Proceedings of the 2020 Winter Simulation Conference
K.-H. Bae, B. Feng, S. Kim, S. Lazarova-Molnar, Z. Zheng, T. Roeder, and R. Thiesing, eds.

MACHINE LEARNING (REINFORCEMENT LEARNING)-BASED STEEL STOCK YARD PLANNING ALGORITHM

Jong Hun Woo
Young In Cho
Sang Hyeon Yu
So Hyun Nam
Haoyu Zhu
Dong Hoon Kwak

Dept. of Naval Architecture and Ocean Eng.
Seoul National University
1 Gwanak-ro, Gwanak-gu
Seoul, 08826, SOUTH KOREA

Jong-Ho Nam

Div. of Naval Architecture and Ocean Syst. Eng.
Korea Maritime & Ocean University
727 Taejong-Ro
Busan, 49112, SOUTH KOREA

ABSTRACT

Steel plates are supplied to shipyards without adhering to the processing schedule because they are ordered in bulk according to the steel market condition and the mid- to long-term production strategy. Hence, steel plates are stacked up in the steel stock yard until they are input to the processing system and then supplied sequentially according to the processing start date. Currently, a steel stock yard, is operated by the experience of field workers, and inefficiencies such as excessive crane use are occurring, so efficient management techniques are required. However, the conventional optimization algorithm has limitations because the input timing of the steel is random. In this study, a study was conducted to determine the order of steel input into steel stock yard using reinforcement learning algorithm. Effective algorithm(A3C) can be identified through tests, and it was validated that proposed method is effective for problems of actual size steel stock yard.

1 INTRODUCTION

Steel plates are supplied to shipyards without adhering to the processing schedule because they are ordered in bulk according to the steel market condition and the mid- to long-term production strategy. Hence, steel plates are stacked up in the steel stock yard until they are input to the processing system and then supplied sequentially according to the processing start date. Because steel plates are input to the steel stock yard in the order of arrival at the port (more or less) regardless of the processing input sequence, they are managed in accordance with the processing input schedule through a separate sorting work. The steel plates in the steel stock yard are stacked up vertically. The steel plates placed above those that are to be shifted must be transferred to another place using a crane. At present, few shipyards manage the initial stacking order. Thus, they incorporate the first and second sorting work so that the steel plates can be input on the planned date. Steel stock yard management cost can be saved if the sorting work is reduced by considering the initial piling sequence. Theoretically, this problem can be solved by stacking up the planned steel plates in the reverse order of input. However, in the actual steel stock yard environment, the order of receipt of steel plates to be planned cannot be determined because the steel plates are unloaded and received in bulk using

a barge or ship. That is, at the time when the stacking location of each steel plate must be determined, each steel plate has a random processing start date that cannot be standardized. Moreover, it is challenging to formulate this as an optimization problem because the stacking status for positioning the next steel plate is altered the instant the steel is deposited in a specific stack. In this study, we conducted an artificial intelligence research to reduce the number of crane movements by minimizing the sorting work in the steel stock yard. We introduce a reinforcement learning algorithm that determines the optimal location (of the file) considering the input date of the steel plates.

2 LITERATURE REVIEW

Lee et al. (2019) conducted a study on scheduling problems based on reinforcement learning for semiconductor manufacturing processes. In this study, SARSA algorithm was applied to learn the model to determine the work to be put into the process equipment among the jobs waiting in the buffer based on the scores for various dispatching rules. Kim (2019) conducted a study on reinforced learning-based scheduling with the aim of learning a planning model applicable to various production environments for semiconductor packaging lines. To this end, this study focused on the robustness of the learning model and studied the scheduling problem using the deep Q network-based reinforcement learning algorithm to which normalized learning was applied. Shin et al. (2010) conducted a study to learn the scheduling method that adaptively changes the dispatching rule as the probability of rework changes with respect to manufacturing systems that have an unstable rework probability. Zhang et al. (2017) conducted a study on sequencing problem of determining the input order of jobs waiting in queues by applying simulation-based value iteration and Q-learning. In the subsequent study of Zhang et al. (2018), reinforcement learning method was applied to sequencing problem of batching in jobshop. Fateme et al. (2013) conducted a study to model the crane scheduling problem related to the selection of a transport truck that minimizes the waiting time of a container transport truck at a container terminal by applying a q-learning-based reinforcement learning algorithm. Hirashima (2008) applied the reinforcement learning algorithm based on q-learning to solve the marshalling plan problem of moving the initially randomly placed containers in an optimal layout for shipping containers in the order in which they are moored at the container terminal. A learning study was conducted. Specifically, the learning algorithm was applied to each marshalling plan by determining the order of containers for relocation and determining the location of the container. Shen et al. (2017) applied the DON reinforcement learning algorithm to the ship stowage planning problem to determine the slot of the ship to which the container will be stacked, so that the availability, reshuffling, and crane usage (yard crane) of the plan were applied. A study was conducted to learn the optimal plan from the perspective of shifting. Verma et al. (2019) conducted a study on applying reinforcement learning to scheduling problem of container loading to minimize the shuffling tasks using crane. In cases of steel stock yard, Kim et al. (2011) conducted a study to formulated the stacking problem in steel stock yard and analyzed several mathematical approaches to solve the problem. In the study of Fechter et al. (2018), Sarsa(λ) algorithm was applied to stacking problem in steel industry to minimize shuffling movements which are required to meet the delivery sequence of produced steel slabs.

