Proceedings of the 2024 Winter Simulation Conference
H. Lam, E. Azar, D. Batur, S. Gao, W. Xie, S. R. Hunter, and M. D. Rossetti, eds.

MATRIX ASSEMBLY SYSTEM SCHEDULING OPTIMIZATION IN AUTOMOTIVE MANUFACTURING: A DEEP Q-NETWORK APPROACH

Whan Lee[1], Seog-Chan Oh[2], Jisoo Park[1], Changha Lee[1], Hua-tzu Fan[2], Jorge Arinez[2], Sejin An[1], and Sang Do Noh[1]

[1] Dept. of Industrial Engineering, Sungkyunkwan University, Suwon-si 16417, REPUBLIC OF KOREA
[2] General Motors Research and Development, 30500 Mound Road, Warren, MI 48090, USA

ABSTRACT

In response to the demand diversification in automobile production, traditional manufacturing processes are transitioning towards more flexible systems with dynamic scheduling methods. The Matrix System (MS) stands out for its utilization of Autonomous Mobile Robots (AMRs) and multi-purposed workstations, enabling a dynamic production environment. Each AMR is tasked with transporting a partially assembled vehicle through multiple workstations until final assembly, adhering to predefined precedence orders. However, determining operation schedules amidst the complexity of multi-model systems poses a significant challenge in minimizing manufacturing time. To address this, we formalize the problem into a Markov Decision Process (MDP) and propose a Deep Q-Network (DQN) based scheduling optimization algorithm for the Vehicles Production Scheduling (VPS) problem. Our approach utilizes discrete event simulation to assess candidate actions suggested by the DQN, aiming to derive an optimal policy. This paper validated the proposed algorithm by comparing with various dispatching rules.

1 INTRODUCTION

As product diversity increases and consumer preferences rapidly change in the automotive market, automobile production systems are also undergoing transformation. In response to this trend, automotive assembly processes are transitioning from traditional conveyor systems to MS (Greschke et al. 2014; Holweg and Pil 2005; Kern et al. 2015; Oh et al. 2022). MS consists of a production system where all workstations are arranged in a matrix format (Kim et al. 2022a), with each workstation manned by workers or automated by robots, performing assembly operations for car production. This MS setup leverages multi-purpose equipment capable of performing various operations within a facility and utilizes computer systems for flexible routing and process adaptability, achieved through the use of AMRs (Kim et al. 2022b; Julaiti et al. 2022; Alatise and Hancke 2020).

However, a challenge arises from the fact that the services provided by AMRs vary depending on the designed algorithms. While MS can employ multiple AMRs to increase automobile throughput, overcrowding AMRs at workstations can lead to increased waiting and cycle times for vehicles, resulting in decreased throughput rates per unit time interval between two states. Hence, MS requires enhanced scheduling algorithms for assigning AMR operation priorities and route planning based on sequential procedures and workstation conditions. The complexity of MS utilizing multiple AMRs presents a challenge in devising algorithms that ensure optimal effectiveness, addressing its dynamic nature and high-dimensional state complexity (Kim et al. 2022b).

To address these challenges, this paper proposes the utilization of the DQN algorithm for the flexible VPS problem within automotive assembly processes. Initially, the paper formalizes the MDP for the system. Subsequently, it employs a DQN algorithm to find the optimal policy. Finally, the effectiveness of the algorithm is validated through simulations, comparing it with various dispatching rules.

2 LITERATURE REVIEW

Enhancing system performance through job priority optimization is crucial in various manufacturing systems. Johnson (1954) initiated extensive research on job shop scheduling models. These models involve processing a series of jobs across multiple machines, requiring diverse operations to follow sequential procedures. Researchers such as Cummings and Egbelu (1998), Gupta et al. (2001), Kubzin et al. (2009), and Ivanov et al. (2016) have significantly contributed to this area. These problems have been proven to be NP-hard, with a few exceptions (Gonzalez and Sahni 1978).

The objective of the job shop scheduling problem is to minimize the makespan (the time to complete all jobs) by determining the optimal processing sequence for each job. Efforts to solve various types of production line scheduling problems have continued using mathematical, heuristic, and metaheuristic techniques (Carlier and Pinson 1989; Park et al. 2002; Rifai et al. 2016). However, ensuring high effectiveness while accounting for the complexity and dynamic nature of real-world problems remains challenging.

Due to the Markovian nature of dynamic scheduling problems, Deep Reinforcement Learning (DRL) is considered suitable for addressing these challenges (Liu et al. 2022). DRL addresses agent learning and decision-making through trial and error, integrating deep learning to enable the agent to make decisions from unstructured input data and approximate state values as nonlinear functions using deep neural networks (Jeon et al. 2022). According to Julaiti et al. (2022), DRL can handle uncertainties such as machine failures or the arrival of emergency queues in dynamic operation scheduling.

Among DRL algorithms, the DQN has been proposed in many studies due to its efficient learning and high stability, allowing effective learning even in high-dimensional and complex state spaces. Waschneck et al. (2018) proposed a DQN approach for scheduling lot movements between locations in semiconductor manufacturing facilities. Shi et al. (2020) proposed a DQN approach for job transfers within individual manufacturing systems. Hu et al. (2020) applied DQN to scheduling problems in Flexible Manufacturing Systems modeled with Petri nets, comparing its efficiency with heuristic methods. Jeon et al. (2022) validated the applicability and effectiveness of a scheduling system developed using DQN by applying situation-specific dispatching rules in re-entrant production lines.

