



OPEN Collaborative optimization of truck scheduling in container terminals using graph theory and DDQN

Shu Cheng¹, Qianyu Liu^{2,3✉}, Heng Jin⁴, Ran Zhang⁵, Longhua Ma² & Chiew Foong Kwong³

The container terminal is a key node in global trade and logistics, where trucks connect quay cranes, storage yards, and vessels. Optimizing truck scheduling is crucial for enhancing port efficiency by addressing issues such as low truck utilization, excessive quay crane waiting times, and extended equipment completion times. This paper develops a container terminal simulation model based on graph theory, with the objective of minimizing the maximum completion time of terminal equipment. A collaborative scheduling algorithm for truck fleets, based on Deep Double Q-Networks (DDQN), is proposed. The algorithm designs five heuristic rules as the action space and refines state features and reward functions to optimize scheduling effectively. Experimental results indicate that this algorithm consistently identifies optimal scheduling strategies, outperforming both the five heuristic rules and the Deep Q-Network (DQN) algorithm. It significantly reduces quay crane waiting times and equipment completion times, improves truck utilization, and enhances overall container terminal efficiency.

Keywords Container terminal, Truck scheduling, Reinforcement learning, DDQN

With the continuous development of global trade, the significance of maritime transportation has become increasingly prominent, serving as an indispensable pillar of international trade¹. Ports, as transfer points between maritime and land transportation, play a critical role in determining the efficiency of the entire logistics supply chain². Among the port operations, container terminals, which are the core areas responsible for handling container cargo, including loading, unloading, storage, and scheduling, have a direct impact on the overall efficiency of port logistics³. Therefore, efficient container terminal operations not only enhance port operational efficiency, reduce vessel and cargo dwell time, and lower logistics costs, but also improve resource utilization, strengthen port competitiveness, and promote the smooth functioning of the global logistics supply chain.

A container terminal is an efficiently operating logistics hub composed of quay cranes, transport vehicles, storage yards, and yard cranes, responsible for the loading, unloading, storage, and transportation of import and export goods. When a vessel arrives at the terminal, quay cranes lift the containers and place them onto horizontal transport vehicles, which then transport the containers from the quay to the designated location in the storage yard⁴. When loading is required, yard cranes lift the corresponding containers onto horizontal transport vehicles, which then transport them to the designated destination. In summary, horizontal transport vehicles are integral to the entire operational process of a container terminal, and the efficiency of their scheduling plays a decisive role in determining the overall terminal efficiency.

However, in traditional container terminals, the absence of real-time information sharing mechanisms across various stages of the operational process impedes terminal management's ability to fully monitor and coordinate the truck scheduling. This inefficiency leads to suboptimal truck utilization, extended quay crane waiting times, and prolonged container loading and unloading processes. Furthermore, optimizing truck scheduling represents a typical NP-hard combinatorial optimization problem⁵. The complexity of coordinating multiple tasks and allocating handling equipment causes the difficulty of solving the problem to grow exponentially as the problem size increases.

To address the challenges in truck scheduling, researchers have proposed methods such as heuristic algorithms, simulation approaches, and reinforcement learning techniques⁶. Heuristic algorithms are computationally efficient but typically provide approximate solutions and rely on experience-based designs,

¹School of Information Science and Engineering, Zhejiang Sci-Tech University, Hangzhou 310018, China. ²School of Information Science and Engineering, NingboTech University, Ningbo 315100, China. ³Department of Electrical and Electronic Engineering, University of Nottingham Ningbo China, Ningbo 315100, China. ⁴School of Mechatronics and Energy Engineering, NingboTech University, Ningbo 315100, China. ⁵Ningbo Daxie Container Terminal Company Ltd, Ningbo 315812, China. ✉email: Qianyu.Liu@nbt.edu.cn

making them less suitable for dynamic and complex terminal environments. Simulation approaches can handle complex scenarios but often generate data that lacks comprehensiveness, failing to accurately reflect the dynamic characteristics of terminal operations. Reinforcement learning has garnered widespread attention for its adaptive learning capability and strong suitability for dynamic environments⁷. However, existing approaches face issues with limited interpretability, making it difficult for operators to understand and trust the scheduling strategies. Moreover, traditional reinforcement learning algorithms, such as Deep Q Networks (DQN), are prone to Q-value overestimation during value function updates due to the maximization operation, which can negatively impact model stability and convergence.

