

Using PER-DQN for Intelligent Control of Dynamic Warehousing and Sorting Systems

Xizhuo Han^{ID} and Chuanchuan Teng*^{ID}

School of Economics and Management, Cangzhou Jiaotong College, China

xizhuoh@czjtu.edu.cn; chuanchuant@czjtu.edu.cn

*Correspondence: chuanchuant@czjtu.edu.cn

Received: 12 November 2025; Accepted: 15 August 2026; Published: 1 October 2026

Abstract: This paper aims to address the lag problem of traditional control strategies caused by order fluctuations and equipment coordination complexity in dynamic warehousing and sorting systems, specifically addressing the limitations of the Deep Q-Network (DQN) algorithm, notably its poor sample efficiency and instability in dynamic environments. It uses PER-DQN (Prioritized Experience Replay-DQN) to build an efficient and stable intelligent control method, which enhances timing adaptability through dynamic replanning and a time-varying reward mechanism to solve existing problems. First, a 10-dimensional state vector is constructed, covering real-time orders and robot status information, and a dynamic environment model is established using the discrete grid division method to achieve a comprehensive perception of the system's operating status. Subsequently, a structured action space was designed that includes task assignment priorities and robot path directions to support multi-dimensional collaborative control. Then, a multi-objective composite reward function is proposed to comprehensively consider the task completion efficiency, dynamically adjust the optimization direction through time-varying weights, and accelerate the convergence of the policy network. Finally, a dual-network training architecture combining the PER mechanism and Double DQN is adopted to improve the utilization of key experience and suppress Q-value overestimation. An online learning mechanism is applied to achieve incremental strategy updates, ensuring that the system can maintain stable performance in sudden order scenarios. Experiments show that PER-DQN has an average delay of 14.35 seconds for small-batch orders, 24.21 seconds for mixed orders, and 34.82 seconds for large-batch orders. PER-DQN also has a path conflict rate of only 1.2% and a sorting accuracy of 99.1%. In the robustness test, PER-DQN's device utilization increases by 25.4% and 27.8% compared to DQN under device failure and high load pressure conditions, respectively. This study verifies the efficiency and robustness of PER-DQN, which improves task scheduling accuracy and resource coordination capabilities through prioritized experience replay and structured action space design.

Keywords: Deep Q-Network; Intelligent Control; Multi-objective Composite Reward Function; Prioritized Experience Replay; Warehousing and Sorting System

1. Introduction

In dynamic warehousing and sorting systems, frequent order changes, concurrent tasks, and complex equipment coordination reveal flaws in traditional control strategies. Under volatile conditions, traditional DQN methods increase order processing latency, lowering equipment use and failing to meet efficiency and stability demands [1,2]. Traditional DQNs struggle in dynamic warehouse settings due to uniform sampling, missing key state transitions, and fixed target networks, causing policy lag. With high-dimensional states and sparse rewards, they overestimate Q-values and have convergence issues, failing against real-time disturbances. Multi-agent reinforcement learning faces high communication overhead, excessive policy space, and "dimensional catastrophe" and convergence problems. Non-stationary agent

policies worsen training instability, harming coordination. Policy gradient methods are good for continuous control but have low sampling efficiency, are sensitive to hyperparameters, and lack exploration for multi-objective discrete tasks [3-5]. These limitations hinder adaptation to dynamic warehouses with real-time and multi-objective demands, limiting optimization. Thus, a reinforcement learning framework that uses experience well, adapts to changes, and supports multi-dimensional collaboration is needed, leading to the proposed PER-DQN method.

The paper's innovation is threefold: First, it creates a high-dimensional state vector with dynamic weighting, incorporating order pool and robot status parameters. A dynamic weight mechanism reflects real-time system load changes, boosting state perception accuracy and adaptability. Second, a structured action space is proposed, covering task priority, robot path direction, and conflict avoidance, enabling real-time performance monitoring for rapid response and efficient collaboration in dynamic settings. Third, by combining a multi-objective reward function with priority experience replay, the algorithm's learning efficiency and strategy generalization in dynamic environments are optimized. Compared to DQN, PER-DQN reduces order delays and improves equipment use, offering a reliable path for intelligent logistics deployment and showing strong robustness under abnormal conditions.

This study presents an Intelligent Control System using the PER-DQN RL algorithm, offering an innovative RL solution for multi-constraint, high-dynamic settings. The environment with a 10-dimensional dynamic feature vector, incorporating order pool queue lengths and robot state info to represent warehouse dynamics. This vector supports a Poisson order model and a dynamic weight mechanism for accurate warehouse characterization. For decision-making, a structured composite action space divides actions into four subspaces, enabling coordinated optimization of robot scheduling and path planning. The optimization includes a time-varying multi-objective reward function and dynamic weight adjustment for adaptive goal balancing. The learning process features an online incremental module in PER and Double DQN, facilitating continuous info acquisition and rapid disruption response, enhancing system adaptability and robustness. The primary contribution of this technology is the systematic integration of the four components of state modeling, action-like designing, reward using mechanisms (in addition to Matching Rewards/Profits), and learning strategies to create a complete intelligent control system for dynamic warehouse applications and not just as a tool for the PER-DQN technology.

The study work in no way attempts to propose an innovative general reinforcement learning approach. Instead, it seeks to formulate a scenario-based PER-DQN scheduling control scheme for dynamic warehousing and sorting operations. The novelty of the study lies in proposing a combination of a warehousing state structure, factorized composite action parameters, PER strategy, Double DQN, online updating, and multi-objective rewards weighting as a holistic scheduling solution. This paper does not attempt to provide individual theoretical contributions; rather, it seeks to combine the identified aspects to improve task scheduling and reduce path collisions in warehouses.

2. Related Work

Research on optimization control for warehousing and sorting systems has evolved from static rules to dynamic intelligence, progressing through three stages: static rule-based heuristic methods, classical meta-heuristic approaches, and deep reinforcement learning. Following this trajectory, this paper critically analyzes the strengths and weaknesses of various methodologies, identifying core contradictions and research gaps.

2.1. Static Rule-Based Heuristic Methods

Early research predominantly employed static rule-based heuristic methods, utilizing simple priority rules for scheduling decisions. The core approach involves simplifying complex scheduling problems into real-time rule-based decision-making. Xu *et al.* [6] proposed order splitting and sub-order batch adjustment strategies for mixed order sorting, constructing an integer programming model aimed at minimizing total completion time, which demonstrated significant effectiveness under specific order structures [6]. Boysen *et al.* [7] studied circular sorter scheduling, deriving parallelized picking algorithms that exhibited good performance under deterministic assumptions. Duque-Jaramillo *et al.* [8] developed an integer linear programming model for SKU warehouse space allocation considering priority levels and order demands,

optimizing static layouts. J. Zhang *et al.* [9] proposed a hybrid rule-based algorithm for the on-line order batching and sequencing problem with multiple pickers, combining priority and heuristic rules for real-time scheduling. However, static rule-based methods have fundamental limitations. Their fixed rules lack dynamic state awareness and neglect environmental dynamics and state transition uncertainties under deterministic assumptions, leading to poor generalization in stochastic dynamic environments and inadequate adaptation to modern logistics systems.

In terms of industrial warehouse automation systems, rule-based and heuristic approaches remain widely used due to their deterministic, interpretable, and computationally inexpensive nature. For example, Kiva-style robotic order fulfillment systems normally incorporate rules of task priorities, shortest path or reservation route planning, queue reasoning, traffic zone management, and congestion avoidance heuristics to synchronize mobile robots in real-time operations. They demonstrate high effectiveness in situations when warehouse configurations, order arrival patterns, and operational constraints remain relatively stable. The main advantage comes from reliability: decisions can be made within strict real-time constraints without the need for online reinforcement learning or continuous optimization procedures. However, rule-based techniques often require human-crafted rules and parameter tuning. In case of drastic changes in the rates of order arrivals, unpredictable robot failures, and dynamic local congestion, rigidly set heuristics might fail to properly adjust decision-making processes. Industrial rule-based systems thus form a reliable baseline for real-time applications, whereas machine learning approaches that would allow for adaptive preference updating and rare event handling become promising directions.

2.2. Meta-Heuristic Optimization Methods

As problem complexity increases, researchers have introduced meta-heuristic optimization methods that simulate natural evolution or swarm intelligence behaviors to search for near-optimal solutions. Zhan *et al.* [10] developed a task operation Pareto optimization model based on cycle time and carbon emissions, considering the impact of repositioning operations and transforming the cycle time model. The nonlinear multi-objective function was solved using the Elite Non-Dominant Sorting Genetic Algorithm II [10]. Ma *et al.* [11] designed an intelligent warehouse robot scheduling system based on an improved non-dominant sorting algorithm, achieving new progress in multi-objective optimization. However, it has core limitations. On one hand, the computational complexity is excessively high, requiring iterative searches across vast solution spaces, which fail to meet real-time requirements for warehouse scheduling. On the other hand, it lacks dynamic adaptability, being essentially an offline optimization approach that cannot adjust strategies in real-time during system operation. This exposes the first fundamental contradiction in logistics optimization: the conflict between optimality and computability. Pursuing global optimal solutions often comes at the cost of real-time computability, while warehouse systems need to make scheduling decisions within milliseconds.

2.3. Deep Reinforcement Learning Methods

In recent years, deep reinforcement learning methods have become a research hotspot due to their end-to-end learning capabilities and environmental adaptability [12-14]. These approaches enable agents to continuously interact with environments, learning optimal strategies through trial and error, theoretically allowing adaptation to dynamic environmental changes. Leon *et al.* [15] pointed out that reinforcement learning can simulate environmental interactions to learn optimal strategies, providing a flexible framework for dynamic decision-making in manufacturing and logistics systems. Haijiang *et al.* [16] integrated Dueling DQN and Double DQN technologies to address dry bulk terminal yard scheduling challenges, significantly improving joint optimization efficiency in transportation planning, yard selection, and equipment scheduling. Fei *et al.* [17] combined reinforcement learning with clustering algorithms to develop a dynamic cigarette logistics transfer point optimization model, reducing transportation costs through real-time transfer area adjustments. Kim *et al.* [18] designed an n-grid sorting system employing collaborative multi-agent reinforcement learning algorithms to achieve dynamic coordinated control between transmitters and sorters.

Theoretically, the problem of warehouse scheduling fits into a Markov Decision Process framework in which the controller observes the state of the warehouse, makes scheduling and routing decisions, gets

feedback from operations, and transitions to another state. However, the approach faces a number of fundamental problems. First of all, the state space explodes very quickly with an increase in the number of robots, shelves, orders, stations, and path nodes. If direct mappings from raw spatial maps, trajectory data of individual robots, inventory states, and characteristics of orders are used, then the learning process will require too much computational power. The second challenge comes from the sparsity or delay in rewards that occur because the results of scheduling become clear only when the order is finished, conflicts have been avoided, or queues have been reduced. The third problem lies in the difficulty of maintaining observability for the controller in practice due to possible local congestion, equipment states, and incoming orders. These problems make state abstraction a key point of warehouse scheduling. Unlike traditional approaches, where the controller works with raw observations of the warehouse, practical reinforcement learning algorithms create abstract semantic states that contain only relevant features. Such features include the length of the order queue, robot states, shelf occupancy, level of congestion, station idleness, and maximal waiting time.

Existing reinforcement learning methods have architectural flaws in dynamic warehouse settings, forming the basis for the proposed PER mechanism analysis. Firstly, uniform empirical sampling leads to low sample efficiency. Standard DQN samples uniformly from the experience pool, ignoring learning value. In sparse reward warehouse scheduling environments, critical state transitions are rare but vital for learning. Uniform sampling often misses these key experiences, weakening learning signals for strategies. Systems must learn from rare conflict scenarios, but uniform sampling makes it hard to use valuable samples, causing slow convergence or suboptimal strategies. Secondly, Q-value overestimation shows significant bias. Standard DQN uses maximization for target values, introducing bias that overestimates Q-values. The complex, high-dimensional multi-robot path combinations in warehouse scheduling worsen this bias, distorting policy evaluation and leading algorithms to overestimate risky paths while ignoring safety strategies, increasing path conflicts. Thirdly, policy non-stationarity and multi-agent coordination remain challenging. In multi-agent warehouses, agent policies evolve simultaneously amid environmental non-stationarity. Independent Q-learning ignores this, harming convergence stability. While multi-agent methods consider coordination, they face communication overhead, expanding joint action spaces, and difficulties in stable training in dynamic settings.

The analysis highlights a core contradiction in logistics reinforcement learning: optimality vs. adaptability. Deep reinforcement learning models can theoretically learn optimal strategies, but their reliance on historical data makes rapid adaptation to sudden or out-of-distribution changes difficult without extensive retraining. This conflict is acute in dynamic warehousing, where real-time responsiveness and robust adaptation to abnormalities are required. The PER-DQN framework addresses this by enhancing key sample use via prioritized experience replay, correcting uniform sampling to learn from rare events; using Double DQN to mitigate overestimation bias; and introducing online incremental learning for rapid adaptation. This balances optimality and adaptability in non-stationary multi-robot settings.

2.4. Core Contradictions in Dynamic Warehouse Control

Dynamic warehouse control inherently represents the trade-off between optimization quality and instantaneous computability of solutions. Rule-based methods such as shortest-path routing, fixed dispatching policies, and priority-based scheduling allow for efficient decision-making because they rely on predetermined procedures that exclude the need for real-time optimization. Still, their adaptability depends significantly on the order arrival pattern, robot availability, machinery malfunction, and dynamic changes in the path traffic. Conversely, meta-heuristic optimization and reinforcement learning can improve scheduling performance considering reward maximization over time, resource management, task delays, and path conflicts; however, they tend to be computationally expensive at the step update phases. The PER-DQN approach cannot eliminate the outlined contradiction entirely. Rather, it presents a balanced compromise between adaptive scheduling quality and the computational aspect. In essence, the major computational effort is shifted to the training and updating processes, where prioritized experience replay makes relearning valuable transitions more likely. The valuable transitions tend to represent important events in the warehouse that happen rarely but have practical significance for decision making, which is demonstrated by transitions associated with path conflicts, sudden spikes in order arrival rates, machinery

failure situations, and late task assignment. However, the advantage comes at the computational expense of increased computation due to priority calculation, priority updating, replay buffer management, and non-uniform sampling in addition to that required by uniform experience replay. The advantages and disadvantages of PER can be quantified when taking into consideration the computation required by PER. In this paper, the computation overhead incurred by PER is mainly reflected in offline training and online learning. At the real-time operation stage, the scheduling controller only needs to conduct a forward propagation and decode the structured action from the trained DQN model. As such, the computational overhead at the online stage is manageable for small to medium robotic warehouses. The experiments support this trade-off. The scheduling decision made by PER-DQN takes about 12.3 ms, slightly larger than 8.5 ms consumed by the heuristic algorithm A*+SJF. But it achieves better average order delay, higher system throughput, higher equipment utilization, and lower path conflict rate. It means that the superiority of PER-DQN does not come from giving the globally optimal real-time scheduling decision, but a practical balance between decision quality and computational efficiency in the online stage. For very large-scale robotic warehouses, the methods based on graph-based learning and value decomposition learning might be more suitable choices.

2.5. Latest Advances

Since 2023, research on intelligent control for warehousing and sorting systems has shifted toward deep integration of multi-agent reinforcement learning and graph neural networks, primarily addressing dimensionality explosion and non-stationarity challenges arising from large-scale robotic coordination. Traditional methods face scalability bottlenecks in scenarios involving thousands of robots. HAC-FRL framework [19] combines auction mechanisms with consensus protocols to achieve distributed task allocation through fine-grained cyclic learning. Its main contribution lies in integrating local policy optimization with auction-consensus coordination, enabling robots to adjust task-allocation strategies dynamically before execution and improving conflict resolution and deadlock recovery in multi-agent warehouse systems [19]. Yuan *et al.* [20] proposed a multi-agent deep reinforcement learning method for multi-robot task allocation in e-commerce robotic mobile fulfillment systems. By improving the DQN algorithm with cooperative mechanisms and prioritized experience replay, the method enhances scalability and adaptability in large-scale dynamic warehouse environments [20]. MARVEL framework integrates graph attention networks with deep reinforcement learning to solve reliable multi-robot navigation in uncertain topological networks. The self-attention mechanism focuses on critical graph features, maximizing punctuality probabilities for robot teams facing road network uncertainties, highlighting the advantages of graph attention mechanisms in dynamic environment adaptation [21]. The RoboBallet system developed by Lai *et al.* [22] integrates graph neural networks with reinforcement learning, transforming robots into nodes on a graph. It learns optimal strategies through simulated cases and generates collision-free motion plans within seconds, demonstrating the advantages of graph-based strategy representation in handling multi-robot spatiotemporal dependencies.

In the field of warehouse automation, MARL and GRL have been extensively studied in recent literature. Specifically, Li *et al.* [23] investigated MARL algorithms applied to order picking robots, proving that robots' cooperation and collaborative behavior improve task allocation and conflict resolution under dynamic order picking conditions. It is worth mentioning that, in this study, robots themselves are seen as individual decision-making agents, while multi-robot interference was resolved by joint policy learning. Furthermore, Chen *et al.* [24] proposed a graph-based MARL algorithm to schedule smart warehouses using graph neural networks and QMIX. By applying graph representation, robots, orders, shelves, and paths were represented by graph elements, and the value decomposition method was used to enable multi-agent cooperation. The importance of scalability, which is well captured in graph models, should not be underestimated since it is one of the key issues in this research area. Recent survey-based studies, e.g., Liu *et al.* [25], show that non-stationarity, credit assignment, communication overhead, partial observability, and scalability are major difficulties in the implementation of MARL in logistics and warehouse automation. In terms of methodology, this proposed approach differs from all of these previous investigations. In particular, HAC-FRL and its variants that utilize an auction or consensus strategy in task allocation focus on distributed solutions, which are better suited for large robot swarms but carry the overhead of

communication and negotiation between agents. The graph-based QMIX and BMG-Q framework represents the interaction between robots, tasks, and shelves by means of graphs and provides better scalability for systems of arbitrary sizes; nevertheless, it needs to build graphs, update them, exchange information, and learn a value decomposition. Finally, MARVEL and RoboBallet primarily concentrate on graph-based navigation and collision-free planning. In other words, this PER-DQN framework is targeted at centralized warehouse scheduling, where task prioritization, robot direction, charging, and conflict resolution are important components of the decision process, as well as multi-objective reward maximization. This method should not be viewed as a substitute for the graph-based multi-agent reinforcement learning (MARL) approach or auction-consensus MARL, which may be more suitable for very large-scale warehouses. Instead, the proposed PER-DQN framework offers a low-complexity approach for scheduling smaller to medium robot swarms, where prioritized experience replay helps improve performance by leveraging less frequent but high-value experiences, e.g., when path conflicts happen, there are urgent orders, or some robots malfunction. For large swarms, graph-based and value-decomposition MARL methods are preferable. It is possible to consider integrating PER into the graph-based QMIX as a good future research direction.

