Proceedings of the 2021 Winter Simulation Conference
S. Kim, B. Feng, K. Smith, S. Masoud, Z. Zheng, C. Szabo and M. Loper, eds.

DEVELOPMENT OF A REINFORCEMENT LEARNING-BASED ADAPTIVE SCHEDULING ALGORITHM FOR BLOCK ASSEMBLY PRODUCTION LINE

Jong Hun Woo
Young In Cho
So Hyun Nam

Department of Naval Architecture and Ocean Engineering
Seoul National University
1 Gwanak-ro, Gwanak-gu, Seoul 08826, SOUTH KOREA

Jong-Ho Nam

Major of Naval Architecture and Ocean Systems Engineering
Korea Maritime and Ocean University
727 Taejong-Ro, Busan, 49112, SOUTH KOREA

ABSTRACT

Rule-based heuristic algorithms and meta-heuristic algorithms have been studied to solve the scheduling problems of production systems. In recent research, reinforcement learning-based adaptive scheduling algorithms have been applied to solve complex problems with high-dimensional and vast state space. A production system in shipyards is a high-variable system, in which various production factors such as space, workforce, and resources are related. Thus, adaptive scheduling according to the changes in the production system and surrounding environment must be performed in shipyards. In this study, a basic reinforcement learning model for scheduling problems of shipyards was developed. A simplified model of the panel block shop in shipyards was assumed and the optimal policy for determining the input sequence of blocks was learned to reduce the flow time. The open source-based discrete event simulation (DES) kernel SimPy was incorporated into the environment of the reinforcement learning model.

1 INTRODUCTION

For manufacturing companies, the introduction of an efficient scheduling method is extremely important to achieve a variety of purposes, such as observing product delivery date, minimizing lead times, and leveling the load on resources. Traditionally, heuristic methods based on various dispatching rules have been developed and applied to solve scheduling problems of production systems (Deane and Moodie 1972, Pierreval and Mebarki 1997, Đurasević and Jakobović 2018). However, as the dispatching rule basically indicates the order of inputs based on the characteristics of the jobs currently waiting in the jobshop, it does not reflect information about the jobs in other jobshops and the characteristics of the production system itself. To overcome this limitation, studies have focused on applying search algorithms such as backtracking search and constraint propagation by modeling the production system scheduling problem as a constraint satisfaction problem and on optimization-based meta-heuristic methods such as genetic algorithms, tabu search, and simulated annealing (Woo et al. 2003, Bae et al. 2007, Hwang et al. 2010, Liu et al. 2011). However, in these methodologies, the computation time required to obtain a solution increases as the production system becomes more complex and the size of the modeled problem is increased, which hinders real-time decision-making. In addition, in these methodologies, when the input information or a state of the production system changes, a new problem must be constructed. To overcome these limitations, a deep reinforcement learning-based scheduling algorithm to target jobshop scheduling problems has been recently applied.

As the shipbuilding industry is a representative order-made production industry, scheduling problems are also critical because accurate production planning to meet the delivery date of the ship is important. Ships, which are products of shipyards, have different specifications according to the requirements of ship owners, even if they are of the same type, and the period from their production to delivery is long. In addition, production plans are frequently changed as the production system of the shipyard is affected by various production factors such as resources, space, and workforce (Woo and Oh 2018, Lee et al. 2020). Owing to these characteristics, the production system of shipyards inevitably has a high variability and the complexity of the state space inherent to the system is significant (Kwak et al. 2020). Therefore, a significant state space needs to be considered for the planning of the shipbuilding production system, and a scheduling method that can respond to dynamic situations by monitoring changes in the internal and external environment of the production system is required.

To measure the applicability of reinforcement learning to the scheduling problem of a production system, many studies have been conducted to apply a reinforcement learning algorithm by modeling the scheduling problem with a dispatching rule selection problem in the beginning. An appropriate dispatching rule should be selected to achieve the purpose of the production system from the predefined dispatching rules.

Aydin and Öztemel (2000) constructed the problem for the agent to select one of three dispatching rules (shortest processing time (SPT), COVERT , and critical ratio) by recognizing the number of jobs in the buffer (or queue) and the average slack time of the buffer as states. This was aimed at minimizing the average tardiness value of the jobshop system by applying the Q-III algorithm modified from the Q-learning algorithm. As a result, it was found that the reinforcement learning-based scheduling method had better performance than applying a single dispatching rule for various scenarios with different average delivery dates for jobs. However, this result cannot be generalized as the reward itself is defined such that it shows the dispatching rule that is the correct answer in a specific state of this study.

Wang and Usher (2004) conducted a study in which the Q-learning algorithm was applied to the scheduling problem for a single machine with one buffer. In this study, to minimize the average tardiness value of a completed job, the problem was constructed for the agent to select one of three dispatching rules: earliest due date (EDD), SPT, and first-in first-out. The state and reward were defined by dividing the estimated average lateness of the jobs in the buffer and the tardiness of the completed jobs, respectively, and setting the corresponding values for each section. The effect of the combination of each parameter on the average tardiness value of the learning result was analyzed using the boundary value dividing the range and section of the state and reward, and the absolute value of the reward set in each section as parameters.

