DOI: 10.2195/lj_proc_mitterer_en_2025_01

A Deep Q-learning Approach for Bin Relocation in Robotic Compact Storage and Retrieval Systems

Katharina Mitterer,
Christophe Senger,
Timo Lehmann

Institut für Fördertechnik und Logistiksysteme (IFL)
Karlsruher Institut für Technologie (KIT)

Robotic compact storage and retrieval systems (RCS/RS) offer space-efficient storage by stacking bins densely and using robots to retrieve them via a grid-based system. While existing operating strategies give fix guidelines on how to store and relocate blocking bins, the literature lacks learning-based strategies. This work closes that gap by applying deep reinforcement learning to optimize bin retrieval and relocation with respect to cycle time. A Deep Q-Learning agent, trained using Double-DQN with prioritized experience replay in a simulated RCS/RS, is evaluated across diverse scenarios. Results show performance gains regarding the cycle time of up to 36.98% over existing operating strategies. These findings demonstrate the potential of reinforcement learning for relocation decisions and suggest promising transferability to real-world systems.

[Keywords: RCS/RS, Relocation, Cycle Time, Reinforcement Learning, Performance Estimation]

1 Introduction

The demand for efficient and innovative storage and order-picking systems is rising steadily. E-commerce revenues are projected to grow by 39% from 2023 to 2027, exceeding eight trillion USD [1]. At the same time, consumers expect both fast and sustainable delivery, increasing the pressure on logistics and supply chains [2]. Warehouses are a major contributor to operational costs, particularly due to their energy consumption. In the U.S., warehouses accounted for 18% of all commercial floor space and 8% of commercial energy use in 2018, with heating and lighting being the largest contributors [3]. As competition grows, companies are seeking ways to reduce costs while improving efficiency. Robotic compact storage and retrieval systems (RCS/RS), such as AutoStore, offer promising solutions. RCS/RS use robots to retrieve bins stacked densely in a grid, achieving high storage density and energy efficiency. To access a target bin, robots must relocate the bins stacked above it. Since relocation is time-intensive, improved control strategies offer great potential for cycle-time reduction. Cycle-time and energy efficiency are linked to each other with the common assumption that a low cycle-time also leads to low energy consumption. However, the application of learning-based methods, to control such logistical systems, remains underexplored.

RCS/RS exhibit dynamic storage configurations, a complex and high-dimensional state representation, and an extensive action space with numerous control options. Learning-based methods such as Deep Q-Learning (DQL), a technique from reinforcement learning (RL), are particularly well-suited for such environments. They can identify complex relationships through interaction with the system that are difficult to model explicitly. Currently, the throughput efficiency (and the linked cycle-time efficiency) of control strategies in RCS/RS systems remains largely unquantified. Moreover, existing RL research in warehouse automation has primarily focused on routing, collision avoidance, and task assignment, while the optimization of bin relocation has received little attention.

This paper addresses this gap by investigating whether DQL can reduce cycle time (and therefore augment throughput) in RCS/RS systems, particularly under varying storage conditions, access patterns, and warehouse dimensions. To this end, the study pursues the following main objectives:

- Develop a simulation-based model environment that accurately replicates RCS/RS dynamics and enables controlled experimentation.
- Implement and train a DQL agent to optimize bin relocation decisions in a defined base scenario, with the aim of minimizing cycle time.

- Evaluate the generalization of the learned strategy across alternative warehouse configurations with varying storage densities, retrieval patterns, and system sizes.
- Quantitatively compare the cycle time of the DQL agent to existing operating strategies, highlighting the improvement potential of learning-based approaches.

The overarching goal is to demonstrate the potential of DQL for intelligent bin relocation in RCS/RS and to lay a solid foundation for future learning-based warehouse control strategies.

2 Related Work

A systematic literature search was conducted using the *Scopus* database (May 2025) to identify relevant work at the intersection of automated warehouse systems and RL. The search combined terms related to RCS/RS and AutoStore-type systems with key RL concepts, including DQL and the Block Relocation Problem (BRP). The objective was to capture existing research and identify gaps in applying DQL to RCS/RS,. Relevance criteria included:

- focus on learning-based methods, particularly DQL,
- application to automated warehouse or AutoStore-like systems,
- investigation of dynamic bin relocation or comparable processes,
- provision of methodological details and validation.