To comprehensively review the research trajectory and technological trends in intelligent control systems for warehousing and sorting, this section synthesizes key prior literature. Table 1 presents comparative analyses of representative studies across four dimensions, highlighting the distinctive features and limitations of different technical approaches.

Table 1. Summary of Key Literature on Intelligent Control of Warehousing and Sorting Systems

References	Algorithm category	Specific algorithm	Status/action representation	Key technical limitations
Boysen [7]	Static rule heuristic	Parallelized picking algorithm	Sorter configuration, order queue (status); picking sequence (action)	Based on deterministic assumptions, performance plummets when equipment fails, and orders fluctuate
Duque-Jaramillo [8]	Static rule heuristic	Integer linear programming	SKU attributes, warehouse layout (status), cargo space allocation (action)	Complete a priori information is required, and the computational complexity increases exponentially with the scale.
Ashok [9]	Meta-heuristic	NSGA-II	Cycle time, carbon emissions (status); task scheduling (action)	Offline optimization is unable to adapt to environmental changes in real time
Li [16]	In-depth intensive learning	Dueling DQN+Double DQN	Yard status, equipment status (status); scheduling plan (action)	Uniform empirical sampling, insufficient learning of key states (such as equipment conflicts)
Kim [18]	Multi-agent intensive learning	Collaborative multi-agent intensive learning	Grid state, transmitter state (state); sorting control (action)	The strategy is not stable, and the training is unstable
Gu [19]	Multi-agent intensive learning	HAC-FRL	Robot status, task queue (status); distributed task allocation (action)	The computational overhead of the auction consensus mechanism increases with the number of agents
Lai [22]	Graph neural network + Intensive learning	RoboBallet	Graph node (robot/task/obstacle) (status); no collision path (action)	Figure scale randomizer number of people square-level growth
Li <i>et al.</i> [23]	Multi-agent reinforcement learning	MARL for order-picking robots	Robot states, order-picking tasks, local path-conflict information; coordinated robot actions	Strong coordination ability, but may suffer from policy non-stationarity and credit-assignment difficulty.
Chen <i>et al.</i> [24]	Graph neural network + MARL	GNN-QMIX / graph-based MARL	Robots, orders, shelves, and paths represented as graph nodes; value-decomposition-based actions	More scalable for variable-size systems, but graph construction and message passing increase computational complexity
Liu <i>et al.</i> [25]	Survey	MARL in logistics and warehouse automation	Taxonomy of agents, tasks, communication structures, and coordination mechanisms	Provides broad field positioning but does not propose a specific warehouse-control algorithm
Present study	In-depth intensive learning	PER-DQN	10-dimensional structured vector (order/robot/shelf); four-dimensional structured action space	Limited extensibility of more than 100 ultra-large-scale scenes

It can be seen from Table 1 that the recent works on MARL and graph-based methods have better scalability than DQN-type algorithms. This is achieved due to the use of explicitly defined inter-agent relations or value function decompositions. In turn, HAC-FRL makes use of auction consensus and, thus, more closely approximates distributed task allocation for large robot fleets. On the other hand, the introduced PER-DQN framework uses centralized scheduling, complemented with priority replay that allows better learning from infrequent but important events related to collisions, emergencies, and robot failure states. Indeed, GNN-based models like GNN-QMIX and BMG-Q are suitable for solving the problem of large-scale robot fleet management since graphs allow modeling different interaction sizes between robots, tasks, and shelves. However, graph-based methods include graph building, updating, propagation, and learning from the decomposed values, which makes them computationally more expensive. MARVEL and RoboBallet focus mainly on graph-based robot navigation and collision-free motion planning. At the same time, PER-DQN addresses task prioritization, direction decision making, charging scheduling, collision prevention, and balancing of the rewards for multiple objectives in centralized scheduling. Thus, the PER-DQN framework appears as an adequate scheduling approach for small- and medium-sized dynamic robot fleets.

3. Construction of Dynamic Warehousing Intelligent Control Framework

The complete framework of the interaction between the PER-DQN agent and the dynamic warehouse environment is shown in Fig. 1. The overall architecture and data interaction flow of the intelligent control system proposed in this paper can be easily illustrated with the complete interaction framework of the intelligent control system illustrated in Fig. 1. The architecture diagram illustrates the entire closed-loop process, including the perception of the environment's state, the decision-making and learning process of the agent, the execution of the action taken, and the feedback received from the environment, from the agent's perspective. The architecture diagram provides a visual display of the collaborative interaction between major modules: PER, structured action space, and dual-network training.

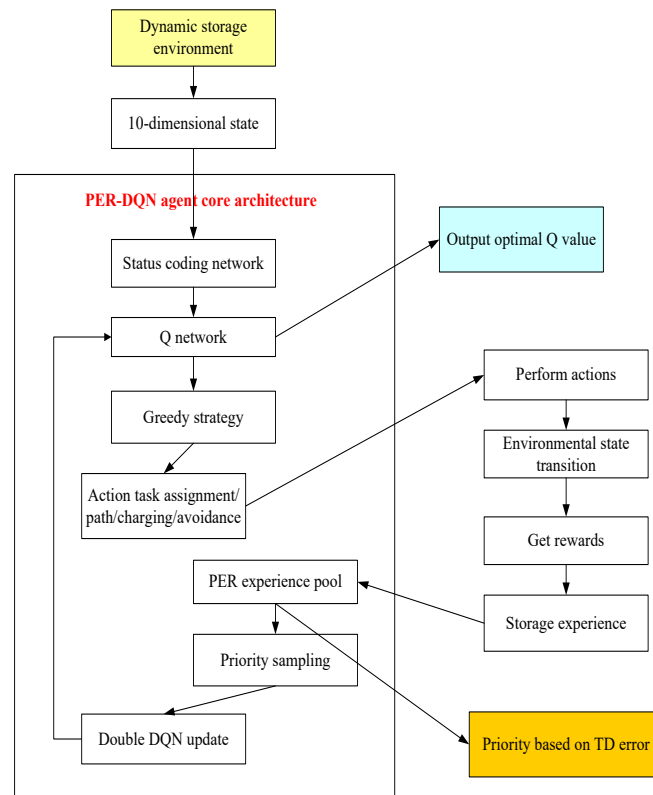


Figure 1. Intelligent Control Architecture of Dynamic Warehousing and Sorting System Based on PER-DQN

The PER-DQN approach is a centralized scheduler rather than a fully decentralized but centrally trained multi-agent paradigm. During both training and deployment phases, one PER-DQN algorithm learns from the full system state to produce a factorized composite action for scheduling purposes. Then the scheduler parses the output to individual actions for each robot, such as task assignments, path

movement, charging operation, and conflict management. Thus, contrary to the QMIX and MAPPO paradigms, one PER-DQN policy or utility function is shared among all robots in the proposed approach. The strength of this scheme is that centralized coordination and prioritization of important experiences during training can be easily achieved in moderately scaled-up warehouses. Nevertheless, the disadvantage is that the scalability to extremely large robot numbers is hindered by the size of the centralized state and action space. This system is formulated as a centralized PER-DQN scheduling controller and not as a CTDE multi-agent system. In other words, there is one PER-DQN model that analyses the overall warehouse situation and generates scheduling decisions. This implies that training and execution remain centralized at the scheduling controller level.

3.1. Dynamic Environment: Modelling and State Representation

Before building a PER-DQN-based intelligent control framework, it is necessary to first model the operating environment of the dynamic warehousing and sorting system. The effectiveness of reinforcement learning algorithms in solving practical problems is highly dependent on the accuracy of environmental modeling and the rationality of state representation. To this end, this paper designs a high-dimensional state vector that combines real-time orders, the operating status of port logistics robots, and the saturation of shelf resources. Based on the warehouse space structure, task flow, and resource occupancy rules, a discrete dynamic environment model that can be used for reinforcement learning training and interaction is constructed. In dynamic warehousing systems, "high" and "low" dimensions of the state representation relate to the problem structure and state information density. The proposed 10-dimensional state vector differs from the typical high-dimensional and sparse state representations in that it is structured as a semantic feature set derived from the logic of warehouse operations. Each of the ten dimensions of the state vector corresponds to key decision-making variables such as order pool size, load pressure, the locations of robots, and the condition of equipment; therefore, the state vector design is intended to compress the raw measurements made at the warehouse into decision-making features that maintain the semantics of scheduling while simplifying both the state description and exploration process. Compared to end-to-end inputs, engineering features using knowledge of the domain allow for improved sample efficiency and more stable training, helping make domain-specific feature engineering suited for use in industrial scheduling problems.

To model fluctuations, the system adopts an order generation mechanism based on the Poisson arrival process, taking into account both sudden and dense orders during peak hours and periodic fluctuations, simulating order loads that are closer to real warehousing scenarios, and embedding order attribute diversity parameters so that scheduling strategies can make dynamic decisions based on order characteristics. In terms of equipment status modeling, the real-time location, energy consumption, and other information of the multi-robot system are applied, combined with path resource occupancy, and indicators such as the channel congestion index are established to dynamically reflect changes in system load and bottlenecks. The state representation integrates statistical features to help the system collaboratively judge scheduling pressure.

This paper formalizes the dynamic warehouse scheduling problem as a Markov Decision Process (MDP), defined by a five-tuple (S, A, P, R, γ) . 1. State Space S The state space is defined as $S \subset \mathbb{R}^{10}$, and the state vector at time t is represented as:

$$s_t = [\sigma_t, a_1, y_1, h_1, a_2, y_2, h_2, \rho_1, \rho_2, \rho_3]^T \quad (1)$$

$\sigma_t \in [0,1]$ denotes the normalized order queue length; (a_i, y_i) represents the position of robot i in the normalized grid coordinate system; $h_i \in \{0,1,2\}$ indicates the robot state code (0 = idle, 1 = working, 2 = charging); and $\rho_j \in [0,1]$ signifies the saturation level of shelf j . This structured representation preserves core scheduling features while avoiding dimensionality explosion.

Action Space A is designed as a Cartesian product of four high-dimensional decision parameters: $A = A_w \times A_p \times A_c \times A_r$, corresponding to task assignment priority, path planning direction, charging trigger threshold, and conflict avoidance rules. The central scheduling action at time t is expressed as:

$$a_t = [a_t^{(w)}, a_t^{(p)}, \theta_{c,t}, a_t^{(r)}]^T \quad (2)$$

This design enables agents to directly optimize scheduling policy parameters rather than underlying motion primitives, thereby enhancing decision-making abstraction levels and generalization capabilities.

The transition probabilities P for states (P, R, γ) are determined by warehouse environment dynamics, while the reward function R employs the proposed time-varying weight multi-objective mechanism. The discount factor $\gamma \in [0,1)$ balances immediate returns with long-term benefits.

The design of the state space must balance information completeness with computational efficiency. This study constructs a 10-dimensional state vector that encompasses three major categories of dynamic parameters: order load, robot status, and resource utilization (see Table 1 for details), enabling comprehensive real-time monitoring of system operations [21,22,26]. The mathematical expression of this vector is:

$$h_t = [o_t, a_1, y_1, h_1, \dots, a_4, y_4, h_4, \rho_1, \rho_2, \rho_3, \rho_4] \in \mathbb{R}^{10} \quad (3)$$

o_t denotes the queue length of pending orders in the system at time t , a non-negative integer. a_i and y_i represent the horizontal and vertical coordinates of the i -th port logistics robot in the two-dimensional warehouse grid ($i=1,2,3,4$), used to track the robot's real-time position. h_i indicates the operational status of the i -th robot. ρ_j denotes the saturation level of the j -th shelf ($j=1,2,3,4$).

The 10-dimensional vector in this paper is based on the knowledge of warehousing and scheduling, and the key decision-making variables extracted after compressing the massive raw data system are the results of structured low-dimensional feature engineering. From the mathematical definition, the state vector $h_t \in \mathbb{R}^{10}$, each dimension has clear semantics: o_t is the normalized value of the queue length of the order pool; (a_i, y_i) is the position of the i -th robot in the normalized grid coordinate system; h_i is the state code of the i -th robot (0 idle, 1 working, 2 charging); ρ_j is the j -th shelf saturation. For multi-robot expansion, the vector adopts a modular design, which can be naturally extended to N robots. $h_t = [o_t, \{a_k, y_k, h_k\}_{k=1}^M, \{\rho_j\}_{j=1}^M]$ Each robot occupies 3 dimensions, and 50 robots are about 150 dimensions, which is still a structured low-dimensional representation. Moreover, this paper adopts a centralized training and distributed execution architecture, and each robot only needs local observation to avoid dimensional explosion. This design balances information completeness and computational efficiency, retains core scheduling characteristics, and has good extensibility.

The state vector architecture (proposed) has a highly scalable design because the dimensional aspects reflect a modular architecture. Even though Formula (4) illustrates the formulation of a state vector in terms of two robots, the design will easily extend to multiple robot systems. In the case of N robots, the state vector would be expressed as $h_t = [o_t, a_1, y_1, h_1, \dots, a_N, y_N, h_N, \rho_1, \rho_2, \rho_3, \rho_M]$, with positions and status codes associated with each robot added to the existing pattern. The number of issues associated with the “shelves” can also be adjusted according to how many shelves there are in actuality. This modularity provides flexibility for the use of robot systems at different scales; path directions and avoidance rules that are defined within a structured action space will be specified independently, relative to the robot ID, which will provide further support for multi-agent, collaborative scheduling, which is an expandable requirement.

$$s_t^{(N)} = [\hat{o}_t, \hat{c}_t, \hat{u}_t, \hat{m}_t, \hat{w}_t, \hat{p}_t, (\hat{x}_{1,t}, \hat{y}_{1,t}, h_{1,t}), \dots, (\hat{x}_{N,t}, \hat{y}_{N,t}, h_{N,t})] \quad (4)$$

$$d_s = 6 + 3N$$

Systematic engineering principles are followed in the 10D state vector's design. The 10D state vector has 10 dimensions that represent selected and key decision variables, which have been identified through semantic abstraction of the various elements that could impact those decision variables. Multi-robot position data has been encoded using normalized coordinates to maintain the relative position of robots while not producing a dimension explosion. Robot status is represented using a 3-digit trinary code defining the availability of equipment, while shelf saturation defines pressure from storage resources, and order pool queue lengths define urgency of tasks based upon their desired maximum wait period. Through this design, all dimensions have a direct influence on scheduling decisions. Normalizing the dimensions eliminates inconsistencies between the dimensions, providing an efficient and accurate environmental representation for subsequent scheduling efforts. Thus, optimizing the level of completeness of information against computational efficiency.

This paper uses the discrete grid partitioning method to model the spatial dimension of the warehousing system. The warehousing space is set as an $M \times N$ two-dimensional grid. Each grid point represents a location unit and can be identified as a different area, such as a storage area. In this space, port logistics robots, shelves, and task targets are all mapped as a set of coordinate points, and the robots perform

tasks by moving between grid points. Spatial region division controls the boundaries and rules of task generation points, target points, and scheduling paths. The system sets path node attributes and occupancy status to avoid conflicts between robot movement and resource access. The system state variable definitions are shown in Table 2.

Table 2. Definition of state variables of a dynamic storage and sorting system

Serial number	State variable name	Dimension	Data type	Meaning description
1	Current order pool queue length	1	Integer	The number of orders waiting to be processed in the system reflects the current load situation.
2	Robot 1 position coordinates (X, Y)	2	Integer	The current two-dimensional coordinate position of robot 1 in the warehouse grid
3	Robot 2 position coordinates (X, Y)	2	Integer	The current moving position of Robot 2 in the warehouse space
4	Robot 1 status	1	Category	Indicates the current status of robot 1 (0-idle, 1-working, 2-charging)
5	Robot 2 status	1	Category	Indicates the current status code of robot 2; the meaning is the same as above
6	Average inventory saturation of the shelf area	1	Floating point	Represents the average inventory ratio of the entire storage area, normalized processing
7	Current system path congestion index	1	Floating point	Represents the density of robot activity in the main channel area, which is used to estimate the degree of path congestion
8	Sorting station idle rate	1	Floating point	Indicates the proportion of free workstations in all sorting stations
9	Number of tasks currently being executed	1	Integer	Indicates the number of orders currently allocated and being processed in the system
10	Current maximum order waiting time steps	1	Integer	The waiting time for the longest queued order in the current waiting pool

The 10 state variables listed in Table 2 describe the real-time operating status of the dynamic warehousing and sorting system, involving multiple key dimensions such as order load, robot location, behavior, and inventory. Among them, the order pool queue length and the maximum order waiting time can reflect the urgency of order processing and system pressure; the path congestion index can intuitively present the main channel traffic efficiency and potential conflict risks. These variables not only provide key inputs for intelligent algorithms to perceive the environment but also lay a comprehensive data foundation for scheduling optimization and strategy evaluation.

The state vector (10D) is subject to observability, relevance, controllability, and feasibility principles. It includes metrics such as order pool size, robot status, shelf saturation, and sorting station idle rate. Each metric relates to orders, robots, and resources. This minimum but complete set of features preserves the dimensionality problems and provides PER-DQN with a structured representation of the environment, thus allowing for real-time decision making in dynamic environments.

