

Optimization of Material Flows in Intralogistics Using Deep Reinforcement Learning for Autonomous Transport Systems

Optimierung von Materialflüssen in der Intralogistik durch Deep Reinforcement Learning für autonome Transportsysteme

Mustafa Jelibaghu, Michael Eley, Oliver Rose, Eleni Leontidou,
Technische Hochschule Aschaffenburg, Aschaffenburg (Germany),
Universität der Bundeswehr München (Germany), oliver.rose@unibw.de,
{mustafa.jelibaghu, michael.eley, eleni.leontidou}@th-ab.de

Abstract: This study explores the use of a Deep Q-Network (DQN) agent to optimize order allocation strategies within warehouse systems, aiming to enhance overall operational efficiency. The DQN agent employs reinforcement learning to improve decision-making processes. Results demonstrate that the DQN agent effectively reduces schedule violations (Terminverletzungen) and buffer overflows (Pufferverletzungen) to very low levels, thereby significantly improving the efficiency of warehouse operations.

1 Introduction

In modern high-bay warehouses, the efficient management of Automated Guided Vehicles (AGVs) for transporting goods represents a critical challenge in intralogistics. The dynamic allocation of transport orders to AGVs introduces complex optimization problems, particularly when balancing operational efficiency with real-time responsiveness. At the core of this issue lies the dynamic order allocation problem, which operates under three key constraints: First, the AGV fleet is limited to single-order processing, meaning each AGV can handle only one transport task at a time due to physical and logistical limitations (e.g., capacity constraints for carrying a single pallet or routing conflicts). Second, orders are subject to strict time windows, requiring completion within predefined deadlines (e.g., "Pallet must be stored by 10:00 AM"). These time windows reflect real-world dependencies such as just-in-time production schedules or time-critical inventory replenishment. Third, the primary optimization objective is the minimization of total travel distance across all AGVs, as reduced distances directly correlate with lower energy consumption, reduced wear-and-tear, and improved system throughput.

However, traditional control strategies such as random dispatching or rule-based heuristics struggle to address these challenges effectively. These methods often prioritize static criteria (e.g., shortest immediate distance) and fail to adapt to dynamic factors like stochastic order arrivals, fluctuating time windows, or uneven AGV workloads. This rigidity leads to suboptimal outcomes, including missed deadlines, excessive travel distances, and imbalanced fleet utilization. To overcome these limitations, this work proposes an AI-driven approach leveraging Deep Reinforcement Learning (DRL), enabling real-time decision-making that dynamically balances distance minimization, deadline adherence, and workload distribution across the AGV fleet.

The remainder of the paper is organized as follows. Section 2 provides a comprehensive literature review on existing approaches to order dispatching for AGVs, with a focus on classical heuristic methods and recent developments in DRL. Based on this analysis, key research gaps are identified, which highlight the need for a novel approach that combines DRL with simulation-based optimization. In section 3, the development and parametrization of the simulated warehouse environment using Siemens Plant Simulation are described in detail. This includes the modelling of dynamic elements such as stochastic order arrivals and variable time windows, creating a realistic basis for evaluating dispatching strategies. Section 4 presents the design and implementation of the DQN-based reinforcement learning agent, including its neural network architecture, reward function, and training methodology. The chapter also explains how the agent is integrated with the simulation environment via the COM interface, enabling real-time interaction and learning. Section 5 evaluates the performance of the proposed DRL approach by comparing it to traditional heuristic methods using quantitative metrics such as travel distance minimization and adherence to time windows. Furthermore, limitations of the current implementation are discussed to provide insights into potential areas for improvement. Finally, section 6 outlines possible future extensions of the model, such as the incorporation of more advanced reinforcement learning algorithms, scalable system architectures for managing larger fleets of AGVs, and the transfer of the developed approach to real-world industrial applications.

2 Literature Review

Research in the field of Reinforcement Learning (RL) and its application to logistical challenges has made significant progress in recent years. Works such as those by Dong et al. (2020) and Wang et al. (2024) lay the theoretical foundations and demonstrate the potential of DRL in complex decision-making environments.