Of the 33 works, most addressed path planning, routing, or multi-robot coordination, with a few on task allocation and battery management. Only two publications met all criteria, both addressing learning-based dynamic bin relocation in RCS/RS,:
[4] propose a Q-Learning–based bin relocation approach for a compact robotic storage system with a ceiling-mounted rail network, aiming to minimize relocation distance. The method uses a distance-based negative reward and a simplified state description of neighboring stacks. However, the state space is insufficiently specified, limiting generalizability to other layouts. Moreover, robot movement is restricted to fixed tracks with dedicated lane-changing robots, contrasting with the more flexible horizontal motion in AutoStore, making DQL promising for the latter's higher-dimensional and continuous state space.
[5] review the Block Relocation Problem (BRP) in AutoStore-like systems, presenting exact, heuristic, and machine learning–based approaches (Table 1). They emphasize that efficient container positioning is critical for overall performance and identify relocation optimization as a promising application area for learning-based methods.

From the relocation-focused studies:

- [6] address the Container Pre-Marshalling Problem (CPMP) with Deep Learning Heuristic Tree Search, trained on near-optimal solutions. While conceptually related, CPMP assumes deterministic, known container moves, unlike the stochastic, continuous operations in AutoStore.
- [7] use ML to improve upper bound estimation for classical search algorithms but do not apply RL directly.
- [8] optimize vehicle relocation in urban mobility systems, which differs structurally from warehouse automation despite using RL.

Few studies target learning-based bin relocation for RCS/RS. Existing works either lack generalization ([4]), focus on surveys without implementation ([5]), or address related but structurally different problems. This work addresses this gap by developing and evaluating a Double Deep Q-Network (DDQN) agent for cycle time-efficient bin relocation in RCS/RS, assessing its generalization to varying storage densities, access patterns, and system sizes.

3 Theoretical Background

This chapter provides the theoretical background necessary for understanding the paper. It introduces RCS/RSs with AutoStore, founded in 1996 and based on the goods-to-person principle, serving as a representative example. Autonomous robots retrieve bins from a densely packed grid and deliver them directly to picking stations. The system has been adopted by major companies, such as Puma, which operates a 60,400 m^2 AutoStore warehouse in Indiana [13]. In contrast to conventional shelf-based systems, AutoStore stores bins vertically stacked without gaps, maximizing storage density. Robots equipped with vertical lift arms access bins from above, and to retrieve deeper bins, all bins above must first be temporarily relocated [14].

Understanding the structure and operation of RCS/RS, as well as its abstraction into a simulation model, is essential for developing and evaluating advanced control strategies. To this end, Section 3.1 describes the core components of RCS/RS and explains how they are represented in the simulation model. Section 3.2 outlines the system's functionality and control strategies, followed by Section 3.3, which introduces the heuristic baselines implemented for comparison. Section 3.4 details the relocation logic governing bin movements in the absence of learning-based control. Finally, Section 3.5 presents the verification and validation of the sim-

Table 1: ML-based approaches for relocation and routing in RCS/RS, adapted from [5].

Reference	Relocation	Routing	Algorithm	Advantage
[6]	✓		Deep Learning Heuristic Tree Search	Reduces relocations significantly.
[7]	✓		ML-driven Upper Bounds	Improves node selection, reduces relocations.
[8]	✓		Deep Deterministic Policy Gradient	Fewer relocations, reduced congestion.
[9]		✓	Deep Reinforcement Learning Framework	Reduces travel distance.
[10]		✓	Adaptive Large Neighborhood Search	Shorter travel distance, improved robustness.
[11]		✓	BERT-based Deep RL	Faster, more accurate pathfinding, reduced travel.
[12]		✓	Multi-Armed Bandits Algorithm	Avoids path conflicts, reduces travel time.

ulation model, ensuring its reliability as the foundation for subsequent experiments.

3.1 Core Components of RCS/RS

Figure 1 shows the schematic structure of an RCS/RS warehouse with 125 storage locations, one robot and one picking station, also referred to as an I/O point (input/output). In practice, however, RCS/RSs usually have significantly more robots and multiple picking stations.

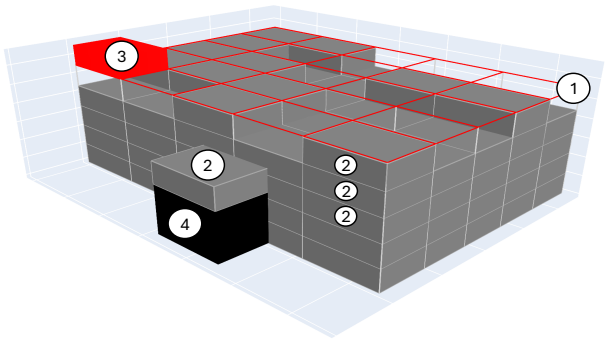

Figure 1: Schematic representation of an RCS/RS consisting of the Grid (1), Bins (2), Robots (3), and I/O points (4). Source: own visualisation from the simulation environment.