The dynamics of the environment are reflected both in spatial changes and in the evolution of the state over time. In reinforcement learning, the state transition function describes the transition of the system from the current state x_t to the next state h_{t+1} after executing the action $P(h_{t+1} | h_t, x_t)$. In dynamic warehousing scenarios, this transfer is affected by factors such as task execution feedback, path planning, inventory updates, and robot position changes. When the port logistics robot completes the pickup and delivery task, its status changes from "busy" to "idle"; its location is updated, the shelf inventory decreases, the saturation changes, and the number of tasks in the system order pool decreases accordingly. The order pool queue update can be expressed as:

$$Q_O(t+1) = Q_O(t) + \lambda_t - \mu_t \quad (5)$$

Q_O is the normalized value of the order queue length, λ_t represents the number of newly generated orders per unit time, which can be modeled as a Poisson distribution, while μ_t represents the number of completed orders within that time unit. The difference between the two reflects the dynamic balance between system workload and service capacity.

To ensure the stability of the input data of the learning model and the convergence of the neural network training, this paper normalizes the original state vector so that it is in the range of [0,1]. For the order queue length, let the maximum number of orders be Q_{max} , and the normalized result is $\hat{o}_t = \frac{o_t}{Q_{max}}$;

similarly, the shelf saturation is already in $[0,1]$ and can be directly input; the coordinate position is mapped to the relative position according to the maximum grid size, and the final state vector is:

$$\hat{\mathbf{h}}_t = [\hat{o}_t, \hat{a}_1, \hat{y}_1, h_1, \dots, \hat{a}_4, \hat{y}_4, h_4, \rho_1, \dots, \rho_4] \quad (6)$$

Although the 10-D structured state vector has been shown to function effectively, improving the representation of the state is crucial for enhancing the realism of the state representation within a system. In complex warehouse environments, features extracted from local maps using convolutional neural networks could provide richer spatial detail regarding obstacle distributions from visual sources than would be possible using traditional sources. These visual features, when integrated into the current state representation, will increase the state-space complexity, thus reducing sample efficiency, training stability, and increasing computation overhead. The production and use of robots in industrial applications requires that both performance and cost be taken into consideration, which necessitates the use of a 10-D state vector, where there is a balance between completeness of information and computational efficiency, while at the same time reducing the effect of noise and redundancy on the core decision variables.

3.2. Adaptive Action Strategy Based on PER-DQN

In dynamic warehousing and sorting systems, intelligent agent behavior decisions must take into account multi-dimensional goals such as scheduling optimization, resource coordination, and path safety [11, 27, 28]. In the reinforcement learning framework, action space design is crucial for the expressiveness of an agent's control strategy. PER-DQN is a deep reinforcement learning method optimized for offline training. Its decision-making strategy consists of a neural network outputting action Q-values, which are then selected using a greedy strategy or ϵ -greedy mechanism. To ensure that PER-DQN has sufficient control accuracy and decision granularity in dynamic warehousing systems, this paper divides the action space into four semantically distinguishable decision dimensions, namely, task dispatch priority, port logistics robot path direction, charging scheduling threshold, and conflict avoidance rules.

3.2.1. Structured action space design

The task dispatch priority is a core scheduling variable used to determine the order in which the agent prioritizes jobs [29-31]. If the system has K tasks to be scheduled, numbered $\{\tau_1, \tau_2, \dots, \tau_K\}$, its priority action is defined as the weight vector $w = [w_1, w_2, \dots, w_K] \in \mathbb{R}_K$ output by the network and normalized. After the network output, it is normalized so that it satisfies:

$$\sum_{k=1}^K w_k = 1, w_k \geq 0 \quad (7)$$

The task with the largest weight vector is the priority task. The action strategy uses this to select tasks that satisfy specific conditions, enabling dynamic task selection. This strategy takes into account order urgency, robot idleness, and path load. The task sequencing logic is learned through neural network training.

The robot's path direction decision is a physical behavioral expression of the task. Given that the warehouse environment is discretized into a two-dimensional grid, each robot can move between four adjacent grid points: up ($\uparrow$), down ($\downarrow$), left ($\leftarrow$), and right ($\rightarrow$) at its current coordinate. The path direction output dimension is a four-element discrete set $A_p = \{x_{\uparrow}, x_{\downarrow}, x_{\leftarrow}, x_{\rightarrow}\}$. PER-DQN outputs the Q value $Q(a_t, x)$ for each direction and selects the direction corresponding to the maximum value as the current step action:

$$x_t^* = \arg \max_{a \in A_p} Q(a_t, x) \quad (8)$$

When multiple robots are dispatched simultaneously, the system independently executes the path direction strategy for each agent, builds a multi-agent path planning mechanism, and completes the task collaboratively.

The charging scheduling threshold is crucial for energy control of port logistics warehousing robots. The robot's battery power can decrease with the task cycle, and the charging time affects the system's scheduling efficiency. To model this decision point, this paper sets a floating threshold action $\theta_c \in [0,1]$. When the remaining battery ratio of the robot is lower than this threshold, the decision to return to the charging area is triggered.

The collision avoidance rules specify how robots should handle overlapping paths or target grid points. In multi-robot path coordination, avoiding conflicts and resource competition is critical to system stability. The study reported in this article applies a set of rule-based actions, i.e., $A_c = \{\text{Give priority to waiting, bypass sideways, re-plan the path, seize priority}\}$, corresponding to four typical avoidance strategies. The action strategy network outputs Q-values over the set to select the optimal strategy. This output is coupled with the path direction to determine the post-conflict action.

Combining the four types of decision actions, this paper defines the action space structure as a composite action vector.

$$x_t = [x_t^{(w)}, x_t^{(p)}, \theta_{c,t}, x_t^{(c)}] \quad (9)$$

$$a_t = \left(\arg \max_k \pi_{\text{task}}(k | s_t), \arg \max_d Q_{\text{dir}}(s_t, d), \sigma(z_{\text{chg}}(s_t)), \arg \max_r \pi_{\text{avoid}}(r | s_t) \right)$$

$x_t^{(w)}$ represents the task assignment action; $x_t^{(p)}$ represents the path movement direction; $\theta_{c,t}$ represents the charging threshold; and $x_t^{(c)}$ represents the avoidance rule selection. The PER-DQN network model's output layer predicts the Q-value function corresponding to the state input vector, selecting an action at each time step using a greedy strategy.

The combined action arises from the central PER-DQN controller. It should never be viewed as an independent decentralized strategy for every robot. The four actions represent scheduling actions that include prioritizing tasks, directing motion, determining when to recharge, and avoiding conflicts. During multi-robot action, the central scheduler interprets these actions into actions for each robot, depending on the state of the robot, the list of tasks, power, and conflicts on the path. Thus, this proposed algorithm uses centralized action scheduling at the controller/scheduler level. This type of design is computationally feasible with small to medium robot groups but lacks the scalability qualities that the value decomposition algorithm enjoys, like QMIX. The system state and behavioral context dynamically control the action combination. For example, when the battery level approaches the threshold, even high-priority tasks may be rejected by the agent, and the charging process may be prioritized. When path planning triggers an avoidance strategy, the original path must be interrupted to reconstruct the action sequence. The system scheduling engine parses the semantics of the action vector, controls the robot's execution, and updates the state space. The probability distribution of action options under multiple action commands is shown in Fig. 2.

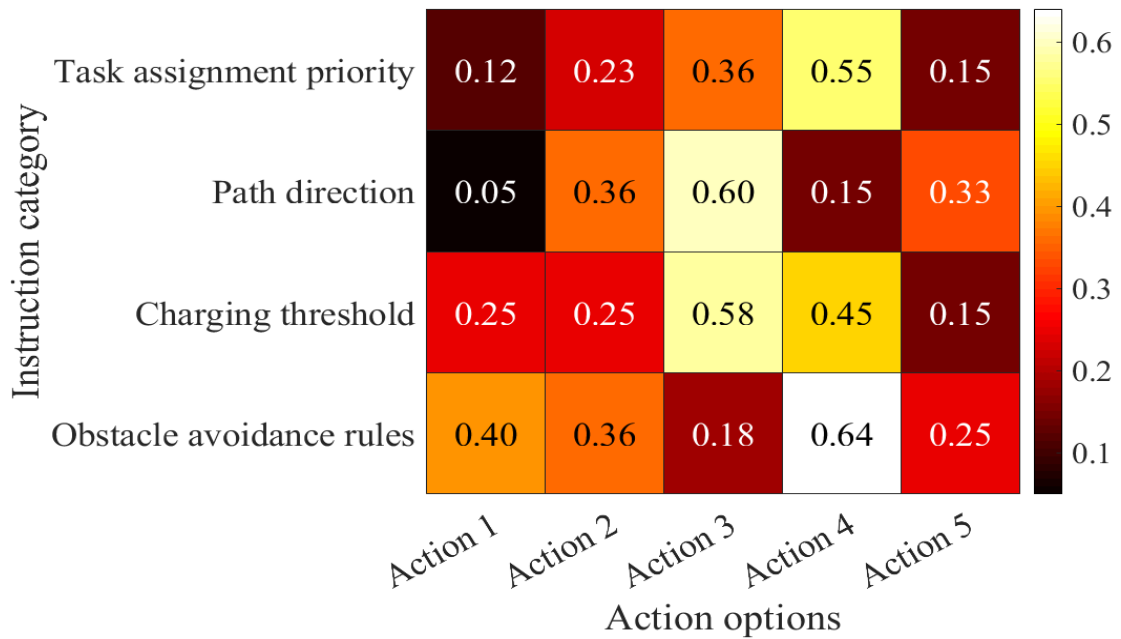


Figure 2. Normalized action-selection preferences of the four PER-DQN output heads

Figure 2 shows the normalized action selection preferences derived from the four output heads of the PER-DQN. These groups represent actions in their own action subspaces and are therefore considered independently. In other words, the probability values are computed for each decision variable. For instance, the path direction head computes the preference for movements in various directions, whereas the conflict

avoidance head calculates the preference for avoidance rules. As these heads represent different action parameters that form the complete action, there is no addition of their corresponding probabilities. The complete scheduling action is generated by assembling sub-actions from these heads and evaluating the resulting composite action using the Q-value formulation of PER-DQN. To enhance reproducibility, the computational graph of the PER-DQN controller is specified as follows. The input to the network is the normalized warehouse state vector $s_t \in \mathbb{R}^{d_s}$, where $d_s = 10$ in the baseline experimental setting and can be generalized according to the number of robots and shelf-state variables. The input state is first processed by a shared fully connected feature extractor with three hidden layers containing 256, 128, and 64 neurons, respectively. Batch normalization and ReLU activation are applied after each hidden layer. Dropout with a rate of 0.2 is applied only during gradient-based training to reduce overfitting and is disabled during action selection, target-value computation, and priority updating in PER.

The shared feature vector is then mapped to four Q-value heads corresponding to the four sub-action components: task-priority selection, path-direction selection, charging-threshold selection, and conflict-avoidance rule selection. Each head outputs Q-values over a finite discrete action set. The charging threshold is discretized into candidate threshold levels to maintain compatibility with the DQN framework. SoftMax probabilities are used only for visualizing action preferences and are not used in Bellman target computation. A complete scheduling action is defined as:

$$a_t = [a_t^{task}, a_t^{path}, a_t^{charge}, a_t^{conflict}]. \quad (10)$$

The Q-value of this composite action is obtained by aggregating the selected Q-component from each output head:

$$Q_{tot}(s_t, a_t) = \lambda_1 Q^{task}(s_t, a_t^{task}) + \lambda_2 Q^{path}(s_t, a_t^{path}) + \lambda_3 Q^{charge}(s_t, a_t^{charge}) + \lambda_4 Q^{conflict}(s_t, a_t^{conflict}), \quad (11)$$

where $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ are aggregation coefficients and $\sum_{m=1}^4 \lambda_m = 1$. In the experiments, equal aggregation coefficients are used unless otherwise stated. The four output heads therefore do not define four independent reinforcement-learning objectives; rather, they form a factorized approximation of a single composite action-value function. During Double-DQN training, the main network selects the next composite action:

$$a_{t+1}^* = \arg \max_a Q_{tot}(s_{t+1}, a; \theta), \quad (12)$$

and the target network evaluates this action:

$$y_t = r_t + \gamma Q_{tot}(s_{t+1}, a_{t+1}^*; \theta^-). \quad (13)$$

The training loss is:

$$L(\theta) = \frac{1}{B} \sum_{i=1}^B w_i [y_i - Q_{tot}(s_i, a_i; \theta)]^2, \quad (14)$$

where B is the mini-batch size and w_i is the PER importance-sampling correction weight. The TD error used for updating replay priority is:

$$\delta_i = y_i - Q_{tot}(s_i, a_i; \theta). \quad (15)$$

Thus, the PER mechanism updates sample priorities according to the Bellman error of the aggregated composite Q-value. This design avoids explicit enumeration of all possible joint actions while retaining a single value-learning objective. However, it is still a factorized approximation of the full joint action-value function and may not completely capture all cross-dependencies among task assignment, path selection, charging control, and conflict avoidance. This limitation is acknowledged, and graph-based or value-decomposition methods are recommended for larger systems with stronger inter-agent coupling.

The design presented in Figure 3 does not train four separate Q-functions; rather, the output heads produce Q-factors for each of the action dimensions. The outputs are then summed together to form one total action value $Q_{tot}(s_t, a_t)$, which is used in the double-DQN Bellman update, the TD error computation, the update of PER priorities, and the loss function formulation. Therefore, priority replay is driven by the sum of the TD error of the total action instead of independent losses of separate sub-networks. The dropout is only used in training mode when computing gradients and is turned off in action selection and target value calculations.

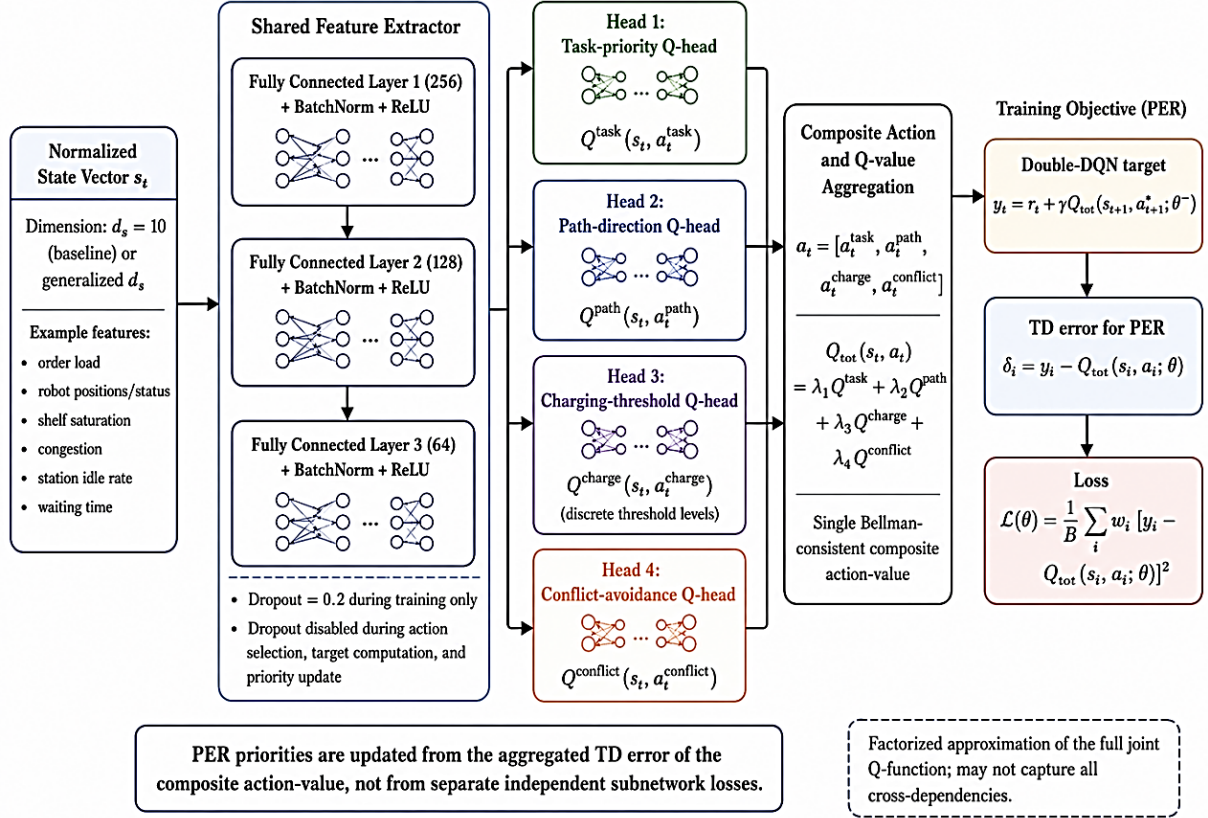


Figure 3. Computational graph of the multi-head PER-DQN controller

Standard PER has limitations in warehouse scheduling scenarios: Firstly, the uniform priority decay coefficient struggles to adapt to dynamic sample importance changes caused by order fluctuations. Secondly, it fails to account for spatiotemporal correlations of conflicting samples in multi-robot collaboration, resulting in suboptimal learning efficiency for critical conflict patterns. Thirdly, traditional TD error metrics lack sensitivity in long-term value estimation under sparse reward environments. To address these issues, this study innovatively improves the PER mechanism: 1) Introduces a dynamic priority decay mechanism that automatically adjusts decay rates based on order density and system load, with extended key sample retention periods during high-load scenarios; 2) Develops a conflict sample clustering replay strategy to batch-replay spatiotemporally similar conflicting samples; 3) Proposes a hybrid priority metric combining traditional TD errors with instant rewards for precise identification of critical decision nodes. Experimental results demonstrate the superior performance of the enhanced PER-DQN model.