Another study by Zhou and Rose (2023) focuses on order dispatching in flow shops. In their work, each machine is equipped with a dispatch agent that receives information about current jobs and the corresponding deadlines by which a job must be completed. When a new job arrives, the dispatch agent decides whether the job is suitable for the machine or not. In this case, Multi-Agent Reinforcement Learning (MARL) is used, as there are multiple machines, each with its own agent.

In contrast, Li et al. (2019) proposes a centralized approach to address the dispatching and pathfinding of automated mobile robots (AMRs). A central dispatching system is used as the main agent. This system collects information from all AMRs, including their current position, planned routes, and the destination of the tasks. By having

- Pallets are received at entry gates.
- Temporarily stored in buffers (WEL1/WEL2).
- Transported to one of six storage aisles by narrow aisle AGVs.

The model implements two strategies for assigning tasks to AGVs:

1. Random Dispatching: Orders are assigned using a uniform distribution (random selection).
2. DQN-Based Dispatching (AI-Driven)

4 Deep Reinforcement Learning Agent

Reinforcement learning is a branch of machine learning that involves an agent learning to interact with an environment to maximize a reward signal (cf. Figure 2). The goal is for the agent to learn an optimal policy, which maps states to actions, that maximizes the expected cumulative reward over time.

Formally, reinforcement learning can be described using the Markov Decision Process (MDP) (Sutton and Barto 2005) framework, which consists of the following components:

- State space (S): All possible conditions that the agent may encounter in the environment.
- Action space (A): a series of possible discrete actions that the agent can perform.
- Transition function (T): A rule that defines how the environment changes when the agent takes an action.
- Reward function (R): a function that specifies the immediate reward received by the agent for taking a particular action in a particular state.
- Discount factor (γ): a value between 0 and 1 that determines the importance of future rewards relative to immediate rewards.

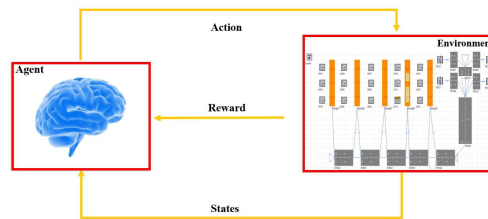


Figure 2: Agent/Environment Interaction

DQN is a specific method in the field of RL, combining classical Q-Learning with deep neural networks. We utilized TensorFlow as the primary software framework. TensorFlow provides a flexible and scalable environment for building deep reinforcement learning architectures, including the necessary tools for designing neural networks, managing experience replay buffers, and handling optimization processes. RL describes a learning approach where an agent learns to make optimal decisions by interacting with an environment to maximize long-term rewards. Overall, the DQN algorithm is a powerful and flexible tool for reinforcement learning and has been used successfully in a variety of applications, including robotics, game playing,

and autonomous driving, e.g., (Lee and Yusuf 2022; Mnih et al. 2013; Friji et al. 2020).

In our simulation model in Figure 1, states s of the AGVs are sent to agents as a matrix input. The states include essential data, enabling the agent to develop an optimal learning behaviour and a goal-oriented strategy.

The full input table is presented in Table 1. In the first column, the five AGVs are listed, to which the subsequent data correspond. These data include the number of orders each AGV currently holds when a new order arrives. Additionally, the distance each AGV requires to reach the new order location is provided, enabling an assessment of travel effort. The final input parameter is the time window, which is calculated based on the estimated time needed for the AGV to reach the new order after completing its current tasks.