In addition, Kong and Wu (2005) conducted a study to train a simulated annealing-based Q-learning agent that selected one of the dispatching rules of SPT, EDD, and round-robin for a single machine problem with one buffer. The agent was trained by constructing the reinforcement learning problem for each of the three objectives of minimizing the average tardiness value, minimizing the maximum tardiness value, and minimizing the number of tardy jobs. As a result, in each case, the dispatching rule predominantly selected by the agent was consistent with the dispatching rule known to be suitable for achieving each purpose. Therefore, the result of this study indicates that the reinforcement learning agent can learn the scheduling policy according to the purpose of the production system.

The implementation of a scheduling algorithm based on reinforcement learning to achieve a specific purpose of a production system has been sufficiently proven through studies on the dispatching rule selection problem. However, the construction method of the problem that makes the agent select a specific dispatching rule has limitations as the scheduling algorithm becomes dependent on the dispatching rule itself. Therefore, studies have also been conducted on an algorithm that determines the order of jobs in the buffer rather than selecting the dispatching rule to apply a more generalized scheduling algorithm.

In the study of Zhang et al. (2017), an agent based on the Q-learning algorithm that minimizes the overall cycle time was trained on a simple jobshop model that produced two types of products consisting of five processes, and it was found that the performance improved compared to the case of applying a single dispatching rule. In addition, the study presented a theoretical basis for structuring the jobshop scheduling problem with a Markov decision process that satisfies the Markovian property, assuming the distribution of job time and arrival time intervals as exponential distributions and calculating the state-transition probabilities.

Lee et al. (2019) applied the SARSA algorithm to the scheduling problem of a process model for a semiconductor process consisting of 160 machines and 50 types of products. A reward function was set to simultaneously consider the three objectives of increasing production, minimizing delivery delays, and minimizing setup changes. As a result, it was found that the scheduling algorithm based on reinforcement learning showed better performance in three indicators than the worker's plan. However, the performance of the reinforcement learning-based scheduling algorithm was low in the changed scenario with a different demand distribution from the learning scenario, which indicates that it is difficult to apply to a generalized scenario.

Waschneck et al. (2018) used a multi-agent reinforcement learning-based scheduling algorithm based on a model that simplified the semiconductor process into three processes. Each process had an independent agent to determine the job order. In the first step, a method was proposed, in which learning was performed for only one agent and the remainder of the process was controlled through a heuristic method. In the second step, a two-step multi-agent learning method was proposed, in which additional learning was performed using all the agents trained in the first step.

Gabel and Riedmiller (2008) conducted a study comparing multi-agent-based scheduling algorithms and conventional heuristic-based dispatching rules based on the total lead time when the final job was completed by targeting various benchmarking problems in the field of jobshop scheduling. Specifically, each resource had an independent agent, and each agent received only the information of the allocated resource as the state, such as to study a decentralized scheduling algorithm to achieve the entire purpose from limited information. The reward was defined considering the entire production system.

These prior studies used reinforcement learning algorithms to overcome the limitations of the conventional scheduling problem, such as the difficulty to reflect the probabilistic variability of input conditions or production environment to the dispatching rule method. Although there are slight differences, prior studies commonly had a buffer (or queue) for intermediate products to wait before each process, and used algorithms that determined the process input priority of intermediate products in the buffer.

The present study was conducted on the shipyard production process and these prior studies were used as benchmarks. As the shipbuilding production process is highly variable, a scheduling algorithm that can derive an optimal plan was developed by reflecting this variability. To this end, an artificial neural network-based reinforcement learning algorithm was used, and a study was conducted on the application of the shipyard panel block shop. Prior studies had a buffer where intermediate products could wait before the process, whereas the panel block shop of the present study does not have a buffer except for the first process part.

In this study, the shortening of the lead time was set as the objective function, and the approximated agent was trained with an artificial neural network to measure the applicability of the reinforcement learning algorithm to the scheduling problem that determines the order of block input in the panel block shop. As an environment for learning, a simulation model implemented with SimPy , an open source discrete event simulation (DES) kernel, was used to simulate the behavior of the production system. In addition, the effectiveness of the reinforcement learning method proposed in this study was verified by analyzing the results of input data (more specifically, randomly generated by using a probabilistic time distribution) with different processing hours from the input blocks used in learning by using the trained agent. However, the problem of the probabilistic model related to changes in the production environment will be covered in a follow-up study.