3 OBJECTIVE

The objective of this study is to develop a reinforcement learning algorithm and environment to determine the optimal stack location to minimize the sorting work (or number of crane movements) until the steel plates received from the outside are input to the process. To achieve this, the selectable actions and the states of the stock yard must be defined so that the problem of determining the steel location in the steel stock yard can be learned. Next, the steel stock yard environment that can be learned according to the defined problem must be configured. The selectable actions, state variation according to the action, and reward for the selected action must be defined for the steel stock yard environment. For the reinforcement learning algorithm, we select an algorithm that exhibits the highest probability of success by applying

established algorithms (e.g., Q-learning, Deep Q-Network (DQN), Advantage Actor-Critic (A2C), and Asynchronous Advantage Actor-Critic (A3C)). Problem Definition

3.1 Operation of Steel Stock Yard

First, we describe the analysis of the process of a steel stock yard, which is the objective of the actual problem. Figure 1 shows a steel stock yard in an actual shipyard. The steel plates are input through the following three-step process.

3.1.1 Unloading Bay → Main stock

When steel plates arrive in a random order at the unloading bay of the shipyard through a ship (or barge), the arrived steel plates are transported to the stock yard in the order of arrival and stacked up. This main stock is composed of approximately 20 stacks.

3.1.2 Main Stock → 1st Sorting Area

When a weekly processing plan is established from the production plan, the corresponding steel plates are transported from the main stock to the first sorting area.

3.1.3 1st Sorting Area → 2nd Sorting Area

The steel plates for a week in the first sorting area are sorted by day and piled in the second sorting area. The second sorting area is close to the conveyer transfer device for transportation to the processing plant. The steel plates stacked in the second sorting area are transported to the processing plant conveyor using a crane according to the input schedule of the processing plant.

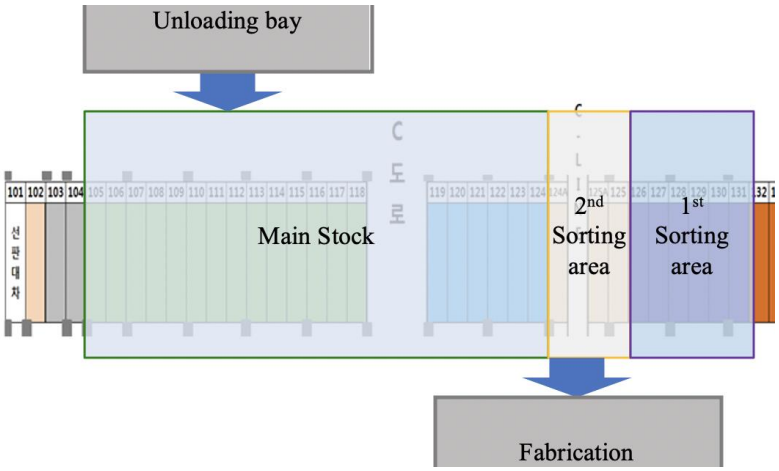

Figure 1: Steel stock yard of shipyard.

3.2 Requirement Analysis

The initial requirement of the shipyard manager is to develop a separate learning algorithm for the three processes described in Section 3.1. We examined the feasibility of separately developing the artificial neural network for determining the stacking location of the main stock and that for sorting in the first and second sorting areas. However, in the sorting in a previous study (reference), this approached yielded a good result for 4–6 stacks, where 10 stacks can be stacked, whereas the learning did not converge for larger stockyard sizes. Because we have not determined a solution for this problem, the artificial neural network learning for sorting was excluded from this study.