Table 1: Input table RL-agent

	integer 1	integer 2	length[m] 3	time 4
string	Vehicle	Number of orders	Route AGV	Time slot
1	1	1	5	1:00.0000
2	2	0	0	1:00.0000
3	3	1	5	57.0588
4	4	2	15	51.1765
5	5	4	160.2	-34.2353

The action space corresponds to the number of available AGVs, where each action represents the selection of a specific AGV for task assignment. The next critical input parameter for the agent is the reward or penalty mechanism. The reward function (cf. Table 2) serves as a cornerstone of the DRL model, as it directly shapes the agent's learning efficiency and decision-making strategy. It was explicitly designed to prioritize key optimization goals within the warehouse logistics scenario. By assigning rewards for favourable outcomes and penalties for unfavourable ones, the function ensures the DQN agent learns to optimize order allocation strategies, thereby improving warehouse operational efficiency. The reward function integrates three core components:

Table 2: Reward function table

For what	Reward	Penalty
Time window	10	15
Shortest distance	20	
Longest distance		25
AGV_leastOrders	25	5
AGV_maxOrders		30

In our work, the reward function was carefully calibrated to balance these factors, ensuring stable learning while prioritizing key objectives. Adjustments would require re-tuning hyperparameters (e.g., learning rate, discount factor) to maintain performance. To enable the agent to optimize this reward function effectively, an appropriate training strategy is required. During training, the exploration rate

gradually decreases to shift the agent's behaviour from exploration to exploitation as learning progresses. We achieved strong results with Epsilon-Greedy through meticulous hyperparameter tuning (e.g., ϵ decay rate, exploration). To learn effectively, the DQN agent needs to both explore the environment and exploit what it has already learned in order to maximize expected rewards e.g., (Sutton and Barto 2005; Mignon and Luis de Azevedo da Rocha 2017).

5 Results

5.1 Small-Scale Model Training and Evaluation

During the training phase, the agent demonstrated a continuous improvement in its strategy because it learned the optimal policy through iterative updates guided by carefully tuned hyperparameters. The DQN algorithm relies on key hyperparameters such as the learning rate (α), discount factor (γ), and ϵ -greedy exploration rate to balance exploration and exploitation. The cumulative reward per episode increased steadily, indicating a successful optimization (on the left is the training phase cf. Figure 3). In the test phase, the agent achieved good results, successfully reducing both travel distance and delivery time violations. The small model was tested using randomly generated data that was set up realistically. The development of the test phase is illustrated on the right in Figure 3.

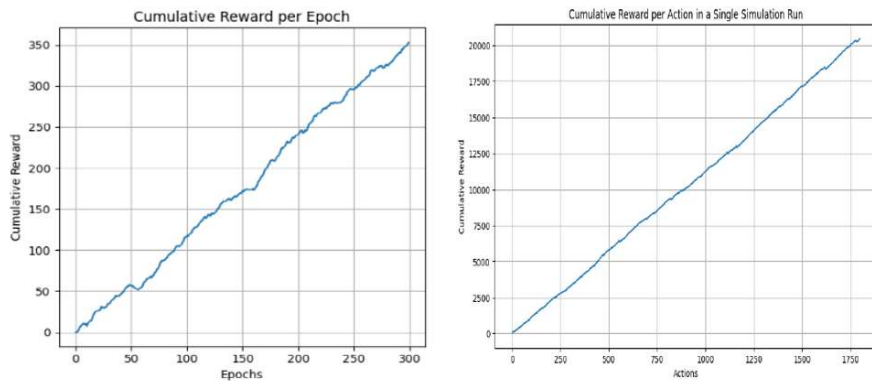


Figure 3: Training and test phase

5.2 Large-Scale Model Implementation and Performance

The resulting inference-capable models were subsequently implemented in a more comprehensive, realistic model defined as the environment in Figure 4, to evaluate their performance and generalizability under more complex conditions. The model simulates an automated warehouse with distinct zones: goods receipt/dispatch (truck gates, sorting), quality control (buffers, inspection), high-bay storage (managed by narrow aisle AGVs), and order fulfilment (packing area). Transport AGVs handle external movements, while narrow aisle AGVs operate in tight storage aisles. Workflow includes: receiving goods, temporary buffering, storage/retrieval, packing, and outbound logistics. Charging stations (BLS 1–10) ensure continuous AGV

operation. A DQN-based control system optimizes task allocation by minimizing travel distance, idle time, and workload imbalance. The system uses 10 AGVs and is tested under three peak load scenarios (Morning, Midday, End-Of-Day), where 70% of daily orders occur in concentrated time windows. This setup evaluates the DQN's robustness in dynamic, high-demand conditions.