To address the issue of Q-value overestimation in the DQN algorithm, this study proposes an internal truck scheduling optimization algorithm based on Double Deep Q Networks (DDQN). Compared to traditional DQN algorithms, DDQN effectively mitigates the overestimation bias by decoupling action selection from value evaluation, thereby significantly improving the stability and convergence performance of the algorithm. This advantage makes DDQN particularly well-suited for handling complex scheduling problems in dynamic environments such as container terminals.

The objective of this study is to optimize internal truck scheduling strategies to minimize the makespan of handling equipment, reduce the waiting time of quay cranes, and improve the utilization rate of internal trucks, thereby enhancing the overall operational efficiency of the terminal. The specific research focuses are as follows:

- A detailed investigation was conducted on the actual layout and handling processes of equipment at the terminals within Ningbo-Zhoushan Port. Based on this analysis, a container terminal mathematical model was developed using graph theory, with simulation parameters derived from real operational data. Compared to traditional mathematical modeling, graph-based modeling offers greater intuitiveness and flexibility, making it better suited to handle the highly dynamic uncertainties present in container terminals.
- The truck scheduling problem is abstracted as a Markov Decision Process, and a deep reinforcement learning method is employed to solve it under a dual-cycle operation strategy (capable of both loading and unloading). This study characterizes the system's state using three node attributes: task processing time, switching time, and mapping entropy. Five different heuristic rule-based scheduling strategies were designed, constructing a flexible scheduling strategy space to enhance the interpretability of reinforcement learning. Additionally, the reward function for the DDQN algorithm is designed based on the mathematical model of the truck scheduling problem.
- This study conducts a comparative analysis with the current mainstream DQN algorithm and five heuristic-based scheduling methods to validate the effectiveness and superiority of the proposed approach.

To the best of the author's knowledge, this is the first study that applies a graph-modeling approach coupled with Double Deep Q-Networks (DDQN) for optimizing scheduling in traditional container terminals. The structure of the paper is as follows: Sect. 2 reviews the relevant literature on truck scheduling at container terminals. In Sect. 3, the truck scheduling problem at container terminals is described in detail, and both a mathematical model and a graph-theoretic model for truck scheduling are established. Section 4 models the horizontal transport vehicle scheduling problem using the Markov Decision Process (MDP) paradigm. In Sect. 5, an algorithm is designed to solve the truck scheduling optimization problem. Section 6 presents numerical experiments to verify the effectiveness of the proposed algorithm. Finally, Sect. 7 provides a summary of the paper and outlines directions for future research.

Literature review

In this section, we briefly review the primary approaches for addressing the optimization of internal truck scheduling in container terminals. Researchers have proposed various techniques to tackle this problem, including heuristic algorithms, simulation-based methods, and reinforcement learning approaches, each offering distinct perspectives for addressing the complexity inherent in truck scheduling.

Heuristic algorithms

Heuristic algorithms are a class of algorithms that utilize experiential rules or specific strategies to quickly solve complex problems. Common heuristic algorithms include genetic algorithms, simulated annealing, particle swarm optimization, and ant colony optimization. These algorithms have been widely applied in fields such as combinatorial optimization^{8–10}, scheduling problems^{11–13}, and path planning^{14–16}.

To address container terminal truck scheduling, Chen et al.¹⁷ proposed a Cooperative Dual-layer Genetic Programming Hyper-heuristic (CD-GPHH) approach, which separates scenarios and computations into two populations, improving multi-scenario handling and solution space continuity. However, CD-GPHH is highly sensitive to parameter settings—like population size, mutation rate, and crossover rate—resulting in limited adaptability and unstable performance across scenarios. To address this, Luo et al.¹⁸ introduced an adaptive heuristic that models the integrated scheduling of quay cranes, yard cranes, and automated guided vehicles (AGVs) as a mixed-integer programming (MIP) problem. This approach dynamically adjusts GA parameters based on performance feedback, enhancing adaptability and optimization across varied scenarios.

