Master's Thesis

석사 학위논문

DQN Learning Approach to Scheduling in Multi-job Production Systems

Jinyoung Kim(김 진 영 金 眞 榮)

Department of Information and Communication Engineering

정보통신융합공학전공

DGIST

2020

Master's Thesis

석사 학위논문

DQN Learning Approach to Scheduling in Multi-job Production Systems

Jinyoung Kim(김 진 영 金 眞 榮)

Department of Information and Communication Engineering

정보통신융합공학전공

DGIST

2020

DQN Learning Approach to Scheduling in Multi-job Production Systems

Advisor: Professor Kyung-Joon Park
Co-advisor: Professor Yongsoon Eun

by

Jinyoung Kim

Department of Information and Communication Engineering
DGIST

A thesis submitted to the faculty of DGIST in partial fulfillment of the requirements for the degree of Master of Science in the Department of Information and Communication Engineering. The study was conducted in accordance with Code of Research Ethics [1]

11. 17. 2020

Approved by

Professor Kyung-Joon Park (Advisor) (Signature)

Professor Yongsoon Eun (Co-Advisor) (Signature)

[1] Declaration of Ethical Conduct in Research: I, as a graduate student of DGIST, hereby declare that I have not committed any acts that may damage the credibility of my research. These include, but are not limited to: falsification, thesis written by someone else, distortion of research findings or plagiarism. I affirm that my thesis contains honest conclusions based on my own careful research under the guidance of my thesis advisor.

DQN Learning Approach to Scheduling in Multi-job Production Systems

Jinyoung Kim

Accepted in partial fulfillment of the requirements
for the degree of Master of Science.

11. 17. 2020

Head of Committee Prof. Kyung-Joon Park (인)

Committee Member Prof. Yongsoon Eun (인)

Committee Member Prof. Jihwan Choi (인)

MS/IC
201922011

김 진 영. Jinyoung Kim. DQN Learning Approach to Scheduling in Multi-job Production Systems. Department of Information and Communication Engineering. 2020 . 25p. Advisor Prof. Kyung-Joon Park. Co-Advisor Prof. Yongsoon Eun

ABSTRACT

Multi-job production is a class of manufacturing systems that produce different products within the same production system. These systems are widely used in production assembly, and becoming a trend with smart factories. In this paper, we propose a Deep-Q reinforcement learning driven scheduling algorithm for multi-job production. In particular, we take into account machine breakdown and production plan change as the inputs of the learning process, which are typically considered as unexpected situations in previous studies. We validate the proposed scheme with real data collected for 6 months between May and October 2019 from a tier-1 vendor of a world top-4 motor company. Our case study shows that the proposed scheme improves the throughput of the production line by 37% compared to the conventional rigid method.

Key words: Multi-job line, production scheduling, reinforcement learning, data processing, flexible production systems.

Contents

List of Figures

List of Tables

Notation, Symbols, and Acronyms

Symbols

M_i	i-th machine in the production line
u	average up-time
d	average down-time
e	efficiency of the machine
N_M	the total number of machines in production line
N_B	the total capacity of buffers in production line
N_J	the number of product models
J_i	i-th product model in N_J
J	the set of all J_i
G_i	output goal of J_i
S_i	current output of J_i
A	the set of product model J_i which is $G_i > S_i$
$P_{(x,y)}$	behavior pattern that machine x and y work
T	time to accomplish the goal G_i for every J_i

Notation

- The difference between the training phase and the validation phase is the presence or absence of exploration.
- In the random scheduling method, model is selected according to ratio of remaining items.

Acronyms

IoT	internet of things
RL	reinforcement learning
AI	artificial intelligence
DQN	deep-q network
AMHS	automated material handiling system
RFID	radio frequency identification
WIP	work in process
NN	neural network

1 Introduction

Smart factories are being widely deployed with the development of the Internet of things, 5G, big data, and most of all, artificial intelligence (AI) and machine learning (ML) [1,2]. Governments are in the process of preparing for Industry 4.0, starting from Germany [3]. Companies are also making efforts to implement smart factories to obtain a variety of benefits such as production flexibility, improved output capacity, reduced set-up cost and fewer errors and machine downtimes, and so on.

Efficient production scheduling is crucial for improving the throughput of the production line without structural change in the line. Since most manufacturing scheduling problems are NP-hard, it is difficult to derive optimal solutions within a reasonable time [4]. Furthermore, scheduling is getting more complex with multi-job production, which produces different products within the same production system. Multi-job production systems are widely used in production assembly, and becoming a trend with smart factories. In order to tackle the complex production scheduling problems, ML techniques are introduced [5]. Among the scheduling problems, those formulated as a Markov decision process can be solved with reinforcement learning (RL) [6]. Recently, the Deep-Q network reinforcement learning is applied to the scheduling problem [7–9].

One of the most critical issues in ML-driven production scheduling is how to deal with change in production schedules and machine breakdown, which are typically considered as unexpected situations. Recently, as the potential for rescheduling due to external factors increases, the importance of coping with changes in production schedules has emerged as a critical issue

In this paper, we investigate a multi-job scheduling problem. More specifically, the contributions of the paper are as follows:

- We propose a Deep-Q reinforcement learning driven scheduling algorithm for multi-job production systems. Unlike existing studies, we additionally take into account both machine breakdown and production plan change as the inputs of the Deep-Q network.
- Based on real factory data collected for 6 months from a tier-1 vendor of a world top-4 motor company, we validate our proposed scheduling scheme. Our case study shows that the proposed algorithm improves throughput performance by 37% compared to the conventional rigid approach.

The rest of the paper is organized as follows. Section II reviews related work. We introduce details of the production line and the data processing and formulate the overall scheduling problem in Section III. We introduce the Deep-Q reinforcement learning model in Section IV. We present performance evaluation in Section V. Finally, our conclusions follow in Section VI.

2 | Related Work

There are multiple approaches to improve the throughput of the production line. One way is to change the structure of the production line such as the size of the buffers [10]. Due to space limitations and budget issues, it is difficult to change the line structure in practice. Another approach for increasing the throughput without structural change is efficient scheduling, which applies to various industries [2, 11–14].

It is possible to derive the optimal solution for scheduling of a simple process involving only one machine [13]. However, the optimal solution for a production line in practice is typically too complex to obtain the optimal scheduling within a reasonable time [5, 12]. Consequently, machine learning is introduced to tackle this problem. Genetic algorithm is a typical example [5]. Recently, after the development of Deep-Q reinforcement learning [15], Deep-Q-based methods are widely used [2, 7, 8, 11, 13].

The main objectives of the existing Deep-Q based scheduling studies are as follows. In job-shop scheduling, the order of the workplace reservation is determined by considering waiting operations, setup status, action history, and utilization history [7]. The automated material handling system (AMHS) scheduling problem considers the remaining processing time, facility states, transportation time and traffic congestion, work-in-process distribution, and intermediate buffer states to determine the appropriate vehicle and route [8]. In the chemical process, the order of work is decided by considering resource and inventory, demand and transition losses [9]. However, all these studies do not explicitly consider machine breakdown, which occurs frequently [10].