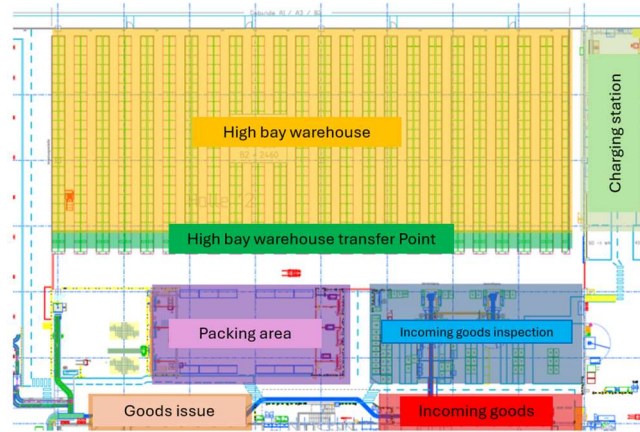


Figure 4: Realistic environment model

To assess the performance of the DQN algorithm, three additional approaches were applied to the same set of test datasets for comparison purposes. Dispatching strategies overview:

1. Random Dispatching:
 - Principle: Random assignment without considering order or AGV data (e.g., location, workload).
 - Purpose: Serves as a benchmark; efficient under high order volumes due to even workload distribution.
2. "Least Orders" Strategy
 - Principle: Assigns new orders to the AGV with the fewest active orders.
 - Focus: Balance workload through dynamic real-time analysis of current task distributions.
3. "Shortest Route" Strategy
 - Principle: Assigns orders to the AGV with the shortest total route (including empty runs between tasks).
 - Goal: Reduce transport time and energy consumption.

Figure 5 illustrates the cumulative delivery time violations in hours for the "Morning Peak" test scenario. In particular, test case 22 demonstrates the clear dominance of the DQN algorithm, with only 62 hours of delivery time violations compared to 1,334 hours for random dispatching. This value corresponds to just 4.65% of the cumulative violation time of random dispatching. Although the "least orders" and "shortest distance" strategies also show a significant reduction in delivery time violations compared to random dispatching, their cumulative values remain substantially higher than those of the DQN algorithm.

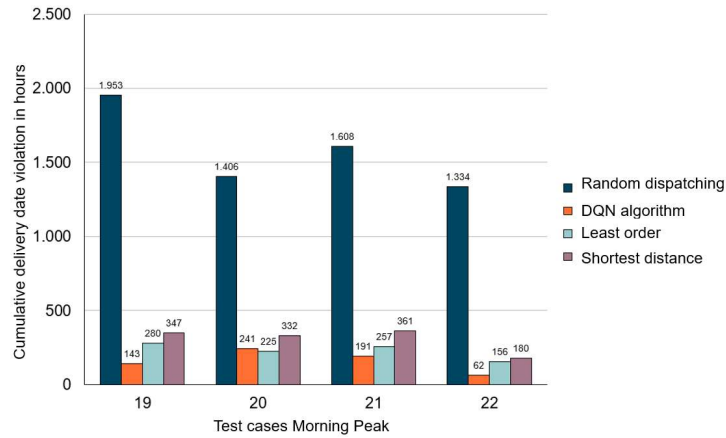


Figure 5: Results of all test cases taking into account the cumulative delivery date violations in hours in the morning peak

Figure 6 evaluates the performance of the tested algorithms in the "Midday Peak" dataset. While the DQN algorithm generally outperforms random dispatching in terms of buffer violations, test case 26 stands out as an exception. In this case, the "least orders" heuristic achieves 31.33% fewer buffer violations than DQN, indicating that alternative methods can temporarily outperform DQN under specific operational conditions. This highlights the context-dependent effectiveness of different algorithms in dynamic scenarios.