To improve the dual-cycle operational efficiency of yard cranes and AGVs, Yue et al.¹⁹ proposed a multi-vessel block sharing strategy. A two-stage mixed-integer programming model was developed to optimize block allocation and quay crane scheduling, as well as the scheduling of yard cranes and AGVs. The problem was solved using an adaptive variable neighborhood immune algorithm and a rule-based heuristic algorithm. Although this method excels in optimizing scheduling efficiency, it falls short in handling uncertainties in the terminal environment, lacking mechanisms for flexibly addressing dynamic changes and random events, which may limit its applicability and robustness in real-world scenarios. To address these issues, Lu et al.²⁰ considered

uncertainties in truck driving speed, yard crane speed, and lifting operation time, proposing an integrated scheduling optimization model based on a particle swarm optimization algorithm to cope with the uncertainties in scheduling.

The operational environment of container terminals involves numerous uncertainties, such as yard congestion, unpredictability of vessel arrival times, fluctuations in equipment operating conditions, and dynamic changes in operational demands. These factors place higher demands on scheduling algorithms. However, heuristic algorithms exhibit notable limitations in addressing these dynamic changes and uncertainties. Firstly, the search methods of heuristic algorithms are often constrained, relying on fixed rules or strategies, making it difficult to dynamically adapt to environmental changes. Additionally, their low sample utilization efficiency hinders effective exploration of complex solution spaces, often leading to suboptimal solutions. These limitations result in less-than-ideal scheduling outcomes in the complex terminal environment.

Simulation methods

Simulation methods construct a virtual model to simulate the interactions between handling equipment and the dynamic changes in various operations at container terminals. Simulations can meticulously replicate the actual scheduling process and predict the effects of different strategies in real-world scenarios.

Castilla-Rodríguez et al.²¹ designed an intelligent system that integrates evolutionary algorithms with simulation tools. This simulation method effectively improves the loading and unloading efficiency of container terminals and provides more comprehensive information support for terminal management. Kastner et al.²² employed simulation methods to successfully address complex and interrelated operational decision-making problems at container terminals. Simulation helps determine the optimal configuration of operational strategies such as truck quantity, scheduling schemes, and yard block allocation, thus enhancing the overall efficiency of the terminal.

From the two aforementioned simulation methods, it can be observed that simulation approaches are capable of modeling various operational scenarios in complex and dynamic terminal environments. They assist in testing and evaluating multiple scheduling strategies, providing comprehensive informational support to terminal operators. However, these methods also have certain limitations. Simulation methods typically require substantial computational resources to run complex models, especially for large-scale scheduling tasks, where computational costs increase significantly, making it challenging to meet real-time optimization demands. Furthermore, the data simulated by these models often lacks completeness, which limits their applicability and scalability in complex and dynamic terminal environments.

Reinforcement learning methods

In the highly dynamic and uncertain environment of terminals, heuristic and simulation algorithms offer some optimization but face limitations in handling frequently changing factors such as vessel arrival times, loading and unloading demands, and equipment scheduling. In contrast, reinforcement learning methods, which can better adapt to complex environmental changes and discover optimal strategies through continuous trial and error, demonstrate significant advantages^{23,24}.

Zheng et al.²⁵ modeled the AGV scheduling problem as a Markov Decision Process (MDP) and proposed a scheduling algorithm based on the Deep Q Network (DQN). This method generates optimal AGV scheduling strategies by leveraging the dynamic characteristics of terminal systems, and its advantages in scheduling performance and computational efficiency were validated through experiments. Similarly, Zhong et al.²⁶ addressed the uncertainties in terminals and proposed a reinforcement learning-based joint scheduling method for quay cranes and AGVs, aiming to minimize vessel makespan. Using data from the automated terminal at Qingdao Port, they developed a real-time simulation system and demonstrated the superiority of their approach. However, these studies predominantly focus on automated terminal equipment, leaving the applicability to conventional container terminals insufficiently explored.

To address the integrated scheduling problem of intelligent handling equipment, Zhang et al.²⁷ proposed a scheduling network method based on reinforcement learning, which integrates network-based heuristic rules with both local and global state information, thereby enhancing the algorithm's generalization capability. Wei et al.²⁸ introduced a novel dynamic AGV scheduling method incorporating a self-attention mechanism, enabling the effective evaluation of the importance of different information in highly dynamic environments, thus alleviating traffic congestion. While these methods partially overcome the limitations of heuristic approaches in handling dynamic changes through the adaptive learning capabilities of reinforcement learning, their application remains primarily focused on the scheduling of automated equipment.