3 PROBLEM DEFINITION

This study addresses the scheduling problem aimed at maximizing vehicle production throughput in the assembly process using a MS that employs multi-purpose equipment and AMRs. The assembly operations at each workstation are predetermined and executed sequentially, as illustrated in Figure 1.

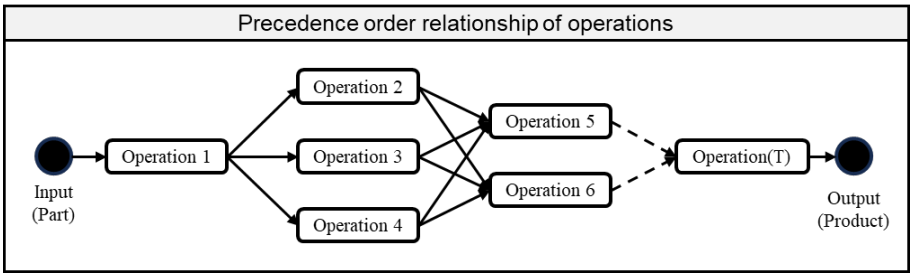

Figure 1: Example of precedence orders to complete operations in MS.

In Figure 1, assembly operations labeled with operation names are performed in sequence according to the step numbers. Specifically, step 1 of the operation must be executed first, followed by steps 2, 3, and 4 in any order as they can be performed in parallel. After steps 5 or 6 are completed, the final operation 'T' is performed. To increase vehicle production throughput in the automotive assembly process, many AMRs are deployed. With the increase in AMRs, the flow of partially assembled vehicles also increases, thereby

enhancing throughput. However, the number of workstations capable of performing operations is limited, leading to constraints. To address this issue, setting objectives to increase throughput in the automotive assembly process is crucial. Typically, the goal is to minimize the makespan, which is the total time required for product production. Minimizing the total time needed to complete the scheduled production volume is a key metric in automotive manufacturing. Accordingly, this paper aims to minimize the makespan associated with the allocated vehicle production target, which is equivalent to the time taken for all AMRs to complete the assembly process. The following summarizes the objectives, decision factors, and constraints for the VPS problem defined in MS.

- Objective:
  - Minimization of the makespan required for product production.
- Decision Factors:
  - Selection of AMR's workstation.
  - Selection of the next operation to be performed by AMR within the workstation.
- Constraints:
  - All operations follow a sequential procedure.
  - Each AMR can transport only one vehicle at a time, and vehicle changeovers are not allowed until all operations are completed.
  - Workstations can perform a limited number of operations.
  - Each workstation can handle only one vehicle per operation, and the AMS holding the next operation must wait in buffer.
  - Workstations with fully occupied buffers are not selectable.

The AMR selects an operation and then chooses a workstation capable of performing that operation. The selection of the workstation follows predefined path selection rules. If the chosen workstation is busy, the AMR moves to a buffer to wait until the preceding operation is completed. If the buffer is full, the workstation is excluded from the path planning, and an alternative workstation is selected. AMRs in the buffer must prioritize the next operation. Figure 2 illustrates this decision-making process of the AMR through an activity diagram. The completion time is significantly affected by which workstation performs the operation and which AMR transfers the finished operation. Therefore, algorithms are needed to plan the paths of AMRs and set operation priorities based on the sequential procedure and overall workstation status. However, achieving an optimal policy within a reasonable computation time is challenging due to the following issues. (1) Despite the use of many AMRs to increase product throughput, the availability of workstations capable of performing operations is limited. (2) As the number of AMRs increases, the number of states to consider grows exponentially. For some products, precedence relationships between tasks require sequential order. To address these issues, an algorithm capable of recognizing large-scale state spaces and making efficient decisions is necessary.

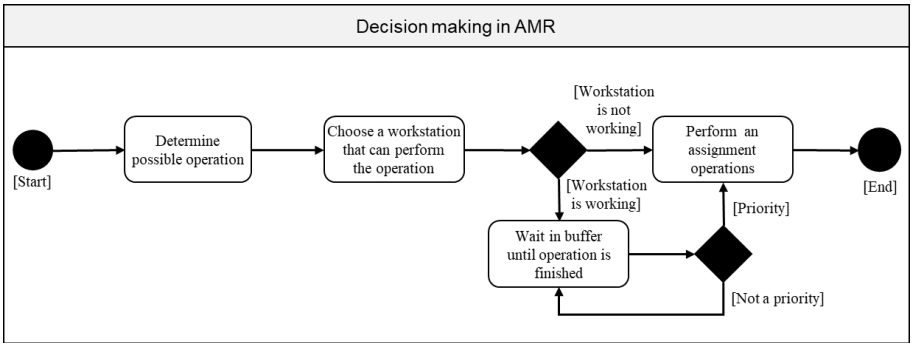

Figure 2: Activity diagram of an AMR.

4 DEEP Q-NETWORK ALGORITHM FOR MINIMIZING MAKESPAN IN MATRIX ASSEMBLY SYSTEM

In this section, we describe the proposed DQN algorithm for addressing the VPS problem to minimize the production makespan in MS. To achieve this, we first model the given problem as an MDP. The notation for MDP modeling is shown in the Table 1.

Table 1: Notation for MDP modeling.