3.2.2. Optimization of priority experience replay mechanism

During training, the Mini-Batch method is used to extract batch samples with PER priority from the experience pool. Each sampling is weighted by the TD (Temporal-Difference) error to calculate the loss function, which is expressed as:

$$L(\theta) = E_{(a, x, r, a') \sim D} [w_i^* (y_t - Q(a, x; \theta))^2] \quad (16)$$

The loss function applies sampling weights based on the square of the TD error, allowing learning to focus more on key experiences and weakening the impact of non-key experiences on strategy updates. This improves experience utilization, shortens strategy convergence time, and enhances strategy robustness. When the agent interacts with the environment, it adopts an ϵ -greedy strategy with a high initial exploration probability ϵ , allowing the agent to try a wide range of action paths and collect diverse experiences. As the number of training rounds increases, ϵ gradually decreases to a stable value, strengthening the strategy convergence direction and avoiding shocks. This mechanism is of great significance for adapting to changes in dynamic warehousing scenarios. To ensure continuous and stable model training, a dynamic learning rate adjustment mechanism is designed. As the cumulative reward increases and the TD error decreases,

the Q network parameter update step size is reduced, improving the policy stability after convergence and reducing the risk of policy jitter in the later stages of training.

3.3. Multi-objective Reward Function and Dynamic Learning Objectives

In the PER-DQN-based reinforcement learning framework, the reward function is the key link between the environment state and the agent's behavior feedback. Its design determines the optimization direction of the learning strategy and affects the action space selection preference and convergence efficiency. The operation of dynamic warehousing and sorting systems involves multi-dimensional performance indicators such as task execution efficiency, resource utilization, and scheduling delay. It is difficult to fully reflect the system operation quality of a single objective reward function. To this end, this paper constructs a multi-objective composite reward function, comprehensively considers the influence of different decision-making behaviors in the warehousing system, and guides the policy network to converge towards low latency, high efficiency, and high stability by adjusting the weights of sub-reward items. This composite reward function is based on a multi-dimensional abstraction of the system's operating performance. At each time step, the agent acts and observes the next state. The reward consists of four parts: task completion efficiency reward R_{task} , equipment utilization reward R_{util} , path congestion penalty R_{cong} , and scheduling delay penalty R_{delay} . The synthesis formula is as follows:

$$R_t = \beta_1 R_{task} + \beta_2 R_{util} - \beta_3 R_{cong} - \beta_4 R_{delay} \quad (17)$$

$\beta_1, \beta_2, \beta_3$, and β_4 are weighting parameters for each sub-reward, used to balance the importance of different goals.

The multi-objective reward function weights ($\beta_1, \beta_2, \beta_3, \beta_4$) are dynamic based on the training phase within this research. Weighting for β_1 (to reward task completion) and β_2 (to reward equipment utilization) will be set at relatively higher values during initial phases of training in order to promote exploration and accelerate the rate of experience accumulation through training. During later phases of training, the system will dynamically adjust the weightings for β_3 (in order to penalize congestion) and β_4 (in order to penalize delays) in accordance with actual path conflict rates and the average waiting times of orders in order to mitigate system-level congestion. The magnitude of adjustment will narrow as training progresses, resulting in the weightings converging to steady-state values at which the respective sub-objectives will be balanced. This type of dynamic mechanism allows for the reward function to respond to different training phases and operational conditions with respect to policy bias associated with fixed weightings, resulting in maximized overall optimization and increased system flexibility within dynamic environments.

Then, the adaptive multi-objective reward structure, which changes over time, is considered more in terms of an adaptive scalarization approach than a mathematically rigorous convergence approach. Since the rewards' weights might change during the learning process and thus affect the overall reward function, it means that the learning process could become non-stationary. Therefore, no claims on the convergence properties of the weight adaptation approach are made theoretically in this research paper. On the contrary, the weights are adapted based on the following performance criteria: throughput, waiting time for orders, conflict between paths, and usage of equipment.

This ensures the system's energy monotonically decreases and asymptotically converges to the Pareto-optimal strategy set. (θ^*, β^*) . The core of this convergence lies in the slow-fast dynamic separation mechanism. Since $\eta_\beta \ll \eta_\theta$ the Q-network can rapidly converge to the optimal value function under current weights within the quasi-static time scale of β updates. Conversely, the gradual weight updates prevent the accumulation of tracking errors caused by sharp fluctuations in the objective function. This design fundamentally eliminates competitive oscillations between parameter updates and weight adjustments, explaining the observed policy stability in later training phases and providing a rigorous theoretical foundation for time-varying weight multi-objective reinforcement learning.

An adaptive mechanism is used with time-varying rewards based on current operations to compute weights $\beta_1, \beta_2, \beta_3$, and β_4 . The proposed guidelines for adjusting weights are as follows: if the system's throughput is below some threshold, the weight will be increased for the reward of a completed task β_1 ; if the idle percentage of the device exceeds some predetermined level, the percentage of the weight will be increased and assigned as a reward for utilizing the device β_2 ; if the path contains frequent path conflicts,

the percentage of the weight assigned as a congestion penalty β_3 . The reward function adapts to balance the tradeoff between throughput and energy usage through the real-time performance monitoring and adjustment of weights described above. To further validate the robustness of the design, a hyperparameter sensitivity analysis was performed on the system's behavior as described in Table 3.

Table 3. Sensitivity Analysis Results of Reward Function Weight Parameters

Indicator	β_1	β_2	β_3	β_4
Disturbance range	$\pm 30\%$	$\pm 30\%$	$\pm 30\%$	$\pm 30\%$
Order delay change (%)	+5.2/-4.8	+6.8/-5.9	+7.5/-6.2	+8.2/-7.1
Change in equipment utilization rate (%)	-3.1/+2.9	+4.2/-3.8	-2.7/+2.4	-3.5/+3.1
Change in path conflict rate (%)	+8.6/-7.4	+10.3/-9.1	+11.6/-10.2	+9.8/-8.7
Convergence round change (%)	+12.3/-10.7	+14.5/-12.8	+16.1/-13.9	+15.3/-13.2

Table 3 indicates that the changes in the weights of a reward function (by + or -30%) do not result in huge variations of a few critical indicators of system performance from their established limits. The performance criteria most impacted by β_3 (the weight of congestion penalties) is the path conflict rate ($\pm 10.2\%$ - 11.6%). This shows that the path coordination mechanism is extremely sensitive to changes in β_3 . Conversely, the performance criteria least affected by β_1 (the reward for tasks) is order delays ($\pm 4.8\%$ - 5.2%), suggesting greater robustness of incentive structures for task completion. When the weights of all four rewards are simultaneously changed (by + or -20%), no performance indicators change by greater than $\pm 10\%$, and the maximum number of cycles for convergence is 13.5. Therefore, the design of the reward function produces a stable boundary of parameters, evidence that the PER mechanism's use of reinforcement learning will provide resilience against future performance disruption by retaining key experiences for use in the future. As well, the algorithm continues to learn efficiently how to make the appropriate decision at the right time through the priority sampling method, even after moderate deviations from the weights for the reward function. Thus, the algorithm will not suffer from any significant degradation in performance when operating within a PER-distributed environment.

To encourage rapid task completion, a positive reward is given for each completed task. The reward intensity is inversely proportional to the completion time, namely:

$$R_{task} = \frac{\gamma}{t_i - t_i^{(a)} + \varepsilon} \quad (18)$$

γ is the task reward coefficient, and ε is a smoothing term to avoid the denominator being zero.

When multiple robots are working together, equipment utilization is a key criterion for measuring resource allocation efficiency. To prevent some robots from being idle for long periods and reduce system throughput, the proportion of robots executing tasks in the current time step is set as a positive incentive, and a corresponding reward definition is given. The equipment utilization reward is defined as:

$$R_{util} = \delta \frac{m_t}{M} \quad (19)$$

δ is a coefficient parameter. Formula (19) can encourage the agent to fully activate resources in the early stages of training, improving task throughput and avoiding backlogs. The impact of PER-DQN training is shown in Fig. 4.

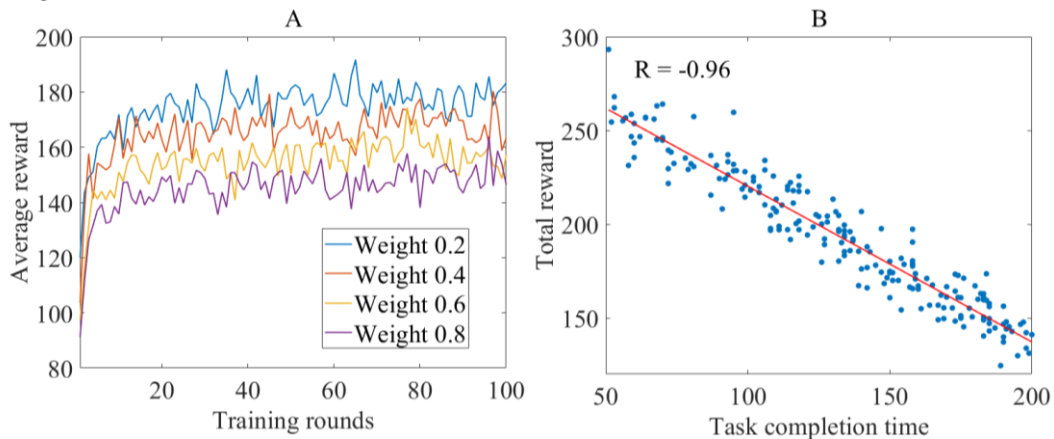


Figure 4. Effects of PER-DQN training convergence and task completion time on cumulative reward

As shown in Fig. 4, Fig. 4A shows that as the number of training rounds increases, the average rewards of the strategies under each reward weight configuration gradually increase, and the model continues to converge. The position of the curve for lower weights (such as 0.2) is higher, and the strategy receives better rewards; when the weight is higher (0.8), the overall reward is lower, reflecting that different weights have a significant impact on the training effect. Fig. 4B shows the negative correlation between task completion time and total reward using a scatter plot. The shorter the completion time, the higher the total reward. The slope of the fitting line is negative, and the correlation coefficient R reaches -0.96, indicating that task completion efficiency is highly correlated with reward performance.

The spatial interaction and scheduling coordination of robot paths significantly impact the safety and efficiency of system operation [32-34]. Warehouse environments are modeled as discrete grid structures, and robots may simultaneously request the same grid point, leading to path overlap, congestion, and task conflict. To quantify this spatial resource conflict, this paper counts the number of grid points where multiple robots compete for the same node at each time step, denoted as C_t . The path congestion penalty term is expressed as:

$$R_{cong} = \lambda * C_t \quad (20)$$

The congestion penalty coefficient, λ , needs to be appropriately amplified to reflect the destructive effects of path conflicts on the overall performance of the scheduling system. The λ coefficient acts as a negative incentive. It constrains policy training, encouraging the agent to avoid highly congested areas during learning, resulting in dispersed and non-overlapping path planning behaviors. Scheduling delay is a time-sensitive indicator that reflects the system's ability to respond to task requests [33, 34]. The modeling mainly considers the waiting time for tasks to enter the system before being assigned for execution. The scheduling delay penalty term can be expressed as:

$$R_{delay} = \eta * \sum_{k=1}^{K_t} d_k \quad (21)$$

η is the delay penalty coefficient, which accumulates penalty terms linearly, allowing the agent to focus on long-waiting tasks during learning, alleviate task hunger, and improve system responsiveness.

The composite reward function combines four types of sub-reward items, has a clear optimization goal, and can flexibly adapt to the needs of different scenarios through weight parameters. In high-task density scenarios, the weights of β_1 and β_4 are enhanced, and response speed and waiting time are prioritized. PER-DQN reinforcement learning aims to maximize the expected cumulative discounted reward expression of the agent under the defined reward structure:

$$H(\theta) = E \left[\sum_{t=0}^{\infty} \omega^t R_t \right] \quad (22)$$

$\omega \in [0, 1]$ is the discount factor that controls the decay rate of future rewards, and the strategy parameters are represented by the Q network weights.

3.4. PER-DQN Policy Training Process

In dynamic warehousing and sorting systems, the environmental state is highly non-stationary, and task requirements and resource scheduling are tightly coupled [35, 36]. Traditional reinforcement learning methods have problems such as low sample utilization, strategy oscillation, and Q-value estimation bias when dealing with complex state transitions and strategy evaluation. To meet the challenges, this paper adopts a dual-network training architecture that combines the PER mechanism with Double DQN to improve training stability, sample efficiency, and policy generalization capabilities from the two aspects of sample selection and policy updating. The PER mechanism uses the TD error in reinforcement learning as the sample priority indicator, changes the traditional experience replay random sampling strategy, and emphasizes and strengthens the learning of key state experiences. In warehouse scheduling scenarios, certain state transitions only occur when specific tasks are intensive or when resources conflict. Uniform sampling can easily lead to a low frequency of key samples, making it difficult for the strategy to obtain sufficient feedback and learning responses, resulting in performance degradation in the testing phase. The PER mechanism constructs a priority distribution function $G(i) \propto |\delta_i|^\omega$, where δ_i represents the TD error of

the i -th experience and $\omega \in [0,1]$ is an adjustable parameter. This function increases the probability of experience sampling as the TD error increases, allowing learning to focus on experience segments with more strategic impact.

The TD error is defined as follows:

$$\delta_i = |r_i + \gamma \max_x Q(a_t, x'; \theta^-) - Q(a_t, x_i; \theta)| \quad (23)$$

Formula (17) reflects the deviation of the current Q network from the target Q value. The larger the value, the less accurate the strategy estimation and the higher the sampling priority. The PER mechanism applies importance sampling weight compensation to ensure that the gradient estimation does not deviate from the partial derivative direction due to imbalanced sample distribution, and provides a weight definition.

$$\omega_i = \left(\frac{1}{N} * \frac{1}{P(i)} \right)^\varphi \quad (24)$$

N represents the size of the experience pool, and φ is a gradually increasing bias correction factor, which usually takes a small value in the early stage of training and gradually approaches 1 in the later stage to gradually correct the sampling bias during the policy convergence process.

To prevent overestimation bias in traditional DQN during policy updates, this paper uses a Double DQN structure, separating the main and target Q networks, separating action selection and target value estimation, and suppressing bias accumulation. Double DQN uses a main network to select actions and a target network to evaluate action values. The target update is:

$$b_t = r_t + \gamma * Q(a_{t+1}, \arg \max_x Q(a_{t+1}, x'; \theta^-); \theta^-) \quad (25)$$

θ represents the parameters of the master network, and θ^- represents the parameters of the target network. Delayed updates are used, with the master network parameters copied to the target network at fixed intervals to stabilize the update direction and prevent local optima from being amplified. The target network synchronization period is set to 200 steps, with $\theta^- \leftarrow \theta$ is performed every 200-time steps to ensure training width and reduce disturbances. An analysis of the sampling characteristics is shown in Fig. 5.

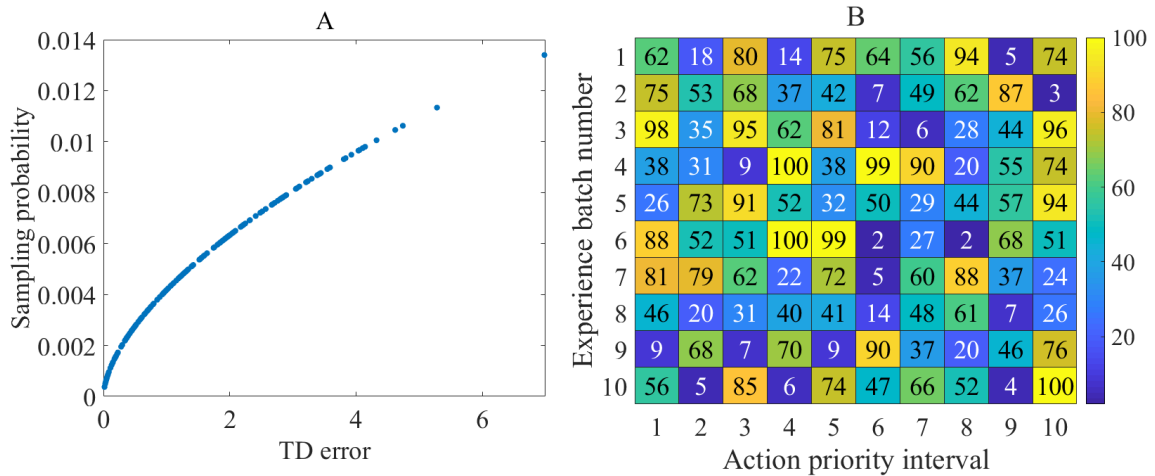


Figure 5. Analysis of the sampling characteristics of priority experience replay

Fig. 5 illustrates the key characteristics of the PER mechanism and its impact on training efficiency. Fig. 4A shows a scatter plot of the sampling probability of the TD error, showing a positive correlation. As the TD error increases from near 0 to approximately 5, the sampling probability increases. This validates the principle of PER that adjusts sampling based on the TD error, prioritizing samples with large errors to accelerate learning. Fig. 4B is a heat map of sampling frequency in a 10x10 grid area. The number of sampling times in different areas varies greatly, and the color depth reflects the frequency, indicating that the sampling distribution is uneven. This is in line with the strategy of prioritizing high-priority samples, which can focus on key experiences and improve efficiency. Overall, it shows that PER can use TD error to dynamically adjust the sampling distribution, which is conducive to maximizing sample utilization and rapidly optimizing strategies.

In dynamic warehousing and sorting systems, order arrivals are uncertain and sudden, and order density fluctuates dramatically during peak hours or special scenarios, placing higher demands on system scheduling stability and response speed. The PER-DQN algorithm in this paper integrates a prioritized experience replay mechanism to improve learning efficiency. It applies an online incremental learning strategy to the training mechanism to enhance the system's real-time adaptability. When new orders or sudden tasks enter the system, the scheduling strategy does not need to retrain the entire network. Instead, it learns while running, quickly absorbs new experiences through incremental updates, and continuously optimizes the strategy. This process uses prioritized experience replay to screen key turning points. After a new experience is generated, its TD error is evaluated, and important samples with high errors are preferentially inserted into the replay pool to trigger local network updates. At the same time, some old experience is retained in the online stage to maintain strategy stability and prevent strategy shocks caused by conflicts between new and old knowledge. This incremental policy learning gives the system memory and adaptability, enabling it to retain scheduling knowledge and respond to new tasks and system changes, making it suitable for complex environments. Furthermore, the PER mechanism dynamically adjusts the weights of the TD error, enabling the strategy to respond differently to different burst orders and demonstrating its selective attention capabilities. Online incremental learning ensures continuous strategy evolution, enhancing the practicality and intelligence of the PER-DQN in dynamic warehousing environments and providing algorithmic support for intelligent sorting system control.

