Transaction assignment policy in double-deep SBS/RSs by using Deep Q-learning

Transaction assignment policy in double-deep SBS/RSs by using Deep Q-learning

Zhun Xu[1], Liyun Xu[1*], Huan Shao[2], Andrea Matta[2]

[1] School of Mechanical Engineering, Tongji University, Shanghai, China
[2] Department of Mechanical Engineering, Politecnico di Milano, Milan, Italy

[*] Corresponding author's e-mail: Lyxu@tongji.edu.cn

Abstract. The double-deep shuttle-based storage and retrieval system (SBS/RS) has gained more attention due to its higher area utilization. However, the additional relocation operations for blocking loads increase the complexity of transaction assignment (TA) challenges. This paper investigates a real-time TA method for a double-deep SBS/RS using the Deep Q-Network (DQN), where shuttles have the flexibility to travel between tiers freely. Using the commonly applied Nearest Free relocation strategy, the system acts as an agent to learn the optimal policy in real-time. It selects a retrieval task from the transaction pool, pairs it with a storage task to create a dual command cycle, and assigns it to an available idle shuttle for execution in the next cycle. The effectiveness of the proposed DQN method was validated through experimental comparisons with various static TA strategies.

1. Introduction

Automated Storage and Retrieval Systems (AS/RSs) have been widely used in large distribution centers due to their efficiency, reliability, and labor-saving advantages [1]. The shuttle-based storage and retrieval system (SBS/RS) is a relatively new type of AS/RS, where the horizontal and vertical movements for loads are performed by shuttles and lifts, respectively. Multiple shuttles work in parallel, coordinating with lifts to complete storage and retrieval transactions. SBS/RS can be categorized into two types: tier-captive and tier-to-tier. In a tier-captive SBS/RS (c-SBS/RS), the number of shuttles equals the number of rack tiers, allowing each shuttle to move only within its designated tier. The tier-to-tier SBS/RS (t-SBS/RS) was started to be researched in 2018, where the number of shuttles is fewer than the number of rack tiers, enabling them to reach any tier through the use of a shuttle-lift [2]. This not only provides greater flexibility but also reduces the initial investment in shuttles. However, the introduction of the shuttle-lift increases the complexity of system operations, particularly in determining which transaction to assign to an idle shuttle for execution next.

The SBS/RS was initially introduced in the early 2000s, featuring a racking subsystem that was solely single-deep for the following decade. By the middle of the 2010s, systems incorporating double-deep racking began to emerge [3]. Compared to single-deep racking, double-deep racks can mostly achieve a 50% decrease in lane space, allowing for more compact storage [4]. However, during the retrieval process, the target load may be blocked by the stock keeping unit (SKU) positioned near the lanes, which requires relocation operations. The relocation process increases the movement and loading/unloading time of shuttles, and affects the dynamic changes in cargo location distribution. Consequently, transaction assignment (TA) problems in double-deep systems are more complex than those in single-deep systems.

According to the operation mode of the shuttle, it can be divided into a single command cycle (SCC) and a dual command cycle (DCC). The DCC refers to the scenario where the shuttle first stores an inbound SKU without returning to the I/O point, followed immediately by retrieving an outbound

SKU. Compared to the SCC, the DCC mode is more efficient as it reduces the number of unnecessary empty movements during operation processes [5]. Thus, it is a challenge for the system to determine which retrieval task from the transaction pool (TP) to combine with the incoming storage to form a DCC and subsequently assign it to an available idle shuttle.

Deep Reinforcement Learning (DRL) utilizes real-time state data collected from the environment to learn how to make intelligent decisions based on the current and potential future states. Deep Q-Networks (DQN) is one type of DRL algorithm that combines classic Q-learning with deep neural networks, enabling agents to learn optimal action strategies in complex environments with high-dimensional state spaces. Numerous studies have demonstrated that DQN performs well in sequential decision-making tasks in manufacturing systems. Reference [6] introduced an adaptive DQN approach with mixed rules to tackle the challenges of real-time scheduling for Automated Guided Vehicles (AGVs) in flexible shop environments to reduce the makespan and delay ratio. A case study on a real-world flexible shop floor validates the efficiency and feasibility of the proposed methodology, showcasing the potential benefits of integrating DQN for optimizing AGV scheduling.