Symbol	Description
i	Index of workstation, $i \in W$, $W = \{0,1,2,\cdots,l\}$
j	Index of operation , $j \in J$, $J = \{1,2,\cdots,m\}$
k	Index of vehicle, $k \in K, K = \{1,2,\cdots,n\}$
b	Index of buffer area,$b \in B, B = \{1,2,\cdots,h\}$
g	Index of job progress,$g \in G, G = \{1,2\}$, 1= True, 2= False
o_i	Remaining progress time of the i-th workstation
w_i	Working time of the i-th workstation between observation points
P_k	Process time of k-th vehicle between observation points
R_k	Remaining total process time of the k-th vehicle
t_{ik}	Remaining progress time of the k-th vehicle at the workstation i-th

4.1 Markov Decision Process Modeling

To apply the DQN algorithm to the VPS problem for minimizing the makespan, the given problem needs to be modeled as an MDP. The MDP model of this study is used to clarify the interaction between the agent and the environment in terms of 3 components such as states, actions, and rewards. Firstly, the state is characterized by three component information:

1. Number of vehicles under production: It represents the number of products currently being processed in the system, the number of deployed vehicles, and is expressed as a scalar value.
2. Vehicles information: It represents the state group of all products. Each product is assigned a vehicle identification number (k) based on the deployment order, and it processes tasks while moving through workstations via the AMR. As the product progresses through tasks, it includes the identification number of the current workstation, the status of progress, the remaining time to complete the current progress (Remaining Progress Time at time t , RPT_t), and the total remaining time to complete all progress (Total Remaining Progress Time at time t, TRPT_t). For example, if n vehicles are scheduled to be deployed on the production line, the state information of each vehicle is represented as shown in Table 2.
3. Workstation information: It represents the state group of all workstations. For each workstation, it includes the remaining time for the ongoing operation (Remaining Progress Time of the i-th Workstation at time t, $\mathrm{RPTW_t}$) and the next task time for the vehicle located in the buffer area of the i-th workstation (Next Progress Time of the vehicle located in the b-th Buffer area of the i-th Workstation, NPTB_b) sequentially. For example, if there are l workstations and b buffers, the workstation information is represented as shown in Table 3.

Table 2: Example of partially assembled vehicle status information table.

Vehicle index	Location	Process	RPT_t	TRPT_t
1	1	1	t_{11}	$R_1 - P_1$
⋮	⋮	⋮	⋮	⋮
n	w	g	t_{in}	$R_n - P_n$

Table 3: Example of workstation status information table.

Workstation index	RPTW_t	NPTB_1	⋯	NPTB_b
1	o_1	t_{1k}	⋯	t_{1k}
⋮	⋮	⋮	⋯	⋮
l	o_l	t_{lk}	⋯	t_{lk}

In Table 2, the position of a vehicle that has not started any operations is represented as 0. At time t, when k out of n vehicles are performing assembly operations, the state vector including the information from Table 2 and Table 3 is expressed as follows:

$$s_t = (k,\ table\ of\ vehicles\ ,\ table\ of\ workstations) \tag{1}$$

At this point, the defined s_t is difficult to apply directly to the proposed scheduling algorithm. The reason is that the main issue with the s_t represented in equation (1) is that the dimensions of each table are different, making it difficult for the algorithm to interpret the state vector. Therefore, to address this problem, we extract the features of each state information. The information of common groups shares one convolutional layer and two fully connected layers. The convolutional layer applies filters to the input data to generate feature maps, enabling the extraction of features from high-dimensional data (Uchida et al. 2018). The transformed state vector obtained through feature extraction can be expressed as follows:

$$f_{s_t} = (n, feature\ value\ of\ vehicles\ table, feature\ value\ of\ workstation\ table) \tag{2}$$

The agent examines the system's state through the transformed state vector (f_{s_t}) obtained via the convolutional layers and then takes actions according to the defined priority rules. The defined priority rules are categorized into two types based on their objectives. The first type of priority rules under consideration are used to determine the path for an AMR to take for movement from a current position to the destination. The second type of priority rules under consideration are required for an AMR to choose as the next destination to visit. Each type of priority rules has three rules. By paring two types of priority rules, actions are defined. The defined priority rules used in actions are summarized as follows:

- Priority rules for path selection:
  - Shortest path first: Select the shortest path to the workstation.
  - Shortest waiting time first: Choose the workstation with the least time left to complete the operation
  - Largest number of operations to process first: Prioritize workstations where the maximum number of operations to process can be performed at once.
- Priority rules for operation selection:
  - First-come, first-start: Start operations in the order of vehicle arrival.
  - Longest processing time first: Start operations with the longest processing time.
  - Highest overall job progress first: Start operations based on highest overall job progress.

Actions are denoted by natural numbers. Note that each action corresponds to a pair of path and operation priority rules. Table 4 describes the all actions.

Table 4: Action set consisting of prioritization rule pairing.