2 REINFORCEMENT LEARNING

2.1 Reinforcement learning and sequential decision-making problems

Reinforcement learning is a field of machine learning that is applied to solving sequential decision-making problems in which the agents learn through interactions with the environment (Sutton and Barto 1998). Solving a sequential decision-making problem is finding the best policy, which is a set of actions that must be taken to achieve a specific goal. To apply reinforcement learning, sequential decision-making problems must be mathematically defined in a framework called Markov decision process (Van Otterlo and Wiering 2012). The main components of the Markov decision process are the state (S_t) that has the information for the agent to take an action, the action (A_t) that the agent can take in each state, the state-transition probability, which is the probability of moving to the next state when a specific action is taken in a specific state, and the reward (R_{t+1}), which is the feedback that the environment gives to the agent. The reward is a factor that is directly connected to the goal of the problem to be solved, and the agent learns to maximize the cumulative reward (Levin et al. 1998). In addition to reinforcement learning algorithms, sequential decision-making problems can be solved by applying optimization algorithms such as genetic algorithms and simulated annealing. The most distinctive feature of reinforcement learning algorithms is that instead of directly searching the policy space by introducing a value function (v_π) or Q-function (q_π), which is an action value function, it searches the state space and finds the optimal policy from it. The Q-function is a function of action and state, which is defined as an expected value for the return value that will be received when a specific action is taken in a specific state. The optimal Q-function can be obtained from the Bellman optimality equation as shown in (1) (Sutton and Barto 1998).

$$q_\pi(s,a) = \max_a \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} q_\pi(S_{t+1}, a') \middle| S_t = s, A_t = a\right] \tag{1}$$

$q_\pi(s,a)$: reward value by action a at state s when agent follow policy π
s: state
a: action
R_t: reward at time t
γ: discount factor
S_t: state at time t
A_t: action at time t

To calculate the expected value in the Bellman optimality equation, it is necessary to know exactly the state-transition probabilities. However, in reinforcement learning, the optimal Q-function can be calculated without an accurate model of the environment by estimating the expected value based on samples obtained while the agent explores the state space (Sutton and Barto 1998).

2.2 DDQN Algorithm

The deep Q-Networks (DQN) algorithm is a reinforcement learning algorithm designed to learn high-dimensional input data, such as images, as stated in the study of Mnih et al. (2013). It is a deep reinforcement learning algorithm that uses an off-policy learning method based on Q-learning and approximates the Q-function with an artificial neural network(Figure 1, Table 1). In addition, a replay memory is introduced to store samples consisting of the current state (s_t), action (a_t), reward (r_t), and next state (s_{t+1}) obtained by the agent while exploring. Samples are extracted from the replay memory in batch units during learning, and the weight value of the Q-function approximated by the artificial neural network is updated to reduce the correlation between samples. However, when the weight is updated, the target value (Q*) includes the maximum value of the Q-function in the next state as shown in (2), which causes a problem of overestimating the value of the current Q-function.

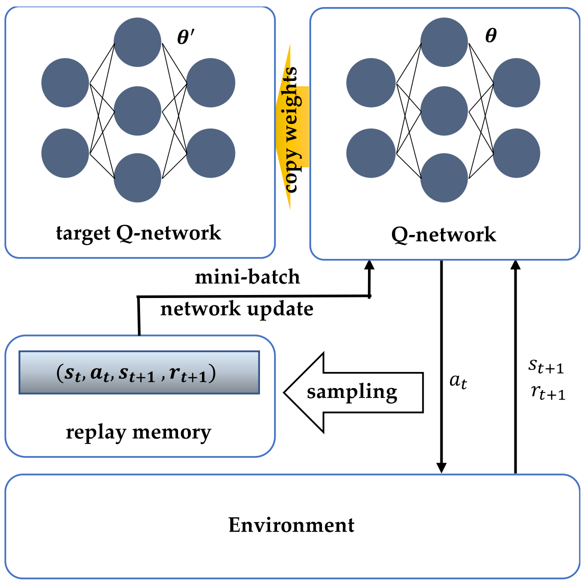


Figure 1: DDQN reinforcement algorithm.

Table 1: Pseudocode for DDQN algorithm.

```
Double Deep Q-network algorithm pseudocode
Initialize replay memory $D$ to capacity $L$
Initialize Q-network $Q_\theta$ and target Q-network $Q_{\theta'}$ with random weights
Copy the weights of Q-network to the weights of target Q-network $Q_{\theta'}$
**for** episode=1, $M$ **do**
    reset state $s_1$
    **for** t=1, $T$ **do**
      Select $a_t$ according to $\varepsilon$-greedy policy $\pi(s) = \begin{cases} \underset{a'}{\operatorname{argmax}} Q_\theta(s_t, a') & \text{with probability } 1-\varepsilon \\ random\ action & \text{with probability } \varepsilon \end{cases}$
       Receive reward $r_t$ and new state $s_{t+1}$
       Store tuple $(s_t, a_t, r_t, s_{t+1})$ in replay memory $D$
       **if** the number of tuples in replay memory is sufficient
          randomly sample mini-batch $N$ of tuples from replay memory
          compute target Q values for each tuple:
             $Q^*(s_t, a_t) = \begin{cases} r_t + \gamma Q_{\theta'}(s_{t+1}, \underset{a'}{\operatorname{argmax}} Q_\theta(s_{t+1}, a')) & \text{for non} - \text{terminal state } s_{t+1} \\ r_t & \text{for terminal state } s_{t+1} \end{cases}$
          update gradients with loss function $L(\theta) = \frac{1}{N}\sum_{i=1}^{N}(Q^*(s_t, a_t) - Q_\theta(s_t, a_t))_i^2$
    **if** target Q-network update
       Copy the weights of Q-network to the weights of target Q-network $Q_{\theta'}$
     **end for**
**end for**
```

$$Q^*(s_t, a_t) = r_t + \gamma \max_{a'} Q(s_{t+1}, a') \quad (2)$$

$Q^*(s_t, a_t)$: Target value of Q function at s_t and a_t
s_t: State at time t
a_t: Action at time t
r_t: Reward at time t
γ: Discount factor
a': Next action