Reference [7] presented a DQN-based method to control multiple AGVs to address the challenges faced by the distribution and manufacturing industries. The methodology demonstrates superior performance in optimizing AGV movement paths and transfers through simulation experiments.

Reference [8] first used a DQN approach for optimizing transaction selection in a t-SBS/RS. The performance of the DQN approach was compared with traditional heuristic methods like first-come-first-served (FCFS) and shortest processing time (SPT) rules. Results indicate that DQN outperforms these rules, showing significant improvements in reducing the average cycle time per transaction. However, they only focused on the problem of transaction sequencing from shuttles' queues in the single-deep t-SBS/RS under the SCC mode. Our study pays attention to TA from the TP of WMS to idle shuttles in a double-deep t-SBS/RS under the DCC mode.

Therefore, this paper investigates the application of DQN for real-time TA to idle shuttles in a double-deep SBS/RS, considering both shuttle relocation and cross-tier operations under a DCC mode. We explore the optimization effects of DQN on two performance metrics: average flow time and average cycle time, comparing the results with three well used static strategies: Random, FCFS, and SPT. The remaining sections of this paper are organized as follows. Section 2 presents the methodology. Section 3 details the DQN implementation for double-deep t-SBS/RSs. An industrial case study is presented in Section 4. Finally, Section 5 concludes the study and provides a preview of future research directions.

2. Methodology

2.1. System description

The configuration of a double-deep t-SBS/RS is illustrated in Figure 1, as shown in [9]. There are four rows of racks on each side of the lane, with the two rows closest to the lane being the first depth, followed by the second depth. The storage-lift and retrieval-lift are located at the same end of the racks, with each tier of the racks having a buffer area connected to these two lifts for buffering loads. The storage-lift is responsible for transporting SKUs from the inbound conveyor to the buffer area on the target tier, while the retrieval-lift transports SKUs from the buffer area to the outbound conveyor. The shuttle-lift is located at the other end of the racks and is responsible for transferring the shuttle to other tiers. Shuttles can reach SKUs through an extendable mechanism attached to them. When the target SKU is blocked, the shuttle needs to perform a relocation operation. Generally, the relocation process involves the shuttle traveling to the bay where the target load is located, relocating the blocking SKU to an available empty bay at its current tier, and then returning to move the target load to the buffer area. It should be noted that shuttles cannot perform the relocation operation across tiers.