As mentioned above, many existing studies contribute to improving production throughput by adopting machine learning techniques. However, the breakdown of the machines, which significantly affects the throughput of the production line, is not properly considered. Therefore, the existing ML-driven scheduling methods have limitations for applica-

tions in practice. Unlike these studies, we collect real factory machine data through RFID sensors and propose a Deep-Q RL based multi-job scheduling algorithm that considers machine breakdown information obtained by processing the collected data.

3 | Target Manufacturing System

The plant covered in this study is that of an auto parts maker, which is a tier-1 vendor for one of the world top-4 motor companies. We investigate a newly introduced assembly line of the plant. We collect data of the line from May to October of 2019. We explain in detail the way we process the collected data. According to the results obtained by processing the data, the scheduling in the plant does not change until the production goal is achieved. This is a simple rigid schedule system, which is actually inefficient in terms of throughput performance. In fact, a multi-job line that produces multiple products in random order may give higher throughput than rigid scheduling [16].

Fig. 3.1 shows an illustrative example. A multi-job line that produces model A, B, and C gives higher throughput when it schedules models in random order (Fig. 3.1(b)) than when it produces models in a fixed order (Fig. 3.1(a)). Fig. 3.2 shows the throughput of random scheduled multi-job line according to the job ratio r_1 and r_2 [16]. The r_1 and r_2 are the ratios of models A and C and models B and C, respectively. The dotted triangle is the throughput of the rigid scheduled multi-job line. As shown in Fig. 3.2 (b), in a random schedule, the job that has the greatest effect on the bottleneck determines the multi-job line's throughput, and has higher throughput than the rigid schedule.

3.1 Data Processing

The data from the assembly line is collected every second by RFID sensors [17]. Fig 3.3 is a part of the raw-data. To comply Non-Disclosure Agreement, information about specifications and models is excluded. The collected data includes information on the machine status, pallet number, inspection status, work required time, logging time, vehicle model, part number, and specifications. Data is saved in a csv file format each hour. Information

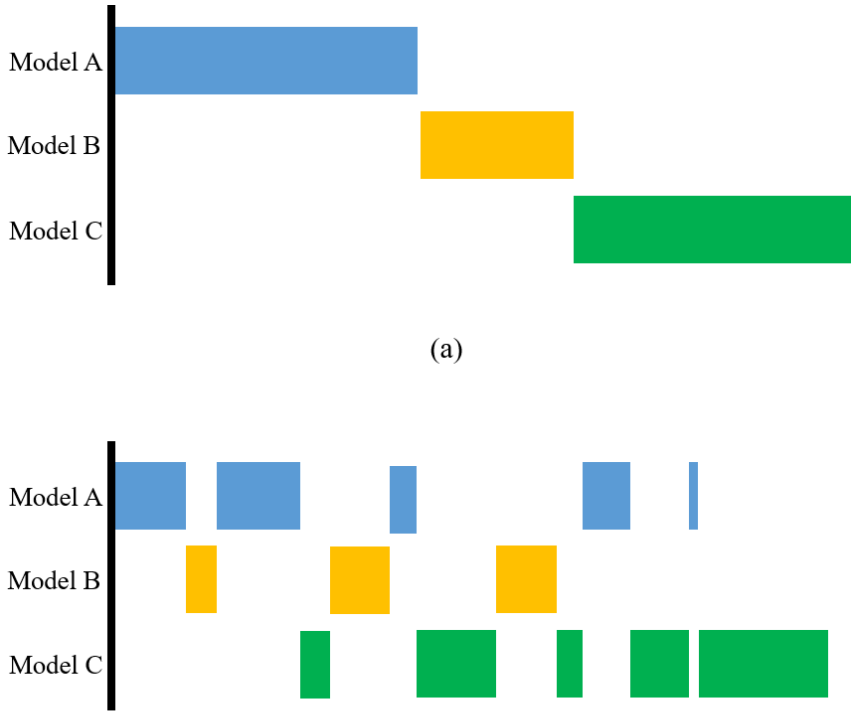

Figure 3.1: Comparison of production schedules. (a) Rigid schedule. (b) Flexible schedule

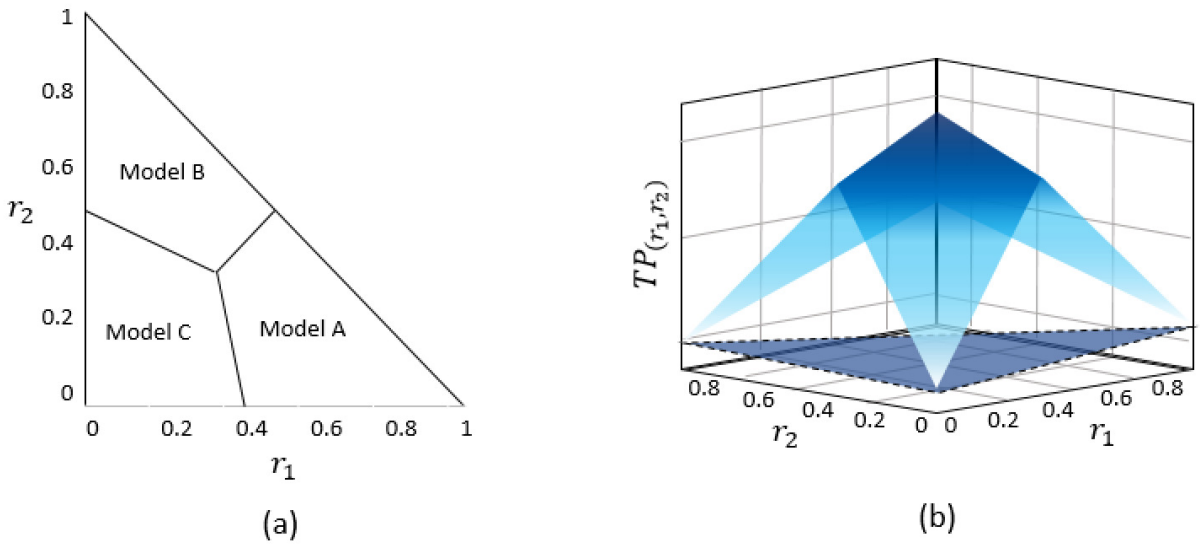

Figure 3.2: Throughput comparison between rigid and random scheduling for multi-job production. (a) Bottleneck simplex. (b) Throughput.