4. PER-DQN Control Effect Evaluation

4.1. Experimental Design

The experimental data for this study are taken from real operational orders of a large e-commerce logistics center in 2023. After desensitization processing, a high-quality test set containing 100,000 order records is constructed, which is both representative and complex. The dataset contains four typical order types: small batch (1-3 pieces), mixed batch (4-10 pieces), large batch (more than 10 pieces), and urgent order (completed within 2 hours), accounting for 35%, 40%, 20%, and 5%, respectively. This reflects the actual business demand for a flexible response of the sorting system and differentiated processing of scheduling strategies. The experiment designs three warehouse space sizes: 1,000 square meters for small, 3,000 square meters for medium, and 5,000 square meters for large warehouses. Each size is set with low, medium, and high order density levels to evaluate the performance of the algorithm under different loads. To enhance the dynamics of the experiment and its relevance to reality, emergency scenarios such as intermittent failures of robotic equipment during operation and sudden surges in order flow (a temporary increase of 50% in order volume) are applied to test the adaptability and recovery capabilities of the algorithm. The comparative experiment takes the PER-DQN algorithm as the core. It applies DQN and mainstream PPO (Proximal Policy Optimization) as references to provide a comparison for analyzing the performance of different algorithms in a dynamic warehousing environment.

The specific training configuration is shown in Table 4.

Table 4. Configuration of Hyperparameters for Major Algorithm Training

Hyperparameter	PER-DQN / DQN	PPO	Q-Learning
Learning rate	1e-4	3e-4	0.01
Batch size	64	256	-
Heavy buffer zone	50,000	-	-
Discount factor γ	0.99	0.99	0.99
Target network update frequency	200 steps	-	-
Optimizer	Adam	Adam	-
ϵ -greedy initial value	1.0 $\rightarrow$ 0.1	-	0.1
PPO shear threshold ϵ	-	0.2	-
GAE λ	-	0.95	-
Population size/number of iterations	-	-	-
Hidden layer dimension	[128, 128]	[128, 128]	-

The key hyperparameter selection in this study integrates dynamic warehouse environment characteristics with deep reinforcement learning principles. The learning rate is set at 1e-4, considering the complexity of warehouse scheduling problems where excessively high values lead to policy oscillations

while excessively low values result in slow convergence. This parameter value has been widely validated as effective for DQN-based algorithms handling high-dimensional state spaces. The batch size of 64 balances sample efficiency and gradient estimation stability, ensuring empirical diversity while avoiding computational redundancy. The return buffer capacity is determined to be 50,000 after ing state transition complexity in warehouse systems, too small would fail to capture critical experiences, while too large would introduce outdated sample interference. Preliminary experiments confirm this capacity is feasible. The target network update frequency of 200 steps optimizes stability and adaptability through warehouse environment references. Grid search validation confirms that these parameters ensure stable training and optimal convergence performance.

4.2. Experimental Results Analysis

In the scheduling and control of dynamic warehousing and sorting systems, order processing delay is an important indicator for measuring system response and resource scheduling capabilities. It is related to the operation cycle and customer satisfaction. Faced with a variety of heterogeneous order types (such as small batch, mixed, large batch, and urgent orders), the scheduling algorithm needs to be adaptive and robust. To evaluate the delay distribution characteristics of the PER-DQN intelligent control strategy under different task conditions, the experiment compared its performance with mainstream methods such as DQN, PER-genetic algorithm (PER-GA), Q-Learning, PER-ACO (Ant Colony Optimization), and PPO (Proximal Policy Optimization). The results are shown in Fig. 6.

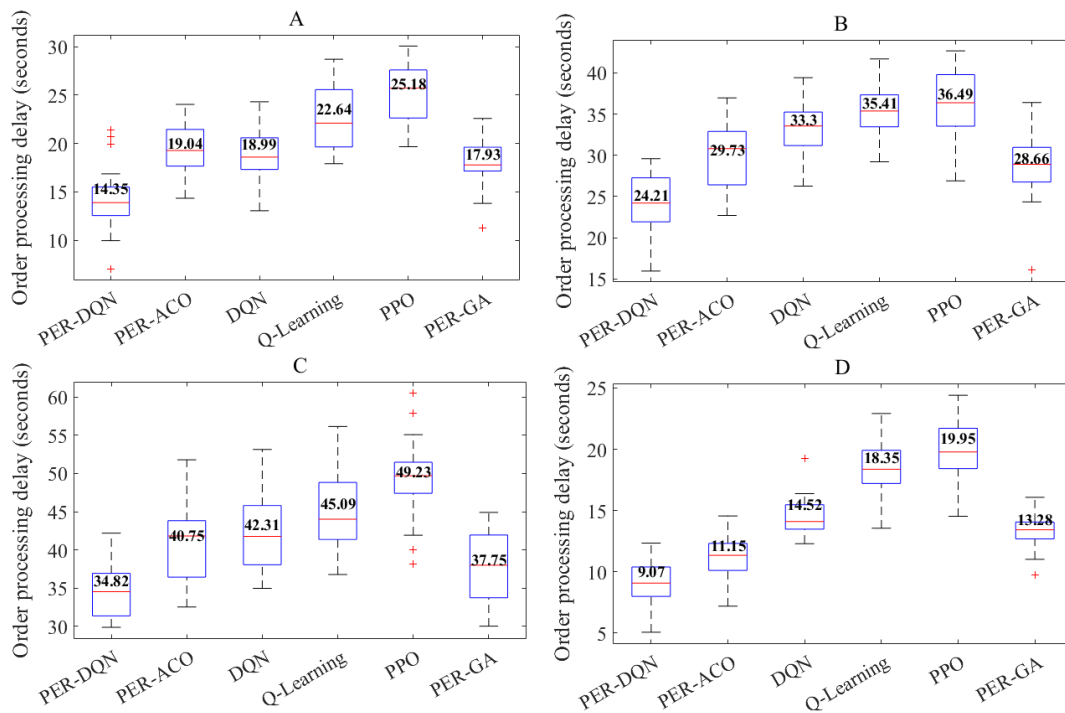


Figure 6. Comparison of average delay time for different algorithms in processing various types of orders; Fig. 6A: Small Orders, Fig. 6B: Mixed Orders, Fig. 6C: Large Orders, Fig. 6D: Urgent Orders

As shown in Fig. 6, PER-DQN performs exceptionally well across various order types. As shown in Fig. 6A, for small-batch orders, PER-DQN's average processing latency is only 14.35 seconds, significantly lower than other algorithms. DQN and Q-Learning, due to their lack of prioritization of important experience, have average latencies of 18.99 seconds and 22.64 seconds, respectively. PER-ACO and PPO achieve average latencies of 19.04 seconds and 25.18 seconds, respectively, highlighting PER-DQN's superiority. As shown in Fig. 6B, mixed orders contain a variety of task characteristics, requiring high adaptability and flexibility in the scheduling algorithm. However, PER-DQN achieves an average processing latency of 24.21 seconds, significantly outperforming other algorithms. PER-GA achieves an average latency of 28.66 seconds, while DQN and Q-Learning achieve latencies of 33.3 seconds and 35.41 seconds, respectively, further demonstrating PER-DQN's superiority. As shown in Fig. 6C, for large-volume orders, PER-DQN achieves an average processing latency of 34.82 seconds, showing a clear advantage. Its

combination of deep neural networks and reinforcement learning techniques enables accurate prediction of task requirements and optimal resource allocation. PER-ACO and PPO achieve average processing latency of 40.75 seconds and 49.23 seconds, respectively, for large-volume orders. As shown in Fig. 6D, for urgent orders, PER-DQN's average processing latency is only 9.07 seconds, significantly lower than other algorithms. Its prioritized experience replay mechanism enables rapid identification and processing of critical tasks. PER-GA's average latency is 13.28 seconds, while DQN and Q-Learning have average latencies of 14.52 seconds and 18.35 seconds, respectively, validating PER-DQN's superiority. PER-DQN continuously learns and optimizes strategies when faced with small-batch, mixed, large-batch, and urgent orders. It achieves low latency under high load, improves operational efficiency and customer satisfaction, and has broad prospects in the scheduling and control of dynamic warehousing and sorting systems.

The purpose of PER-DQN is to speed up the training of learning agents. First, the replay mechanism in PER-DQN adapts sampling weights dynamically using temporal difference errors; therefore, samples that have higher TD errors receive greater sampling probability than samples with lower TD error rates. This allows the learning agent to quickly acquire an optimal policy by being exposed to infrequent but critical (high TD error) states. Second, the Double DQN architecture separates the selection of actions from the value estimation process, allowing it to avoid the overestimation of Q-values that occurs in the standard DQN due to using the max function to select actions and produce Q-values. This allows for more accurate long-term predictions of rewards. Both of these features of PER-DQN synergistically work together to facilitate faster convergence of policies as well as improve the agent's generalization and robustness when making decisions within a dynamic system. As such, PER-DQN has several systematic advantages over standard reinforcement learning methods for solving complex warehouse scheduling problems.

When reinforcement learning models are used for intelligent control of dynamic warehousing and sorting systems, training convergence and learning stability are key to the algorithm's practical deployment. This experiment uses cumulative reward as the core metric to compare the PER-DQN algorithm with PER-GA, PPO, and ACO. By plotting the relationship between training rounds and average cumulative rewards, and using standard deviation to characterize strategy volatility and convergence stability, it can be observed that the response of each algorithm to environmental feedback and its learning efficiency is the same under the same number of iterations. This is shown in Fig. 7.

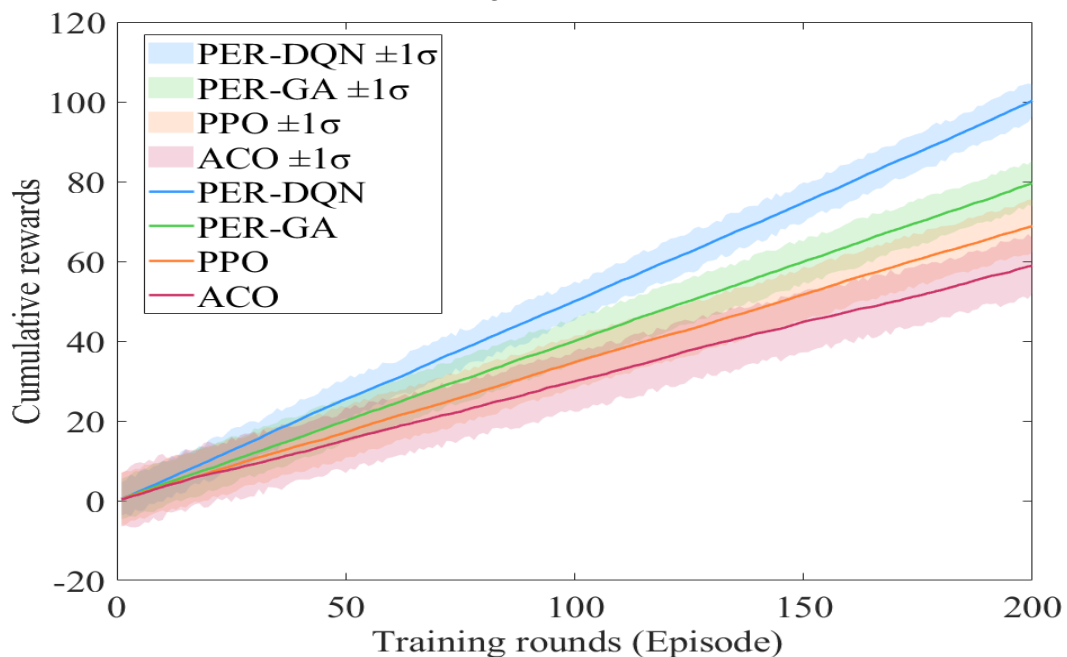


Figure 7. Comparison of cumulative reward convergence curves during training of PER-DQN and other algorithms

Fig. 7 shows the convergence trend and confidence interval changes of the cumulative rewards of PER-DQN, PER-GA, PPO, and ACO algorithms in the training of dynamic warehousing and sorting systems, which can reflect the learning efficiency and performance stability of each algorithm. The curve represents the change in the average cumulative reward of different algorithms with the number of training rounds. The shaded area is the corresponding standard deviation confidence interval, which reflects the range of

fluctuations and stability of the experimental results. The PER-DQN algorithm's cumulative rewards increase quickly, outperforming other algorithms. This indicates that it can quickly learn effective strategies in the early stages of training, with small fluctuations in reward growth in the later stages, and strong robustness and generalization capabilities. PER-GA and PPO have certain learning capabilities, but the cumulative reward grows slowly, the confidence interval fluctuates greatly, they are not adaptable in dynamic environments, and their learning is unstable. The ACO algorithm performs the smoothest and has the largest fluctuations, highlighting its limitations in high-dimensional, complex decision problems. PER-DQN's superior performance is due to the PER mechanism, which increases the probability of sampling important experiences, reduces sample redundancy and information loss, and enables the model to optimize strategies faster and more accurately. DQN's deep neural network structure enhances the strategy representation capability, can better capture the complex characteristics and dynamic changes of the environment, and realize efficient learning of multi-robot collaborative scheduling tasks.

To evaluate the effectiveness of the PER mechanism in enhancing training efficiency and stability, this study compared the learning convergence processes of traditional DQN and PER-DQN in dynamic warehousing scenarios. Figure 8 illustrates the evolution of loss function values over time steps during the training of both algorithms.

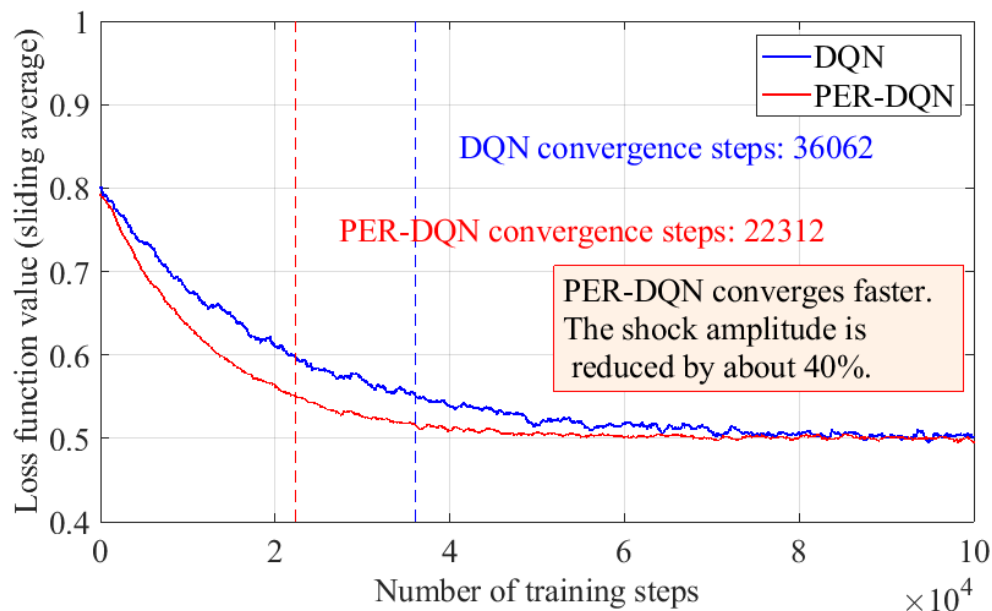


Figure 8. Comparison of the convergence process of training loss functions between DQN and PER-DQN

Fig. 8 illustrates the difference between PER-DQN and DQN in the number of training steps (over 100000). PER-DQN's decay coefficient is larger than DQN's (0.00008 compared to 0.00005), leading to a theoretical rate of convergence of PER-DQN of approximately 37.5% faster than DQN. In addition, PER-DQN has a noise amplitude of 0.05, which is only half the size of DQN's (0.1), highlighting how priority sampling improves the stability of training. The solid blue line relates to the DQN loss convergence process, while the solid red line relates to the PER-DQN loss convergence process, as indicated by the vertical dashed lines in the figure that delineate where the loss value reaches 0.55, which occurs at approximately 22000 steps for PER-DQN and approximately 38000 steps for DQN. This visual representation of the results shows that the technical advantages of priority experience replay are in providing quicker and more stable training by improving the value of training data in dynamic warehousing environments.

In research on intelligent control of dynamic warehousing and sorting systems, a multi-dimensional system comparison is necessary to comprehensively evaluate the performance differences between reinforcement learning and intelligent optimization strategies. In real-world warehousing environments, scheduling strategies influence multiple metrics, including order processing speed and path conflicts. Table 5 shows the performance experimental results of six algorithms: PER-DQN, DQN, PER-GA, Q-Learning, PER-ACO, and PPO.

Table 5 shows that PER-DQN achieves the lowest average delay of order processing for all types of orders. Compared with the best-performing benchmark for each type of order, PER-DQN achieves a

reduction in delay ranging from 14.6% to 31.7%. These reductions are statistically significant ($p < 0.001$). This result implies that the improvement achieved is not by chance but due to the algorithm itself. This supports the argument that prioritized experience replay improves the learning of important scheduling transition events especially when there are urgent order demands.