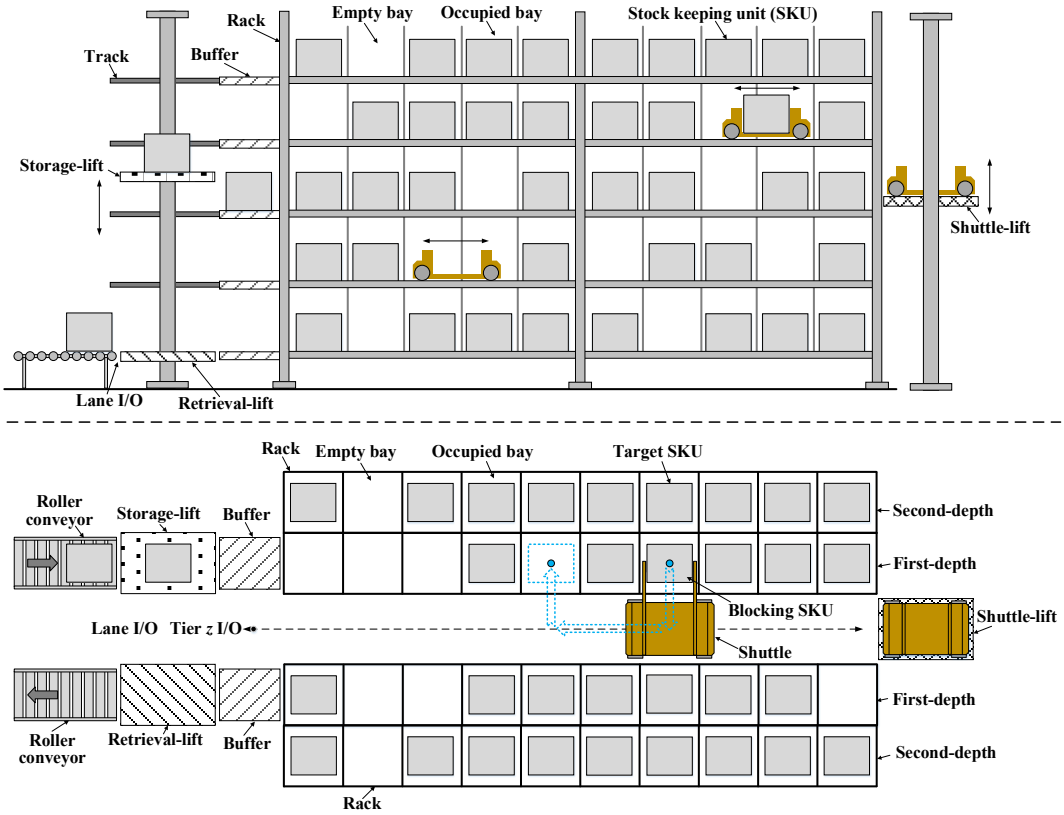

Figure 1. Installation of a double-deep t-SBS/RS.

2.2. Simulation

Table 1. Main notations.

Notation	Unit	Description
T	-	Number of tiers in each rack
B	-	Number of bays in each tier of the rack
S	-	Number of shuttles
N	-	Transaction pool size
v_{S}	m/s	The maximum moving speed of the shuttle
v_{L}	m/s	The maximum speed of the lift
v_m	m/s	The moving speed of the extendable mechanism of the shuttle
a_{S}	$\mathrm{m/s}^2$	The acceleration/deceleration of the shuttle
a_{L}	$\mathrm{m/s}^2$	The acceleration/deceleration of the lift
w	m	The width of a bay (rack unit)
h	m	The height of a bay
l	m	The length of a bay
τ_{ld1}	s	The time of loading/unloading SKUs of the shuttle
τ_{ld2}	s	The time of loading/unloading SKUs of the lift
τ_{ld3}	s	The time of loading/unloading shuttles of the lift
t_c	s	The average time between transactions
C_{avg}	s	The average cycle time of transactions
F_{avg}	s	The average flow time of transactions

Table 1 presents the main notations in this paper. A professional simulation software - Tecnomatix Plant Simulation, was used to build the simulation model. In the simulation, both storage and retrieval transactions arrive at the WMS sequentially, forming queues Q_{S} and Q_{R}. Once any transactions are assigned to an idle shuttle, the same transactions are synchronously replicated into the queues of storage-lift and retrieval-lift. If a transaction requires the shuttle to move across tiers, it is also duplicated into the queue of shuttle-lift. This approach aims to preserve the order of tasks assigned to the shuttle, ensuring consistency in the execution sequence of the same transaction across all service devices. As the inbound conveyor can only transport one SKU at a time to the storage-lift, transactions in the storage queue Q_{S} are required to perform following the FCFS principle. Transactions in the retrieval queue Q_{R} enter the WMS TP based on the FCFS principle. When a shuttle is available, the WMS selects a retrieval transaction from the TP using policies such as FCFS, SPT and DQN. This retrieval is paired with the earliest arriving storage transaction in Q_{S}, forming a DCC allocated to the shuttle for the next execution cycle. The system operation abides by the following assumptions:

- Transaction arrivals follow a Poisson distribution, with equal average arrival rates for storage and retrieval.
- Storage operations adhere to the Nearest Free principle. It means SKUs are stored in the nearest available empty bay to the I/O point, and if multiple options exist, they are stored at the deepest bay.
- Retrieval operations follow the principle of Least Relocation Time First, where the retrieval location assigned to the transaction prioritizes the option with the fewest relocation times.
- Relocation operations follow the Nearest Free principle, which means moving blocking SKUs to the nearest available empty bay on the current tier during retrieval operations. If multiple options exist, priority is given to deeper locations.
- Shuttles and lifts perform transactions following the Point-of-Service-Completion principle, meaning all devices remain at the current location after completing each transaction.