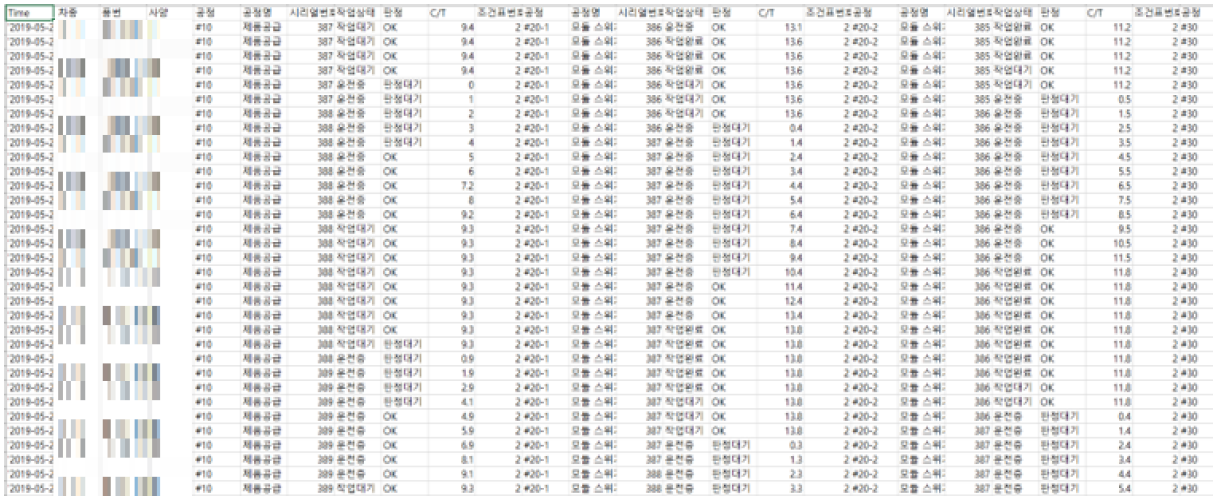

Figure 3.3: An example of raw-data

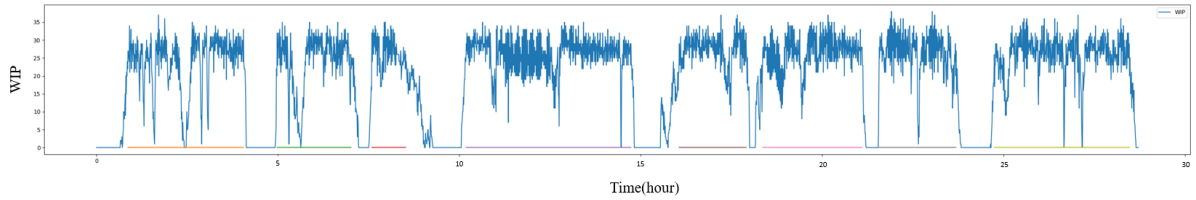

Figure 3.4: WIP of the assembly line.

about each of the machines on the line, the produced item and buffer was obtained by processing these csv files.

3.1.1 Work Time

Because data is collected 24 hours a day, it also includes information when the plant is shut down. Fig. 3.4 shows work-in-process (WIP) of the assembly line for a day. from the figure, we can find when the line is interrupted. Causes of interruption include worker break times, machine breakdowns, and product changes. We need to distinguish whether the plant interruptions are intentional or not. This is because intentional interruptions are not taken into account when calculating the efficiency of machines. By comparing the work hours of several work days, we distinguish actual work time. Interruptions within the common work schedule are judged to be unintentional interruptions. The line at the bottom of the graph in Fig. 3.4 is the work time of the factory. Therefore, it is an unintentional stoppage of the production line that the work-in-process becomes zero above this work time.

3.1.2 Machine Parameters

There are several parameters that can specify the nature of the machine [18]. In this paper, cycle time, average up-time, average down-time, and efficiency are extracted from factory data and used for the regularization of production items and simulation environments.

The products are produced during the corresponding work performed on each machine for a certain period of time. This time is called the cycle time. Different types of production items have different cycle times on the machine. In real data, the amount of time that an item stays on the same machine, even when the same operation is repeated, is not an

Index	average_uptime	average_downtime	machine	e	ct	product
0	0 days 00:24:03.0000…	0 days 00:00:00.000000…	#10	1	0 days 00:00:14.000…	135
1	0 days 00:24:03.0000…	0 days 00:00:00.000000…	#20-1	1	0 days 00:00:16.000…	135
2	0 days 00:24:10.0000…	0 days 00:00:00.000000…	#20-2	1	0 days 00:00:12.000…	136
3	0 days 00:24:20.5000…	0 days 00:00:00.000000…	#30	1	0 days 00:00:17.000…	137
4	0 days 00:24:29.5000…	0 days 00:00:00.000000…	#40	1	0 days 00:00:13.000…	137
5	0 days 00:24:40.0000…	0 days 00:00:00.000000…	#50-1	1	0 days 00:00:19.000…	138
6	0 days 00:24:48.0000…	0 days 00:00:00.000000…	#50-2	1	0 days 00:00:13.000…	138
7	0 days 00:24:57.5000…	0 days 00:00:00.000000…	#60	1	0 days 00:00:14.000…	138
8	0 days 00:11:59.7500…	0 days 00:00:45.666666…	#70-1	0.940338	0 days 00:00:15.000…	139
9	0 days 00:12:04.2500…	0 days 00:00:45.666666…	#70-2	0.940686	0 days 00:00:17.000…	140
10	0 days 00:25:29.5000…	0 days 00:00:00.000000…	#80	1	0 days 00:00:12.000…	140
11	0 days 00:26:28.5000…	0 days 00:00:00.000000…	#90	1	0 days 00:00:16.000…	141
12	0 days 00:26:36.5000…	0 days 00:00:00.000000…	#100	1	0 days 00:00:18.000…	141
13	0 days 00:26:45.5000…	0 days 00:00:00.000000…	#110	1	0 days 00:00:17.000…	141
14	0 days 00:26:54.5000…	0 days 00:00:00.000000…	#120	1	0 days 00:00:25.000…	71
15	0 days 00:27:10.5000…	0 days 00:00:00.000000…	#130	1	0 days 00:00:16.000…	141
16	0 days 00:27:16.5000…	0 days 00:00:00.000000…	#140	1	0 days 00:00:34.000…	71
17	0 days 00:27:42.5000…	0 days 00:00:00.000000…	#150	1	0 days 00:00:30.000…	142
18	0 days 00:27:54.5000…	0 days 00:00:00.000000…	#160	1	0 days 00:00:16.000…	143
19	0 days 00:46:35.5000…	0 days 00:00:00.000000…	#170	1	0 days 00:00:10.000…	0

Figure 3.5: An example of the machine parameters

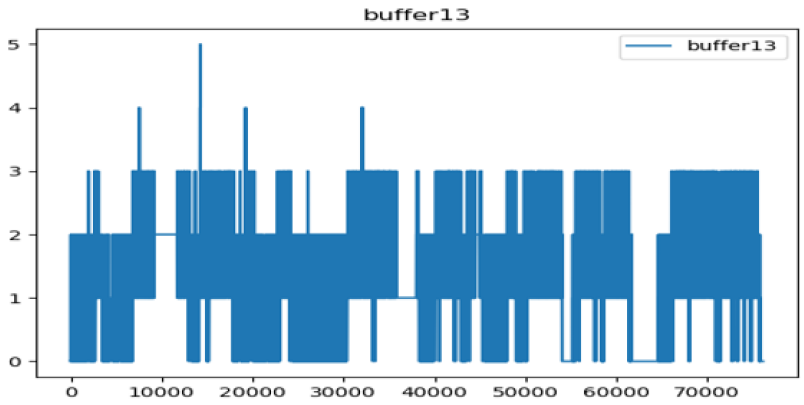

Figure 3.6: An example of buffer data

exact constant value. The cause may be defective product, machine breakdown, blockage, etc. In order to deduce an accurate cycle time, the minimum value was obtained from the data when no issues such as machine breakdown or blockage.