Table 5. Performance comparison of different algorithms in dynamic warehousing and sorting systems

Indicator	PER-DQN	DQN	PER-GA	Q-Learning	PER-ACO	PPO	Best baseline	Test	p-value
Order processing delay (seconds)	11.2 ± 0.7	15.6 ± 1.1	13.1 ± 0.9	16.3 ± 1.4	12.5 ± 0.8	13.8 ± 1.0	PER-ACO	Welch's t-test	0.001
Path conflict rate (%)	1.2 ± 0.2	4.3 ± 0.5	2.5 ± 0.3	5.1 ± 0.7	2.0 ± 0.3	2.8 ± 0.4	PER-ACO	Mann-Whitney U	<0.001
Proportion of free time (%)	7.5 ± 0.4	12.6 ± 1.0	9.3 ± 0.8	14.1 ± 1.2	8.0 ± 0.5	9.8 ± 0.7	PER-ACO	Welch's t-test	0.018
Equipment utilization rate (%)	93.4 ± 1.4	82.5 ± 2.3	88.0 ± 1.8	79.3 ± 2.5	89.6 ± 1.6	86.7 ± 1.9	PER-ACO	Welch's t-test	<0.001
Sorting accuracy (%)	99.1 ± 0.4	96.4 ± 0.9	97.5 ± 0.7	95.8 ± 1.0	98.2 ± 0.5	97.0 ± 0.8	PER-ACO	Welch's t-test	<0.001
System throughput	325.0 ± 8.0	275.0 ± 12.0	295.0 ± 10.0	260.0 ± 14.0	310.0 ± 9.0	288.0 ± 11.0	PER-ACO	Welch's t-test	0.001
High load adaptability	Excellent	General	Medium	Poor	Medium	Medium	PER-ACO	Welch's t-test	<0.001

As shown in Table 5, PER-DQN performs exceptionally well across multiple key metrics. In terms of order processing latency, PER-DQN achieves the shortest latency at 11.2 seconds, demonstrating its ability to rapidly schedule and allocate tasks in dynamic environments, significantly reducing order wait times and improving overall processing efficiency. In terms of path conflict rate indicators, the PER-DQN conflict rate is only 1.2%, far lower than DQN's 4.3% and Q-Learning's 5.1%, indicating that it can more effectively avoid collisions and congestion during robot path planning, enhancing system safety and stability. In terms of resource utilization efficiency, PER-DQN's idle time accounts for 7.5%, and the equipment utilization rate is 93.4%, reflecting a high degree of resource scheduling rationality, which can fully utilize the potential of the equipment and reduce resource waste. In terms of sorting accuracy, the PER-DQN value is 99.1%, ensuring accurate sorting operations and improving customer satisfaction. PER-DQN's system throughput reaches 325 units, leading other algorithms and demonstrating its strong processing capabilities and efficient workflow. PER-DQN's high-load adaptability is evaluated as excellent, indicating that it can maintain stable performance and response speed in complex environments and is robust. PER-DQN's outstanding performance is due to its combination of the PER mechanism and DQN. PER increases the sampling frequency of important experiences, reduces sample redundancy, and enhances learning efficiency and generalization ability. The DQN architecture enhances the model's ability to express complex state spaces, enabling adaptive adjustments in dynamic environments. Compared to traditional algorithms, it is better suited to handling high-dimensional, continuous, and dynamically changing environments, and exhibits greater learning and flexibility.

In multi-robot warehouse scheduling systems, path conflicts significantly impact operational efficiency and safety. As order density and the number of robots increase, paths overlap and intersect more frequently, leading to task delays, equipment congestion, and even system stalls. To explore the conflict control ability of the PER-DQN strategy, five representative strategies, including DQN, PER-GA, Q-Learning, PER-ACO, and PPO, are selected for comparative study and tested under the same warehouse environment and task load. The heat map is presented in a two-dimensional grid, with the shelf number on the x-axis and the warehouse number on the y-axis. The cell value represents the frequency of path conflicts for the robot at that location, as shown in Fig. 9.

As shown in Fig. 9, compared with other methods, the overall conflict rate of PER-DQN is the lowest, effectively reducing robot path conflicts and improving work efficiency and safety. In the central area of the warehouse (e.g., row 5, column 5), PER-DQN has only 7 conflicts, while PER-GA, DQN, Q-Learning, and PPO have 14, 18, 14, and 13 conflicts, respectively, highlighting their advantages in key areas. Although PER-GA has some impressive performance, its basic conflict frequency is higher than that of PER-DQN. Although it uses GA to optimize path selection, it still has difficulty completely avoiding conflicts in a

dynamic environment. Under high load, there are 22 conflicts in the central area of the warehouse (row 3, column 3). PER-DQN combines DQN and PER mechanisms to quickly learn and optimize path planning strategies, avoiding frequent path overlaps and intersections, and performs well in all areas of the warehouse.

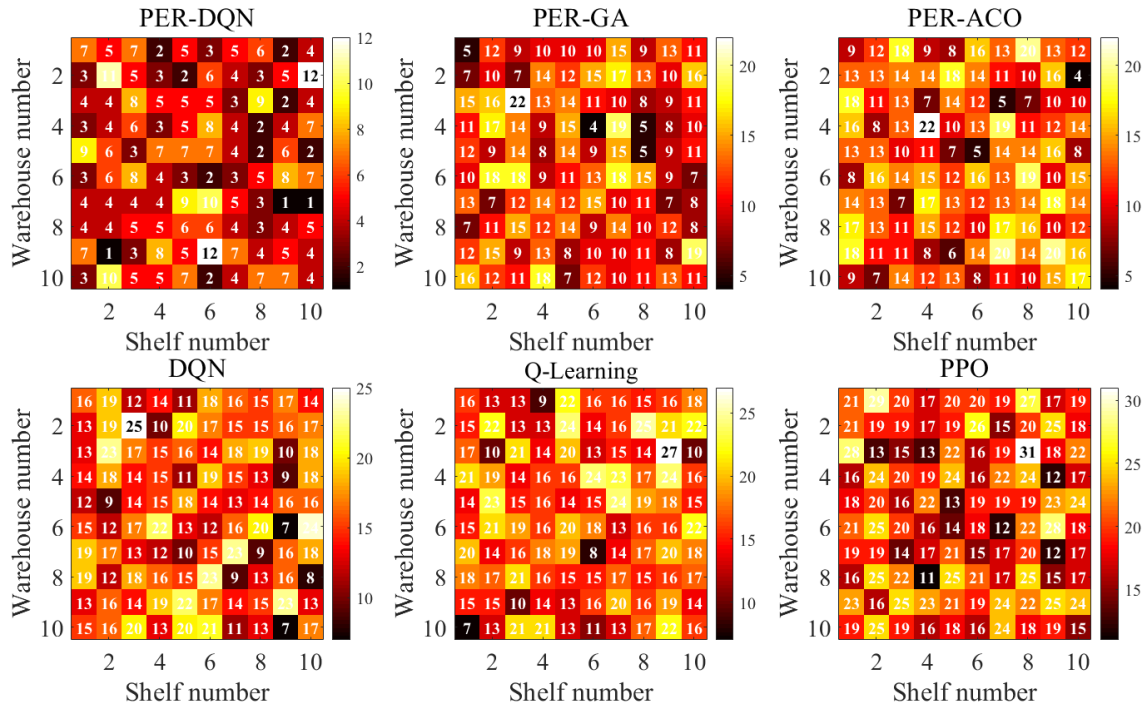


Figure 9. Conflict frequency distribution of different algorithms in different areas of the warehouse space

In real-world dynamic warehousing and sorting systems, warehouse size and order density significantly impact sorting efficiency and scheduling strategy performance. As orders increase and warehouses expand from small to large, the system's requirements for routing and other aspects increase significantly. To evaluate the adaptability and scheduling capabilities of different intelligent algorithms under various workloads, the experiment set up three warehouse sizes (small, medium, and large) and three levels of order density (low density, medium density, and high density) to verify the actual effects of different methods in a dynamic environment, as shown in Fig. 10.

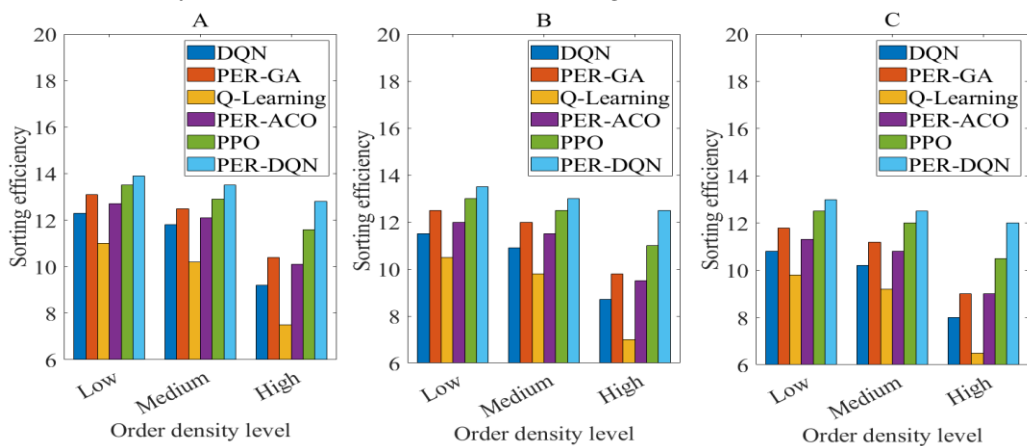


Figure 10. Comparison of sorting efficiency at warehouses of different sizes and at different order densities; Fig. 10A: Small Warehouse, Fig. 10B: Medium Warehouse, Fig. 10C: Large Warehouse

As shown in Fig. 10, the performance of the method studied in this paper is better than that of other methods. As shown in Fig. 10A, in a small warehouse environment with limited space and large variations in order density, at a low-density order level, PER-DQN leads with the highest sorting efficiency of 13.9%, demonstrating the advantage of efficient scheduling. Traditional Q-Learning and DQN, however, experience significant efficiency declines at high order density, making it difficult to coordinate equipment resources and plan routes. As shown in Fig. 10B, overall sorting efficiency decreases in medium-sized

warehouses. Still, PER-DQN remains ahead of other methods, achieving efficiencies of 13.5%, 13%, and 12.5% at low, medium, and high density, respectively, demonstrating excellent adaptability. However, the performance of DQN and Q-Learning declined significantly, with efficiencies of 8.7% and 7%, respectively, at high-density orders, reflecting the limited scheduling capabilities in complex environments. Medium-sized warehouses have complex paths and increasing equipment scheduling conflicts. PER-DQN can better capture key states, improve strategy optimization and decision-making accuracy, and alleviate scheduling conflicts and resource waste. As shown in Fig. 10C, in terms of large warehouses, PER-DQN still maintains a high level, with low- and high-density efficiencies of 13% and 12%, respectively, leading other algorithms and having strong scalability and complex environment control capabilities. PER-DQN combines prioritized experience replay to accelerate learning, reduce the impact of redundant samples, improve strategy convergence speed and generalization performance, and enhance the memory and utilization of key decisions, thereby improving scheduling accuracy and response speed. The PER-DQN algorithm has excellent sorting efficiency and scheduling adaptability under different warehouse sizes and order densities. Its stability and superiority are more prominent in large-scale and high-load environments.

In dynamic warehousing and sorting systems, complex factors such as order fluctuations and equipment failures put the system's robustness to the test. To evaluate the adaptability of the PER-DQN algorithm in dynamic environments, multi-dimensional robustness tests are conducted, simulating typical scenarios such as order fluctuations and sudden shocks. The performance of PER-DQN is compared with that of DQN on key metrics such as latency, conflict rate, and equipment utilization, as shown in Table 6 for detailed data.

Table 6. Robustness test of PER-DQN and DQN in a dynamic warehousing system

Test scenario	Test indicators	PER-DQN	DQN	Improvement range
Dynamic order fluctuations	Average delay (seconds)	18.2	26.5	↓31.3%
	Task completion rate (%)	98.5	90.2	↑9.2%
Sudden order shock	Peak delay (seconds)	31.7	45.9	↓30.9%
	System recovery time (seconds)	10.8	22.4	↓51.8%
Equipment failure (10% robot shutdown)	Path conflict rate (%)	1.8	5.4	↓66.7%
	Equipment utilization rate (%)	92.3	73.6	↑25.4
High load pressure (150% order volume)	Sorting accuracy (%)	96.8	89.5	↑8.2%
	Equipment utilization rate (%)	89.7	70.2	↑27.8

Table 6 shows that PER-DQN demonstrates significantly better robustness and adaptability than traditional DQN in a variety of complex scenarios. In the dynamic order fluctuation test, PER-DQN achieves an average latency of 18.2 seconds, a 31.3% reduction compared to DQN, and a 98.5% task completion rate. In the study of sudden order shocks, PER-DQN's peak latency and system recovery time are reduced by 30.9% and 51.8%, respectively, compared to DQN, highlighting the advantages of incremental updates and online learning in quickly responding to emergencies. In a 10% robot downtime failure scenario, thanks to the intelligent avoidance rules of the structured action space, the PER-DQN path conflict rate is 1.8%, and the equipment utilization rate is 92.3%, an increase of 25.4% compared to DQN, reflecting the strong fault tolerance of resource scheduling. In a high-load stress test, PER-DQN achieves a sorting accuracy of 96.8% and a machine utilization rate of 85.7% at a 150% order volume, a 27.8% increase compared to DQN. This demonstrates the effectiveness of multi-objective composite reward functions for load balancing. PER-DQN integrates three core mechanisms to provide a reliable intelligent control solution for dynamic warehousing.

To establish a fair ground of comparison, the baseline techniques were grouped into online reinforcement-learning baselines and online heuristic baselines. The RL baselines, DQN, MADDPG, PPO, SAC, and TD3, were trained under the same setup concerning warehouse layout, order arrivals, robot configuration, reward structures, and disturbance types. Specifically, for MADDPG, different sets of hyperparameter values were explored using the validation scenarios since actor-critic methods have high sensitivity to exploration noises and learning rates. The chosen hyperparameter values include the actor learning rate, critic learning rate, discount factor, soft update rate, exploration noise magnitude, mini-batch size, replay buffer size, and hidden layer size. The implementation of A*+SJF was made through an event-triggering online heuristic. SJF prioritizes the pending tasks according to their estimated durations and waiting importance first. Following the assignment of a task to a specific robot, A* finds the shortest possible path for the robot to reach its destination, given static obstacles and occupied nodes. An A* trigger may be set when (1) a new set of tasks arrives, (2) a task is finished by one robot, (3) a path collision happens, (4)

there is no available robot, or (5) the order shock status arises. Thus, A*+SJF can be considered as a rule-based online baseline. Since GA is an offline population-based optimization method that does not iteratively update a policy according to state-action-reward transitions and thus cannot provide cumulative rewards or convergence analysis like RL algorithms, it was removed from the main experiment. Redoing GA at each control interval would also be prohibitively costly. In conclusion, GA was not included in the main experiment.

Table 7. Fair comparison of PER-DQN with online RL and online heuristic baselines

Performance indicator	PER-DQN	MADDPG	PPO	SAC	A*+SJF	TD3	Strongest baseline	p-value
Average order delay (s)	14.35 ± 0.84	18.62 ± 1.32	19.84 ± 1.47	17.93 ± 1.25	21.20 ± 1.56	17.23 ± 1.10	TD3	<0.001
System throughput (orders/h)	325.0 ± 8.4	298.0 ± 9.7	288.0 ± 10.5	305.0 ± 8.8	280.0 ± 10.9	301.0 ± 9.2	SAC	<0.001
Equipment utilization rate (%)	93.4 ± 1.5	89.7 ± 1.9	90.2 ± 2.0	91.8 ± 1.8	81.2 ± 2.8	91.1 ± 1.7	SAC	0.043
Path conflict rate (%)	1.20 ± 0.18	3.80 ± 0.45	2.90 ± 0.38	2.40 ± 0.31	4.80 ± 0.60	2.10 ± 0.28	TD3	<0.001
Convergence rounds	3200 ± 185	4800 ± 310	5200 ± 360	3900 ± 250	N/A	4500 ± 290	SAC	<0.001
Single-step decision time (ms)	12.3 ± 0.6	28.7 ± 1.8	35.2 ± 2.1	19.6 ± 1.3	8.5 ± 0.4	25.8 ± 1.5	A*+SJF	<0.001
Burst recovery time (s)	10.8 ± 0.9	18.3 ± 1.5	22.4 ± 1.8	15.7 ± 1.2	34.6 ± 2.8	18.9 ± 1.6	SAC	<0.001
Training time (h)	6.8 ± 0.4	9.2 ± 0.6	8.5 ± 0.5	7.2 ± 0.4	N/A	4.5 ± 0.3	TD3	<0.001
Average cumulative reward	285.6 ± 7.8	262.4 ± 9.5	247.3 ± 10.7	268.5 ± 8.4	271.2 ± 6.8	N/A	A*+SJF	0.002

Table 7 reports mean ± standard deviation values over 10 independent simulation runs. According to Table 7, the improved results show that PER-DQN produces the best average order delivery delay, the largest number of delivered orders (throughput), the highest equipment utilization rate, and the least path conflict ratio when compared to the other baselines. PER-DQN has a faster convergence speed and higher cumulative reward when compared with MADDPG, PPO, SAC, and TD3, implying that the new method is more sample efficient in a discrete warehouse control system. The performance of A* + SJF in terms of average delay, throughput, equipment utilization rate, conflict ratio, and burst scene recovery time is relatively poor compared to PER-DQN since A* + SJF does not apply any neural network prediction or learning process. However, A* + SJF has the fastest decision-making speed in a single step due to the lack of the use of neural networks in making predictions and learning policies (GA is not included in the main comparison because it is an offline population-based optimization method and does not update a scheduling policy through state-action-reward transitions. Therefore, GA cannot provide comparable convergence episodes or cumulative reward values.

The results highlight the tradeoff that PER-DQN tries to solve. A*+SJF provides the shortest decision time since it does not require any neural computation or policy training. However, its scheduling ability declines with increased disturbance. On the other hand, PER-DQN takes longer to make decisions online, but it outperforms A*+SJF in terms of latency, throughput, utilization, and conflict control. Thus, the benefit of PER-DQN is not just about sample efficiency but also about maintaining computational feasibility with improved adaptability.