To address the limitations of traditional heuristics in handling system uncertainties, Zhang et al.²⁹ proposed a hyper-heuristic framework based on deep reinforcement learning. This approach integrates a data-driven heuristic selection module, significantly enhancing the capability of low-level heuristic algorithms to manage uncertainties, and demonstrating the potential of combining reinforcement learning with traditional methods. Additionally, Jin et al.³⁰ proposed an optimization method for container terminal truck scheduling based on deep reinforcement learning. By employing a dual-network structure that incorporates expert knowledge and state information, their method exhibited strong generalization capabilities. However, its single-cycle strategy only considers loading or unloading tasks, which limits its applicability in comprehensive scheduling scenarios. Studies have shown that dual-cycle operation strategies can significantly improve the utilization of quay cranes and horizontal transportation equipment, thereby enhancing overall terminal efficiency³¹. Nevertheless, current scheduling optimization research predominantly focuses on single-cycle strategies, with limited exploration of dual-cycle operation strategies in the dynamic environments of conventional container terminals.

Based on this, this study proposes a DDQN-based reinforcement learning optimization method for internal truck scheduling. By enabling continuous interaction between the environment and the agent, the proposed

method can dynamically perceive environmental changes and adjust decisions in real time, effectively adapting to the dynamic characteristics and stochastic variations of port operations. Furthermore, a dual-cycle operation strategy is integrated into the reinforcement learning framework, exploring its potential applications in internal truck scheduling to address existing research gaps. Additionally, this study models the terminal operating environment using graph theory and configures simulation parameters based on real terminal operation data, effectively overcoming the limitations of simulation methods caused by incomplete data within the reinforcement learning framework. Leveraging the dual-network architecture and experience replay mechanism of DDQN, the proposed method significantly improves sample efficiency and decision robustness while dynamically addressing multi-task coordination demands in complex environments. This approach offers an efficient and intelligent solution aimed at comprehensively enhancing terminal operational efficiency.

Truck scheduling problem and model

Description of the truck fleet scheduling problem

The layout of the container terminal at the port, as shown in Fig. 1, consists primarily of the quay crane operational area, the truck transportation area, and the yard crane operational area. Upon the arrival of a vessel, the quay crane lifts the containers and places them onto the trucks, which then transport the containers to the yard, where the yard crane positions them in their designated locations. The loading process is similar; the yard crane places the containers to be loaded onto the waiting trucks, which then transport them to the quay crane for placement into the vessel.

The problem investigated in this study can be described as follows: during the unloading process, the quay crane operational tasks are known, and during the loading process, the yard crane operational tasks are also known. Under these conditions, a dual-cycle operation strategy (capable of both loading and unloading) is employed to schedule the trucks for executing the corresponding container tasks. The objective is to minimize the vessel's port stay time while minimizing the maximum completion time of various loading and unloading equipment under reasonable truck scheduling operations.

Mathematical model for truck scheduling

The mathematical model provides a clear and precise description of the truck scheduling problem, defining key variables, constraints, and objective functions. This helps in systematically understanding the complexity of the scheduling problem and its critical influencing factors.

Assumptions

Assumption 1 The vessel stowage plan, quay crane operation sequence, and yard container positions are known in advance.

Assumption 2 Each truck can only transport one container at a time.

Assumption 3 The number of quay cranes, trucks, and yard cranes is known.

Assumption 4 The empty and loaded speeds of quay cranes, trucks, and yard cranes are known, and all trucks are identical.

Assumption 5 Each quay crane, truck, and yard crane can handle only one container per operation.