Blockage, starvation, machine breakdown can stop the machines, but among these, blockage and starvation are flow problems so do not affect the down-time of the machine. Hence, only machine interruptions due to machine breakdown are counted as down-time. Up-time is defined as the period between down-time and the next down-time. All down-times and up-times are measured for each machine, and the average down-time and average up-time are calculated. A machine's efficiency e is calculated as $e = u/(u + d)$ when u is the average up-time and d is the average down-time.

Fig 3.5 shows an example of machine parameters form the processed data. There are average up-time, average down-time and ct for each machine. The e is efficiency which is calculated from average up-time and average down-time.

3.1.3 Buffers

The buffer is the space between machines where the product can wait. The fig 3.6 shows an example of buffer data. It show how many units stay in the buffer according to time(s). The maximum capacity that can store the product for each buffer is determined. The capacity of the buffer can be inferred through the difference between the serial numbers of the products in progress for each machine. The following is the structure of the factory

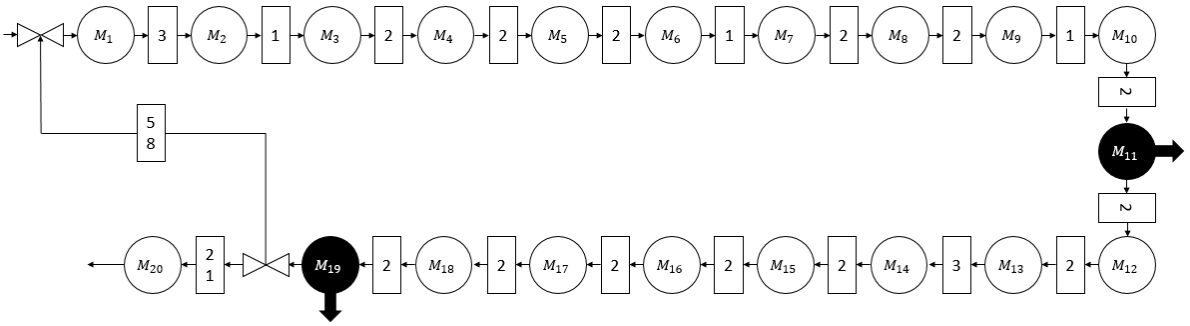

Figure 3.7: Structure of the assembly line.

and the capacity of the buffer obtained by analyzing the data.

3.2 Factory Description

The assembly line consists of 20 machines and 19 buffers. Workers put products into M_1 and collect finished products from M_{20}. The product is transported by a pallet, and the pallet is collected at M_{19} and moved to M_1. There is no loss of productivity due to insufficient or excessive number of pallets. Defective products are removed from M_{11} and M_{19}.

Fig. 3.7 shows the structure of the assembly line. The circles and squares indicate machines and buffers, respectively. The numbers in the circles and squares denote the order of work and the capacity of the buffer. M_i means the i-th machine. The machine colored black eliminates defective products. The valve symbol between M_{19} and M_{20} represents the branches of the pallet carrying goods.

This assembly line is capable of producing various models. When the model is changed, the RFID tag containing information about the target production model is put on the pallet and passed through the machines, and the machine reads the information and automatically changes the settings.

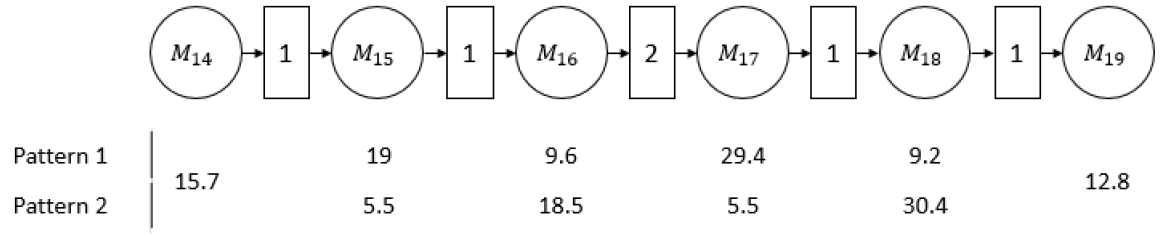

Figure 3.8: Distinctive process pattern between M_{15} and M_{18}.

3.3 Distinctive Feature of the Line

A distinctive behavior pattern on the assembly line exists between M_{15} and M_{18}. Because the assembly line is not designed to operate in parallel due to budget and space issues, instead it is designed by connecting machines doing the same work in series. M_{15} and M_{16}, M_{17} and M_{18} are pairs that each handle the same task. Fig. 3.8 shows the cycle times of machines $M_{14}, \cdots, M_{19}$. Except M_{14} and M_{19}, all the other machines M_{15}, M_{16}, M_{17}, and M_{18} have two cycle times. In Pattern 1, the products are assembled at M_{15} and M_{17}, and just pass M_{16} and M_{18}. The opposite occurs in Pattern 2. Since the same task is performed, the cycle time of M_{15} in Pattern 1 and the cycle time of M_{16} in Pattern 2 are the same.

3.4 Scheduling Problem

The problem dealt with in this paper is to select the model to be the next input and the working machine between M_{15} and M_{19}. The total number of machines in the assembly line is expressed as N_M, the total capacity of the buffer N_B, and the number of product models is expressed as N_J. The i-th product model in N_J is expressed as J_i, and the cycle time of J_i in M_k is expressed as $ct(J_i, M_k)$. When the output goal of J_i is G_i and the current output of J_i is S_i, the set consisting of all J_i is defined as J, and the set consisting of J_i, which is $G_i > S_i$, is defined as A.

Behavior patterns are organized into four categories. When we define the pattern working in M_x and M_y in $M_{15}, \cdots, M_{18}$ as $P_{(x,y)}$, the possible pattern is $P_{(15,17)}$,$P_{(15,18)}$,$P_{(16,17)}$

or $P_{(16,18)}$. This pattern is applied after M_{15}, but since the machine's setting in the factory is determined by the RFID tag inserted in M_1, the production model selection and pattern selection must be made at the same time. In summary, the problem to be solved in this paper is to select the model and pattern to be invested to minimize T, which is the time to accomplish the goal G_i for every J_i.