RCS/RS consists of five main components, which are numbered in Figure 1. Unless otherwise indicated, the descriptions of the system components are based on information from the official AutoStore website [13]. In our work, these components were abstracted into a discrete-event simulation model, focusing on processes relevant to bin relocation. The implementation of each component is as follows:

- **1 Grid:** Represented as a three-dimensional array of stacks, each position holding a vertical sequence of bins. Each bin in a stack is characterized by an ID, height, and ABC category. The grid size in our model varies from a $5\times5\times5$ system to a $20\times20\times10$ with a filling level from 60% to 93.6%.
- **2 Bins:** Modeled as individual items, each containing one unique SKU. No partial picks or replenishments are considered. Retrieval probabilities follow different distributions.
- **3 Robot:** A single robot moves along the top layer of the grid in the X- and Y-directions using Manhattan routing. Both horizontal and vertical motions are time-based with acceleration and lifting dynamics: horizontal speed $3.1\,\mathrm{m/s}$, acceleration $0.8\,\mathrm{m/s^2}$, and lift/lower speed $1.6\,\mathrm{m/s}$. The robot executes tasks autonomously without downtime or charging cycles.
- **4 I/O Port:** Implemented as a single, bottom-centered input/output location for all order processing operations. Each order is assigned to this port.
- **5 Controller and WES:** Modeled as a simplified decision layer that assigns retrieval tasks, manages bin locations, and schedules robot actions. Order generation is continuous with a rolling horizon of ten known upcoming requests.

In summary, the simulation cycle abstracts RCS/RS operations into a sequence of order arrival, bin retrieval, and temporary relocations. This representation captures the dynamic interaction of the core components while remaining sufficiently tractable for analyzing alternative control strategies.

3.2 Functionality and Control Strategies of RCS/RS

The following description of RCS/RS's operational logic is mainly based on Meller [14]. The system coordinates all components to efficiently handle storage and retrieval requests. Upon order arrival, the controller assigns one or more robots to retrieve the target bin. If the bin is buried within a stack, all bins above it must first be removed and temporarily stored in predefined free locations called "holes." Once accessible, the target bin is transferred to an intermediate buffer and subsequently delivered to an I/O port. The previously removed bins are returned to their original stack in reverse order, shifted one level lower, which creates a new hole at the top. Transport to the I/O port is often performed by a different robot to increase throughput. After picking, bins are not returned to their original location but are instead stored in any available top-level hole. This mechanism results in *natural slotting*, where frequently ac-

cessed bins remain closer to the surface while less frequently used bins drift deeper into the grid. Inventory management is handled through Stock Keeping Units (SKUs), typically assigned one per bin. A Warehouse Execution System (WES) oversees SKU assignment, ensures bin availability for order fulfillment, and communicates with the RCS/RS controller. AutoStore in particular also supports *dual-command* cycles, where a robot combines retrieval and storage within a single trip to minimize idle travel time [15]. While basic processes such as bin relocation are documented in the literature, most decision-making rules remain proprietary. To date, no evidence exists of RL in the core system, although AI-driven add-ons like the CarouselAI™ station have recently been introduced.

Within the simulation model, these operational rules are simplified to emphasize bin handling and relocation. Orders arrive continuously at a single I/O port, and robot actions are coordinated through either heuristic or learning-based decision policies. The rule-based heuristics serve as the baseline against which the RL agent can be systematically evaluated.

3.3 Reference Operating Strategies

To establish meaningful baselines for later evaluation, two rule-based operating strategies are implemented in the simulation:

- AS_M**:** Following [14], bins are randomly stored in upper grid layers, prioritizing stacks with the lowest fill height.
- AS_{NN}**:** Nearest Neighbor (NN) strategy, placing bins close (in travel time) to the next retrieval target, in line with the RCS/RS-NN approach from [16].

3.4 Relocation Logic

Bin relocations, including temporary placement of the target bin, follow the NN principle, whereby stacks are chosen to minimize the expected travel time from the relocation site to the next retrieval task. This rule-based relocation mechanism represents the operational baseline against which the performance of the RL strategy will be benchmarked.