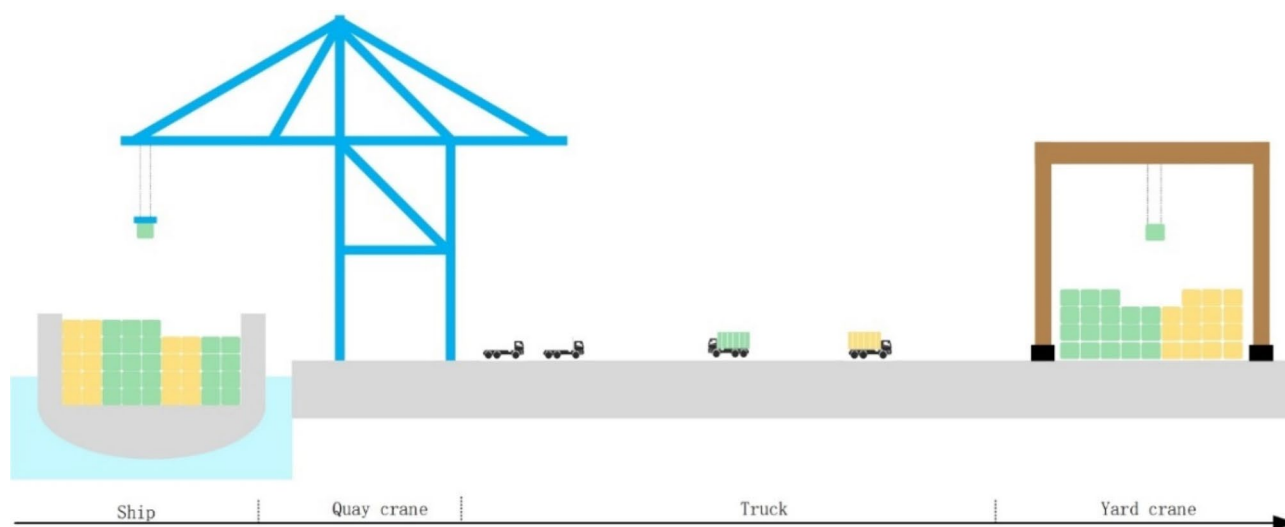


Fig. 1. Layout of the container terminal.

Symbol	Annotation
J	Set of trucks, $j \in J$
Y	Set of yard cranes, $y \in Y$
Q	Set of quay cranes, $q \in Q$
I	Set of import containers
O	Set of export containers
C	Set of container tasks, $c' \in C = I \cup O$
V_j	Empty travel speed of trucks
$V_{j'}$	Loaded travel speed of trucks
V_y	Travel speed of yard cranes
C_q^k	Number of tasks handled by the k th quay crane, $q \in Q$

Table 1. Constant symbols and annotations.

Symbol	Annotation
t_{cq}	The time taken by the quay crane to complete a container task c
$t_{cc'q}^s$	The preparation time for the quay crane q to transition to the next container handling task c' after completing the current task c
t_{cj}	The time taken by the yard truck j to complete a container handling task c
$t_{cc'j}^s$	The empty travel time for the yard truck j to reach the next container handling task c' after completing the current task c
t_{cy}	The time taken by the yard crane y to complete a container handling task c
$t_{cc'y}^s$	The preparation time for the yard crane y to switch to the next container handling task c' after completing the current task c

Table 2. Variable symbols and annotations.

Symbol	Annotation
X_{cj}	If container task c is assigned to yard truck j , its value is 1; otherwise, it is 0
Y_{cq}	If container task c is assigned to quay crane q , its value is 1; otherwise, it is 0
Z_{cy}	If container task c is assigned to yard crane y , its value is 1; otherwise, it is 0
$X_{cc'j}$	If yard truck j performs container task c' after completing container task c , its value is 1; otherwise, it is 0
$Y_{cc'q}$	If quay crane q performs container task c' after completing container task c , its value is 1; otherwise, it is 0
$Z_{cc'y}$	If yard crane y performs container task c' after completing container task c , its value is 1; otherwise, it is 0

Table 3. 0–1 variable symbols and annotations.

Notation definition

To establish a mathematical model for container terminal truck scheduling, the relevant symbols are defined as shown in Tables 1, 2 and 3.

Yard truck scheduling mathematical model

With the objective of minimizing the maximum completion time of each loading and unloading equipment, the yard truck scheduling mathematical model is designed as follows:

$$M_1 = \sum_{q=1}^Q \sum_{c=1}^C t_{cq} Y_{cq} + \sum_{q=1}^Q \sum_{c=1}^C \sum_{c'=1}^C t_{cc'q}^s Y_{cc'q} \tag{1}$$

$$M_2 = \sum_{j=1}^J \sum_{c=1}^C t_{cj} X_{cj} + \sum_{j=1}^J \sum_{c=1}^C \sum_{c'=1}^C t_{cc'j}^s X_{cc'j} \tag{2}$$

$$M_3 = \sum_{y=1}^Y \sum_{c=1}^C t_{cy} Z_{cy} + \sum_{y=1}^Y \sum_{c=1}^C \sum_{c'=1}^C t_{cc'y}^s Z_{cc'y} \tag{3}$$