4 | Deep-Q Scheduling

In this section, we introduce a scheduling method based on deep Q-network. By processing the data obtained from the factory, we build a virtual production line. Then, we train the deep Q-network in the virtual production line. Fig. 4.1 shows the structure of the overall process. The virtual production line is a discrete-time event simulator, which provides the state of the production line to the Q-network and receives an appropriate model and pattern.

The reinforcement learning method used here, the deep Q-network, is introduced in [15]. The states and rewards of the production line caused by the production models and behavior patterns determined through the Q-network are stored as a set in the replay buffer. When the amount stored in the replay buffer exceeds a certain level, a randomly selected set is used for learning. For the stationary target network, the weight of the Q-network is periodically copied to the target Q-network. The model with the best performance in the learning process is saved and used in the simulation phase to evaluate performance.

4.1 Agent and Environment of RL

In reinforcement learning, the agent takes action on the environment and learns by feedback on it. Here, the environment is production line which consists of 20 machines and 19 buffers as introduced in Section III, and has the same constraints as the actual production line. Based on the machine efficiency obtained by processing the data obtained from the factory, the breakdown event of the machine is implemented according to the exponential model.

In this section, we explain the process of deep-Q learning. The agent selects the ac-

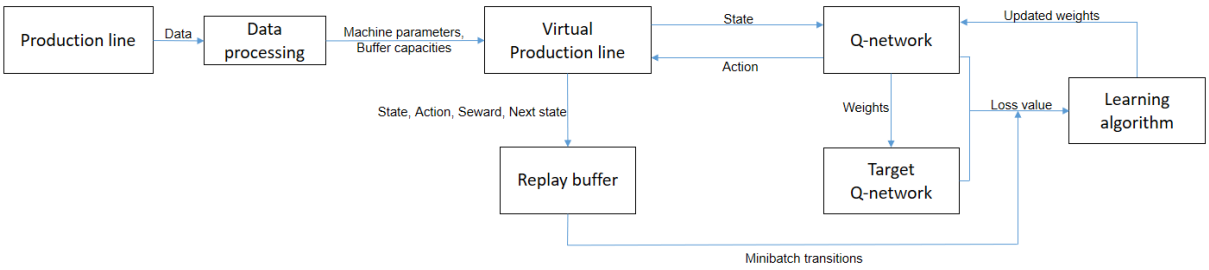

Figure 4.1: Overall framework of the learning process.

tion with the highest Q-value by entering the state of the environment into the neural networks (NN) of the Q-network. The NN is suitable for expressing the possible states of environment. After taking the action a_i in the state s_i, the state s_{i+1} which is returned by the environment and the reward r_i, d_i indicating whether production is over are all saved in the replay buffer as a batch $(s_i, a_i, s_{i+1}, r_i, d_i)$.

4.2 States, Actions, and Rewards

The agent determines the model and pattern to put on M_1. Actions can be taken when M_1 is empty and in up-time. So, the environment returns the next actionable state when the agent takes an action. For example, if the action is $a_i = (J_k, P_{(15,17)})$, the k model is put in M_1, and the put items are operated in M_{15}, M_{17}. And it just go through M_{16}, M_{18}. If an action is possible when the environment is s_i, the set of actions that can be selected is expressed as $A(s_i)$. Completed models are excluded from A.

Here, s_i contains information about the plant components. Table 4.1 represents the state components. The state consists of information on the production items in progress for each machine, whether the machine is operating, and information on the production items staying in the buffer. The item's model and pattern information is expressed as ct for each machine. With this notation, additional learning or regularization is not required even if there are unlearned items in the line. Also, this can be applicable to buffers.

If there is no item remaining in the buffer or machine, it is expressed as a zero vector of the same dimension. Residual time means the additional amount of time the machine has to work. When residual time is zero, the machine discharges the item to the next buffer. If the machine is in down-time, residual time does not decrease. Residual time is

Table 4.1: Components of a state

	Features	**Descriptions**	**Dimension**
Machines	Item	Information on item being worked on the machine containing ct with patter applied	$(N_M)^2$
	Residual time	The time left for the machine to finish its job	N_M
	Breakdown	Whether the machine is broken	N_M
Buffers	Item	Information on item being stay in the buffer containing ct with pattern applied	$N_M \times N_M$
Models	Stock	Remaining work by model	N_M

also expressed as zero when the machine can not discharge items because of a blockage. Breakdown is expressed by 0 or 1, indicating whether the machine is broken or not, respectively.

The reward is determined according to the item released by M_{20} between s_i and s_{i+1}. Rewards are determined for items discharged by M_{20} between s_i and s_{i+1}. The sum of the rewards must fit well with the objective function. Therefore, the sum of rewards should be designed to minimize T. Hence, we have the following relations:

$$CT_a = \sum_{l=1}^{N_M} ct(J_a, M_l), \tag{4.2.1}$$

$$r_i = \begin{cases} 0, & N_{out} = 0 \\ \sum_{k=1}^{N_{out}} -(\tau_{out}(J_k) - \tau_{in}(J_k) - CT_k) & N_{out} > 0. \end{cases} \tag{4.2.2}$$

Equation (4.2.1) corresponds to the sum of J_a's every cycle time for each machine. N_{out} is the number of items discharged between s_i and s_{i+1}. In (4.2.2), we can notice that r_i is a negative value of the sum of the total delay times of the items emitted between s_i and

s_{i+1}. The sum of rewards R is represented as follows:

$$R = \sum_{i=1}^{N_J} \sum_{k=1}^{G_i} -(\tau_{out}(J_{i,k}) - \tau_{in}(J_{i,k}) - CT_{i,k}). \tag{4.2.3}$$

The total production time is the sum of the total CT and the delay time. So, minimizing the delay time minimizes the total production time T.

4.3 Training Method

We use a fully connected NN to Q-network in the learning phase. Furthermore, to solve the problem of correlations between samples and non-stationary targets, a replay buffer is used and the target network is separated from the Q-network [15]. The state containing the machine and line information is put into the Q-network as an input, and the Q-value for each action to be put next time is returned as an output. The Q-value of a_i in s_i is represented as $Q(s_i, a_i; \theta)$ in the Q-network with weight θ. As mentioned in Section IV, it is designed by a single agent, since the model and pattern should be decided simultaneously. Thus the dimensions of the Q-network's output is the multiplication of N_J and N_B.

Algorithm 1 describes the learning method used to solve the selection problem. The process of Lines 6–16 of Algorithm 1, which determines the production models and patterns to be put in, is performed in a state when M_1 accepts the next item. We implement a virtual production line based on the openAI qym's structure [19]. The ϵ-greedy policy is introduced in Line 4. As the episode progresses, ϵ decreases, but the minimum of ϵ is set to keep exploring. In Lines 6-16, the action which has the highest Q-value is selected and progressed. In Lines 9–12, previous state s_i, next state s_{i+1}, reward r_i and mask d_i are stored in replay buffer B as a transition $(s_i, a_i, r_i, s_{i+1}, d_i)$ after taking an action.

