42	**−5.46**	−4.97	−5.2	−4.99	−4.78
3	−3.85	−3.52	**−4.11**	−3.41	−2.29	**−5.02**	−4.74	−4.86	−4.92	−4.76
4	−4.47	−4.23	**−5.46**	−4.14	−2.26	−6.64	−6.57	**−6.74**	−6.48	−6.24
5	−2.09	−0.80	**−2.12**	−0.99	0.12	−4.24	−3.97	**−4.38**	−3.90	−3.59
6	−0.61	0.02	**−0.88**	−0.18	2.62	**−3.76**	−3.45	−3.67	−3.59	−3.49
7	0.94	0.58	**−0.37**	1.00	4.67	−3.22	−3.16	**−3.59**	−3.00	−2.89
8	0.00	0.05	**−0.91**	0.46	4.78	**−4.89**	−4.77	−4.83	−4.65	−4.48
9	**0.28**	1.00	0.31	0.79	5.26	**−4.71**	−4.34	−4.52	−4.48	−4.32
10	−1.01	−0.11	**−1.18**	−0.46	4.45	**−6.46**	−5.94	−6.27	−6.02	−5.82

Table 11
Makespan (C_{max}) of systems according to setup time.

Time window	Experiments with setup time					Experiments without setup time				
	SMS	SPT	EDD	LPT	LIS	SMS	SPT	EDD	LPT	LIS
1	**1.16**	1.66	1.50	1.4	2.10	0.92	1.19	1.22	**0.90**	1.12
2	**1.67**	2.43	1.88	2.03	3.04	**0.93**	1.36	1.19	1.01	1.31
3	**2.29**	3.01	2.5	2.69	4.27	**1.01**	1.39	1.26	1.09	1.35
4	**3.68**	4.94	4.00	4.21	6.12	1.97	2.48	2.23	**1.96**	2.37
5	**5.06**	5.99	5.10	5.65	8.19	**2.16**	2.94	2.35	2.40	2.69
6	**5.53**	6.11	5.88	6.08	8.65	**2.34**	2.87	2.82	2.48	2.73
7	**6.96**	7.57	7.32	7.11	11.37	3.05	3.49	3.29	**3.01**	3.37
8	**8.00**	8.65	8.28	8.58	13.00	**3.29**	3.64	3.67	3.36	3.58
9	**8.57**	9.25	8.87	8.99	13.42	3.60	3.99	3.89	**3.59**	3.92
10	**9.49**	10.12	9.60	9.62	14.48	**3.97**	4.39	4.24	4.01	4.29

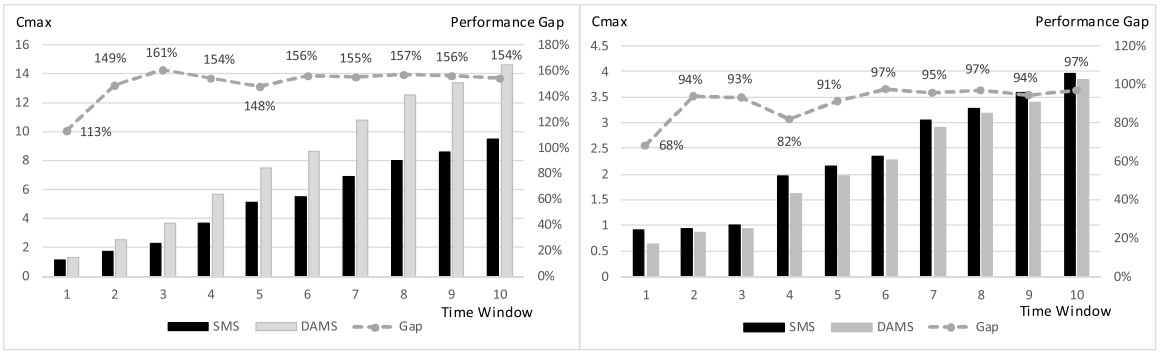
Table 12
Performance comparison with DMAS.