Meanwhile, the sorting work can be minimized if the stacking location is determined appropriately (so that the steel plates are stacked in the order of processing date) when steel plates are shifted from one area to the next, in which case sorting becomes a secondary problem. Therefore, to summarize the requirement, although the decision on stock location and sorting of steel plates are independent tasks, the sorting work can be minimized by optimizing the decision on stock location. Hence, we decided to focus on learning about the decision on stock location. The learning for each transport section can be performed by altering the state definition and steel plate input using the same algorithm. Therefore, we set the research direction to design the environment using parameters and develop an artificial neural network for decision on stock location for various states.

4 REINFORCEMENT LEARNING FOR DECISION ON STOCK LOCATION

Figure 2 shows a diagram of reinforcement learning for decision on steel location and summarizes the content of this paper. Reinforcement learning at the top is the algorithm for learning. For this, we used DQN, A2C, and A3C in this study. This learning algorithm delivers the selected action information to the environment in the middle, receives the state and reward feedback from the environment, and updates the weight of the neural network. The environment in the middle is implemented to interoperate with the learning algorithm by extracting only the part (stacking location) required for learning from a real steel stock yard.

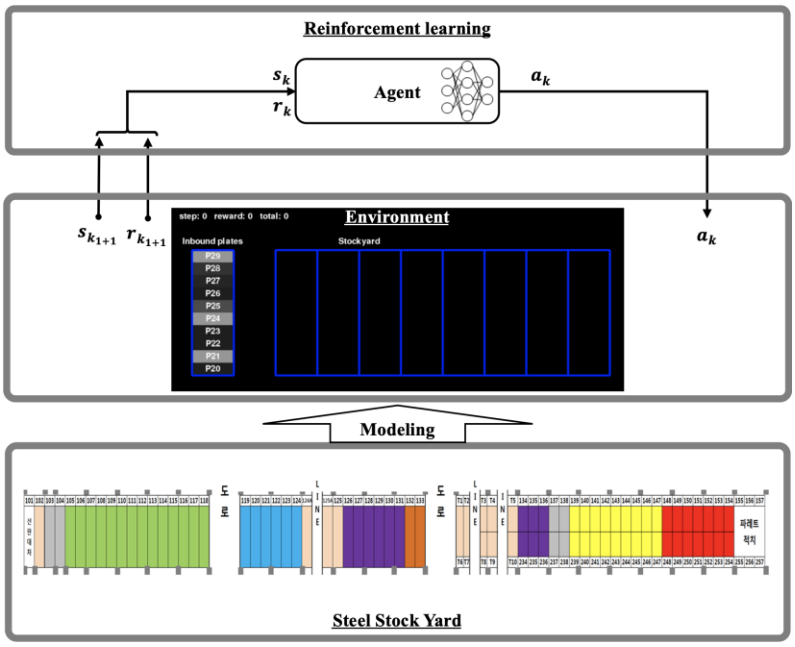

Figure 2: Diagram of reinforcement learning for decision of steel location.

4.1 Reinforcement Learning Algorithm

Next, the reinforcement learning algorithms used in this study are introduced. In this study, we compared the results of learning using DQN, A2C, and A3C.

4.1.1 DQN

The DQN algorithm was developed from the Deep SARSA(State-Action-Reward-State-Action) algorithm. It approximates the Q function for action to a neural network called Q-Network and updates the weight of

the Q-Network to receive the maximum reward as learning progresses. Here, undertaking an action (a) in the current state (s), receiving the reward (r) and the next state (s') from the environment, and undertaking the next action (a') are used as one sample (s, a, r, s', a'). Then, the weight(θ) of the Q-Network is updated using a loss function for the mean square error (MSE) defined by the following (1) (γ is a discounting ratio):

$$MSE = (r + \gamma \max_{a'} Q(s', a'; \theta) - Q(s, a; \theta))^2 \quad (1)$$

4.1.2 A2C/A3C

The A3C algorithm (which was developed from the existing A2C algorithm) updates the global network with the independent learning result of each of several agents for learning using A2C. A2C approximates the value function to a value neural network called Critic and also approximates the policy to a policy neural network called Actor. In A2C, the differential value of the policy neural network loss function is determined by multiplying the cross entropy function by the Q-function. Owing to the large fluctuations in the Q-function values, the value obtained by subtracting the value function from the Q-function, with the value function as the baseline, is defined as an advantage function (as shown in (2)). The advantage function is used rather than the Q-function. (S_t and A_t respectively mean state and action at time step t)

$$A(S_t, A_t) = Q_w(S_t, A_t) - V_V(S_t) \quad (2)$$

However, in actual learning, for more efficient calculation, the Q-Network for approximating the Q-function is not defined separately. In addition, the value approximated from the value neural network is used for the Q-function. The Q-function for the action undertaken in the current state is approximated by adding the reward to the value function of the next state, to which the discounting ratio(γ) was multiplied, as shown in (3). (R_t means a reward at time step t)

$$\delta_V = R_{t+1} + \gamma V_V(S_{t+1}) - V_V(S_t) \quad (3)$$

Finally, the equation for updating the parameters(θ_t) of the policy neural network($\pi_\theta(a|s)$), including the approximate advantage function, is expressed as follows by the following (4) (α is a learning rate for updating weights):

$$\theta_t \leftarrow \theta_t + \alpha[\nabla_\theta \log \pi_\theta(a|s)\delta_V] \quad (4)$$