Lines 17–25 correspond to a learning process in which the weight θ of the Q-network is updated. For acceleration, learning takes place N_{tr} times for an episode. For effective learning according to Line 18, the size of the replay buffer needs to be large enough. Randomly selected transitions in the replay buffer are used for learning, due to the problem of correlations between samples (Line 20). The largest Q-value in the sampled transition is q_u (Line 21). The next state s_i's maximum Q-value is calculated using the target network for a stationary target (Line 22). Loss l is obtained by using a smooth L1 function

Algorithm 1 Selection With Q-network

Input: Selection problem

Output: Q-network

Initialization: Set Q-network with random weight θ, target network $\hat{Q}$ with $\hat{\theta}$ and size N_B buffer B

for $e = 1, 2, ..., N_E$ **do**

Reset line consisting of N_M,N_B and N_P

$\epsilon = \max(0.01, 0.08 - 0.01(e/200))$

$i = 0$

while $N_P \neq N_G$ **do**

Observe s_i

$x \leftarrow$ random value between 0 and 1

if $x < \epsilon$ **then**

Select a_i in randomly in A

else

$a \leftarrow \text{argmax}Q(s_i, a_i; \theta_i)$

end if

Put a_i in line

Observe r_i, s_{i+1}, d_i

Store transition $(s_i, a_i, r_i, s_{i+1}, d_i)$ in B

end while

if Size of $B > N_{tr}$ **then**

for $t = 1, 2, ..., N_{tr}$ **do**

Sample transitions $(s_u, a_u, r_u, s_{u+1}, d_u) \in B$

$q_u \leftarrow max_{a_u} Q(s_u, a_u; \theta_u)$

$y_u = r_u + \gamma * \max_{a_{i+1}} \hat{Q}(s_{i+1}, a_{i+1}; \theta_u) * d_i$

Calculate loss L from (4)

Perform gradient descent step on L with respect to θ

end for

end if

replace $\hat{Q} = Q$ every N_u episodes

end for

return Q-Network =0

(Line 23) [20] as follows:

$$f(y_u, q_u) = \begin{cases} 0.5(y_u - q_u)^2, & \text{if } |y_u - q_u| < 1 \\ |y_u - q_u| - 0.5, & \text{otherwise.} \end{cases} \tag{4.3.1}$$

The weight θ is updated with l (Line 24). For target stationary, target network is copied Q-network at intervals of N_u episodes not every episodes (Line 27). Learned Q-network model is returned after learning is completed. Here, we evaluate the method with the model which gives the best performance in the learning process.

5 Experimental Results

5.1 Data Sets and Training Details

The data used for training is based on the actual factory production schedule. The production line produces 41 kinds of models. The Q-network is trained by ADAM [21] by a gradient decent algorithm. In addition, the hyperparameters used in the experiment can be found in Table 5.1.

The training process is divided into training and validation segments. In the validation process, since ϵ is 0, no exploration is performed and only the performance of the Q-network is verified. Thus, the results of validation are not used for training. Fig. 5.1 shows the learning and validation results with the accumulated episodes. As learning progresses, it can be seen that the completion time tends to decrease, and also it can be seen that the performance of the training result including exploration is better at the beginning of training. However, after ϵ decreases, no difference is observed between training and validation. In the process of training 4,000 episodes, the 3884-th model, the network

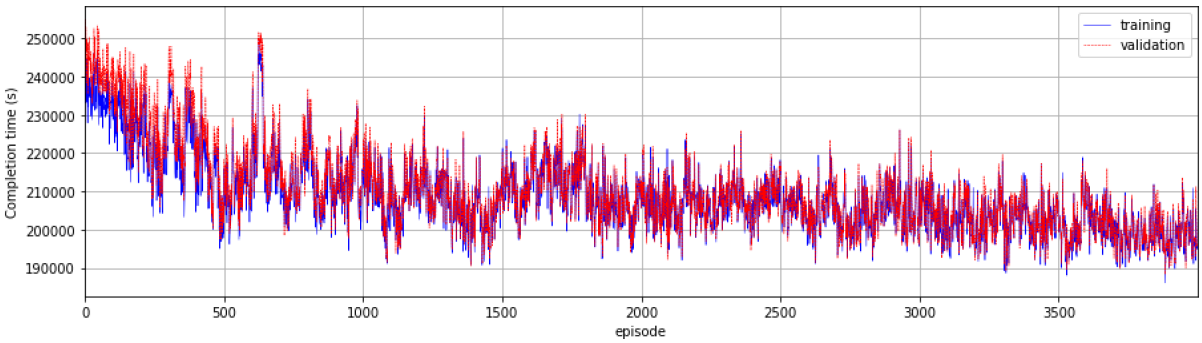

Figure 5.1: Training and validation results of episode accumulation.

Table 5.1: Hyperparameters for RL

Hyperparameters	**Value**
Numbers of hidden layers	5
Numbers of nodes in each hidden layers	1240,930,610,320,160
Learning rate of Adam	5×10^{-4}
Discount rate	0.98
Target Q-network update frequency	20 episodes
Replay buffer size	50,000
Minibatch size, $N_t r$	32
Epsilon convergence	0.01

Table 5.2: Throughput of rigid, random, and proposed scheduling

Method	Rigid	Random	Proposed
Throughput (units/min)	1.43	1.68	1.96
Improvement over rigid	0 %	17.48 %	37.06%

model that performed the best, is used for performance evaluation.

5.2 Performance Evaluation

Table 5.2 shows the throughput of each method to complete the same goal of production. The rigid method does not change the model until the production goal for each model is achieved. Random method means that a random model is put into the production line. The probability that a model is selected depends on the remaining amount for each model. The results are averaged values over 100 simulations. The proposed Q-learning-based scheduling method increases throughput by about 37% compared to the rigid method and 17% compared to the random method.

This result indicates that our algorithm makes appropriate choices for each given state of the line. The result of the random method can be explained as multi-job line analysis [16]. This assembly line consists of more than 4 machines. In the random method case, it works as a multi-job line as defined in [16]. A line with more than 4 machines performs better than the average of simple continuous operations, like the rigid method.