Time window	Experiments with setup time					Experiments without setup time				
	L_{max}		C_{max}			L_{max}		C_{max}		
	SMS	DMAS	SMS	DMAS	GAP	SMS	DMAS	SMS	DMAS	GAP
1	**−7.06**	−6.91	**1.16**	1.31	113%	−7.38	**−7.39**	0.92	**0.63**	68%
2	**−4.89**	−4.63	**1.67**	2.49	149%	**−5.46**	−5.37	0.93	**0.87**	94%
3	**−3.85**	−2.78	**2.29**	3.68	161 %	−5.02	**−5.09**	1.01	**0.94**	93%
4	**−4.47**	−2.40	**3.68**	5.68	154%	**−6.64**	−6.39	1.97	**1.62**	82%
5	**−2.09**	0.35	**5.06**	7.47	148%	**−4.24**	−4.19	2.16	**1.97**	91 %
6	**−0.61**	2.25	**5.53**	8.62	156%	**−3.76**	−3.74	2.34	**2.28**	97 %
7	**0.94**	4.67	**6.96**	10.81	155%	**−3.22**	−3.10	3.05	**2.91**	95%
8	**0.00**	4.57	**8.00**	12.57	157%	**−4.89**	−4.82	3.29	**3.18**	97 %
9	**0.28**	5.03	**8.57**	13.38	156%	−4.71	**−4.73**	3.60	**3.40**	94%
10	**−1.01**	4.27	**9.49**	14.61	154%	−6.46	**−6.52**	3.97	**3.85**	97 %



(a) Experiments with setup time (b) Experiments without setup time

Fig. 5. L_{max} results of SMS and DAMS.



(a) Experiment with setup time (b) Experiment without setup time

Fig. 6. C_{max} results of SMS and DAMS.

but the opposite is observed when setup time is not considered. It is shown that the performance of each approach depends on setup time, but the performance gap behavior is different under each condition. With setup time, the performance gap ranges between 115 and 161 %, with an average of 150 %. Meaning that, on average, SMS performs 50 % better than DMAS. Without setup time, even though DMAS performs better than SMS, the gap has a narrower range (65–97 %) and a lower average gap (91 %). These results show the effectiveness of the proposed SMS.

5. Conclusion

In this study, we presented a smart manufacturing system composed of machines with intelligent agents that have autonomy of decision making, sociability to interact with each other, and the intelligence to learn dynamically changing environments using MAS and RL. The differences between this study and existing studies on smart manufacturing system are as follows: i) applying autonomous distributed decision making of machines to a scheduling problem, ii) implementing a method to evaluate the importance of job by learning dynamically changing environment, iii) implementing the function that machines drop themselves in negotiation on job for the benefit of the entire system, iv) implementing the function that a particular machine requests other machines to drop in the negotiation on the job, v) and implementing the ability to learn, adapt and make appropriate decisions on dynamically changing environments by applying RL to both functions mentioned in iii) and iv).

The significance of the proposed system through the experimental results of this study can be summarized as follows. First, we observed that the function has the effect of evaluating the importance of a job, that is, its priority in the sequence-dependent setup time environment. Second, the performance of the proposed SMS is compared with dispatching rules (SPT, EDD, LPT, and LIS) and a benchmarking distributed MAS algorithm. Numerical results show that SMS is effective and capable of making schedules for a personalized production system.

This study also suggests possibilities for future research. We can first implement a smart manufacturing system that can respond to uncertain situations such as machine failure, order change, cancellation, or rush order. In addition, the various objective of each order can be considered in the personalized production system.

Declaration of Competing Interest

The authors report no declarations of interest.

Acknowledgements

This work was supported by Electronics and Telecommunications Research Institute (ETRI) grant funded by the Korean government [20ZR1100, Core Technologies of Distributed Intelligence Things for solving Industry and Society Problems].