Rule numbers applied to each action	Priority rules for path selection	Priority rules for operation selection
Rule no.1	Shortest Path First(SPT)	First-Come, First-Start (FCFS)
Rule no.2	Shortest Waiting Time First(SWTF)	First-Come, First-Start (FCFS)
Rule no.3	Largest number of Operations to Job First (LOJF)	First-Come, First-Start (FCFS)
Rule no.4	Shortest Path First(SPT)	Longest Processing Time First(LPTF)
Rule no.5	Shortest Waiting Time First(SWTF)	Longest Processing Time First(LPTF)
Rule no.6	Largest number of Operations to Job First(LOJF)	Longest Processing Time First(LPTF)
Rule no.7	Shortest Path First(SPT)	Highest overall Job Progress First(HJF)
Rule no.8	Shortest Waiting Time First(SWTF)	Highest overall Job Progress First(HJF)
Rule no.9	Largest number of Operations to Job First(LOJF)	Highest overall Job Progress First(HJF)

Agent applies the selected action to the environment, transitions to the next state, and evaluates the performance of the action selection based on the received reward. Therefore, the reward must be given based only on the point in time when the action is applied. This study considers two evaluation metrics for calculating the reward. The first evaluation metric is the average job completion rate at time t (AJCR_t), which calculates how many operations, on average, the vehicles in the system have completed during a unit time between observation points. It is calculated as follows:

$$AJCR_t = \frac{\sum_1^n P_k}{(A\ unit\ time)*n} \tag{3}$$

The second evaluation metric is the average operating rate of workstations (AORW_t). When a vehicle, whose previous task has been completed, selects the next workstation for the subsequent operation, it must wait at its current location if the buffer space at the chosen workstation is full until it can move to the next position. During this waiting period, the workstation is inoperative as it cannot proceed with any operations. Therefore, the AORW_tconsidering the average inoperative rate of workstations, is calculated as follows:

$$AORW_t = \frac{\sum_1^l w_i}{(A\ unit\ time)*l} \tag{4}$$

The reward r_t at time t, considering $\mathrm{AJCR}_t$and AORW_t, can be expressed as follows:

$$r_t = AJCR_t + AORW_t \tag{5}$$

As the sum of AJCR_t and AORW_t increases, the time to complete all tasks decreases, so it is evaluated as a reward. The DQN algorithm proposed in this study observes the system's state at predefined intervals and selects the action that maximizes the reward, thereby minimizing the makespan, which is the time taken for all operations of the deployed vehicles to be completed.

4.2 Deep Q-Network Algorithm

In this section, the DQN algorithm applied to solve the VPS problem using the defined MDP model is described. The DQN algorithm combines Q-learning, one of the reinforcement learning algorithms, with deep learning. In traditional Q-learning, the values of state-action pairs, denoted as $Q(s, a)$, are stored in a table format and learned over time (Watkins and Dayan 1992) . However, this approach requires significant memory and exploration time as the state and action spaces grow larger. To address this challenge, DQN approximates the Q-function using deep neural networks (Mnih et al. 2015). Additionally, DQN improves learning stability through techniques like experience replay and target Q-Network. The agent stores

experiences, represented by tuples $Q(s, a, r, s')$ in an experience replay buffer (Nair et al. 2015). These tuples consist of the current (s), taken action (a), resulting reward (r), and next state (s'). These experiences are randomly sampled from previously stored experiences to predict approximate Q-values. The target Q-network is updated by copying the weights of the current Q-network. Using the fixed weights of the current Q-network, the target Q-value, and max Q value is computed. DQN minimizes the difference between the current predicted Q-value, the target predicted Q-value computed from the target Q-Network, and the actual value, which is the reward, by considering a loss function. Figure 3 illustrates this training process of DQN.

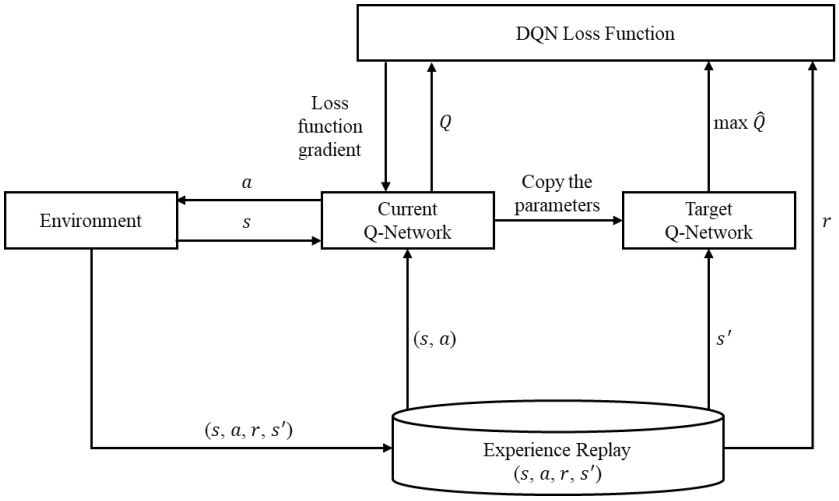

Figure 3: Learning flow diagram of DQN (modified from Nair et al. 2015).

5 NUMERICAL EXPERIMENT AND DISCUSSION

In this section, simulation experiments are conducted to compare the proposed DQN with various rule combinations. The simulation environment used for the experiments and DQN training were implemented with Siemens Plant Simulation (version 23.02).

To evaluate the performance of the proposed DQN, the makespan obtained using DQN was compared with the makespan obtained using each of the 9 predefined priority rule combinations. For this purpose, a small-scale layout design of the matrix system was created using operation, time, position, and precedence information data obtained from industry partners. Figure 4 depicts the layout of the MS implemented as a simulation model.