Furthermore, for the value neural network, the weight is updated by using the MSE as a loss function (as in (5)). This is performed in a manner similar to that of updating the Q-Network in the DQN.

$$MSE = (R_{t+1} + \gamma V_V(S_{t+1}) - V_V(S_t))^2 \quad (5)$$

4.2 Environment modeling

The environment plays the role of feeding back the new state and reward obtained through simulation after receiving the selected action from the learning algorithm. The environment for this study was configured as follows.

4.2.1 State

The states of environment input to the artificial neural network for reinforcement learning were defined, as shown in Figure 3. The left single column in Figure 3 indicates the loaded steel plates. Each number

represents the time remaining until the steel plate is supplied to the processing system. The grid on the right side is the environment of the steel stock yard. The horizontal direction indicates the number of piles that can be piled up, and the vertical direction indicates the maximum number of steel plates that can be piled in each stack. In this study, an identical maximum height was defined for all the piles. The number inside each grid is initially zero. The grid is filled with the residual period of steel input as the stacking-up simulation progresses. Furthermore, the process was simplified so that the steel plates are deleted when the loading time arrives as time progresses (time progresses step-by-step in the algorithm) without movement of the stacked steel plates. In addition, the state updated by the neural network was implemented to include the loaded steel plate and the stacking information of the steel stock yard.

4.2.2 Functions for generating input into environment

Table 1 presents the information of the steel plates used in the learning. The plate number, loading date, and fabrication date are loaded to the environment as steel plate information, as illustrated in Table 1. The number of steel plates included in the learning was adjusted according to the problem size. The learning scope was adjusted according to the learning strategy. The example of input functions used in this study are listed in Table 2.

4.2.3 Reward

To explain the reward applied in this study, we assume a steel stock yard problem that has 4 × 4 state spaces with four steel plates that can be stacked in each one of four piles. In each pile, the stacks were numbered sequentially from the bottom. That is, it was assumed that the stack agent no. 0 undertook action to stack a steel plate at the bottom stack in the first pile.

Table 1: Input data (partially selected).

Plate number	Loading date	Fabrication date
2467ALP611NG002	2019.3.8	2019.5.16
2469ALP182NG025	2018.12.27	2019.2.21
2477ALP629FZ028	2019.2.21	2019.3.22

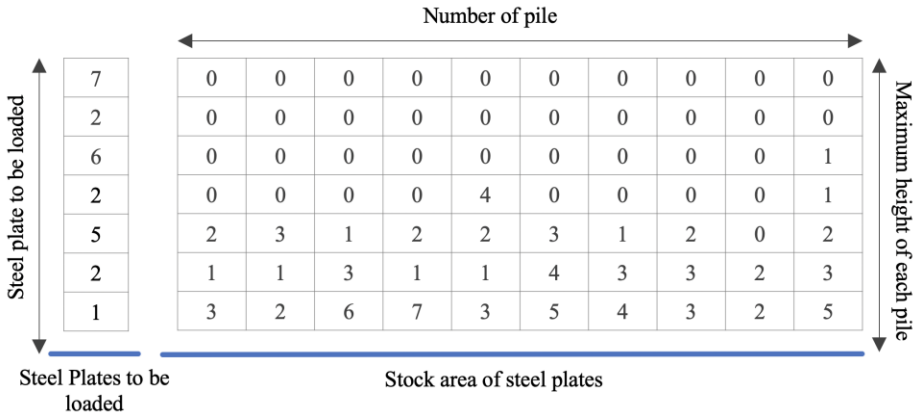

Figure 3: Environmental parameters and example of states of environment.

Table 2: Functions for generating input.

Generation function	Description
import_plates_schedule_rev(file)	• Input function that receives data on all the steel plates and applies them to learning. • Removes the missing value in the input data and converts the date information (loading date and release date) of all the steel plates into an integer format based on the loading date. • Generates objects for each steel plate, sorts them in the order of process loading date, and returns them as a list.
generate_schedule(num_plate)	• Input function that generates random steel plate data for all the steel plates. • The loading date and release date of steel plates are generated based on the statistics obtained by analyzing the input data. The loading date interval between steel plates and the piling period are assumed to follow exponential and beta distributions, respectively. • Generate steel plate objects for each steel plate in the order of loading date and returns them as a list.