References

[1] Panwalkar SS, Iskander W. A survey of scheduling rules. Oper Res 1977;25:45–61. https://doi.org/10.1287/opre.25.1.45.
[2] Baker KR. Sequencing rules and due-date assignments in a job shop. Manag Sci 1984;30:1093–104. https://doi.org/10.1287/mnsc.30.9.1093.
[3] Potts CN, Van Wassenhove LN. A branch and bound algorithm for the total weighted tardiness problem. Oper Res 1985;33:363–77. https://doi.org/10.1287/opre.33.2.363.
[4] Laguna M, Barnes JW, Glover FW. Tabu search methods for a single machine scheduling problem. J Intell Manuf 1991;2:63–73. https://doi.org/10.1007/BF01471219.
[5] Li L, Qiao F, Wu QD. ACO-based multi-objective scheduling of parallel batch processing machines with advanced process control constraints. Int J Adv Manuf Technol 2009;44:985–94. https://doi.org/10.1007/s00170-008-1904-8.
[6] Vepsalainen AP, Morton TE. Priority rules for job shops with weighted tardiness costs. Manag Sci 1987;33:1035–47. https://doi.org/10.1287/mnsc.33.8.1035.
[7] Hall LA, Schulz AS, Shmoys DB, Wein J. Scheduling to minimize average completion time: off-line and on-line approximation algorithms. Math Methods Oper Res 1997;22:513–44. https://doi.org/10.1287/moor.22.3.513.
[8] Mourtzis D. Challenges and future perspectives for the life cycle of manufacturing networks in the mass customisation era. Logist Res 2016;9:2. https://doi.org/10.1007/s12159-015-0129-0.
[9] Bi Z, Da Xu L, Wang C. Internet of things for enterprise systems of modern manufacturing. IEEE Trans Ind Inform 2014;10:1537–46. https://doi.org/10.1109/TII.2014.2300338.
[10] Denno P, Dickerson C, Harding JA. Dynamic production system identification for smart manufacturing systems. J Manuf Syst 2018;48:192–203. https://doi.org/10.1016/j.jmsy.2018.04.006.
[11] Park JW, Shin M, Kim DY. An extended agent communication framework for rapid reconfiguration of distributed manufacturing systems. IEEE Trans Ind Inform 2018;15:3845–55. https://doi.org/10.1109/TII.2018.2883409.
[12] Tao F, Qi Q, Liu A, Kusiak A. Data-driven smart manufacturing. J Manuf Syst 2018;48:157–69. https://doi.org/10.1016/j.jmsy.2018.01.006.
[13] Karnouskos S, Leitao P. Key contributing factors to the acceptance of agents in industrial environments. IEEE Trans Ind Inform 2016;13:696–703. https://doi.org/10.1109/TII.2016.2607148.
[14] Leitão P, Rodrigues N, Turrin C, Pagani A. Multiagent system integrating process and quality control in a factory producing laundry washing machines. IEEE Trans Ind Inform 2015;11:879–86. https://doi.org/10.1109/TII.2015.2431232.
[15] Leung CW, Wong TN, Mak KL, Fung RY. Integrated process planning and scheduling by an agent-based ant colony optimization. Comput Ind Eng 2010;59:166–80. https://doi.org/10.1016/j.cie.2009.09.003.
[16] Wong TN, Zhang S, Wang G, Zhang L. Integrated process planning and scheduling–multi-agent system with two-stage ant colony optimisation algorithm. Int J Prod Res 2012;50:6188–201. https://doi.org/10.1080/00207543.2012.720393.
[17] Owliya M, Saadat M, Anane R, Goharian M. A new agents-based model for dynamic job allocation in manufacturing shopfloors. IEEE Sys J 2012;6:353–61. https://doi.org/10.1109/JSYST.2012.2188435.
[18] Huang CJ, Liao LM. A multi-agent-based negotiation approach for parallel machine scheduling with multi-objectives in an electro-etching process. Int J Prod Res 2012;50:5719–33. https://doi.org/10.1080/00207543.2011.617394.
[19] Wang S, Wan J, Zhang D, Li D, Zhang C. Towards smart factory for industry 4.0: a self-organized multi-agent system with big data based feedback and coordination. Comput Netw 2016;101:158–68. https://doi.org/10.1016/j.comnet.2015.12.017.