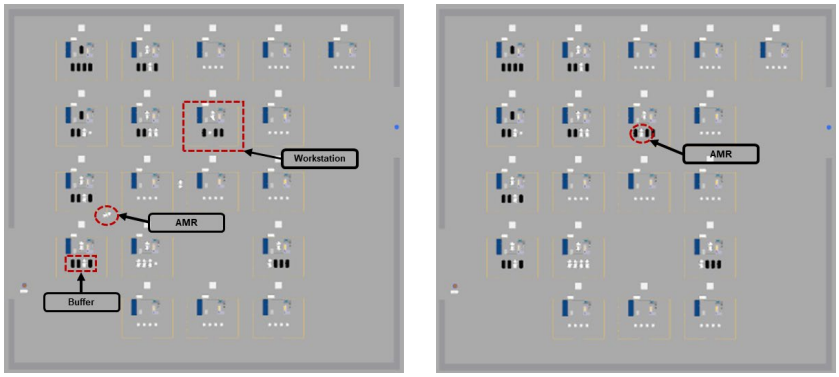

Figure 4: A layout of matrix system used in the training and experiment.

The layout implemented for the experiment consists of 19 workstations. Each workstation has space to accommodate up to 4 AMRs in addition to the workspace and is capable of performing between 4 and 14 operations. The AMRs operate at a speed of 1.5 m/s, transporting products, with one vehicle introduced every 30 seconds. Each product requires a total of 20 operations for assembly. The left image(shows an AMR heading towards the selected workstation for the next operation. The right image illustrates the scenario where the AMR waits in the buffer until the ongoing operation at the arrived workstation is completed.

In this study, the proposed DQN algorithm was trained through 310 simulation experiments. The agent interacts with the simulation environment and selects one of the predefined combination numbers as an action every 1,800 seconds (30 minutes). This 30-minute interval is defined as the unit time interval between two states. The ε-greedy method was used as the action selection strategy, where a random action is chosen if ε is less than the threshold value, otherwise, the action with the maximum Q-value is selected. The initial ε value was set to 1 and decreased at a fixed rate with each iteration to maintain a balance between exploration and exploitation throughout the simulation repetitions. Figure 5 shows the learning curve of the trained DQN algorithm, with the x-axis representing the number of training iterations and the y-axis representing the makespan in seconds. The proposed DQN algorithm achieved a makespan of 26,771 seconds by selecting the action with the maximum Q-value at each moment, starting from the experiment with ε reaching 0 in the 300s range.

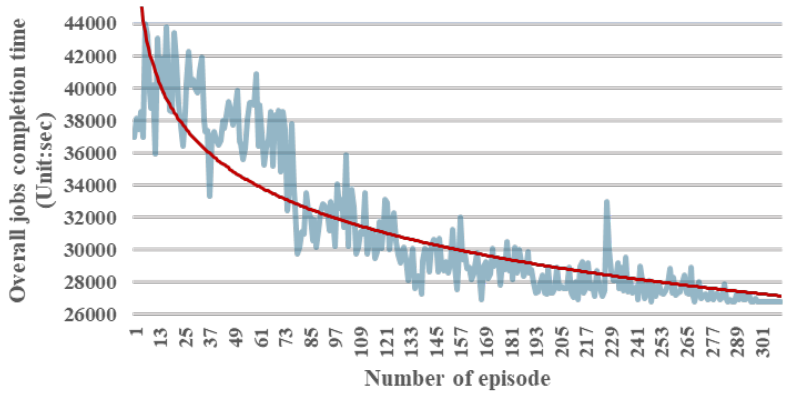

Figure 5: Learning curve for DQN algorithm.

To evaluate the performance of the algorithm, we applied a single rule to each action selection and calculated the average makespan. We repeated the experiment five times with a goal of producing 60 vehicles and computed the average of the results. Figure 6 illustrates the average makespan values obtained for each combination of priority rules. Among the disposable priority rules chosen as the control group, the second rule (Combination of SPT, FCFS) showed the shortest makespan at 27,880 seconds, while the ninth rule (Combination of LOJF, HJF) exhibited the longest makespan at 55,924 seconds. Comparing the makespan achieved by the DQN algorithm with those obtained when using static priority rules alone, the DQN algorithm demonstrated the shortest makespan, indicating its superior performance compared to other priority rules. The AORW (during the entire operation time) of the DQN algorithm and static priority rules over the entire operation time is shown in Figure 7. This was found to be at least 3% to a maximum of 31.4% higher compared to static priority rules combination.

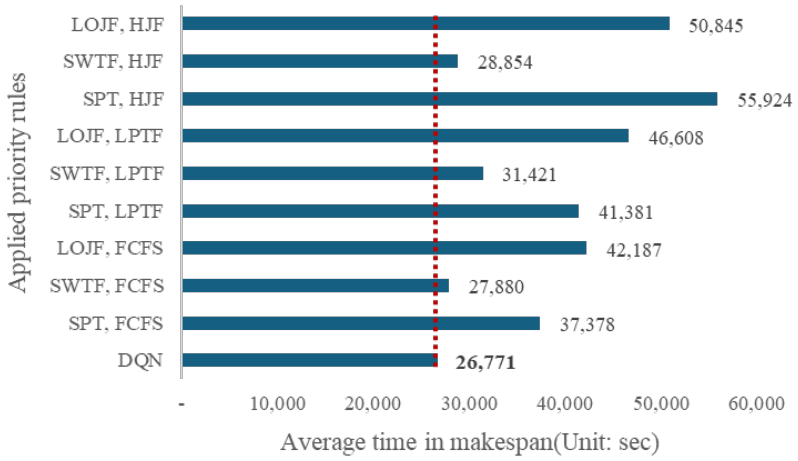

Figure 6: Makespan of running priority rules and DQN algorithm.