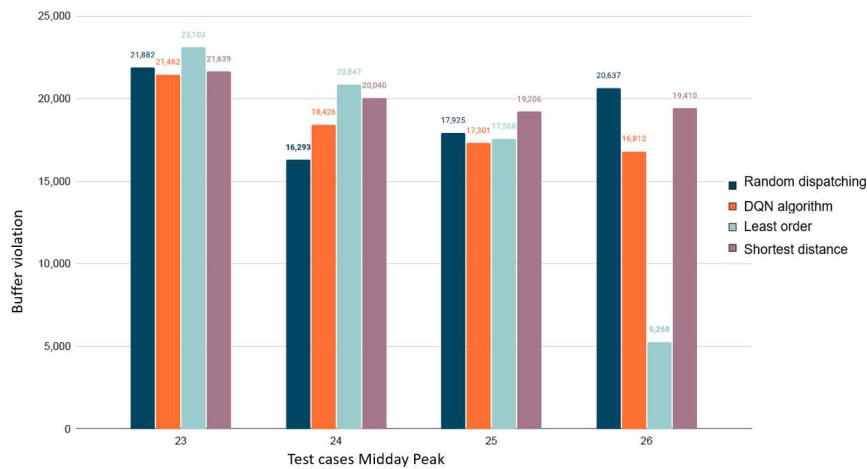


Figure 6: Results of all test cases taking into account the buffer violations in hours in the midday peak

Figure 7 presents the results of the "End-of-Day Peak" test dataset, which simulates a shift in main delivery times to the afternoon. While the DQN algorithm outperforms random dispatching, its efficiency in this specific scenario is lower than the "Least

Orders “and “Shortest Distance “variants. This suggests a potential weakness in the DQN algorithm when handling a high concentration of orders near the end of the simulation period. Although the DQN algorithm achieves fewer buffer violations than random dispatching, its performance still significantly lags behind the two alternatives methods.

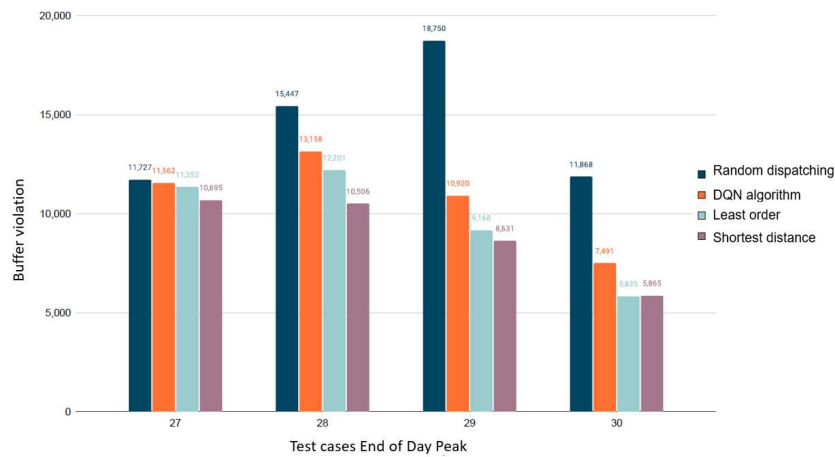


Figure 7: Results of all test cases taking into account the buffer violations in hours in the end-of-day peak

The comprehensive evaluation using diverse test datasets confirms the DQN algorithm’s robust performance under various, including extreme, load conditions. It significantly outperforms random dispatching and heuristic methods (“least orders “, “shortest distance “) in most scenarios. The following figure shows the percentage of test cases in which the DQN algorithm achieved better results than the other variants.

As shown in Table 3 in 90% of the test cases, the DQN algorithm outperformed random dispatching. It also proved superior to the other heuristic methods: in approximately 73.33% of the cases, it delivered better results than dispatching based on the least orders, and in 63.33% of the cases, it outperformed dispatching based on the shortest distance. These results highlight the adaptability and effectiveness of Reinforcement Learning (RL) for solving complex order assignment problems in intralogistics systems, even under challenging conditions like high load peaks and frequent arrivals.