3 PROBLEM DEFINITION

3.1 Panel block shop

The panel block shop was set as the process model for this study, which is the representative process of shipyards. Hwang et al. (2010) developed a production planning system based on simulated annealing for the panel block shop. The authors emphasized that in addition to the uncertainty inherent to the process, the panel block shop becomes a critical obstruction in shipyards where large quantities of ships are concentrated, because all ships produced in the shipyard go through the panel block shop. Therefore, it is extremely important to plan a schedule to shorten the lead time of the completed block. Thus, this study aimed to learn a scheduling algorithm based on reinforcement learning to minimize lead time as a basis.

The panel block shop modeled for learning consists of a total of seven detailed processes: panel arrangement, frontside welding, plate turn-over, backside welding, longitudinal mounting, longitudinal welding, and grand assembly, as shown in Figure 2, and each process has a finish-to-start connection in the order presented. Each detailed process of the panel block shop does not have a buffer, and the block that was worked in the preceding process waits in this process until the work in the subsequent process is completed. The time in each process for each block is pre-planned and given as input data as shown in Table 2. As a result, in this study, the scheduling problem of the panel block shop was modeled as a problem of determining the order of blocks to be input to the panel measurement process, which is the first process.

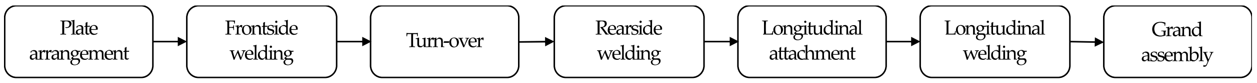


Figure 2: Production process of panel block.

Table 2: Block data for panel block shop (partially selected, unit: hour).

Block	Plate arrangement	Frontside welding	Turn-over	Rearside welding	Longitudinal attachment	Longitudinal welding	Grand assembly
236	3.0	2.6	0.5	2.6	1.6	2.0	2.5
226	3.0	2.6	0.5	2.6	1.6	2.0	3.5
632	2.0	1.9	0.5	1.9	1.4	1.7	2.0
622	2.0	1.9	0.5	1.9	1.4	1.7	2.0
203	3.0	3.5	0.5	2.6	1.8	2.0	2.5
633	2.0	1.9	0.5	1.9	1.4	1.7	2.0
623	2.0	1.9	0.5	1.9	1.4	1.7	2.0
433	3.3	2.8	0.5	2.8	4.7	4.0	7.0

3.2 Markov decision process

In this study, the summary of the scheduling problem of the panel block shop structured with the reinforcement learning framework of the Markov decision process is shown in Figure 3. The reinforcement learning model consists of an agent that determines the input order of blocks and an environment implemented to perform production simulation by modeling the panel block shop using SimPy. The environment inputs the block into the process model and executes the DES simulation until the next decision-making point. In other words, the agent selects a block to be input, and in the point in which the work on the block is completed in the panel measurement process, the decision for the next block to be input needs to be made.