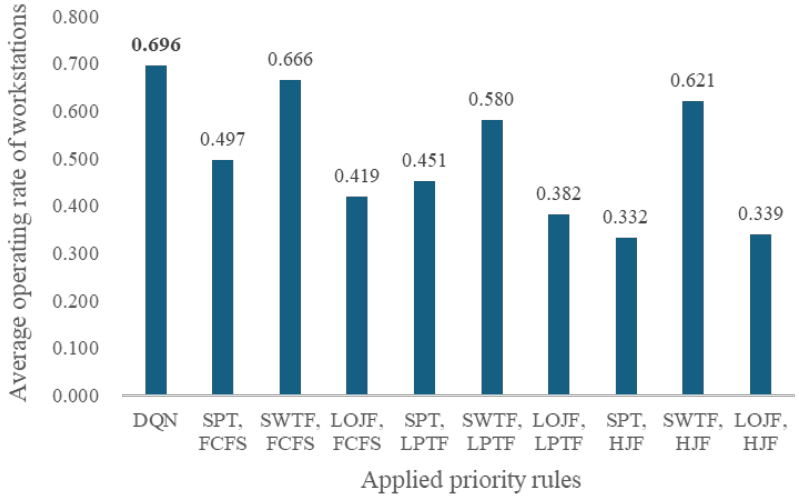

Figure 7: Comparison of AORW_t Across Different Rules During the Entire Operation Time.

The reason why the DQN algorithm achieves a shorter makespan compared to static priority rule combinations is that it dynamically determines the utilization of each combination. The DQN algorithm uses the sum of AORW_t and AORW_t as a reward to minimize the makespan. Consequently, it continuously updates actions to learn the action that maximizes rewards according to the state. The rule combinations selected by the DQN algorithm to maximize rewards are shown in Table 5. Rule no.2 (Combination of SPT, FCFS) and rule no.8 (Combination of SWTF, HJF) are the two best-performing rule combinations among the remaining priority rules, excluding the DQN algorithm.

Table 5: Action selection of the DQN algorithm.

Application times for each action (Unit : Time)	Combinations of Priority Rules
00:00:00 ~ 02:30:00	Rule no.8 (Combination of SWTF, HJF)
02:30:00 ~ 07:26:11	Rule no.2 (Combination of SPT, FCFS)

Table 6 compares the AORW_tover a 2-hour period when applying a single rule versus the DQN algorithm's rule changes. The points where the rules change are indicated in bold. Initially, the DQN algorithm selects rule 8, maintaining a high utilization rate for the first 2 hours and 30 minutes. After this period, it switches to rule no.2, allowing for a flexible response to changes in the workstation status. By selecting the optimal priority rule based on the workstation's status at this point, the DQN algorithm maintains a high overall utilization rate.

Table 6: Difference in AORW_t over time according to applied rules (Unit: Hour).

Applied Rule	0.5 (%)	1 (%)	1.5 (%)	2 (%)	**2.5 (%)**	3 (%)	3.5 (%)	4 (%)	4.5 (%)
DQN	0.35	0.64	0.78	0.79	**0.78**	0.74	0.80	0.78	0.81
Rule no.2	0.35	0.67	0.81	0.76	**0.73**	0.68	0.75	0.79	0.80
Rule no.8	0.35	0.64	0.78	0.79	**0.78**	0.69	0.73	0.72	0.80

These rule changes can maximize overall throughput by applying the most suitable priority rule at each stage of the process. Table 7 shows a comparison of the reduction in WIP over time due to the rule changes by the DQN algorithm. The DQN algorithm begins to quickly reduce WIP from the 4-hour mark, achieving a greater reduction in WIP by the 6-hour mark compared to fixed priority rules. This indicates that the dynamic priority decisions made by the DQN algorithm are more effective during intermediate stages.

Table 7: Difference in WIP over time according to applied rules (Unit: Hour).

Applied Rule	1 (WIP)	2(WIP)	3(WIP)	4(WIP)	5(WIP)	**6(WIP)**	7(WIP)	7.5(WIP)
DQN	60	60	60	58	54	**35**	11	0
Rule no.2	60	60	60	60	59	**42**	15	3
Rule no.8	60	60	60	59	59	**49**	24	8

The common feature of the rules selected by the DQN algorithm is the use of the SWTF rule when selecting workstations. This helps minimize waiting time and increase processing speed. Additionally, when initially selecting task priorities, the DQN algorithm chooses HJF from rule no.8 to prioritize vehicles nearing completion. In other words, the combination of rules in rule no.8 is effective in quickly reducing the initial workload. Subsequently, based on the workstation status, the algorithm switches to rule no.2, applying FCFS to minimize waiting times across all workstations and enabling flexible task processing. Thus, the dynamic priority decisions made by the DQN algorithm can flexibly respond to the variability of the work environment by evaluating the workstation status at the observation point and selecting the optimal rule. This allows for the optimization of the workflow.