Table 3: Overview of DQN performance compared to the other methods

Compared Method	DQN better performance
DQN vs. random dispatching	90.00%
DQN vs. Least Orders	73.33%
DQN vs. Shortest Distance	63.33%

6 Outlook

Future research should test the developed approach in dynamic scenarios and investigate Multi-Agent Systems (MAS) with specialized agents for load management, dispatching, and deadlock resolution to enhance efficiency and prevent system blockages (deadlocks). A coordinated approach of these agents or, alternatively, a combined agent for dispatching and deadlock avoidance offers promising avenues for further development but requires adapted control logics and training processes

References

- Alpaydin, E.: Frontmatter Maschinelles Lernen (3. Aufl.). Berlin, Boston: De Gruyter Oldenbourg. <https://doi.org/10.1515/9783110740196-fm> (2022).
- Dong, H.; Ding, Z.; Zhang, Sh.: Deep Reinforcement Learning: Fundamentals, Research and Applications. Springer (2020).
- Friji, H.; Ghazzai, H.; Besbes, H.; Massoud, Y.: A DQN-based autonomous car-following framework using RGB-D frames. IEEE Global Conference on Artificial Intelligence and Internet of Things (GCAIoT) 12-16 December 2020 Dubai (2020), pp. 1-6. <https://doi.org/10.1109/GCAIoT51063.2020.9345899>.
- Jung, A.: Maschinelles Lernen: Die Grundlagen. Singapore: Springer Nature Singapore. (2024), <https://doi.org/10.1007/978-981-99-7972-1>.
- Kuhnle, A.: Adaptive order dispatching based on reinforcement learning application in a complex job shop in the semiconductor industry. Ph.D Thesis, Karlsruher Institut für Technologie, Karlsruhe (Germany). (2020). <https://doi.org/10.5445/IR/1000127740>.
- Lee, M.; Yusuf, H.: Mobile robot navigation using deep reinforcement learning. Processes 10 (2022) 12. <https://doi.org/10.3390/pr10122748>.
- Li, Maojia P.; Sankaran, P.; Kühl, M.; Ptucha, R.; Ganguly, A.; Kwasinski, A.: Task Selection by Autonomous Mobile Robots in A Warehouse Using Deep Reinforcement Learning in: Winter Simulation Conference (WSC) (2019), S. 680–689, URL: <https://ieeexplore.ieee.org/abstract/document/9004792>.
- Li, X.; Zhang, C.; Yang, W.; Qi, M.: Multi-AGVs conflict free routing and dynamic dispatching strategies for automated warehouse. In: Kim, K.; Kim, H. (Eds.) Mobile and Wireless Technology 2018. ICMWT 2018. Lecture Notes in Electrical Engineering 513 (2019). https://doi.org/10.1007/978-981-13-1059-1_26.
- Mignon, A.; Luis de Azevedo da Rocha, R.: An adaptive implementation of epsilon-greedy in reinforcement learning. Procedia Computer Science 109 (2017), pp. 1146-1151, <https://doi.org/10.1016/j.procs.2017.05.431>.
- Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M.: Playing Atari with deep reinforcement learning. (2013). <https://doi.org/10.48550/arXiv.1312.5602>.
- Sutton, R.; Barto, A.: Reinforcement learning: an introduction. IEEE Transactions on Neural Networks 16 (2005), pp. 285-286. <https://doi.org/10.1109/TNN.1998.712192>.
- Wang, X.; Wang, S.; Liang, X.; Zhao, D.; Huang, J.; Xu, X.: Deep Reinforcement Learning: A Survey. IEEE Transactions on Neural Networks and Learning Systems (2024). Zhou, Z.; Rose, O.: Deep reinforcement learning for workload balance and due date control in wafer fabs, in: 20. ASIM Fachtagung Simulation in Produktion und Logistik (2023), S. 383–392.