[20] Barenji AV, Barenji RV, Roudi D, Hashemipour M. A dynamic multi-agent-based scheduling approach for SMEs. Int J Adv Manuf Technol 2017;89:3123–37. https://doi.org/10.1007/s00170-016-9299-4.
[21] Zhang S, Wong TN. Flexible job-shop scheduling/rescheduling in dynamic environment: a hybrid MAS/ACO approach. Int J Prod Res 2017;55:3173–96. https://doi.org/10.1080/00207543.2016.1267414.
[22] Zhang S, Wong TN. Integrated process planning and scheduling: an enhanced ant colony optimization heuristic with parameter tuning. J Intell Manuf 2018;29:585–601. https://doi.org/10.1007/s10845-014-1023-3.
[23] Barbosa J, Leitão P, Adam E, Trentesaux D. Dynamic self-organization in holonic multi-agent manufacturing systems: the ADACOR evolution. Comput Ind 2015;66:99–111. https://doi.org/10.1016/j.compind.2014.10.011.
[24] Maoudj A, Bouzouia B, Hentout A, Kouider A, Toumi R. Distributed multi-agent scheduling and control system for robotic flexible assembly cells. J Intell Manuf 2019;30:1629–44. https://doi.org/10.1007/s10845-017-1345-z.
[25] Ma A, Nassehi A, Snider C. Anarchic manufacturing. Int J Prod Res 2019;57:2514–30. https://doi.org/10.1080/00207543.2018.1521534.
[26] Ma A, Nassehi A, Snider C. Balancing multiple objectives with anarchic manufacturing. Procedia Manuf 2019;38:1453–60. https://doi.org/10.1016/j.promfg.2020.01.142.
[27] Ma A, Nassehi A, Snider C. Embracing complicatedness and complexity with Anarchic Manufacturing. Procedia Manuf 2019;28:51–6. https://doi.org/10.1016/j.promfg.2018.12.009.
[28] Ma A, Frantzén M, Snider C, Nassehi A. Anarchic manufacturing: distributed control for product transition. J Manuf Syst 2020;56:1–10. https://doi.org/10.1016/j.jmsy.2020.05.003.
[29] Metan G, Sabuncuoglu I, Pierreval H. Real time selection of scheduling rules and knowledge extraction via dynamically controlled data mining. Int J Prod Res 2010;48:6909–38. https://doi.org/10.1080/00207540903307581.
[30] Hammami Z, Mouelhi W, Said LB. On-line self-adaptive framework for tailoring a neural-agent learning model addressing dynamic real-time scheduling problems. J Manuf Syst 2017;45:97–108. https://doi.org/10.1016/j.jmsy.2017.08.003.
[31] Wang YC, Usher JM. Application of reinforcement learning for agent-based production scheduling. Eng Appl Artif Intell 2005;18:73–82. https://doi.org/10.1016/j.engappai.2004.08.018.
[32] Shahrabi J, Adibi MA, Mahootchi M. A reinforcement learning approach to parameter estimation in dynamic job shop scheduling. Comput Ind Eng 2017;110:75–82. https://doi.org/10.1016/j.cie.2017.05.026.
[33] Shiue YR, Lee KC, Su CT. Real-time scheduling for a smart factory using a reinforcement learning approach. Comput Ind Eng 2018;125:604–14. https://doi.org/10.1016/j.cie.2018.03.039.
[34] Ou X, Chang Q, Chakraborty N. Simulation study on reward function of reinforcement learning in gantry work cell scheduling. J Manuf Syst 2019;50:1–8. https://doi.org/10.1016/j.jmsy.2018.11.005.