A simulation environment built in Python using a discrete grid has been created for this study to provide both abstract scheduling logic and physical accuracy. Since the kinematics of the robots have been purposely simplified within the simulation, important elements of warehouse scheduling still exist: multi-agent collision volumes, sequencing of tasks, resource exclusivity, and Poisson distribution-based order generation. The environment was validated through four categories of tests, including logical consistency, parameter sensitivity, physical benchmarking, and scalability testing, as shown in Table 8.

To validate that the simulation environment captures the behavior of the actual warehouse, a physical testbench experiment was conducted as a reference baseline. The testbench was designed to include four robots driven in the ROS environment, working on a warehouse grid layout. The grid layout, location of shelves, order generation, and locations of sorting stations have been set according to the simulation

scenario. The robots in the physical testbench used the same scheduling policy computed by the PER-DQN controller, while the order arrival policy was set identically to that in the simulation environment. In particular, the orders were assumed to follow the Poisson distribution used in the simulation scenario. The proportion of small batch orders, mixed batch orders, large batch orders, and urgent orders was set according to the experimental configuration. As a physical testbed experiment, 30 experiments were conducted under similar conditions for the task loads. In all of the experiments, order delays and the rate of path conflicts between the robots were observed both in the simulation and the physical experiments.

Table 8. Simulation Environment Verification Experiment Results

Verification category	Test item	Revised test method	Evaluation indicators	Standard requirements	Measured results	Conclusion
Logical consistency	State transition verification	Traverse all possible combinations of state actions	State transition accuracy rate	100%	100%	Pass
	Resource exclusivity verification	Concurrent resource request test	Incidence of conflict	0%	0%	Pass
	Order processing chain verification	End-to-end task flow testing	Task completion rate	100%	100%	Pass
Parameter sensitivity	Sensor noise test	Position error $\pm 5\%$ random disturbance	Path planning success rate	$\geq 95\%$	96.80%	Pass
	Communication delay test	100-300ms random delay	Instruction execution accuracy	$\geq 90\%$	93.20%	Pass
Physical benchmarking	Order delay benchmarking	Deploy the same PER-DQN scheduling strategy on a scaled physical testbed with four ROS-based robots; compare simulation and physical results over 30 independent runs.	Delay difference rate	$\leq 20\%$	12.30%	Pass
	Path conflict rate benchmarking	Repeat the same task-load and grid-layout scenarios in simulation and physical testbed; compare mean conflict-rate deviation over 30 runs.	Conflict rate difference rate	$\leq 25\%$	14.70%	Pass
Scalability test	Expansion of the number of robots	2 units $\rightarrow$ 6 units gradually increased	Decision time growth rate	$\leq$ Linear growth	1.8 times (theoretical 3 times)	Excellent

The self-developed customized simulation platform validates the core operational logic of dynamic warehousing systems, demonstrating high accuracy with low computational cost as outlined in Table 8. The three Indicators for Logical Consistency Verification have met 100%, demonstrating a strong correlation to the previously defined environment model ruleset. The sensitivity of operational parameters during testing shows that even with disturbances of $\pm 5\%$ positional error and 100-300ms of communication delay, the success rate of path planning and execution accuracy remains above 96.8% and 93.2%, respectively. This reflects the strong capabilities of the simulation system to withstand interference from outside sources. Comparing physical benchmark results demonstrates order delay and path conflict rate differences being kept below 15%, and the increase of decision time being 1.8 times greater (still less than theoretical linear growth) for increasing numbers of robots. These results provide evidence that this simulation environment serves not only as an accurate mapping tool of real-world scenarios but also as a very scalable simulation platform. The extensive amount of data resulting from this comprehensive validation effort supports the use of this simulation platform as a sound basis for developing algorithms and testing performance.

This study was conducted to investigate the contribution of the PER mechanism and multi-objective composite reward function to overall system performance through an ablation experiment, which used three control models: the PER-DQN model (full model), DQN + multi-objective reward (removing only the

PER component), and DQN + the PER (removing only the weight dynamic reward - i.e., with fixed weights). As can be seen in Table 9. Further progress past simple descriptive comparison and quantification of the individual contribution of each algorithmic element involved conducting an ablation study. While the previous ablation considered the PER algorithm and multi-objective reward function, one more variant was added to consider only the effect of Double DQN: PER + DQN. Thus, in this case, only PER is used for conflict sample replay along with the multi-objective reward, while the Q target update is performed according to the standard DQN approach using the maximum Q value among actions. This way, there are three effects isolated in the comparison between the algorithms: PER effect on the use of conflict samples for replay; multi-objective reward effect on trade-off scheduling; and Double DQN effect on preventing overestimation in Q values and stabilizing learning. All experiments were run with various random seeds, with all reported values being mean $\pm$ standard deviation.

Table 9. Extended ablation analysis of PER, Double DQN, and multi-objective reward components

Performance indicator	Full PER-DQN	PER + standard DQN	DQN + multi-objective reward, no PER	DQN + PER, fixed reward	Standard DQN	p-value vs. PER + standard DQN
Order delay (seconds)	14.35 $\pm$ 0.42	15.47 $\pm$ 0.51	16.82 $\pm$ 0.63	17.23 $\pm$ 0.66	18.99 $\pm$ 0.74	p < 0.01
Path conflict rate (%)	1.20 $\pm$ 0.09	1.74 $\pm$ 0.13	2.10 $\pm$ 0.18	2.80 $\pm$ 0.22	4.30 $\pm$ 0.31	p < 0.01
Equipment utilization (%)	93.40 $\pm$ 1.10	91.85 $\pm$ 1.24	90.10 $\pm$ 1.36	88.70 $\pm$ 1.45	82.50 $\pm$ 1.82	p < 0.05
Burst recovery time (s)	10.80 $\pm$ 0.37	12.64 $\pm$ 0.49	16.40 $\pm$ 0.71	14.90 $\pm$ 0.66	22.40 $\pm$ 1.05	p < 0.01
Training convergence	3200 $\pm$ 115	3610 $\pm$ 148	4100 $\pm$ 186	3800 $\pm$ 171	4800 $\pm$ 225	p < 0.01
Strategy stability variance (	0.024 $\pm$ 0.003	0.033 $\pm$ 0.004	0.041 $\pm$ 0.006	0.035 $\pm$ 0.005	0.067 $\pm$ 0.008	p < 0.01
Q-value overestimation error	0.096 $\pm$ 0.011	0.184 $\pm$ 0.019	0.207 $\pm$ 0.024	0.176 $\pm$ 0.021	0.263 $\pm$ 0.031	p < 0.01

The analysis of the results from extended ablation experiments clearly demonstrates how prioritized experience replay (PER) and Double DQN compensate for different flaws of regular DQN. By removing the PER part, one can observe a significant rise in the percentage of path conflicts (1.20% to 2.10%) and burst recovery time (10.80 s to 16.40 s), which confirms the crucial role of prioritized replay for learning from rare and important conflict/disturbance samples. A comparison between the entire PER-DQN network and its modified version, which lacks Double DQN, highlights the unique function of the latter. Without Double DQN, the duration of order delay increases by more than a second (from 14.35 s to 15.47 s), and the percentage of path conflicts jumps to 1.74%. Moreover, the magnitude of overestimation errors in Q-values becomes even more pronounced (0.096 versus 0.184). Therefore, it is evident that Double DQN minimizes overoptimistic evaluations of actions in terms of rewards; this characteristic might be especially useful in warehouse scheduling, as highly optimistic Q-values would induce agents to choose dangerous routes with a higher probability of conflicts. In summary, Double DQN boosts reliability and safety in value estimation. In addition, statistical analysis shows that the differences between PER-DQN and its version without Double DQN are significant and cannot be attributed to numerical coincidence. According to the results of t-tests, the former algorithm outperforms the latter on measures of order delay, path conflicts, burst recovery time, convergence rounds, strategy stability, and overestimation errors. Hence, the updated findings offer robust experimental evidence that PER facilitates replay of rare events and Double DQN alleviates overoptimistic estimates of actions' values. To ensure a fair interpretation of the multi-agent comparison, the implementation difference between PER-DQN and QMIX is clarified. PER-DQN is implemented as a centralized controller with a global state input and a factorized composite action output. In contrast, QMIX is implemented as a value-decomposition-based CTDE algorithm. Each robot is assigned a local utility network with shared parameters. The local observation of robot i is defined as:

$$o_{i,t} = [\hat{x}_{i,t}, \hat{y}_{i,t}, h_{i,t}, b_{i,t}, \hat{d}_{i,t}, \hat{q}_t, \hat{c}_{i,t}, \hat{p}_{i,t}] \quad (20)$$

where $(\hat{x}_{i,t}, \hat{y}_{i,t})$ denotes the normalized robot position, $h_{i,t}$ is the robot status, $b_{i,t}$ is the normalized battery level, $\hat{d}_{i,t}$ is the normalized distance to the assigned or nearest task, $\hat{q}_t$ is the normalized local task-

queue pressure, $\hat{c}_{i,t}$ is the local congestion index around robot i , and $\hat{p}_{i,t}$ is the local shelf or station saturation feature. The local action set contains movement direction, task-acceptance decision, charging decision, and local avoidance rule. The individual utility values $Q_i(o_i, a_i)$ are combined by a monotonic mixing network conditioned on the global state, producing Q_{tot} . The mixing network consists of two hidden layers with 64 and 32 neurons, ReLU activation, and non-negative mixing weights generated by hypernetworks. This setting allows decentralized robot-level execution in QMIX, while PER-DQN remains a centralized scheduling baseline. To evaluate the performance of the proposed method in multi-robot collaborative scenarios and address the gap in comparative studies with mainstream multi-agent reinforcement learning paradigms, this research introduces comparative experiments with two representative algorithms: QMIX and MAPPO. QMIX employs a value decomposition network architecture, combining agents' local Q-values through a hybrid network to achieve centralized training and distributed execution. MAPPO extends proximal policy optimization to multi-agent scenarios, leveraging parameter sharing and centralized value function learning to develop collaborative strategies. Experiments were conducted under identical warehouse environments and task loads with 8 robots and moderate order density, with results presented in Table 10.

Table 10. Performance Comparison Between PER-DQN and Multi-Agent Reinforcement Learning Algorithms

Performance indicators	PER-DQN	QMIX	MAPPO
Order processing delay (seconds)	11.2	14.6	15.3
Path conflict rate (%)	1.2	2.5	2.8
Equipment utilization rate (%)	93.4	90.2	88.7
Training convergence rounds	3200	4500	5100
4 robots expanded to 8 performance attenuation rates (%)	8.3	7.1	9.5
8 robots expanded to 16, performance attenuation rate (%)	21.6	15.4	18.2

Table 10 demonstrates that in medium-scale scenarios with 8 robots, PER-DQN outperforms QMIX and MAPPO across three core metrics: order processing latency (11.2 seconds), path conflict rate (1.2%), and equipment utilization (93.4%). This advantage stems from PER's prioritized sampling of critical collaborative experiences, enabling agents to rapidly learn conflict-avoidance strategies. However, QMIX maintains superior scalability: when the robot count increases from 4 to 8, PER-DQN's performance declines by 8.3% (higher than QMIX's 7.1%); at 16 robots, PER-DQN shows a 21.6% performance drop compared to QMIX's 15.4%. These results indicate that explicit multi-agent algorithms demonstrate better scalability in ultra-large-scale scenarios, where their value decomposition architecture effectively handles collaborative relationships. PER-DQN employs a centralized training framework with distributed execution, and its structured action space design reduces joint action space complexity, delivering excellent performance in small-to-medium scale scenarios. However, increased robot numbers exacerbate the challenge of expanding centralized value function input dimensions. Future research could integrate PER mechanisms with QMIX architecture to enhance scalability and collaborative efficiency in large-scale scenarios.

To evaluate the applicability boundaries of PER-DQN in warehouse scenarios of varying scales, this section conducts scalability experiments with robot quantities incrementally increasing from 10 to 50 and 100 units. The experiments were performed in a discrete grid warehouse environment (50×50 grid) with moderate order density. Comparative algorithms included PPO, SAC, and the multi-agent reinforcement learning algorithm QMIX, with particular focus on analyzing trends in processing latency and path conflict rates. Experimental results are presented in Table 11.

Table 11. Performance changes of various algorithms under different robot numbers

Scenario	Metric Performance I	PER-DQN, mean ± SD	DQN, mean ± SD	Improvement	Test	P-value
Dynamic order fluctuation	Average delay (s)	18.2 ± 1.1	26.5 ± 1.8	↓31.3%	Welch's t-test	<0.001
Dynamic order fluctuation	Task completion rate (%)	98.5 ± 0.7	90.2 ± 2.0	↑9.2%	Mann-Whitney U	<0.001
Sudden order shock	Peak delay (s)	31.7 ± 2.3	45.9 ± 3.4	↓30.9%	Welch's t-test	<0.001

	Recovery time (s)	10.8 ± 0.9	22.4 ± 2.2	↓51.8%	Welch's t-test	<0.001
10% robot shutdown 10% robot shutdown	Path conflict rate (%)	1.8 ± 0.3	5.4 ± 0.7	↓66.7%	Mann-Whitney U	<0.001
	Equipment utilization rate (%)	92.3 ± 1.6	73.6 ± 3.1	↑25.4%	Welch's t-test	<0.001
150% order volume 150% order volume	Sorting accuracy (%)	96.8 ± 0.9	89.5 ± 1.8	↑8.2%	Welch's t-test	<0.001
	Equipment utilization rate (%)	89.7 ± 1.8	70.2 ± 3.3	↑27.8%	Welch's t-test	<0.001

Table 11 confirms the robustness of PER-DQN under stochastic disturbance conditions. Across 10 independent runs, PER-DQN consistently outperforms DQN under dynamic order fluctuation, sudden order shock, robot shutdown, and high-load pressure. The improvements in delay reduction, recovery speed, conflict suppression, equipment utilization, and sorting accuracy are statistically significant ($p < 0.001$). The scalability results need to be considered in consideration of differences between PER-DQN and QMIX architectures. Since PER-DQN utilizes centralized state representation and centralized scheduling action selection, both input and coordination complexity will increase in proportion to the increase in the number of robots. Thus, PER-DQN's superiority when the number of robots lies between 10 and 30 and the drop-off in performance at 100 robots can be explained by this factor. On the other hand, QMIX takes into account local robot observation and a value decomposition network to solve the multi-agent learning problem, providing better scalability for ultra-large robotic agents. Therefore, the suggested PER-DQN controller would best fit a smaller fleet of robots in warehouse environments, while larger robotic environments will benefit from PER-QMIX controllers.

As shown in Table 11, in medium-to-small-scale scenarios with 10-30 robots, PER-DQN outperforms PPO and SAC in latency and collision rate while matching QMIX's performance, thanks to its PER mechanism prioritizing key collaborative experience sampling. When robot numbers increase to 50, PER-DQN shows a latency of 35.81 seconds and 8.6% collision rate, while QMIX reaches 34.96 seconds latency and 8.1% collision rate, demonstrating comparable performance. At 100 robots, QMIX demonstrates superiority with 62.38-second latency (68.54 seconds for PER-DQN) and 15.2% collision rate (18.3% for PER-DQN). This indicates QMIX's value decomposition architecture handles collaborative relationships more effectively in ultra-large-scale scenarios, highlighting its scalability advantages. PER-DQN experiences approximately 4.8-fold latency increase and 15.3-fold collision rate escalation, while QMIX shows 3.7-fold latency growth and 7.2-fold collision rate increase, revealing limitations of centralized value function approaches in large-scale systems. At 100 robots, PER-DQN still maintains lower collision rates than PPO and SAC, but QMIX achieves even lower rates, demonstrating explicit multi-agent architecture advantages. PER-DQN is suitable for 10-50 robot scenarios, while the QMIX framework or PER mechanism-value decomposition architecture integration should be considered for over 100 robots. Future research may explore PER-QMIX fusion frameworks to enhance scalability.

5. Conclusion

This study focuses on dynamic warehousing systems. To address the problems of low task scheduling efficiency and difficulty in adapting to changes in dynamic environments caused by traditional control methods, an intelligent control method based on PER-DQN is proposed. In a highly dynamic warehousing environment, traditional methods have shortcomings such as incomplete state perception and a single decision-making dimension, which leads to delayed system response and unreasonable resource allocation. To address these challenges, this paper first constructs a high-dimensional state vector with 10-dimensional features, integrates key parameters such as the order pool queue length, and accurately characterizes and monitors the operating status of the warehousing system in real time. In the design of the action space, an innovative structural concept is adopted, dividing it into four interrelated subspaces, such as task assignment priority. Each subspace is optimized for specific control requirements, improving the system control accuracy and decision-making flexibility. To optimize the performance of the algorithm, a multi-objective composite reward function is studied and designed, which comprehensively considers multiple key performance indicators such as task completion efficiency and equipment utilization, and balances and

coordinates different optimization objectives by dynamically adjusting the weight coefficients. At the same time, the PER mechanism is applied to determine the importance weight based on the sample TD error, prioritize replaying key experience samples, and improve sample utilization efficiency and algorithm convergence speed. In model training, the Double DQN architecture is used to separate action selection and evaluation, suppress the Q-value overestimation bias, and enhance learning stability. Experimental results show that in a medium-sized warehouse, PER-DQN achieves efficiencies of 13.5%, 13%, and 12.5% at low, medium, and high densities, respectively. In a scenario with equipment failure (10% robot downtime), PER-DQN achieves an equipment utilization rate of 92.3%. PER-DQN effectively solves the complex scheduling issues of dynamic warehousing systems, improves system operational efficiency, and enhances robustness to environmental changes, providing a reliable technical solution for the practical application of intelligent logistics systems.

Authorship Contribution Statement

Chuanchuan Teng: Writing-Original draft preparation, Conceptualization, Supervision, Project administration; Xizhuo Han: Methodology, Software.