3. DQN Implementation

3.1. DQN

DQN is a reinforcement learning algorithm developed from Q-Learning. In traditional reinforcement learning algorithms, Q-Learning utilizes a Q-table to store a finite set of state-action-value pairs. However, the problem of "dimensional explosion" arises when the dimension of the state space becomes large. To address this issue, DQN was introduced [10]. DQN uses a deep neural network to approximate the Q-function instead of a Q-table, enabling the agent to learn the optimal policy in complex environments. To improve training efficiency and enhance stability, DQN introduces an experience replay mechanism where the agent's experiences in the environment are stored in a replay buffer. Experience replay randomly samples data from the buffer for iterative training to update Q-values, as shown in Equation (1).

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha[r_t + \gamma\, max_a\, Q(S_{t+1}, a)]. \quad (1)$$

where s_t and a_t represent the current state and the selected action at time t, respectively; $\alpha\epsilon[0,1]$ stands for the learning rate; r_t represents the reward received for taking action a_t, and $\gamma\epsilon[0,1]$ is the discount factor for future rewards.

To reduce the volatility of Q-value prediction during training and enhance training stability, DQN employs two separate networks: one is the prediction network $Q(s, a; \theta)$, responsible for controlling the agent and gathering experience. The other is the target network $Q(s, a; \theta^-)$, which calculates the target Q-value y_t using Equation (2). During training, the parameters θ of the prediction network $Q(s, a; \theta)$ are updated via gradient descent to minimize the loss function $L(\theta)$, as computed in Equation (3). After a certain number of updates, the parameters θ of the updated prediction network are copied to the target network $Q(s, a; \theta^-)$ with parameters θ^-.

$$y_t = r_t + \gamma \, max_a \, Q(S_{t+1}, a). \tag{2}$$

$$L(\theta) = [y_t - Q(s_t, a_t)]^2. \tag{3}$$

3.2. Agent

The system triggers decision-making when a shuttle is idle. The system selects a retrieval transaction from the TP, which needs to form a DCC with the earliest arriving storage transaction, allocated to the idle shuttle for its next execution cycle. Therefore, the system is treated as the agent in this study. Retrieval transactions are continuously added to the TP following the FCFS principle, with the pool size fixed at n. When the system triggers decision-making, it conveys the observed current state s_t to the DQN, which returns the action a_t through the prediction network $Q(s, a; \theta)$. The system then obtains r_t and s_{t+1} based on s_t and the returned a_t, and feeds them back to the DQN. The experience (s_t, a_t, r_t, s_{t+1}) from this decision is stored in the experience replay buffer, constituting one training step. The system simulation will operate in a non-episodic, long-run manner until a certain number of training steps are reached.

3.3. State space

The state of the environment for the double-deep SBS/RS is defined as follows, where t represents the moment when the decision-making process is triggered while a shuttle is idle; $k \in \{1,2, \dots, S\}$ denotes the shuttle; $\tilde{k} \in \{1,2, \dots, S\}$ represents the shuttle which triggers the decision at t, and $i \in \{1,2, \dots, N\}$ indicates transactions in the TP.

$s(t)$=(the bay where shuttle $\tilde{k}$ storage the SKU, the number of waiting transactions in the storage-lift queue, the number of waiting transactions in the retrieval-lift queue, the number of waiting transactions in the shuttle-lift queue, the stop tier of the shuttle-lift after the completion of its last transaction, the availability of shuttle k, the stop tier of shuttle k, the depth, tier, and bay of transaction i's target location, the tier difference between task transaction i and $\tilde{k}$, the number of blocking SKUs for transaction i, the total number of bays to be passed for transaction i's relocation operation, the feasibility of shuttle $\tilde{k}$ executing transaction i, and the number of transactions in the TP that are on the same tier as transaction i's target location).