[35] Leng J, Jin C, Vogl A, Liu H. Deep reinforcement learning for a color-batching resequencing problem. J Manuf Syst 2020;56:175–87. https://doi.org/10.1016/j.jmsy.2020.06.001.
[36] Hu L, Liu Z, Hu W, Wang Y, Tan J, Wu F. Petri-net-based dynamic scheduling of flexible manufacturing system via deep reinforcement learning with graph convolutional network. J Manuf Syst 2020;55:1–14. https://doi.org/10.1016/j.jmsy.2020.02.004.
[37] Valckenaers P, Van Brussel H, Wyns J, Bongaerts L, Peeters P. Designing holonic manufacturing systems. Robot Comput Integr Manuf 1998;14:455–64. https://doi.org/10.1016/S0736-5845(98)00020-9.
[38] Van Brussel H, Wyns J, Valckenaers P, Bongaerts L, Peeters P. Reference architecture for holonic manufacturing systems: PROSA. Comput Ind 1998;37:255–74. https://doi.org/10.1016/S0166-3615(98)00102-X.
[39] Balasubramanian S, Brennan RW, Norrie DH. An architecture for metamorphic control of holonic manufacturing systems. Comput Ind 2001;46:13–31. https://doi.org/10.1016/S0166-3615(01)00101-4.
[40] Wang L. Integrated design-to-control approach for holonic manufacturing systems. Robot Comput Integr Manuf 2001;17:159–67. https://doi.org/10.1016/S0736-5845(00)00050-8.
[41] Blanc P, Demongodin I, Castagna P. A holonic approach for manufacturing execution system design: an industrial application. Eng Appl Artif Intell 2008;21:315–30. https://doi.org/10.1016/j.engappai.2008.01.007.
[42] Leitão P, Mařík V, Vrba P. Past, present, and future of industrial agent applications. IEEE Trans Ind Inform 2012;9:2360–72. https://doi.org/10.1109/TII.2012.2222034.
[43] Brauer W, Weiß G. Multi-machine scheduling-a multi-agent learning approach. In: Proceedings Int Conference on Multi Agent Systems; 1998 July 3-7; 2002. https://doi.org/10.1109/ICMAS.1998.699030.
[44] Maturana F, Shen W, Norrie DH. MetaMorph: an adaptive agent-based architecture for intelligent manufacturing. Int J Prod Res 1999;37:2159–73. https://doi.org/10.1080/002075499190699.
[45] Paternina-Arboleda CD, Das TK. A multi-agent reinforcement learning approach to obtaining dynamic control policies for stochastic lot scheduling problem. Simul Model Pract Theory 2005;13:389–406. https://doi.org/10.1016/j.simpat.2004.12.003.
[46] Shen W, Maturana F, Norrie DH. Learning in agent-based manufacturing systems. In: Proceedings of AI & Manufacturing Research Planning Workshop; 1998 Aug 31 - Sep 2; 1998.
[47] Lee YH, Bhaskaran K, Pinedo M. A heuristic to minimize the total weighted tardiness with sequence-dependent setups. IIE Trans 1997;29:45–52. https://doi.org/10.1080/07408179708966311.
[48] Lee YH, Pinedo M. Scheduling jobs on parallel machines with sequence-dependent setup times. Eur J Oper Res 1997;100:464–74. https://doi.org/10.1016/S0377-2217(95)00376-2.
[49] Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, et al. Playing atari with deep reinforcement learning. arXiv:1312.5602v1 [Preprint] 2013 [cited 2013 Dec 19]: [9 p.]. Available from: https://arxiv.org/abs/1312.5602.
[50] Kaggle. Bosch production line performance. 2016. https://www.kaggle.com/c/bosch-production-line-performance.