$$M = \min(\max(M_1 + M_2 + M_3)) \tag{4}$$

$$\sum_{c=1}^C X_{cj} = 1; \forall j \in J \tag{5}$$

$$\sum_{j=1}^J X_{cj} = 1; \forall c \in C \tag{6}$$

$$\sum_{j=1}^J X_{cc'j} = 1; \forall c' \in C \tag{7}$$

$$\sum_{q=1}^Q Y_{cq} = 1; \forall c \in C \tag{8}$$

$$\sum_{q=1}^Q Y_{cc'q} = 1; \forall c' \in C \tag{9}$$

$$\sum_{y=1}^Y Z_{cy} = 1; \forall c \in C \tag{10}$$

$$\sum_{y=1}^Y Z_{cc'y} = 1; \forall c' \in C \tag{11}$$

The completion times for the quay crane, yard truck, and yard crane are represented by Eqs. (1), (2), and (3), respectively. Equation (4) defines the objective function, where MMM denotes the minimization of the maximum completion time of each piece of loading and unloading equipment; Eq. (5) ensures that each yard truck completes only one container task; Eqs. (6) and (7) ensure that each container is handled by only one yard truck; Similarly, Eqs. (8) and (9) ensure that each container is handled by only one quay crane; Eqs. (10) and (11) ensure that each container is managed by only one yard crane.

Graph modeling

Following the development of the mathematical model for yard truck scheduling, this paper introduces a graph theory approach to further model the port environment. The constraints from mathematical model are mapped into a graph-theoretic framework, providing a more intuitive and flexible representation of the actual terminal operations. Compared to traditional mathematical modeling, graph theory offers distinct advantages. Firstly, graph theory uses nodes and edges to represent the terminal environment, offering a structured and intuitive modeling approach. This makes it easier to visualize and concretize the complexity of the terminal environment, particularly when addressing the intricate network structures involved in yard truck scheduling processes³². Additionally, graph theory possesses strong scalability and adaptability, allowing for the seamless integration of various optimization algorithms to tackle complex scheduling problems. Therefore, adopting a graph-based model provides a more systematic and visual tool to further enhance the optimization of yard truck scheduling.

Figure 2 illustrates the container task network modeled using graph theory. To ensure clarity in both visualization and detailed description, the diagram displays only 10 container loading and unloading tasks. These tasks are handled by two quay cranes, three trucks, and two yard cranes. As depicted, the first five tasks correspond to unloading operations, while the latter five represent loading operations. For simulation purposes, it is assumed that Quay Crane 1 completes the first five unloading tasks, while Quay Crane 2 handles the last five loading tasks. In the diagram, the red, blue-purple, and yellow nodes represent specific operational tasks, with