3.5 Simulation Model Verification and Validation

For this work, no real RCS/RS data was available. The simulation model was therefore verified and validated using a combination of established methods as proposed by [17, S. 95ff]. The AS_{NN} strategy was used as reference, since both baseline strategies share the same model. Verification confirmed structural correctness through animation runs, fixed-value tests, and boundary value checks. For extreme fill levels, results behaved as expected: at 0.8% fill level, the average cycle time was $10.91\,\mathrm{s}$, while at the maximum valid 93.6% fill level, it increased to $19.64\,\mathrm{s}$. Monitoring over $1{,}000$ dual-command cycles revealed an initial transient phase of about 350 cycles, after which performance stabilized. A conservative warm-up of $1{,}000$ cycles was applied in all experiments. To assess robustness, 30 replications were performed for both identical and random initial states. Mean cycle times were $17.29\,\mathrm{s}$ and $17.88\,\mathrm{s}$, respectively, with overlapping 95% confidence intervals in both cases. Minor significant deviations in a few replications indicate some sensitivity to initial bin layouts, but overall results were consistent and stable. The model reliably reproduces the expected operational behavior and provides a sound basis for the subsequent evaluation of the DDQN approach.

4 Method

This chapter describes the methodological framework for developing and evaluating a DQL approach in the context of an RCS/RS. The approach builds directly on the assumptions, model simplifications, and heuristic strategies introduced in Section 3, using the validated simulation environment as the basis for training and testing RL agents.

The chapter is structured as follows. Section 4.1 formalizes the RCS/RS control task as a RL problem by defining the agent, environment, state space, action space, and reward function. Section 4.2 introduces the proposed Double Deep Q-Network (AS_{DDQN}), which extends the classical DQL framework with stability-enhancing mechanisms such as prioritized experience replay. Section 4.3 discusses the selection of suitable hyperparameters and the network architecture.

4.1 Reinforcement Learning Problem Formulation

This section formalizes the RCS/RS as a RL problem for developing a DQL approach aimed at reducing energy consumption by minimizing average dual command cycle time. The agent, environment, state space, action space, and reward function are defined, enabling the agent to learn effective bin relocation strategies through simulation.

4.1.1 Agent and Environment

RL is a machine learning paradigm in which an agent learns to make sequential decisions by interacting with an environment to maximize cumulative reward over time [18]. In our formulation, the **agent** is the robot responsible for storage and retrieval, while the **environment** is the validated simulation model (see Section 3).

In classical Q-Learning, the goal is to approximate the optimal action-value function $q^*(s, a)$, which represents the expected return of taking action a in state s and following the optimal policy thereafter. However, due to the very large state space of RCS/RS (see Section 4.1.2), a tabular representation is infeasible. Instead, DQL [19] employs a neu-

ral network $Q(s, a; \theta)$ to approximate Q-values, enabling generalization across high-dimensional states. To stabilize training, two standard techniques are employed:

- **Experience Replay:** Transitions are stored in a replay buffer and sampled randomly to reduce correlation in updates.
- **Target Network:** A periodically updated target network is used to compute training targets, preventing feedback instabilities.

The agent interacts with the environment by observing its current state, selecting actions, and receiving a reward signal that reflects efficiency. Storage follows the AS_{NN} heuristic for non-agent-driven placements, while the DQL agent specifically learns strategies for relocating blocking bins.

4.1.2 State Space and Features

The state space $\mathcal{S}$ comprises all possible warehouse states. Each state $s \in \mathcal{S}$ is represented numerically to capture features relevant for decision-making. Features are divided into:

- **Stack-unspecific (SU):** Global features describing the robot and the currently handled bin.
- **Stack-specific (SS):** Local features for each stack, e.g., number of free slots.

Most features are normalized to $[0, 1]$, ensuring scalability across different system sizes. The SU set includes relative positions of the target bin and robot, distances to key points (e.g., I/O), retrieval probabilities, weighted height indices, binary accessibility indicators, and priority values. The SS set includes relative stack distances, heights, space availability, retrieval-demand features, identifiers for stack roles, waiting-bin indicators, and neighbor information.

For a system of size $L \times B \times H$ with K bin categories, the estimated state space size grows exponentially (e.g., $\sim 7.9 \cdot 10^{25}$ states for $L = B = H = 5$ and $K = 3$), which makes tabular Q-Learning impractical. Neural approximation of the Q-function via DQL is therefore necessary.

4.1.3 Actions and Action Space

The action space $\mathcal{A}$ consists of all placement decisions available to the agent. Actions are defined at the level of selecting a destination stack for a bin, not low-level robot movements. The agent makes decisions only when blocking bins must be relocated or when processing previously displaced bins.

Two modeling options exist: restrict actions to valid moves or allow invalid actions and penalize them. This work adopts the latter, letting the agent learn to avoid invalid decisions. The number of possible actions equals the number of stacks (25 for a 5×5 grid, 100 for a 10×10 grid). Invalid actions include placing bins on full stacks, moving already-blocked bins, or returning bins to temporary stacks. They are not executed but incur a fixed negative reward, ensuring that the replay buffer contains examples of both good and bad decisions.