- **The reward is zero when the number of steel plate on the pile is one.**

When only one steel plate is tacked on a pile, as shown in Figure 4, it is not necessary to use a crane for sorting. There is no standard for assessing whether to use an additional crane because there is no steel plate stacked in the lower part. In this study, we defined the reward for this case as 0.

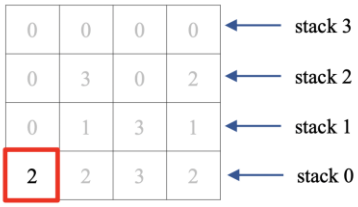

Figure 4: Case of reward is 0.

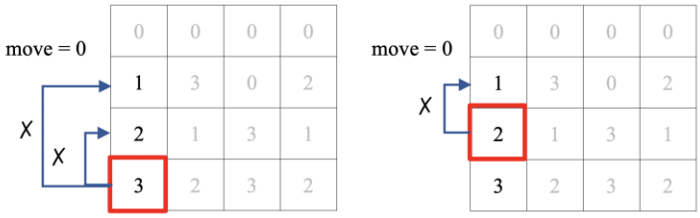

Figure 5: Case of maximum number of crane transportation is 0.

- **Stocking two or more steel plates in one pile: the number of additional crane movement is zero.**

The release date for the steel plates in stack 0 is 3, as shown in Figure 5. The number of steel plates above it that have a later release date is zero. Therefore, the number of additional crane transportations required to release the steel plate in stack 0 on the release date is zero. Next, the release date of the steel plate in stack 1 is 2, and the number of steel plates above it that have a later release date is zero. Thus, the number of additional crane transportations required to remove the steel plate in stack 1 on the release date is also zero. Consequently, the number of additional crane transportations required to release the steel plates from the corresponding pile becomes zero. This scenario corresponded to a desirable case, and the highest reward value (2) was assigned.

- **Stocking two or more steel plates in a pile: the number of additional crane movement is not zero.**

The release date of the steel plate in stack 0 is 1, as shown in Figure 6. The number of steel plates above stack 0 that have a later release date is three. Therefore, the number of additional crane transportations required to remove the steel plate in stack 0 on the release date is three. Next, the release date of the steel plate in stack 1 is 2. Thus, there are two steel plates above stack 1 that have a later release date: the steel plate in stack 2 (release date: 4) and that in stack 3 (release date: 3). Therefore, the number of crane transportations required to remove the steel plates in stack 1 (release date: 2) on the released date is two. Concerning the steel plate in stack 2, there is no steel plate above stack 2 which have a later release date, so that the number of unnecessary crane transportations required to remove the steel plates in stack 2 on the

release date is 0. Finally, the reciprocal of the maximum number of additional crane transportations (i.e., of three) was assigned as the reward.

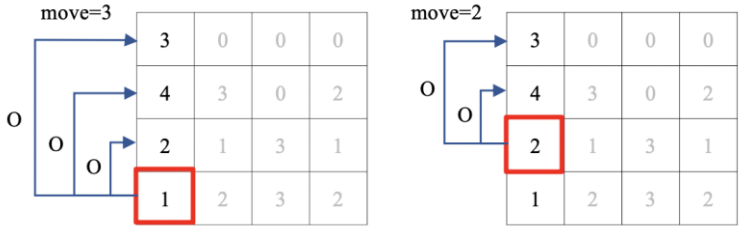

Figure 6: Case when the maximum number of required crane transportation is not zero.

5 LEARNING RESULTS

In this study, the DQN, A2C, and A3C algorithms were selected based on validation and performance. These algorithms are well-known algorithms that have been verified through various problems, and . DQN weakens the correlation between samples and enhances the stability of convergence by using reply memory and target network. A2C, a policy-based algorithm, overcomes the limitations of REINFORCE algorithm that learning can be only done at the end of episodes, and allows the agent to learn at each step rather than episode. A3C runs A2C algorithm asynchronously, not only weakening the correlation between samples but also being able to use the latest data for updating neural network. These algorithms were applied to sorting problem in steel stock yard using environment, state, and reward, which were introduced in Chapter 4. In addition, the results were analyzed.