References

- [1] Janka Saderova, Andrea Rosova, Marian Sofranko and Peter Kacmary, "Example of Warehouse System Design Based on the Principle of Logistics", *Sustainability*, Online ISSN: 2071-1050, 17 April 2021, Vol. 13, No. 8, p. 4492, Published by MDPI, DOI: 10.3390/su13084492, Available: <https://www.mdpi.com/2071-1050/13/8/4492>.
- [2] Riccardo Accorsi, Riccardo Manzini and Fausto Maranesi, "A Decision-Support System for the Design and Management of Warehousing Systems", *Computers in Industry*, Print ISSN: 0166-3615, Online ISSN: 1872-6194, January 2014, Vol. 65, No. 1, pp. 175–186, Published by Elsevier, DOI: 10.1016/j.compind.2013.08.007, Available: <https://www.sciencedirect.com/science/article/abs/pii/S0166361513001875>.
- [3] Enoch Oluwademilade Sodiya, Uchenna Joseph Umoga, Olukunle Oladipupo Amoo and Akoh Atadoga, "AI-Driven Warehouse Automation: A Comprehensive Review of Systems", *GSC Advanced Research and Reviews*, Online ISSN: 2582-4597, 3 January 2024, Vol. 18, No. 2, pp. 272–282, Published by GSC Online Press, DOI: 10.30574/gscarr.2024.18.2.0063, Available: <https://gsonlinepress.com/journals/gscarr/content/ai-driven-warehouse-automation-comprehensive-review-systems>.
- [4] Fatma Gzara, Samir Elhedhli, Ugur Yildi and Gohram Baloch, "Data-Driven Modeling and Optimization of the Order Consolidation Problem in E-warehousing", *INFORMS Journal on Optimization*, Print ISSN: 1545-8490, Online ISSN: 1545-8504, 2 October 2020, Vol. 2, No. 4, pp. 273–296, Published by INFORMS, DOI: 10.1287/ijoo.2019.0039, Available: <https://pubsonline.informs.org/doi/abs/10.1287/ijoo.2019.0039>.
- [5] Joakim Kembro and Andreas Norrman, "The Transformation from Manual to Smart Warehousing: An Exploratory Study with Swedish Retailers", *The International Journal of Logistics Management*, Print ISSN: 0957-4093, Online ISSN: 1758-6550, 22 June 2022, Vol. 33, No. 5, pp. 107–135, Published by Emerald Publishing Limited, DOI: 10.1108/IJLM-11-2021-0525, Available: <https://www.emerald.com/ijlm/article/33/5/107/514534/The-transformation-from-manual-to-smart>.
- [6] Yuze Xu, Linxuan Zhang, Hui Li, Ming Ge and Wanyi He, "Modeling and Optimization of Smart Warehouse Order Sorting Considering Splitting Strategy", *Journal of System Simulation*, Online ISSN: 1004-731X, 2 October 2024, Vol. 36, No. 3, pp. 564–577, Published by China Simulation Federation, DOI: 10.16182/j.issn1004731x.joss.23-0422, Available: <https://dc-china-simulation.researchcommons.org/journal/vol36/iss3/4/>.
- [7] Nils Boysen, Stefan Schwerdfeger and Marlin W Ulmer, "Robotized Sorting Systems: Large-Scale Scheduling under Real-Time Conditions with Limited Lookahead", *European Journal of Operational Research*, Print ISSN: 0377-2217, Online ISSN: 1872-6860, 16 October 2023, Vol. 310, No. 2, pp. 582–596, Published by Elsevier, DOI: 10.1016/j.ejor.2023.03.037, Available: <https://www.sciencedirect.com/science/article/pii/S0377221723002643>.
- [8] Juan C. Duque-Jaramillo, Juan M. Cogollo-Flórez, Cristian G. Gómez-Marín and Alexander A. Correa-Espinal, "Warehouse Management Optimization Using a Sorting-Based Slotting Approach", *Journal of Industrial Engineering and Management*, Print ISSN: 2013-8423, Online ISSN: 2013-0953, January 2024, Vol. 17, No. 1, pp. 133–150, Published by Universitat Politècnica de Catalunya (UPC), DOI: 10.3926/jiem.5661, Available: <http://jiem.org/index.php/jiem/article/view/5661>.
- [9] Zhang Jun, Wang Xuping, Chan Felix T.S. and Ruan Junhu, "On-line order batching and sequencing problem with multiple pickers: A hybrid rule-based algorithm", *Applied Mathematical Modelling*, Print ISSN: 0307-904X, Online ISSN: 1872-8480, May 2017, Vol. 45, pp. 271–284, Published by Elsevier B.V., DOI: 10.1016/j.apm.2020.05.028, Available: <https://www.sciencedirect.com/science/article/pii/S0307904X16306655>.

- [10] Xiangnan Zhan, Liyun Xu and Xufeng Ling, "Task Scheduling Problem of Double-Deep Multi-Tier Shuttle Warehousing Systems", *Processes*, Online ISSN: 2227-9717, 26 December 2020, Vol. 9, No. 1, p. 41, Published by MDPI, DOI: 10.3390/pr9010041, Available: <https://www.mdpi.com/2227-9717/9/1/41>.
- [11] Jia Ma, Shujun Yang and Hao Jing, "Intelligent Warehouse Robot Scheduling System Using a Modified Nondominated Sorting Algorithm", *Discrete Dynamics in Nature and Society*, Print ISSN: 1026-0226, Online ISSN: 1607-887X, 15 June 2022, Vol. 2022, No. 1, p. 2021535, Published by Wiley, DOI: 10.1155/2022/2021535, Available: <https://onlinelibrary.wiley.com/doi/full/10.1155/2022/2021535>.
- [12] Pei-Ju Wu and Yu-Chin Tai, "Artificial Intelligence-Based Food-Quality and Warehousing Management for Food Banks' Inbound Logistics", *Journal of Enterprise Information Management*, Print ISSN: 1741-0398, Online ISSN: 1758-7409, 29 January 2024, Vol. 37, No. 1, pp. 307–325, Published by Emerald Publishing Limited, DOI: 10.1108/JEIM-10-2022-0398, Available: <https://doi.org/10.1108/JEIM-10-2022-0398>.
- [13] Zefang Zong, Tao Feng, Jingwei Wang, Tong Xia and Yong Li, "Deep Reinforcement Learning for Demand-Driven Services in Logistics and Transportation Systems: A Survey", *ACM Transactions on Knowledge Discovery from Data*, Print ISSN: 1556-472X, Online ISSN: 1556-4681, 9 May 2025, Vol. 19, No. 4, pp. 1–42, Published by ACM Digital Library, DOI: 10.1145/3708325, Available: <https://dl.acm.org/doi/full/10.1145/3708325>.
- [14] Zhenhua He, Liang Chen and Bin Liu, "Application of Integrating Reinforcement Learning and Intelligent Scheduling in Logistics Distribution", *Intelligent Decision Technologies*, Print ISSN: 1872-4981, Online ISSN: 1875-8843, 1 February 2024, Vol. 18, No. 1, pp. 57–74, Published by Sage Journals, DOI: 10.3233/IDT-230528, Available: <https://journals.sagepub.com/doi/full/10.3233/IDT-230528>.
- [15] Jonas F. Leon, Yuda Li, Xabier A. Martin, Laura Calvet, Javier Panadero *et al.*, "A Hybrid Simulation and Reinforcement Learning Algorithm for Enhancing Efficiency in Warehouse Operations", *Algorithms*, Online ISSN: 1999-4893, 27 August 2023, Vol. 16, No. 9, p. 408, Published by MDPI, DOI: 10.3390/a16090408, Available: <https://www.mdpi.com/1999-4893/16/9/408>.
- [16] Haijiang Li, Jiapeng Zhao, Peng Jia, Hongdong Ou and Weili Zhao, "Optimization of Bulk Cargo Terminal Unloading and Outbound Operations Based on a Deep Reinforcement Learning Framework", *Journal of Marine Science and Engineering*, Online ISSN: 2077-1312, 8 January 2025, Vol. 13, No. 1, p. 105, Published by MDPI, DOI: 10.3390/jmse13010105, Available: <https://www.mdpi.com/2077-1312/13/1/105>.
- [17] Fei Wang, Honghai Zhang, Sen Du, Mingzhuang Hua and Gang Zhong, "C-SPPO: A Deep Reinforcement Learning Framework for Large-Scale Dynamic Logistics UAV Routing Problem", *Chinese Journal of Aeronautics*, Print ISSN: 1000-9361, Online ISSN: 2588-9230, May 2025, Vol. 38, No. 5, p. 103229, Published by Elsevier, DOI: 10.1016/j.cja.2024.09.005, Available: <https://www.sciencedirect.com/science/article/pii/S1000936124003662>.
- [18] Ju-Bong Kim, Ho-Bin Choi, Gyu-Young Hwang, Kwihoon Kim, Yong-Geun Hong *et al.*, "Sortation Control Using Multi-Agent Deep Reinforcement Learning in N-Grid Sortation System", *Sensors*, Online ISSN: 1424-8220, 16 June 2020, Vol. 20, No. 12, p. 3401, Published by MDPI, DOI: 10.3390/s20123401, Available: <https://www.mdpi.com/1424-8220/20/12/3401>.
- [19] Hanjie Gu, Yanyong Feng, Deke Yu, Junwei Fang, Yuliang Sun *et al.*, "HAC-FRL: A Learning-Driven Distributed Task Allocation Framework for Large-Scale Warehouse Automation", *Information Processing & Management*, Online ISSN: 0306-4573, March 2026, Vol. 63, No. 2, p. 104430, Published by Elsevier, DOI: 10.1016/j.ipm.2025.104430, Available: <https://www.sciencedirect.com/science/article/abs/pii/S0306457325003711>.
- [20] Ruiping Yuan, Jiangtao Dou, Juntao Li, Wei Wang and Yingfan Jiang, "Multi-robot task allocation in e-commerce RMFS based on deep reinforcement learning", *Mathematical Biosciences and Engineering*, ISSN: 1551-0018, 8 November 2022, Vol. 20, No. 2, pp. 1903–1918, Published by AIMS Press, DOI:10.3934/mbe.2023087, Available: <http://www.aimspress.com/article/doi/10.3934/mbe.2023087>.
- [21] Zhuoyuan Yu, Hongliang Guo, Chee-Meng Chew, Albertus Hendrawan Adiwahono, Jianle Chan *et al.*, "Multi-Robot Reliable Navigation in Uncertain Topological Environments with Graph Attention Networks", *IEEE Robotics and Automation Letters*, Online ISSN: 2377-3766, May 2025, Vol. 10, No. 5, pp. 5082–5089, Published by IEEE, DOI: 10.1109/LRA.2025.3557751, Available: <https://ieeexplore.ieee.org/abstract/document/10948339>.
- [22] Matthew Lai, Keegan Go, Zhibin Li, Torsten Kroger, Stefan Schaal *et al.*, "RoboBallet: Planning for Multirobot Reaching with Graph Neural Networks and Reinforcement Learning", *Science Robotics*, Online ISSN: 2470-9476, 3 September 2025, Vol. 10, No. 106, p. eads1204, Published by AAAS, DOI: 10.1126/scirobotics.ads1204, Available: <https://www.science.org/doi/full/10.1126/scirobotics.ads1204>.
- [23] Baiyu Li and Hang Ma, "Double-Deck Multi-Agent Pickup and Delivery: Multi-Robot Rearrangement in Large-Scale Warehouses", *IEEE Robotics and Automation Letters*, Online ISSN: 2377-3766, 1 May 2023, Vol. 8, No. 6, pp. 3701–3708, Published by IEEE, DOI: 10.1109/LRA.2023.3272272, Available: <https://ieeexplore.ieee.org/abstract/document/10113719>.
- [24] Zhuo Chen, Bowen Zhu and Chuan Zhou, "Container Cluster Placement in Edge Computing Based on Reinforcement Learning Incorporating Graph Convolutional Networks Scheme", *Digital Communications and Networks*, Print ISSN: 2468-5925, Online ISSN: 2352-8648, February 2025, Vol. 11, No. 1, pp. 60–70, Published by

- Elsevier, DOI: 10.1016/j.dcan.2023.02.012, Available: <https://www.sciencedirect.com/science/article/pii/S2352864823000470>.
- [25] Guozhi Liu, Weiwei Lin, Haotong Zhang, Jianpeng Lin, Shaoliang Peng *et al.*, “Public Datasets for Cloud Computing: A Comprehensive Survey”, *ACM Computing Surveys*, Print ISSN: 0360-0300, Online ISSN: 1557-7341, 7 March 2025, Vol. 57, No. 8, pp. 1–38, Published by ACM Digital Library, DOI: 10.1145/3719003, Available: <https://dl.acm.org/doi/full/10.1145/3719003>.
- [26] Huwei Liu, Li Zhou, Junhui Zhao, Fan Wang, Jianglong Yang *et al.*, “Deep-Learning-Based Accurate Identification of Warehouse Goods for Robot Picking Operations”, *Sustainability*, Online ISSN: 2071-1050, 20 August 2025, Vol. 14, No. 13, p. 7781, Published by MDPI, DOI: 10.3390/su14137781, Available: <https://www.mdpi.com/2071-1050/14/13/7781>.
- [27] Shaohui Zhang, Qiuying Han, Hai Zhu, Hongfeng Wang, Huiling Li *et al.*, “Real Time Task Planning for Order Picking in Intelligent Logistics Warehousing”, *Scientific Reports*, Online ISSN: 2045-2322, 1 March 2025, Vol. 15, No. 1, p. 7331, Published by Springer Nature, DOI: 10.1038/s41598-025-88305-9, Available: <https://www.nature.com/articles/s41598-025-88305-9>.
- [28] Nils Boysen, Konrad Stephan and Felix Weidinger, “Efficient Order Consolidation in Warehouses: The Product-to-Order-Assignment Problem in Warehouses with Sortation Systems”, *IIE Transactions*, Online ISSN: 2472-5854, 10 January 2022, Vol. 54, No. 10, pp. 963–975, Published by Taylor & Francis, DOI: 10.1080/24725854.2021.2004336, Available: <https://www.tandfonline.com/doi/abs/10.1080/24725854.2021.2004336>.
- [29] Thomas De Lombaert, Kris Braekers, René De Koster and Katrien Ramaekers, “Is it Good to Have a Choice? The Value of Participatory Order Assignments in Warehousing”, *International Journal of Operations & Production Management*, Print ISSN: 0144-3577, Online ISSN: 1758-6593, 16 September 2024, Vol. 45, No. 3, pp. 783–806, Published by Emerald Publishing Limited, DOI: 10.1108/IJOPM-11-2023-0882, Available: <https://www.emerald.com/ijopm/article-abstract/45/3/783/1242491/Is-it-good-to-have-a-choice-The-value-of?redirectedFrom=fulltext>.
- [30] Zhang Honglin, Wu Yaohua, Hu Jinchang and Wang Yanyan, “Collaborative Optimization of Task Scheduling and Multi-Agent Path Planning in Automated Warehouses”, *Complex & Intelligent Systems*, Online ISSN: 2198-6053, 15 April 2023, Vol. 9, No. 5, pp. 5937–5948, Published by Springer, DOI: 10.1007/s40747-023-01023-5, Available: <https://link.springer.com/article/10.1007/s40747-023-01023-5>.
- [31] Amir Zarinchang, Kevin Lee, Iman Avazpour, Jun Yang, Dongxing Zhang *et al.*, “Adaptive Warehouse Storage Location Assignment with Considerations to Order-Picking Efficiency and Worker Safety”, *Journal of Industrial and Production Engineering*, Print ISSN: 1015-2168, Online ISSN: 1023-2168, 9 October 2023, Vol. 41, No. 1, pp. 40–59, Published by Taylor & Francis, DOI: 10.1080/21681015.2023.2263009, Available: <https://www.tandfonline.com/doi/full/10.1080/21681015.2023.2263009>.
- [32] Tetiana Likhousova and Yuliia Demianova, “Robot Path Optimization in Warehouse Management System”, *Evolutionary Intelligence*, Print ISSN: 1864-5909, Online ISSN: 1864-5917, 11 May 2021, Vol. 15, No. 4, pp. 2589–2595, Published by Springer Nature, DOI: 10.1007/s12065-021-00614-w, Available: <https://link.springer.com/article/10.1007/s12065-021-00614-w>.
- [33] Kaushlendra Sharma and Rajesh Doriya, “Coordination of Multi-Robot Path Planning for Warehouse Application Using Smart Approach for Identifying Destinations”, *Intelligent Service Robotics*, Print ISSN: 1861-2776, Online ISSN: 1861-2784, 27 March 2021, Vol. 14, No. 2, pp. 313–325, Published by Springer, DOI: 10.1007/s11370-021-00363-w, Available: <https://link.springer.com/article/10.1007/s11370-021-00363-w>.
- [34] Mun-Kit Ng, Yung-Wey Chong, Kwang-man Ko, Young-Hoon Park and Yu-Beng Leau, “Adaptive Path Finding Algorithm in Dynamic Environment for Warehouse Robot”, *Neural Computing and Applications*, Print ISSN: 0941-0643, Online ISSN: 1433-3058, 4 February 2020, Vol. 32, No. 17, pp. 13155–13171, Published by Springer, DOI: 10.1007/s00521-020-04764-3, Available: <https://link.springer.com/article/10.1007/s00521-020-04764-3>.
- [35] HyeokSoo Lee and Jongpil Jeong, “Mobile Robot Path Optimization Technique Based on Reinforcement Learning Algorithm in Warehouse Environment”, *Applied Sciences*, Online ISSN: 2076-3417, 8 January 2021, Vol. 11, No. 3, p. 1209, Published by MDPI, DOI: 10.3390/app11031209, Available: <https://www.mdpi.com/2076-3417/11/3/1209>.
- [36] Antonio Maria Coruzzolo, Francesco Lolli, Elia Balugani, Elisa Magnani and Miguel Afonso Sellitto, “Order Picking Problem: A Model for the Joint Optimisation of Order Batching, Batch Assignment Sequencing, and Picking Routing”, *Logistics*, Online ISSN: 2305-6290, 11 September 2023, Vol. 7, No. 3, p. 61, Published by MDPI, DOI: 10.3390/logistics7030061, Available: <https://www.mdpi.com/2305-6290/7/3/61>.