4.1.4 Reward Function

The RL objective is to minimize energy consumption by reducing dual-command cycle time. Each agent action causes physical movement, with costs for travel time, starts, and direction changes. While short moves may appear efficient, they can create future blockages. The reward function therefore penalizes each relocation proportionally to its time cost:

$$r_{\text{valid}} = -t_{\text{um}}, \quad r'_{\text{valid}} = -\left(t_{\text{um}} + \frac{t_{\text{zwl}}}{n_{\text{wl}}}\right)$$

where t_{um} is relocation time, t_{zwl} is the waiting-list storage time, and n_{wl} the number of bins in the list.

Invalid actions are penalized by a constant $r_{\text{invalid}} = -10$, exceeding the worst-case relocation time ($\hat{t}_{\text{um}}^{\max} \approx 6.59\,\text{s}$ in the base scenario), thereby discouraging infeasible choices.

4.1.5 Episode Definition

Episode length affects whether the agent optimizes short- or long-term performance. To encourage strategies that reduce cycle time over many retrievals rather than a single relocation, episodes are defined as 5,000 cycles with continuous value updates during training.

4.2 Deep Q-Learning Approach

The RCS/RS system is a dynamic, high-dimensional control problem, where bin relocation decisions have both immediate and long-term consequences. Classical rule-based heuristics (AS_M, AS_{NN}) capture only static placement logic. By contrast, DQL enables adaptive, policy-based control through simulation-based interaction, learning strategies that balance immediate relocation efficiency with long-term benefits such as energy and cycle time reduction and improved bin accessibility. Moreover, extensions like Double DQN [20] mitigate overestimation bias, further improving stability. These characteristics make DQL a particularly suitable approach for optimizing RCS/RS control strategies. Building on the modeled RL environment, a (DDQN) [20] is therefore used to approximate Q-values, following the DQN guidelines in [19]. Stability is enhanced with prioritized experience replay [21], modified to prioritize transitions by low sampling frequency rather than TD-error, ensuring balanced sampling and reducing the influence of stale experiences.

The network employs fully connected shared layers to generalize across warehouses of different sizes while limiting parameters. For each stack, stack-unspecific and stack-specific features (see Section 4.1.2) are concatenated into a single input vector, enabling the model to learn generalizable patterns (e.g., placing a bin on a full stack is always invalid).

Key training components:

- **Dropout** and **L2 regularization** to prevent overfitting (values selected during hyperparameter tuning).
- **Activation functions**: evaluated among *swish*, *tanh*, *leaky ReLU*, and *ReLU*.
- **Optimization**: Adam optimizer with gradient clipping (1.0) to prevent unstable updates.
- **Loss function**: modified Huber loss, masking all Q-values except the executed action to update only relevant weights.
- **Epoch-like training**: training frequency increases with episode progress (1/3/10 updates per cycle for early/mid/late phases), reusing the same mini-batch within a cycle.
- **Replay buffer**: minimum of 2,000 stored experiences before updates; prioritized sampling as described above.
- **Network updates**: online network updated every 5 cycles; target network every 20 cycles.

4.3 Hyperparameter & Network Architecture Selection

Hyperparameters were tuned in the base scenario using Optuna [22], with 30 trials evaluated on average dual command cycle time and number of invalid actions over 1,000 evaluation cycles. The found configuration is displayed in Table 2.

Table 2: Performant hyperparameter configuration found with Optuna [22].

Hyperparameter	Value
Learning rate	0.005993
Training duration*	10 000
Discount factor γ	0.9687
Replay buffer size	30 000
Batch size	64
Epsilon decay	0.985
Network architecture	[64, 32]
Activation function	*tanh*
Dropout rate	0.030
L_2 regularization λ	5×10^{-6}

* Measured in number of dual command cycles.

5 Experiments & Results

This section presents the results of the computational experiments, evaluating the trained DQL agent. The DDQN agent (AS_{DDQN}) is compared separately with two RCS/RS reference strategies from Section 3.3: the standard heuristic ASM by [14] and a nearest-neighbor variant $ASNN$.

5.1 Definition and Selection of KPIs for Efficiency Measurement

To evaluate efficiency in RCS/RS, this work focuses on a small set of key performance indicators (KPIs). The two central measures are the average dual-command cycle time, capturing the total time for storage and retrieval including relocations. As a complementary measure, the direct access rate indicates the proportion of retrievals that can be completed without relocations. Together, these KPIs provide a consistent basis for comparing alternative relocation strategies both at system level and across different bin categories.