5.1 DQN

Starting with an artificial neural network that has a small number of layers, the model was modified in the direction that was effective for learning, by adding layers as follows:

- **Case 1**: Artificial neural network consisting of a hidden layer with 10 nodes
- **Case 2**: Two convolution layers added (4 × 4 filter, 2 × 2 filter)
- **Case 3**: Same artificial neural network as case 2 with an increased number of steel plate data

The reward did not converge even for small-sized problems while learning with the neural network consisting of a hidden layer as in case 1 (Figure 7). To improve this problem, two CNN layers were added as in Case 2. The reward converged, and learning was feasible for the same model as in case 1 (Figure 8). However, the reward did not converge when the problem size increased as in case 3 (Figure 9). This means that agent could not find the optimal solution as the complexity of state space increased. This rendered the DQN learning algorithm inappropriate for the steel stock yard problem.

5.2 A2C

Next, we applied the A2C learning algorithm. As with DQN, starting with an artificial neural network having two layers, the model was modified in the direction of addition of the layers.

- **Case 1**: Artificial neural network consisting of two hidden layers with 15 nodes each
- **Case 2**: Two convolution layers added (4 × 4 filter, 2 × 2 filter) with an increased number of steel

As with the reward result of case 1 in Figure 10, the learning progressed in the desired direction for A2C as in DQN when the number of steel plates was small. However, as shown in the reward result for

case 2 in Figure 12, the reward did not converge when the problem size increased (increased number of loaded steel plates). Thus, it was determined that A2C also could not be applied to real problems.

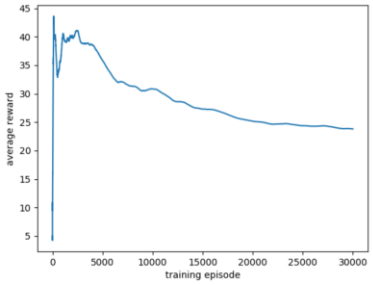

Figure 7: Reward result of case 1 (DQN learning)

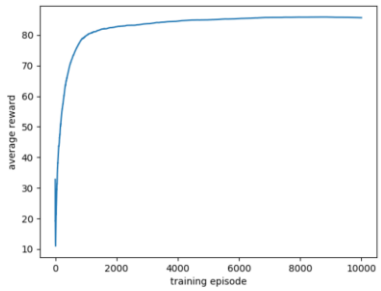

Figure 8: Reward result of case 2 (DQN learning)

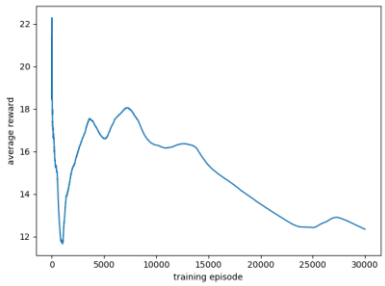

Figure 9: Reward result of case 3 (DQN learning)

5.3 A3C

In the case of A3C, learning was performed effectively for the case of many loaded steel plates. It had failed for DQN and A2C. Therefore, the case of learning for a fixed sequence of input and that of learning for a random sequence of input were compared for A3C learning. The results from A3C were output and compared in graphic interchange format (GIF). It facilitates the visual observation of the behaviors for each episode of the learning process of the environment.

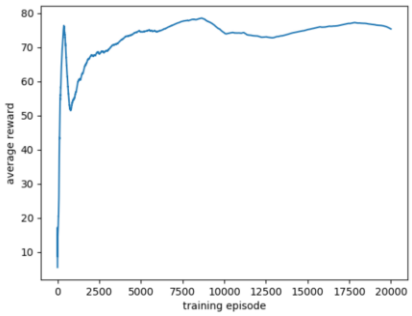

Figure 10: Reward result of case 1 (A2C learning)

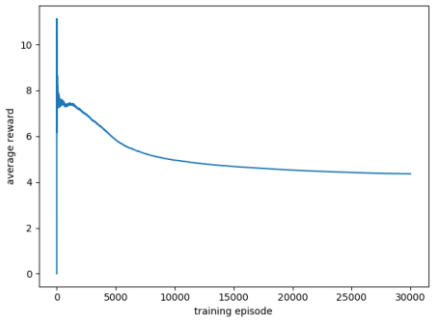

Figure 11: Reward result of case 2 (A2C learning)

5.3.1 Learning for Fixed Sequence of Input

We performed learning for the case of stacking 38 steel plates in a stock yard in six piles and a maximum stack height of 10 (first sorting area of the targeted shipyard). The data used in this learning scenario is real data of weekly supply of steel plates obtained from shipyards. Then, thousand episodes were performed, with the stacking of 38 steel plates as one episode. The sequence of the 38 steel plates was fixed. When learning was performed under these conditions, the reward almost converged after approximately 6,000 episodes. Furthermore, the GIF image of each episode revealed that the steel plates with a long stock period1 were stacked in the lower positions of the pile (Figure 12–Figure 13).