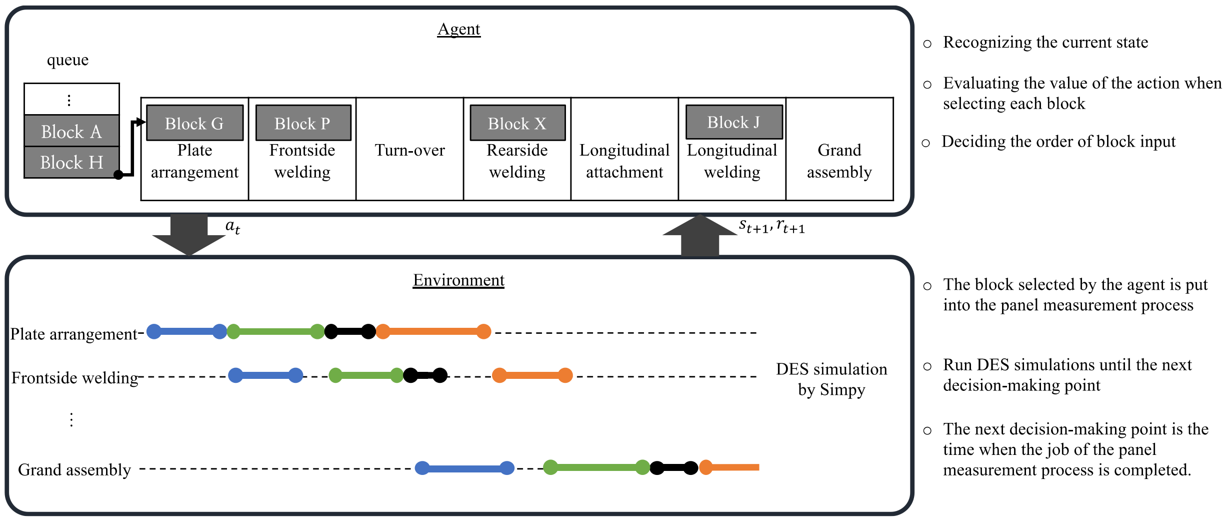

Agent
queue
Block A
Block H
Block G
Plate arrangement
Block P
Frontside welding
Turn-over
Block X
Rearside welding
Longitudinal attachment
Block J
Longitudinal welding
Grand assembly
a_t
s_{t+1}, r_{t+1}
Environment
Plate arrangement
Frontside welding
Grand assembly
DES simulation by Simpy
Recognizing the current state
Evaluating the value of the action when selecting each block
Deciding the order of block input
The block selected by the agent is put into the panel measurement process
Run DES simulations until the next decision-making point
The next decision-making point is the time when the job of the panel measurement process is completed.

Figure 3: Reinforcement learning for scheduling problem of panel block shop.

3.2.1 State

The state is composed of information on each process and blocks loaded in the queue, as shown in Figure 4. The information on each process was defined as the time remaining from the current point in time until the work on the block in progress is completed. In addition, the information on the blocks was defined as work time information in the seven processes of each block, which is planned and provided in advance. If the block queue length set in the problem is 10, the state becomes a one-dimensional vector with a length of 77 consisting of 7 values for the process and 70 values for the block, as the modeled panel block shop has seven detailed processes.

3.2.2 Action

The agent's action is to select a block to be put into the panel measurement process. In this study, a virtual queue was set before the panel measurement process. Thus, a predefined number of blocks were randomly loaded into the queue, and the loaded blocks formed a set of actions. Therefore, the action set was defined for the agent to select one of the randomly loaded blocks rather than the entire block.

3.2.3 Reward

For the successful learning of reinforcement learning agents, rewards need to be defined to be directly related to the goal of minimizing lead time. To this end, the reward (r) was defined as shown in (3), including the predicted lead time that increases when a specific block is input in this study.

$$r = C - \Delta LT \tag{3}$$

In equation (3), $\Delta LT(LT_{t+1} - LT_t)$ means an increase in the predicted lead time, and C is a constant. In this study, the constant C was set as the maximum value for the total working time (sum of working hours in all seven processes) of the block data to be planned. For example, the predicted final lead time at a certain decision-making point t was 122 assuming that the maximum value of the total working hours in the planning block data is 25. If the final lead time is 134 predicted again after the agent decides to input a specific block, the increase in predicted lead time becomes 12, and in this case, the reward becomes 13.

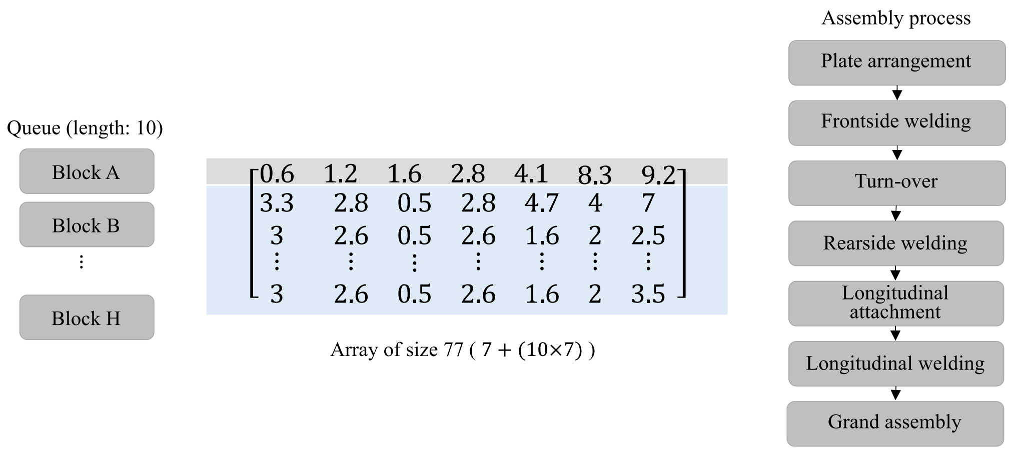


Figure 4: State configuration of environment.

4 LEARNING RESULTS

4.1 Input data

In the agent's learning process, input data consisting of a total of 61 blocks was used. The format of the input data is as shown in Table 2. In other words, the working hours in the seven detailed processes were given in advance for each block. For each episode, the end condition was set to complete the work of all 61 blocks. In addition, the order in which blocks were input was randomly set for each episode, and randomness was added to the agent's learning process.

4.2 Hyperparameter setting

The length of the queue, which defines the set of actions that the agent can select, was set to 10. In addition, the hyperparameter for the artificial neural network model to approximate the Q-function was set as shown in Table 3, and the set value of the hyperparameter set to apply the DDQN algorithm was as shown in Table 4.