5.2 Objective and Setup of Computational Experiments

The experiments aim to evaluate whether the decision-making strategy learned by the DDQN agent outperforms established heuristic approaches in terms of cycle time and energy consumption under varying system configurations. The heuristic baselines follow RCS/RS control logic from the literature: blocking bins are relocated to the nearest available location, later returned to their original stack when possible, while target bins are temporarily stored before transport to the I/O point.

A further objective is to assess the generalization capability of an agent trained in a **5×5×5** grid environment when applied to different operating conditions, including larger storage dimensions, varying fill levels, and altered retrieval probabilities for ABC-classified bins.

Each method was evaluated in 30 replications with identical initial warehouse states to eliminate random start effects; robustness to varying states was tested in an additional 30 replications. A replication comprised 2,000 dual-command cycles (1,000 warm-up and 1,000 for analysis), with fixed parameters and layout but varying random seeds for order generation. For AS_M and AS_{NN}, relocation rules were fixed ("Return" = 1, "Repeated Relocation" = 0), while in AS_{DDQN} relocation behavior was learned through rewards. Energy consumption, directly linked to cycle time, is reported separately. Global KPIs were compared using Welch's t-test ($\alpha = 0.05$), while category-specific KPIs were analyzed descriptively.

The experimental procedure consisted of:

1. **Baseline Test:** Evaluation in the original training environment (5×5×5 grid, 125 positions, 90 % fill rate, ABC retrieval probabilities) against the heuris-

tic baselines. Performance is measured using the KPIs defined in Section 5.1.

2. **Generalization Tests:** Application of the trained agent to environments deviating from the training setup:
    - **Fill rate variation:** 60 % and 93.6 % (maximum feasible for 5×5×5).
    - **Retrieval pattern variation:**
        - Equal retrieval probability and stock share across all categories.
        - High retrieval probability concentrated in a rarely stocked category.
    - **System size variation:**
        - 10×10×10 grid (1 000 positions, 6.49 m × 4.49 m × 2.2 m).
        - 20×20×10 grid (4 000 positions, 12.98 m × 8.98 m × 2.2 m).

If the agent's performance under altered system conditions falls significantly below that of the heuristic baselines (AS_{NN} or AS_M), a transfer learning approach is applied. In this case, the pre-trained network weights serve as initialization for a short adaptation phase, during which the agent is fine-tuned to the new environment.

5.3 Results

Table 3 provides a consolidated overview of the main scenarios, reporting global KPIs for both operating strategies and the DDQN agent across baseline, fill-level, retrieval-pattern and system-size variations accompanied with a fine-tuning of the agent. The table highlights consistent efficiency gains of AS_{DDQN} in terms of cycle time, typically accompanied by reduced direct access rates. The baseline, fill rate variations and retrieval pattern variations show the following results:

- **Baseline**: The AS_{DDQN} yields the overall lowest cycle times, but the direct access is also lower. This means that the driving distance for the robots is smaller, but this strategy produces lower number of direct accesses compares compared to the AS_M and AS_{NN} operating strategies. Category-wise, A-items show lower direct access and higher blocking frequency, while B/C-items achieve large cycle-time reduction (up to –42%); trends are robust across initial states.
- **Fill rates**: AS_{DDQN} reduces cycle time compared to AS_M and AS_{NN} for both fill rate variations. However, the direct access is also lower compared to both operating strategies.
- **Retrieval pattern**: AS_{DDQN} reduces the cycle time compared to the AS_M and AS_{NN} for both variation cases. The direct access is either higher (equal retrieval probabilities) or lower (high retrieval probabilities) for both AS_M and AS_{NN} compared with the AS_{DDQN}.

For the variation of system size and transfer learning, we find that direct transfer of the 5×5×5 agent to larger grids did not surpass AS_{NN} in efficiency. We therefore applied fine-tuning in the target environment with (i) initialization from existing weights, (ii) rescaled reward and stronger invalid-action penalties, and (iii) light hyperparameter retuning within a short budget ($\leq$5,000 cycles).

Table 4 shows that fine tuning leads to significant lower cycle times compared to all other operating strategies. This includes the agent trained with the baseline system without fine tuning.

Table 4: Compact comparison of fine-tuned (FT) vs. non-finetuned and heuristics.