[1] A darker GIF image indicates a steel plate with a longer time until the loading date.

5.3.2 Learning random sequence of input

Next, we performed learning for the same environment in previous section. However, in this case, the input sequence for the 38 steel plates was varied randomly in each episode to achieve the universality of the artificial neural network. The data used in this learning scenario is generated data of weekly supply of steel plates based on analysis of actual data. In this case, 50,000 episodes of learning were performed because the problem had become more complex than the one in 5.3.1 owing to the random sequence of input.

Figure 12: Locating result after 1,000 episodes (fixed sequence of input).

Figure 13: Locating result after 10,000 episodes (fixed sequence of input).

As a result of the learning, the reward converged after approximately 40,000 episodes. This implies that it requires computation approximately six or seven times that required to perform learning for the fixed sequence of input. As with the case in 5.3.1, the GIF image of each episode revealed that the steel plates with a longer stock period were stacked in the lower part of the pile (Figure 14).

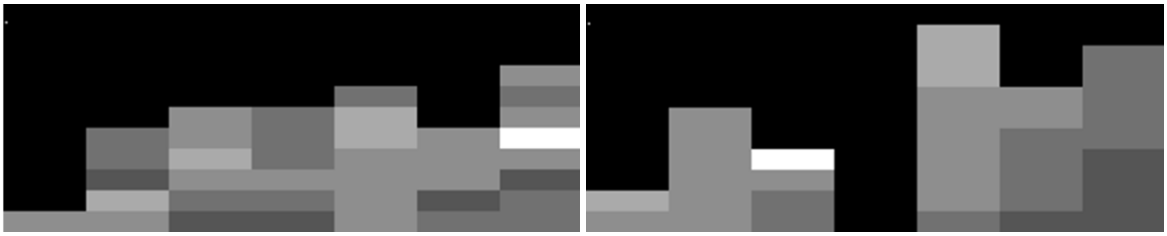

Figure 14: Locating result after 5,000 episodes and 50,000 episodes each (random sequence of input).

5.3.3 Learning for random sequence of input in main stock

Finally, we tested learning for stacking 254 steel plates in the main stock yard (20 piles, maximum stack height: 15). The number of input steel plates was increased to 254 (number of input steel plates for approximately 15~16 week). The data used in this learning scenario is data of steel plates generated based on real data. Furthermore, 50,000 episodes were performed, with the stocking of 254 steel plates as one episode. The reward continued to increase even after approximately 50,000 episodes were performed. However, as shown in Figure 15, the steel plates were arranged in the desired sequence to a certain degree. However, it was not optimum until step 100 (100th steel plate input) in the 50,000th episode. However, when it was close to the final step, the steel plates with a long time until the loading date were positioned

above the middle. This is an undesirable result. This result can be analyzed as follows. First, sufficient learning in the height direction was not achieved because the stack height was insufficient. Furthermore, because the reward curve continues to rise without converging, it can be presumed that sufficient learning is achieved only if over 50,000 episodes are performed. However, in this case, the calculation will consume over 30 min when the calculation time is divided in units of 250 episodes. Hence, additional calculations could not be performed in this study because of the limitation on calculations using a general desktop computer (it requires 5 days for 50,000 episodes of learning).

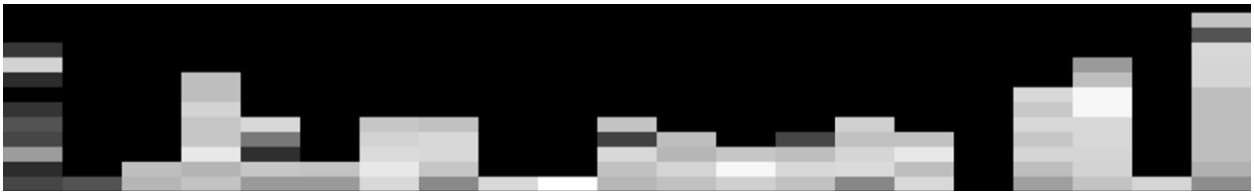

Figure 15: Snapshot of step 100 in episode 50,000 (random sequence of input).

6 CONCLUSION

We developed an artificial neural network that can determine the optimal positions of steel plates (to minimize the number of additional crane transportations) loaded in a steel stock yard of a shipyard, using reinforcement learning. The result revealed that A3C exhibits a higher performance than A2C and DQN. Although the neural network learning methods of A3C and DQN are different, they use various empirical data. However, the samples in the replay memory of DQN also include old data on the previous states. This is because they were collected by an agent while following the steps. However, A3C exhibited better learning results because it receives data from agents that perform steps simultaneously and displays the advantage of learning with more recent data. However, we intend to introduce a dedicated computer or use external services for calculation in future research because the computational capacity of general computers becomes inadequate when the problem size increases.