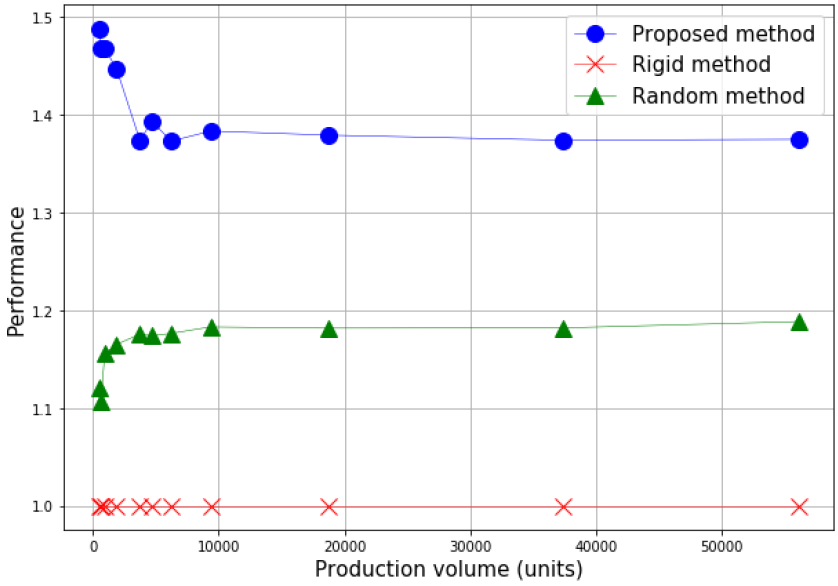

Figure 5.2: Change of performance according to the volume of production.

Fig. 5.2 shows the performance according to the production volume. The performance was calculated by dividing the throughput of each scheduling method volume by the throughput of the rigid method at the same production volume. When production exceeds a certain amount, performance converges. The proposed scheduling method shows maximum performance when the production volume is less than about 1800 units. When producing less than 1800 units, the improvement effect is close to 50. The actual assembly line produces less than 300 units while it is start and stop. Thus, if the proposed scheduling method is applied to an actual factory, a higher throughput can be expected.

We compare the blockage and starvation time of each machine to analyze the performance of the proposed model in detail. A blockage occurs when an item cannot be released because the next buffer is full, even though the machine has completed its operation. A starvation is a state in which the machine cannot perform any work even though the machine is up-time and empty, because the buffer is empty. Fig. 5.3 and Fig. 5.4 are comparisons of the blockage and starvation for each machine, respectively.

Overall, the machines with larger numbers tend to have smaller blockage time and larger starvation time. In most of the machines, our methods result in less blockage and starvation. In the proposed method, the blockage in M_{15} and M_{17} are larger than those

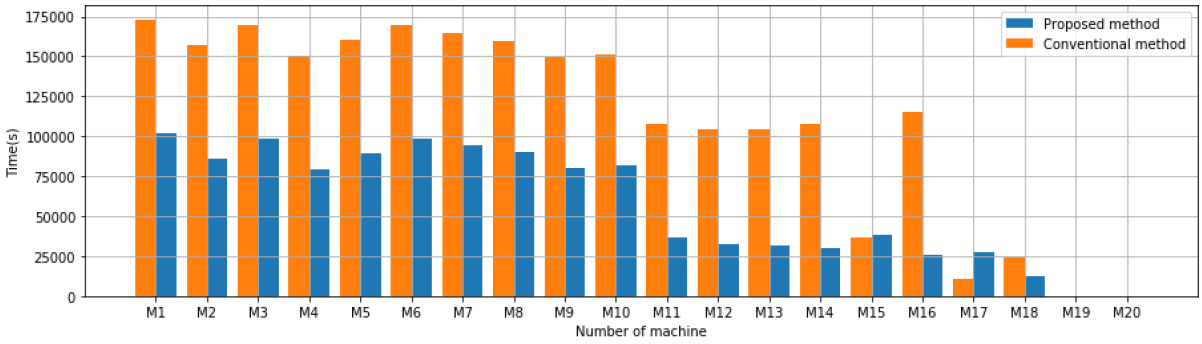

Figure 5.3: Blockage comparison for each machine by method.

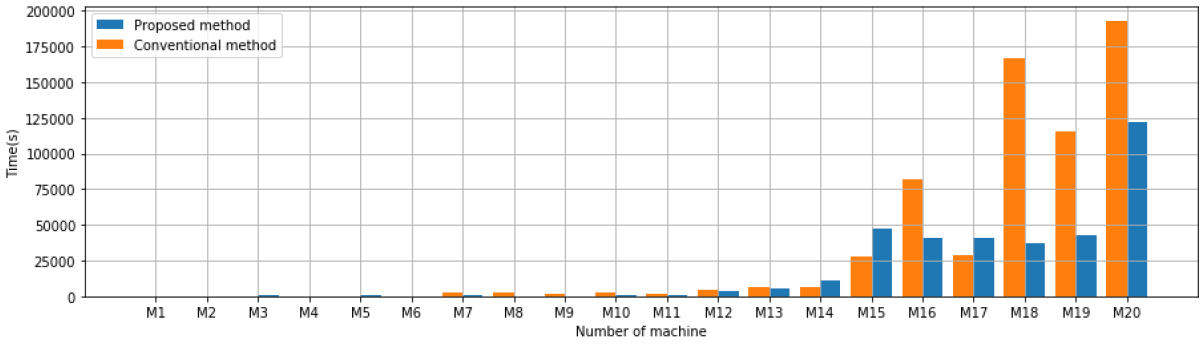

Figure 5.4: Starvation comparison for each machine by method.

in the existing rigid method. The same is true for starvation. The starvation time in our method is longer than the rigid method in M_{15} and M_{17}. Also, M_{16} and M_{18} show the largest reduction rates among all the machines in both blockage and starvation.

M_{15}, M_{16}, M_{17} and M_{18} are machines involved in the behavior pattern of this line. The method we propose reduces the overall blockage and starvation by selecting the appropriate model to be used. On the other hand, by using behavioral patterns, the blockage and starvation of M_{15} and M_{17} are sacrificed to decrease M_{16} and M_{18} dramatically.

6 Conclusions

In this paper, we have studied the problem of multi-job production. In particular, we have presented a selection problem that combines input model selection and production line behavior patterns. We have applied the deep-Q reinforcement learning (RL) method to attack the problem. Based on the real factory data collected for 6 months, we have established an experimental environment for training and evaluating the Q-network model.

Bibliography

[1] J. Wan, S. Tang, D. Li, M. Imran, C. Zhang, C. Liu, and Z. Pang, "Reconfigurable smart factory for drug packing in healthcare industry 4.0," *IEEE transactions on industrial informatics*, vol. 15, no. 1, pp. 507–516, 2018.

[2] C.-C. Lin, D.-J. Deng, Y.-L. Chih, and H.-T. Chiu, "Smart manufacturing scheduling with edge computing using multiclass deep Q network," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 4276–4284, 2019.

[3] G. Büchi, M. Cugno, and R. Castagnoli, "Smart factory performance and industry 4.0," *Technological Forecasting and Social Change*, vol. 150, no. 119790, 2020.

[4] C. Dimopoulos and A. M. Zalzala, "Recent developments in evolutionary computation for manufacturing optimization: Problems, solutions, and comparisons," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 2, pp. 93–113, 2000.