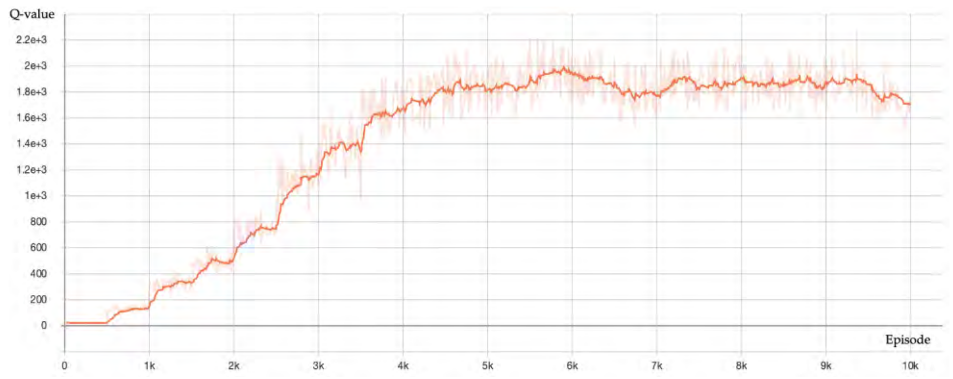
Table 3: Hyperparameters of Q-network.

Layer	HYPERPARAMETER	VALUE
FCN 1	Number of nodes	512
	Activation function	ReLU
	Weight initialization	He Normal
FCN 2	Number of nodes	256
	Activation function	ReLU
	Weight initialization	He Normal
FCN 3	Number of nodes	256
	Activation function	ReLU
	Weight initialization	He Normal
FCN 4 (Output)	Number of nodes	10
	Activation function	Linear

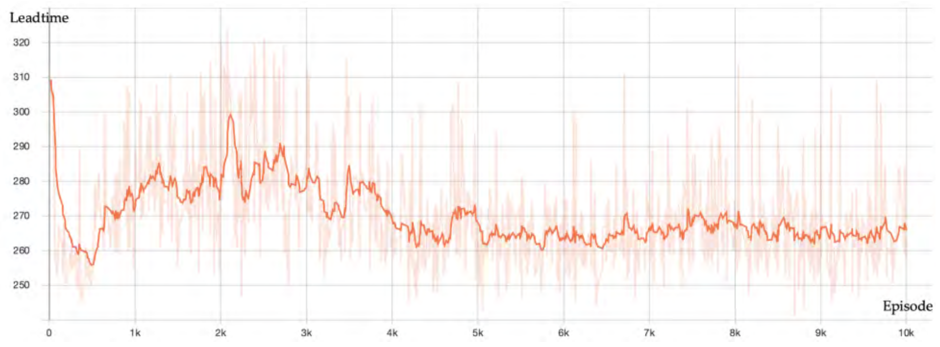
FCN: fully convolutional network

Table 4: Hyperparameters of reinforcement learning algorithm.

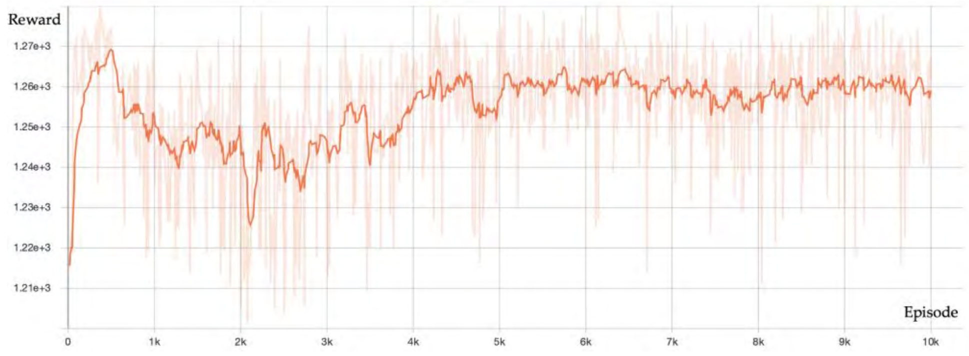
	HYPERPARAMETER	VALUE
OPTIMIZER	Type	Adam
	Learning rate	1e-5
DDQN	Batch size	64
	Discount ratio	0.99
	Target network update iteration	500
	Size of replay memory	2000



< Average Maximum Q-value of each episode>



<Lead time of each episode>



<Reward of each episode>

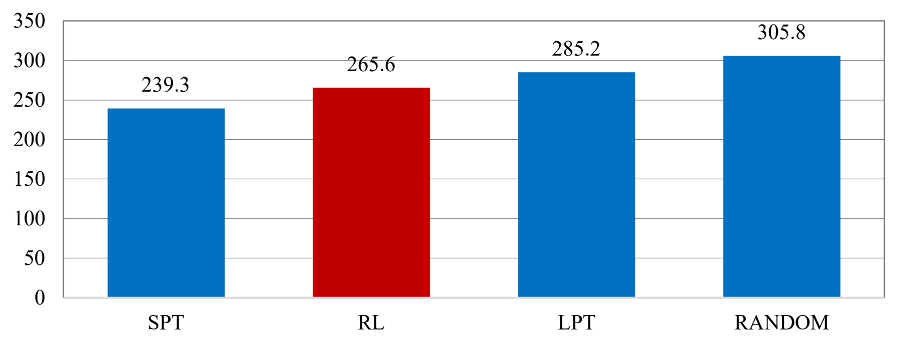
Figure 5: Measurements of performances during the learning phase.