6 CONCLUSION

This study proposes a scheduling method based on the DQN algorithm that adapts to dynamic changes to apply optimal priority rules. Makespan, the time to complete all operations until the last vehicle, was used as the operational performance indicator. To apply the DQN algorithm, states, actions, and rewards were defined to suit the problem characteristics. To train and evaluate the scheduling model, a discrete event simulation was used to model and utilize a virtual matrix layout that represents the actual manufacturing environment. The proposed scheduling model demonstrated superior performance compared to fixed priority rules, validating the applicability and effectiveness of DQN-based scheduling. However, this study has several limitations. First, the applied DQN algorithm relies on predefined priority rules and may not fully reflect various production conditions. Although three rules were considered based on decision-making types, the actual production environment may have more diverse rules and complex situations. Second, this study targeted a small-scale matrix production system, so it is necessary to verify the proposed model's effectiveness in large-scale systems. Actual production lines are much larger and more complex than the

simulation model used in this study. Therefore, additional research targeting large-scale systems is needed, and experiments in large-scale environments are required to evaluate the model's scalability and generalization capability. Lastly, further research on real-time integration is needed. In this study, the DQN algorithm was set to select actions every 1,800 seconds (30 minutes). In actual production environments, it is crucial to explore the feasibility of real-time application using systems such as Internet of Things (IoT) devices, which can respond to production situations changing at the scale of minutes or seconds.

REFERENCES

Greschke, P., M. Schönemann, S. Thiede, and C. Herrmann. 2014. "Matrix Structures for High Volumes and Flexibility in Production Systems." *Procedia CIRP*, 17: 160-165.

Holweg, M. and Pil, F. K. 2005. *The Second Century: Reconnecting Customer and Value Chain through Build-to-Order Moving Beyond Mass and Lean in The Auto Industry*. MIT Press Books.

Oh, S. C., J. W. Wells, and J. Arinez. 2022. "Conveyor-Less Urban-Car Assembly Factory with VaaC and Matrix System." *Smart Cities*, 5(3): 947-963.

Kern, W., F. Rusitschka, W. Kopytynski, S. Keckl, and T. Bauernhansl. 2015. "Alternatives to Assembly Line Production in The Automotive Industry". In *Proceedings of the IFPR Conference*, August 2^{th}-5^{th}, Manila, Philippines.

Kim, M. S., S. C. Oh, E. H. Chang, S. Lee, J. W. Wells, J. Arinez et al. 2022a. "A Dynamic Programming-Based Heuristic Algorithm for a Flexible Job Shop Scheduling Problem of a Matrix System in Automotive Industry," In *Proceedings of the CASE Conference*, August 20^{th}-24^{th}, Mexico City, Mexico, 777-782.

Kim, M. S., S. C. Oh, E. H. Chang, J. W. Wells, J. Arinez, and Y. J. Jang. 2022b. "Performance Evaluation of Conveyor-Less Matrix Assembly System Using Simulation and Mathematical Models". In *Proceedings of the ASME Conference*, October 30^{th}- November 3^{th}, Columbus, Ohio, USA, Vol. 86649, V02BT02A043.

Julaiti, J., S. C. Oh, D. Das, and S. Kumara. 2022. "Stochastic parallel machine scheduling using reinforcement learning." *Journal of Advanced Manufacturing and Processing*, 4(4): e10119.

Alatise, M. B. and Hancke, G. P. 2020. "A Review on Challenges of Autonomous Mobile Robot and Sensor Fusion Methods." *IEEE Access*, 8, 39830-39846.

Johnson, S. M. 1954. "Optimal Two-and Three-Stage Production Schedules with Setup Times Included." *Naval Research Logistics Quarterly*, 1(1): 61-68.

Cummings, D. H. and Egbelu, M. P. 1998. "Minimizing Production Flow Time in A Process and Assembly Job Shop." *International Journal of Production Research*, 36(8): 2315-2332.

Gupta, J. N., V. R. Neppalli, and F. Werner. 2001. "Minimizing Total Flow Time in A Two-Machine Flow Shop Problem with Minimum Makespan." *International Journal of Production Economics*, 69(3): 323-338.

Kubzin, M. A., C. N. Potts, and V. A. Strusevich. 2009. "Approximation Results for Flow Shop Scheduling Problems with Machine Availability Constraints." *Computers & Operations Research*, 36(2): 379-390.

Ivanov, D., A. Dolgui, B. Sokolov, F. Werner, and M. Ivanova. 2016. "A Dynamic Model and An Algorithm for Short-Term Supply Chain Scheduling in the Smart Factory Industry 4.0." *International Journal of Production Research*, 54(2): 386-402.

Gonzalez, T. and Sahni, S. 1978. "Flow Shop and Job Shop Schedules: Complexity and Approximation." *Operations Research*, 26(1): 36-52.

Park, Y., S. Kim, and Jun, C. H. 2002. "Mean Value Analysis of Re-Entrant Line with Batch Machines and Multi-Class Jobs." *Computers & Operations Research*, 29(8): 1009-1024.

Carlier, J. and Pinson, É. 1989. "An Algorithm for Solving the Job-Shop Problem." *Management Science*, 35(2): 164-176.

Rifai, A. P., S. Z. M. Dawal, A. Zuhdi, H. Aoyama, and K. Case, 2016. "Reentrant FMS Scheduling in Loop Layout with Consideration of Multi Loading-Unloading Stations and Shortcuts." *The International Journal of Advanced Manufacturing Technology*, 82: 1527-1545.

Liu, R., R. Piplani, and C. Toro. 2022. "Deep reinforcement Learning for Dynamic Scheduling of a Flexible Job Shop." *International Journal of Production Research*, 60(13), 4049-4069.

Jeon, S. W., D. Lee, S. C. Oh, K. T. Park, S. D. Noh, and J. Arinez. 2022. "Design and Implementation of Simulation-Based Scheduling System with Reinforcement Learning for Re-Entrant Production Lines." *Machines*, 10(12): 1169.