ACKNOWLEGEMENT

This research was supported by following research projects:

1) The new faculty startup fund from Seoul National University.
2) IoT and AI based development of Digital Twin for Block Assembly Process (20006978) of the Korean Ministry of Trade, Industry and Energy
3) ‘Mid-sized shipyard dock and quay planning integrated management system (20007834)) of the Korean Ministry of Trade, Industry and Energy
4) Education program in design and regulation experts in autonomous ships (KITECH 2019-0259-01)

REFERENCES

Fateme, F., N. Huynh, J. M. Vidal, and Y. Xie. 2013. “Modeling Yard Crane Operators as Reinforcement Learning Agents”. *Research in Transportation Economics* 42:3-12.

Fechter, J., A. Beham, S. Wagner, and M. Affenzeller. 2018. “Approximate Q-learning for Stacking Problems with Continuous Production and Retrieval”. *Applied Artificial Intelligence* 33(1):1-19

Hirashima, Y. 2008. “An Intelligent Marshalling Plan using a New Reinforcement Learning System for Container Yard Terminals”. In *New Developments in Robotics Automation and Control*, edited by A. Lazinica, 181–194. London: IntechOpen.

Kim, B., J. Koo, and H. Sambhajirao. 2011. “A Simplified Steel Plate Stacking Problem”. *International Journal of Production Research* 49(17):5133-5151.

Kim, J.K. 2019. "Enhancing Robustness of Deep Reinforcement Learning based Semiconductor Packaging Lines Scheduling with Regularized Training". Master's degree thesis, Department of Industrial Engineering, Seoul National University, Seoul, South Korea.

Lee, W., B. Kim, K. Ko, and H. Shin. 2019. "Simulation Based Multi-Objective Fab Scheduling by Using Reinforcement Learning". In *Proceedings of the 2019 Winter Simulation Conference*, edited by N. Mustafee, M. Rabe, K. Bae, C. Szabo, and S. Lazarova-Molnar, 1672-1683. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Shin, H., and J. Ru. 2010. "An Adaptive Scheduling Algorithm for Manufacturing Process with Non-stationary Rework Probabilities". *Journal of the Korea Academia-Industrial Cooperation Society* 11(11):4174-4181.

Shen, Y., N. Zhao, M. Xia, and X. Du. 2017. "A Deep Q-learning Network for Ship Stowage Planning Problem". *Polish Maritime Research* S3(95) 24:102-109.

Verma, R., S. Saika, H. Khadilkar, P. Agarwal, and G. Shroff. 2019. "A Reinforcement Learning Framework for Container Selection and Ship Load Sequencing in Ports". In *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems*, May 13th-17th, Montreal, Canada, 2250-2252.

Zhang, T., S. Xie, and O. Rose. 2017. "Real-time Job Shop Scheduling based on Simulation and Markov Decision Processes". In *Proceedings of the 2017 Winter Simulation Conference*, edited by V. Chan, A. D'Ambrogio, G. Zacharewicz, and N. Mustafee, 3899-3907. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Zhang, T., S. Xie, and O. Rose. 2018. "Real-time Batching in Job Shops based on Simulation and Reinforcement Learning". In *Proceedings of the 2018 Winter Simulation Conference*, edited by M. Rabe, A. Skoogh, N. Mustafee, and A. Juan, 3331-3339. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

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.

SANG HYUN YU is a bachelor's degree student in the Department of Naval Architecture and Ocean Engineering at Seoul National University. His research interest is in DES simulation, APS(Advanced Planning and Scheduling), machine learning and queuing. His email address is yush0123@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.

HAOYU ZHU is a graduate student in the Department of Naval Architecture and Ocean Engineering at Seoul National University. He received a bachelor's degree in Naval architecture and ocean engineering from the University of Ulsan. He is currently researching supervised learning and deep learning in machine learning. His email address is zhuhaoyu838@snu.ac.kr

DONG HUN KWAK is a master's course in the Department of Naval Architecture & Ocean Engineering at Seoul Nat'l University. He received his bachelor's degree in the same department. His research interests include stochastic modeling, queueing theory, data analytics, and machine learning. His email address is s2arta2s@snu.ac.kr.

JONG-HO NAM is professor in the Division of Naval Architecture and Ocean Systems Engineering at Korea Maritime & 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 computational ship design and production, geometric modeling, and application of virtual environments. His email address is jhnam@kmou.ac.kr.