Scenario	Comparison	Diff. % Cycle Time	Diff. % Direct Access
10×10×10	FT vs. non-FT	–2.30%	–13.12%
	FT vs. AS_M	–3.64%	–38.28%
	FT vs. AS_{NN}	–3.00%	–37.93%
20×20×10	FT vs. non-FT	–5.93%	–14.10%
	FT vs. AS_M	–6.93%	–36.10%
	FT vs. AS_{NN}	–1.25%	–35.10%

6 Limitations & Conclusion

6.1 Limitations

This study is subject to several limitations. First, the simulation relies on multiple modeling assumptions due to the lack of publicly available details on RCS/RS and particularly on AutoStore's internal control logic and real operational data. Comparisons with reference strategies (ASM, $ASNN$) are based on heuristics from literature, which may not fully reflect actual system behavior. The model includes simplifications in robot movement (e.g., no braking phase, constant lifting speed), a single-robot, single I/O-point configuration, and excludes factors such as load-dependent speed use or multi-robot coordination. Second, the work focuses solely on a DDQN-based relocation policy without comparison to other RL approaches. Training was performed on a fixed initial warehouse state without testing alternative initializations. The warm-up phase for AS_{DDQN} was aligned with the heuristics without adjustment to the agent's learning process, and the reward function was only empirically tuned.

Table 3: Global KPIs across baseline, fill-level, and retrieval-pattern scenarios (mean over 30 replications and confidence intervals)

Scenario	AS_M		AS_{NN}		AS_{DDQN}	
	Cycle [s]	Direct	Cycle [s]	Direct	Cycle [s]	Direct
Baseline (90% fill, ABC)	17.37 [17.19; 17.55]	0.53 [0.52; 0.53]	17.25 [17.08; 17.43]	0.55 [0.54; 0.55]	14.40 [14.27; 14.52]	0.42 [0.41; 0.42]
Low fill (60%)	15.02 [14.92; 15.12]	0.61 [0.60; 0.61]	14.27 [14.18; 14.37]	0.60 [0.59; 0.60]	11.75 [11.68; 11.83]	0.53 [0.52; 0.53]
High fill (93.6%)	19.19 [19.00; 19.39]	0.51 [0.50; 0.51]	19.33 [19.08; 19.58]	0.53 [0.52; 0.54]	16.15 [16.00; 16.30]	0.38 [0.37; 0.38]
Equal retrieval prob.	26.75 [26.59; 26.91]	0.21 [0.20; 0.21]	27.76 [27.57; 27.95]	0.22 [0.22; 0.23]	17.50 [17.40; 17.59]	0.23 [0.22; 0.23]
High retrieval prob.	16.02 [15.91; 16.14]	0.66 [0.65; 0.67]	15.46 [15.31; 15.61]	0.69 [0.68; 0.69]	13.22 [13.13; 13.31]	0.54 [0.54; 0.55]

Finally, the absence of external validation with real-world data limits the generalizability of the results. While the DDQN agent achieved significant energy savings in simulated scenarios, transferring these findings to operational RCS/RS requires further research under realistic multi-robot and variable operating conditions.

6.2 Conclusion

This work explored the use of a DDQN-based agent for container relocation in RCS/RS to reduce cycle time (and consequently energy consumption) for dual command cycles compared to rule-based heuristics. The agent achieved significant savings, with up to 36.98% reduction in scenarios with uniform retrieval patterns and an average of 20.89% in the 5×5×5-grid training environment, while maintaining robustness under varying fill levels and retrieval distributions. Even in larger warehouse configurations, fine-tuning preserved notable efficiency gains, though at lower percentages. The strategy avoided time-intensive returns, trading off direct access rates for overall time efficiency, highlighting the potential of learning-based approaches in realistic settings where classical ABC-based heuristics are less effective.

Future research should address current limitations by extending to multi-robot coordination, scaling to larger systems with adapted reward functions, and comparing with alternative RL algorithms such as PPO. External validation with real operational data is crucial to assess practical applicability. Explainable RL techniques could increase transparency and acceptance, while systematic tuning of reward functions and simplified state representations may improve efficiency and reduce computational demands. These directions offer a broad scope for advancing intelligent control in RCS/RS.

REFERENCES

[1] S. Chevalier, "Global retail e-commerce sales 2014–2027," Statista, 2024. [Online]. Available: https://www.statista.com/statistics/379046/worldwide-retail-e-commerce-sales/

[2] D. Coppola, "Sustainability in e-commerce," Statista, 2024. [Online]. Available: https://www.statista.com/topics/8200/sustainability-in-e-commerce/#topicOverview

[3] U.S. Energy Information Administration, "2018 cbecs: Commercial buildings energy consumption survey," U.S. Energy Information Administration, 2018. [Online]. Available: https://www.eia.gov/consumption/commercial/pba/warehouse-and-storage.php

[4] R. Wang, P. Yang, Y. Gong, and C. Chen, "Operational policies and performance analysis for overhead robotic compact warehousing systems with bin reshuffling," *International Journal of Production Research*, vol. 62, no. 14, pp. 5236–5251, 2024.