4.3 Learning environment and outcomes

Using the configured environment and algorithm, learning was performed in an environment comprised of an Intel(R) CPU Core(TM) i7-10700 and 16 GB RAM. The computation time spent on learning was 5 h 53 min based on 10,000 episodes.

In the learning process, the agent's performance index according to the episode is described as follows. The first graph in Figure 5 is an index for the Q-function, and the Average Max Q index is the average value of the maximum Q-functions in all states visited by the agent during one episode. As shown in the graph, Average Max Q increased at the beginning of learning, which means that the agent was learning to maximize the cumulative reward. In addition, Average Max Q converged to a constant value of approximately 1,800 from the 5,000th episode, which means that the agent's policy converged. The lead time graph shows how the lead time, which is the time taken for all blocks to complete the large assembly process (final process), changed as learning progressed. The reward graph shows the change in the cumulative sum of the rewards received by the agent for each episode. Although there were some variations in the graphs, they converged in the section where the Average Max Q graph converged.

Figure 6 is a graph comparing the reinforcement learning-based scheduling algorithm developed in this study with other dispatching rules. It was found that the results were improved by approximately 13.1% compared to the case of inputting blocks in a random order, and by approximately 6.87% compared to longest processing time. In addition, the performance was approximately 9.90% lower than that of SPT. As a result, it was found that the reinforcement learning-based scheduling algorithm showed satisfactory performance in terms of shortening lead time, considering that the SPT was the optimal dispatching rule, showing excellent performance in the panel block shop.



SPT: shortest processing time
RL: reinforcement learning (this study)
LPT: longest processing time

Figure 6: Comparison of lead time with various dispatching algorithm.

5 CONCLUSION AND FUTURE STUDIES

In this study, a single-purpose reinforcement learning-based scheduling algorithm was developed for the panel block shop, which is a bottleneck process for shipyards, to minimize the total lead time. Specifically, the agent performing the scheduling receives the information of the blocks loaded in the queue and the predicted lead time information for each process and decides the order of the input blocks. The environment implemented a reinforcement learning problem to present a reward to the agent based on the calculation of the increase in predicted lead time for the agent's decision. In addition, in the process of modeling the scheduling problem of the panel block shop as a reinforcement learning problem, a simulation model was implemented using SimPy, an open-source DES simulation kernel, for simulation of the panel block shop, which was linked to the reinforcement learning environment.

As a result, it was found that the scheduling algorithm based on reinforcement learning in terms of lead time performed approximately 9.90% lower than the SPT dispatching rule, which showed the best performance in the panel block shop. Therefore, it is necessary to further study the hyperparameter optimization analysis and the state and reward setting method to improve the performance of the developed reinforcement learning scheduling algorithm. In addition, the reinforcement learning algorithm has a considerable advantage in more complex situations in which there are probabilistic variations and the characteristics of the environment change over time. Therefore, a study to develop a multipurpose reinforcement learning-based scheduling algorithm that additionally considers the uncertainty inherent in the panel block shop such as machine failure will be conducted in the future. In addition, proposed research was conducted under the assumption that all information of the target system can be known, but in actual problems, some information of the system may not be known, so future research on the partially observable problem is necessary.

ACKNOWLEDGMENTS

This research was supported by of the Ministry of Trade, Industry and Energy, and is the result of projects for the development of digital twin technology for IoT and AI-based block assembly processes (20006978), the development of an integrated system for optimizing the arrangement of ships at dock and quay on mid-sized shipyards (20007834) and the development of master data system of lead time for precision enhancement of shipbuilding production management" sponsored by the Ministry of Trade, Industry, and Energy of Korea (20000208).

REFERENCES

Deane, R. H., Moodie, C. L. 1972. "A dispatching methodology for balancing workload assignments in a job shop production facility". *AIIE Transactions*: 4(4):277–283.

Pierreval, H., Mebarki, N. 1997. "Dynamic scheduling selection of dispatching rules for manufacturing system". *International Journal of Production Research* 35(6): 1575–1591.

Đurasević, M., Jakobović, D. 2018. "A survey of dispatching rules for the dynamic unrelated machines environment". *Expert Systems with Applications* 113: 555–569.

Woo, S. B., Ryu, H. G., Hahn, H. S. 2003. "Heuristic algorithms for resource leveling in pre-erection scheduling and erection scheduling of shipbuilding". *IE Interfaces* 16(3): 332–343.

Bae, H. C., Park, K. C., Cha, B. C., Moon, I. K. 2007. "Scheduling of shipyard sub-assembly process using genetic algorithms". *IE Interfaces* 20(1): 33–40.