3.4. Action space

In this study, the system assigns a retrieval from the TP to idle shuttles. All retrievals in the TP are available. The action space can be defined as: $A(t)$ = (Retrieval transaction i in the transaction pool at time t), where $i \in \{1,2, \dots, \mathrm{N}\}$. The selected retrieval, along with the earliest arrived storage, forms a DCC for the next execution by the shuttle. If a feasible retrieval is selected, it is canceled, and removed from the TP, and a new retrieval is replenished from the retrieval task queue following the FCFS principle.

3.5. Reward function

The design of an appropriate reward function is crucial for training stability and performance for DQN. In this study, our main objective is to minimize the average cycle time. The cycle time refers to the total time a transaction spends in the system from arrival to completion, encompassing all waiting times within the system [8]. The flow time represents the time a transaction takes from the initiation of execution by the shuttle to completion, excluding the waiting time in the task queue. To reduce the state space complexity, we focus on flow time in the reward design, excluding arrival times and task indices. In addition, to enhance the sensitivity of the reward to actions, we use an exponential form. To scale the reward appropriately, accelerate the learning process of the agent, and balance the exploration-exploitation trade-off, we multiply the reward by a coefficient a, constraining the reward to the range (0, a]. We set a =500 and b=-0.05 in this paper.

$$R_t = a * \exp\{b * [flowtime(a_t) - \min(flowtime)]\}. \tag{4}$$

4. Case study

4.1. Case description

A small-scale double-deep SBS/RS with $T = 10$, $B = 20$, was used to verify the proposed approach. The total number of bays (C) is calculated as $T * B * 2 * 2 = 800$. The system's filling degree (F) is 0.7, resulting in a total of 560 stored locations. The number of cargo types stored in the system is 186. The number of shuttles $S = 2$. The TP size $N = 5$, and t_c is set as 18 to achieve a shuttle utilization rate of around 80%. The remaining parameters of the system are detailed in Table 2.

Table 2. System parameters.

Notation	Value	Notation	Value	Notation	Value
v_S	4	a_L	1.5	τ_{ld1}	6
v_L	2	w	0.85	τ_{ld2}	2
v_m	0.5	h	0.75	τ_{ld3}	7
a_S	1	l	0.75		

4.2. Parameter calibration

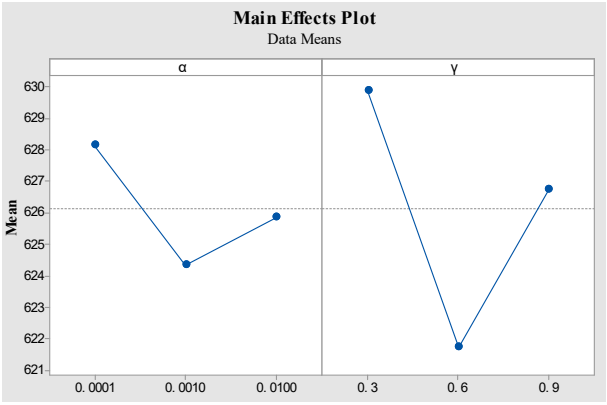

Figure 2. Main effects plot for DQN.

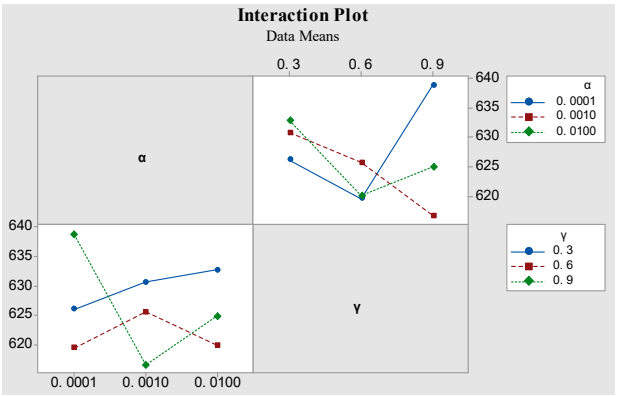

Figure 3. Interaction plot for DQN.