Verma, A., S. C. Oh, J. W. Wells, J. Arinez, and S. Kumara. 2022. "Conveyer-Less Matrix Assembly Layout Design to Maximize Labor Productivity and Footprint Usage." In *Proceedings of the ASME 2022 Conference,* October 30^{th}- November 23^{th}, Columbus, Ohio, USA, Vol. 86649, p. V02BT02A020.

Waschneck, B., Reichstaller, A., Belzner, L., Altenmüller, T., Bauernhansl, T., Knapp, A et al. 2018. "Optimization of Global Production Scheduling with Deep Reinforcement Learning." *Procedia Cirp*, 72: 1264-1269.

Shi, D., W. Fan, Y. Xiao, T. Lin, and C. Xing. 2020. "Intelligent Scheduling of Discrete Automated Production Line via Deep Reinforcement Learning." *International Journal of Production Research*, 58(11): 3362-3380.

Hu, L., Z. Liu, W. Hu, Y. Wang, J. Tan, and F. Wu. 2020. "Petri-Net-Based Dynamic Scheduling of Flexible Manufacturing System via Deep Reinforcement Learning with Graph Convolutional Network." *Journal of Manufacturing Systems*, 55: 1-14.

Uchida, K., M. Tanaka, and M. Okutomi. 2018. "Coupled Convolution Layer for Convolutional Neural Network". *Neural Networks*, 105: 197-205.

Watkins, C. J. and Dayan, P. 1992. "Q-learning". *Machine Learning*, 8: 279-292.

Mnih, V., K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare et al. 2015. "Human-Level Control through Deep Reinforcement Learning". *Nature*, 518(7540): 529-533.

Nair, A., P. Srinivasan, S. Blackwell, C. Alcicek, R. Fearon, A. De Maria et al. 2015. "Massively Parallel Methods for Deep Reinforcement Learning". *arXiv preprint arXiv:1507.04296.*

Dann, C., Y. Mansour, M. Mohri, A. Sekhari, and K. Sridharan. 2022. "Guarantees for Epsilon-Greedy Reinforcement Learning with Function Approximation". In *Proceedings of the PMLR 2022 Conferences*, July 17th-23th, Baltimore, Maryland, USA, 4666-4689.

AUTHOR BIOGRAPHIES

WHAN LEE received his Bachelor's degree in Industrial Engineering from Cheongiu University, Republic of Korea. He is currently a Master's degree student at Sungkyunkwan University, Republic of Korea. His major research areas are production planning&control, simulation modeling, digital twin. His email address is leewahn0202@g.skku.edu.

SEOG-CHAN OH received his PhD in Industrial Engineering from Pennsylvania State University in 2006. He currently works as Staff Researcher at General Motors R&D in Warren, Michigan since 2007. His research areas include optimization, simulation, data analytics, AI/ML, and digital twins. He is passionate about solving challenging problems in the areas of smart manufacturing systems, energy and environment, labor productivity, and stochastic and dynamic scheduling and control. His email address is seog-chan.oh@gm.com.

JISOO PARK received her Bachelor's degree in Industrial Management Engineering from Daejin university, Republic of Korea. She is currently a Master's degree student at Sungkyunkwan University, Republic of Korea. Her major research areas are cyber-physical system, digital twin, data analytics, smart manufacturing. Her email address is jisoo7589@g.skku.edu.

CHANGHA LEE received hus Bachelor's degree in Mechanical Engineering from Hanbat University, Republic of Korea. He is currentrly a Master's degree student at Sunkyunkwan University, Republic of Korea. His major research areas are cyber-physical system, digital twin, simulation modeling, simulation-based optimization. His email address is lddl0585@g.skku.edu.

HUA-TZU FAN received his PhD in Mechanical Engineering in 1992 from the University of Michigan. He is currently a staff Researcher at General Motors R&D in Warren, Michigan since 1994. His current research areas include Innovative Painting Technologies for Next Generation Automotive Assembly and machine learning applications in First-time Quality Improvement. He is passionate in developing alternative automotive painting technologies that can simplify the painting process and deliver reductions in carbon emission, energy consumption, facility footprint, and manufacturing cost per unit. His email address is charles.fan@gm.com.

JORGE ARINEZ is a Technical Fellow and Lab Group Manager for Manufacturing Systems and Controls Research at GM Global Research and Development. Dr. Arinez's current work focuses on Smart Manufacturing Systems and real-time, decision support tools that enable GM's plants to meet productivity and quality goals. Dr. Arinez's technical contributions include development of system models and diagnostic methods to identify real-time throughput bottlenecks by use of stochastic/continuous flow models. Dr. Arinez holds a Bachelor of Applied Science in Mechanical Engineering from the University of Toronto, and a Masters and Ph.D. in Mechanical Engineering from the Massachusetts Institute of Technology. His email address is jorge.arinez@gm.com.

SEJIN AN received her Bachelor's degree in Industrial System Engineering from Gyeongsang National University, Republic of Korea. She is currently a Master's degree student at Sungkyunkwan University, Republic of Korea. Her major research areas are cyber-physical system, simulation modeling, digital twin, smart manufacturing. Her email address is whoareyou88@g.skku.edu.

SANG DO NOH received his Ph.D. in mechanical design and production engineering from Seoul National University, Republic of Korea. He currently works as professor in Department of Industrial Engineering at Sungkyunkwan University, Republic of Korea. His major research areas are CAD/CAM/PLM, modeling&simulation of manufacturing system, smart manufacturing, smart factory, cyber-physical system and digital twin. His email address is sdnoh@skku.edu.