[5] F. Pezzella, G. Morganti, and G. Ciaschetti, "A genetic algorithm for the flexible job-shop scheduling problem," *Computers & Operations Research*, vol. 35, no. 10, pp. 3202–3212, 2008.

[6] R. S. Sutton, A. G. Barto *et al.*, *Introduction to reinforcement learning*. MIT press Cambridge, 1998, vol. 135.

[7] I. Park, J. Huh, J. Kim, and J. Park, "A reinforcement learning approach to robust scheduling of semiconductor manufacturing facilities," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 3, pp. 1420–1431, 2020.

[8] H. Kim, D.-E. Lim, and S. Lee, "Deep learning-based dynamic scheduling for semiconductor manufacturing with high uncertainty of automated material handling system

capability," *IEEE Transactions on Semiconductor Manufacturing*, vol. 33, no. 1, pp. 13–22, 2020.

[9] C. D. Hubbs, C. Li, N. V. Sahinidis, I. E. Grossmann, and J. M. Wassick, "A deep reinforcement learning approach for chemical production scheduling," *Computers & Chemical Engineering*, no. 106982, 2020.

[10] X. Xie and J. Li, "Modeling, analysis and continuous improvement of food production systems: A case study at a meat shaving and packaging line," *Journal of Food Engineering*, vol. 113, no. 2, pp. 344–350, 2012.

[11] X. Ou, Q. Chang, J. Arinez, and J. Zou, "Gantry work cell scheduling through reinforcement learning with knowledge-guided reward setting," *IEEE Access*, vol. 6, pp. 14 699–14 709, 2018.

[12] M. Saidi-Mehrabad and P. Fattahi, "Flexible job shop scheduling with tabu search algorithms," *The International Journal of Advanced Manufacturing Technology*, vol. 32, no. 5-6, pp. 563–570, 2007.

[13] J.-H. Lee and H.-J. Kim, "Analysis of backward sequence for single-armed cluster tools with processing time variations," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 4, pp. 2167–2174, 2020.

[14] Y. Fang, C. Peng, P. Lou, Z. Zhou, J. Hu, and J. Yan, "Digital-twin-based job shop scheduling toward smart manufacturing," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 12, pp. 6425–6435, 2019.

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[16] P. Alavian, P. Denno, and S. M. Meerkov, "Multi-job production systems: Definition, problems, and product-mix performance portrait of serial lines," *International Journal of Production Research*, vol. 55, no. 24, pp. 7276–7301, 2017.

[17] J. Feng, F. Li, C. Xu, and R. Y. Zhong, "Data-driven analysis for rfid-enabled smart factory: A case study," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 1, pp. 81–88, 2020.

[18] J. Li and S. M. Meerkov, *Production Systems Engineering*, 2008.

[19] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[20] P. J. Huber, "Robust estimation of a location parameter," *Ann. Math. Statist.*, vol. 35, no. 1, pp. 73–101, 1964.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

요 약 문

멀티잡 생산라인 시스템에서의 Deep-Q 강화학습 기반 스케줄링 : 사례 연구

인공지능, IoT, 5G, AR 등 기술들의 발전으로 인해 스마트 팩토리는 전세계적인 추세이며, 4차 산업혁명의 핵심 요소이다. 이에 따라 각 정부들은 다양한 정책을 앞다투어 발표하고 목표를 설정한다. 이 중 다품종 소량 생산은 산업계의 전반적인 트렌드로써 4차 산업혁명의 핵심 목표 중 하나이다. 이를 위한 다양한 생산품을 생산할 수 있는 복수 작업 생산시스템 (multi-job line) 이나 작업장의 구조나 예약 변경이 자유로운 유연한 생산 시스템 (flexible production systems) 이 각광받고 있다.

스케줄링 문제는 이러한 생산 시스템에서의 중요한 문제이다. 기존에는 운송 시스템이나 작업장 예약과 같은 스케줄링 문제가 주로 다뤄졌다. 그러나 생산 시스템의 자동화 됨에 따라 작업 교체에 따른 지체 시간을 무시할 수 있게 되었고, 작업 스케줄링 문제 해결을 통한 생산량의 향상을 기대할 수 있게 되었다. 기존 생산 시스템과 관련한 스케줄링 문제는 대부분 NP-hard 문제로써 최적의 솔루션을 빠른 시간 안에 도출하기 어렵기 때문에 머신러닝이 많이 활용되기 시작했다. 특히 마르코프 결정 프로세스로 표현 가능한 특성 때문에 강화학습과 딥러닝을 결합한 DQN(Deep-Q Network) 가 많이 이용된다.

실제 생산 현장에서 머신의 고장은 재가동 후 정상 상태까지의 상당한 손실을 초래하는 등 반드시 고려되야 할 요소이며, 실재하는 요소이다. 그러나 지금까지의 생산 환경에서의 강화학습에서 고장을 고려하여 학습시킨 사례는 거의 없다. 공장 머신의 고장을 구현하기 위해서 실제 공장 데이터가 요구되기 때문이다. 이에 따라 본 연구에서는 실제 공장의 데이터를 이용하여 머신의 고장을 구현하여 이를 고려한 DQN 기반 스케줄링 문제 해결 메소드를 제시한다.

실제 공장 생산라인에서 RFID로 수집한 데이터를 가공하여 가상의 생산환경을 구현하기 위한 파라미터를 추출해냈다. 생산라인의 머신의 평균 고장 시간과 작동 시간, 작업에 따른 작업 시간, 버퍼의 용량 등이 계산되었다. 이를 통해 생산라인에서 특정 머신에서 특이 행동 양식을 갖는 것을 발견하였다. 이러한 행동양식을 포함한 생산 스케줄링 문제를 정의하였다. 생산 스케줄링 문제는 머신의 고장이 구현된 가상의 생산라인에서 DQN 알고리즘을 기반으로 학습을 진행하여 최종적으로 약 37%의 생산량 향상을 이뤄냈다.

제시한 메소드의 생산량 향상은 전체 생산량의 규모에 의존해 변한다. 각 에피소드는 총 목표 생산량의 규모가 커짐에 따라 일정 생산량에 수렴하였으며, 본 연구에서 제시된

메소드는 규모가 작을수록 더 큰 생산효율 향상을 보였다. 실제 공장의 생산 일정에 따르면 제시된 메소드는 최대 37% 이상의 생산 효율 개선을 이룰 수 있을 것으로 보인다.

생산량 개선 메커니즘을 파악하기 위해 각 머신별 starvation과 blockage가 분석되었다. 분석결과 특이 행동패턴 양식을 갖는 머신에서의 starvation과 blockage가 전체 머신에 재분배를 함으로써 전체 생산라인의 생산 효율을 향상시켰다고 유추할 수 있었다.

주요어휘: 멀티잡 라인, 생산 스케줄링, 강화학습, 데이터 프로세싱, 유연한 생산 시스템.