The DQN algorithm was coded in Python language under the Pytorch framework and integrated with Plant Simulation software in real-time via the COM interface. The performance of the DQN algorithm is significantly influenced by its parameters, particularly the learning rate α and the discount factor γ. To calibrate these parameters, we utilized the Design of Experiment method. Considering three levels of α=0.01, 0.001, 0.0001 and γ=0.3, 0.6, 0.9, the objective was to minimize the average cycle time. Each experiment was run for 500,000 steps, with the exploration phase constituting 70% of the total steps. During the initial 350,000 steps, the exploration rate ε linearly decayed from 1 to a minimum value of

0.1. The training batch size was set to 128. We observed that the simulation was halted after running for 200 days, and the average C_{avg} of the final 30 days were considered. Each run repeats three times and use the mean as the metric. The main effects and interaction plots of the experiments are presented as Figure 2 and Figure 3.

From the main effects plot, it is evident that the performance of DQN is notably superior at γ=0.6 compared to γ=0.3 and 0.9; hence, γ=0.6 was chosen. Furthermore, based on the interaction plot, when γ=0.6, the performance at α=0.0001 outperformed that at α=0.01. Therefore, the final parameter settings were determined as α=0.0001 and γ=0.6.

4.3. Comparison experiment

We utilized the optimal parameter settings, α=0.0001 and γ=0.6, along with the remaining parameters mentioned in Section 5.2, to conduct experiments on the case study generated in Section 5.1. The training consisted of 500,000 steps, repeated five times for each experiment. Subsequently, the mean and 95% confidence interval of the final 30 days of each run were calculated. To assess the performance of the DQN, it was compared with three commonly used static policies - Random, FCFS, and SPT. In the comparison, Random denotes a policy where a task is randomly selected from the TP, FCFS indicates selecting the task that arrived first in the TP, and SPT involves selecting the task with the shortest flow time in the TP based on the position of the shuttle that triggered the decision. We maintained consistency in the random number seeds used to simulate the arrival of transactions. The static policies were run for the same duration as the DQN in the simulation, and the averages of the final 30 days were used for comparison. The experimental results are shown in Table 3.

As shown in Table 3, it is evident that the DQN outperforms the three static policies in terms of C_{avg} and F_{avg}, showing particularly significant advantages over the Random and FCFS rules. In terms of F_{avg}, DQN shows optimizations of 9.74%, 10.01%, and 3.74% compared to the Random, FCFS, and SPT policies, respectively. As for C_{avg}, the optimizations are notably higher at 69.65%, 73.35%, and 24.24% when compared to the same set of policies. This indicates that the improvement achieved by DQN on C_{avg} is significantly more pronounced than on F_{avg}.

Table 3. Experiment results.

No.	Policy	Metric		
		C_{avg}	F_{avg}	W_{avg}
1	Random	2015.32	80.89	967.25
2	FCFS	2295.27	81.13	1107.08
3	SPT	807.49	75.85	365.82
4	DQN	611.74±4.35	73.01±0.02	269.37±2.18

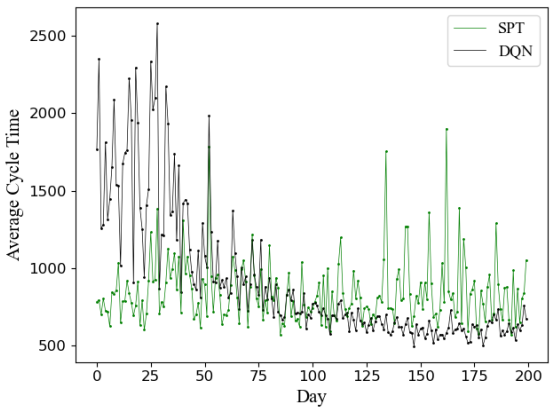

Figure 4. Comparison results for average cycle time.