Hwang, I.H., Noh, J.-K., Lee, K.-K., Shin, J. G. 2010. "Short-term scheduling optimization for subassembly line in ship production using simulated annealing". *Journal of the Korea Society for Simulation*. 19(1): 73–82.

Liu, Z., Chua, D. K. H., Yeoh, K. W. 2011. "Aggregate production planning for shipbuilding with variation-inventory trade-offs". *International Journal of Production Research* 49(20): 6249–6272.

Woo, J. H., Oh, D. 2018. "Development of simulation framework for shipbuilding". *International Journal of Computer Integrated Manufacturing* 31(2): 210–227.

Lee, Y. G., Ju, S., Woo, J. H. 2020. "Simulation-based planning system for shipbuilding". *International Journal of Computer Integrated Manufacturing* 33(6): 1–16.

Kwak, D. H., Yu, S. H., Woo, J. H., Park, J. G. 2020. "Analysis of production and procurement plan of shipbuilding based on queueing theory with variability". *Journal of the Korean Institute of Industrial Engineers* 46(6): 673–682.

Aydin, M. E., Öztemel, E. 2000. "Dynamic job-shop scheduling using reinforcement learning agents". *Robotics and Autonomous Systems* 33(2): 169–178.

Wang, Y. C., Usher, J. M. 2004. "Learning policies for single machine job dispatching". *Robotics and Computer-Integrated Manufacturing* 20(6): 553–562.

Kong, L. F., Wu, J. 2005. "Dynamic single machine scheduling using Q-learning agent". In *Proceedings of the 2005 International Conference on Machine Learning and Cybernetics*, August 18th-21st, Guangzhou, China, 3237-3241.

Zhang, T., Xie, S., Rose, O. 2017. "Real-time job shop scheduling based on simulation and Markov decision processes". In *Proceedings of the 2017 Winter Simulation Conference*, edited by W.K.V. Chan, A. D'Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. Page, 3899-3907. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Lee, W. J., Kim, B. H., Ko, K., Shin, H. 2019. "Simulation based multi-objective fab scheduling by using reinforcement learning". In *Proceedings of the 2019 Winter Simulation Conference*, edited by N. Mustafee, K.-H. Bae, S. Lazarova-Molnar, M. Rabe, C. Szabo, P. Haas, and Y.-J. Son, 2236-2247. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Waschneck, B., Reichstaller, A., Belzner, L., Altenmüller, T., Bauernhansl, T., Knapp, A., Kyek, A. 2018. "Optimization of global production scheduling with deep reinforcement learning". *Procedia CIRP* 72(1): 1264–1269.

Gabel, T., Riedmiller, M. 2008. “Adaptive reactive job-shop scheduling with reinforcement learning agents”. *International Journal of Information Technology and Intelligent Computing* 24(4): 14–18.
Sutton, R. S., Barto, A. G. 1998. *Introduction to reinforcement learning*. Vol 135:MIT press Cambridge.
Van Otterlo, M., and Wiering, M. 2012. *Reinforcement Learning*. Berlin: Springer.
Levin, E., Pieraccini, R., Eckert, W. 1998. “Using Markov decision process for learning dialogue strategies”. In *Proceedings of the 1998 IEEE International Conference on Acoustics*, May 15th, Seattle, Washington, USA, 201-204.
Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M. 2013. “Playing atari with deep reinforcement learning”. In *NIPS Deep Learning Workshop 2013*, December 9th, Lake Tahoe, USA, 1-9.
Van Hasselt, H., Guez, A., Silver, D. 2016. “Deep reinforcement learning with double q-learning”. In *Proceedings of the AAAI Conference on Artificial Intelligence*, February 12th-17th, Phoenix, Arizona, USA, 2094-2100.

AUTHOR BIOGRAPHIES

JONG HUN WOO is an associate professor in the Department of Naval Architecture and Ocean Engineering at Seoul National University. He holds a PhD in Naval Architectur and Ocean Engineering from Seoul National University. His research interest is in DES simulation, APS(Advanced Planning and Scheduling), machine learning and queuing, and he has relevant research experiences in the application areas of shipbuilding. His email address is j.woo@snu.ac.kr.

YOUNG IN CHO is a graduate school student in the Department of Naval Architecture and Ocean Engineering at Seoul National University. He has a Bachelor of Science in Naval Architecture and Ocean Engineering. His research interests include DES (Discrete Event Simulation) and reinforcement learning. His email address is whduddlsi@snu.ac.kr.

SO HYUN NAM is a bachelor's degree student in the Department of Naval Architecture and Ocean Engineering at Seoul National University. Her research interest is in DES simulation, Reinforcement learning and queuing. Her email address is sohyon525@snu.ac.kr.

JONG-HO NAM is a professor in the Major of Naval Architecture and Ocean Systems Engineering at Korea Maritime and Ocean University. After obtaining a bachelor degree from Seoul National University, he continued for the master degree at MIT and the doctoral degree at the University of Michigan, Ann Arbor. His research interests include computerized ship design and production, geometric modeling, and application of virtual environments. His email address is jhnam@kmou.ac.kr.