[5] Y. Liu and X. Dong, "Bins relocation problem in autostore: an overview," in *International Conference on Algorithms, High Performance Computing, and Artificial Intelligence (AHPCAI 2024)*, vol. 13403. SPIE, 2024, pp. 1013–1026.

[6] A. Hottung, S. Tanaka, and K. Tierney, "Deep learning assisted heuristic tree search for the container premarshalling problem," *Computers & Operations Research*, vol. 113, p. 104781, 2020.

[7] C. Zhang, H. Guan, Y. Yuan, W. Chen, and T. Wu, "Machine learning-driven algorithms for the container relocation problem," *Transportation Research Part B: Methodological*, vol. 139, pp. 102–131, 2020.

[8] H. Li, Q. Luo, and R. Li, "Optimizing urban carsharing systems based on geospatial big data and machine learning: A spatio-temporal rebalancing perspective," *Travel Behaviour and Society*, vol. 38, p. 100875, 2025.

[9] Y. Wu, W. Song, Z. Cao, J. Zhang, and A. Lim, "Learning improvement heuristics for solving routing problems," *IEEE transactions on neural networks and learning systems*, vol. 33, no. 9, pp. 5057–5069, 2021.

[10] J. Kallestad, R. Hasibi, A. Hemmati, and K. Sörensen, "A general deep reinforcement learning hyperheuristic framework for solving combinatorial optimization problems," *European Journal of Operational Research*, vol. 309, no. 1, pp. 446–468, 2023.

[11] Q. Wang, K. H. Lai, and C. Tang, "Solving combinatorial optimization problems over graphs with bert-

based deep reinforcement learning," *Information sciences*, vol. 619, pp. 930–946, 2023.

[12] K. Li, T. Liu, P. R. Kumar, and X. Han, "A reinforcement learning-based hyper-heuristic for agv task assignment and route planning in parts-to-picker warehouses," *Transportation research part E: logistics and transportation review*, vol. 185, p. 103518, 2024.

[13] AutoStore. (2025) Autostore – the world's fastest as/rs solution. [Online]. Available: https://www.autostoresystem.com

[14] R. D. Meller, "Considerations when designing an autostore™ system," in *Proceedings of the 16th International Material Handling Research Colloquium (IMHRC)*. Dresden, Germany: Progress in Material Handling Research, 2023. [Online]. Available: https://digitalcommons.georgiasouthern.edu/pmhr_2023/12

[15] P. Trost and M. Eder, "A performance calculation approach for a robotic compact storage and retrieval system (rcs/rs) serving one picking station," *Production & Manufacturing Research*, vol. 12, no. 1, p. 2336056, 2024.

[16] T. Lehmann, "Throughput models and performance evaluation for multi-deep automated storage systems," Ph.D. dissertation, Karlsruher Institut für Technologie (KIT), 2025. [Online]. Available: https://doi.org/10.5445/IR/1000180971

[17] M. Rabe, S. Spiekermann, and S. Wenzel, *Verifikation und Validierung für die Simulation in Produktion und Logistik: Vorgehensmodelle und Techniken*. Springer, 2008.

[18] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA: MIT Press, 2018.

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

[20] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.

[21] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[22] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 2623–2631.

Katharina Mitterer, M.Sc., Master Thesis, Institute for Material Handling and Logistics (IFL), Karlsruhe Institute of Technology (KIT).
E-Mail: katiemitterer@outlook.com

Christophe Senger, M.Sc., Research Associate at Institute for Material Handling and Logistics (IFL), Karlsruhe Institute of Technology (KIT).
E-Mail: christophe.senger@kit.edu

Timo Lehmann, Dr.-Ing., Postdoctoral Researcher at Institute for Material Handling and Logistics (IFL), Karlsruhe Institute of Technology (KIT).
E-Mail: timo.lehmann@kit.edu

Address: Gotthard-Franz-Str. 8 Building 50.38, 76131 Karlsruhe, Germany

Acknowledgments: This research was funded by the Ministry of Economic Affairs, Labour and Tourism Baden-Württemberg (Ministerium für Wirtschaft, Arbeit und Tourismus Baden-Württemberg) under the call "Klimaneutrale Produktion mittels Industrie 4.0-Lösungen" – project KLIMAförderer (Funding code: WM33-42-47/140/5).