Figure 4 depicts the variation of the C_{avg} over days for DQN and SPT. From Figure 4, it is evident that SPT demonstrates consistent fluctuations within a defined range for C_{avg}. Conversely, DQN exhibits pronounced oscillations during the initial exploration phase; however, as the training steps progress, the C_{avg} gradually stabilizes with minor fluctuations. Compared to SPT, DQN shows both smaller values and reduced fluctuations in C_{avg} during the last 50 days, indicating improved performance and better stability.

5. Conclusions

This study investigates the TA problem using DQN in a double-deep SBS/RS in which shuttles are capable of freely moving between tiers. Unlike single-deep SBS/RSs, shuttles in this configuration need to perform additional relocation operations for blocking SKUs, improving the complexity of operation process and making transaction assignment to idle shuttles more challenging for minimizing the average cycle time. In this context, the system is modeled as an agent, employing DQN to address the TA problem. Operating under a DCC mode, the system selects a retrieval and the earliest arrival storage from the TP, which is then allocated to the idle shuttle as its next cycle to execute. Based on a commonly used relocation strategy – the Nearest Free, the system explores utilizing DQN to learn optimal TA policies. Carefully design are made for defining the state space, action space, and reward function of the DQN algorithm. Through parameter tuning, the performance of the proposed approach is evaluated against classical TA policies on a small-scale case. Experimental results demonstrate that DQN outperforms the static policies in terms of average cycle time.

In the future study, the impact of DQN on TA in multi-deep SBS/RSs under various relocation strategies can be investigated. Additionally, exploring the performance of DQN under different system configurations and equipment operating properties may be a focus. Moreover, studying the learning of optimal relocation policies for different scale cases using DQN is also a promising direction for future study.

Acknowledgment

This work was supported by the Shanghai Science and Technology Innovation Action Plan (21DZ1209700), the National Natural Science Foundation of China (51975417), and the China Scholarship Council (20230620257).

References

[1] Z. Xu, L. Xu, X. Ling, and B. Zhang, “Data-driven hierarchical learning and real-time decision-making of equipment scheduling and location assignment in automatic high-density storage systems,” International Journal of Production Research, vol. 61, no. 21, pp. 7333-7352, 2022.
[2] Y. Ha and J. Chae, “Free balancing for a shuttle-based storage and retrieval system,” Simulation Modelling Practice and Theory, vol. 82, pp. 12-31, 2018.
[3] N. Kosanić, J. Marolt, N. Zrnić, and T. Lerher, “Travel time model for multiple-deep shuttle-based storage and retrieval systems,” International Journal of Production Research, pp. 1-34, 2023.
[4] T. Lerher, “Travel time model for double-deep shuttle-based storage and retrieval systems,” International Journal of Production Research, vol. 54, no. 9, pp. 2519-2540, 2015.
[5] J. Marolt, N. Kosanić, and T. Lerher, “Relocation and storage assignment strategy evaluation in a multiple-deep tier captive automated vehicle storage and retrieval system with undetermined retrieval sequence,” The International Journal of Advanced Manufacturing Technology, vol. 118, no. 9-10, pp. 3403-3420, 2021.
[6] H. Hu, X. Jia, Q. He, S. Fu, and K. Liu, “Deep reinforcement learning based AGVs real-time scheduling with mixed rule for flexible shop floor in industry 4.0,” Computers & Industrial Engineering, vol. 149, 2020.
[7] K. Takahashi and S. Tomah, “Online optimization of AGV transport systems using deep reinforcement learning,” Bulletin of Networking, Computing, Systems, and Software, 2020.
[8] B. Arslan and B. Y. Ekren, “Transaction selection policy in tier-to-tier SBSRS by using Deep Q-Learning,” International Journal of Production Research, pp. 1-14, 2022.

[9] D. Li, J. S. Smith, and Y. Li, “Design and Control of Shuttle-Based Storage and Retrieval Systems Using a Simulation Approach,” presented at the 2022 Winter Simulation Conference (WSC), 2022.
[10] V. Mnih et al., “Human-level control through deep reinforcement learning,” Nature, vol. 518, no. 7540, pp. 529-33, Feb 26 2015.