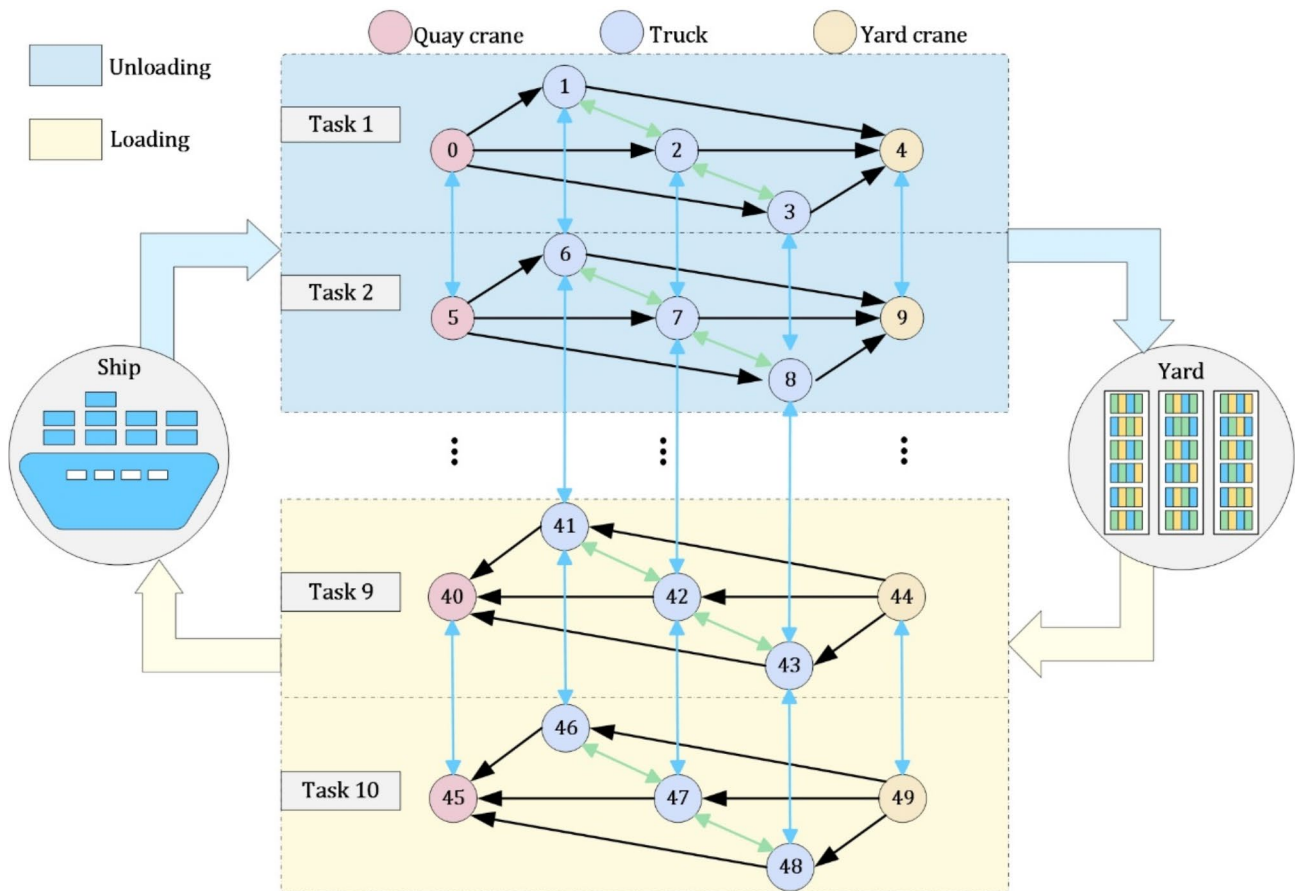


Fig. 2. Container loading and unloading task network diagram.

the lines between the nodes indicate the dependencies between and the workflow of the loading and unloading equipment. For instance, Node 0 represents Quay Crane 1, Nodes 1, 2, and 3 represent three different trucks, and Node 4 corresponds to Yard Crane 1. Together, these five task nodes form a single container handling task.

In the graph theory model, the constraints from the mathematical model are represented by edges and nodes. Specifically, the uniqueness constraint in the truck scheduling mathematical model, which dictates that each container task must be assigned to only one piece of equipment (such as a truck, quay crane, or yard crane), is incorporated through blue and green bidirectional edges. The corresponding constraint conditions are (6), (7), (8), (9), (10), and (11) as mentioned above. These bidirectional edges ensure that each container task is exclusively handled by the selected equipment during a specific phase, preventing simultaneous processing by multiple pieces of equipment.

For instance, in Fig. 2, Nodes 0 and 5 represent tasks assigned to Quay Crane 1, while Nodes 40 and 45 correspond to tasks handled by Quay Crane 2. This ensures that a single piece of handling equipment can manage only one container task at a time and cannot process multiple tasks simultaneously. Equipment must be released after completing one task before it can be reassigned to another. The blue bidirectional edges reflect the constraints in the mathematical model (specifically, constraints (5), (8), (9), (10), and (11)). These constraints collectively ensure the uniqueness of assignments and the rational allocation of resources within the scheduling model.

The green bidirectional edges in the figure connect tasks within the same truck transport phase, indicating that each container can only be transported by a single truck. These green bidirectional edges reflect the constraints in the mathematical model (see constraints (6) and (7)). The black unidirectional edges in the figure represent the execution sequence of the same container task, highlighting the dependencies between tasks, ensuring that container tasks are performed in the correct order.

The graph theory model exhibits strong scalability, enabling the seamless addition of new task nodes and edges. In the event of equipment malfunctions or temporary adjustments to the operational plan within the terminal environment, nodes and their corresponding edges can be removed from the scheduling network. Conversely, when new equipment is introduced or new operational plans are implemented, nodes and edges can be added accordingly. This flexible modeling approach enables better adaptation to the dynamic nature of terminal environment. Furthermore, the graph theory model provides enhanced flexibility in defining the terminal's state space for reinforcement learning, enabling it to more effectively capture real-time environmental changes.

Markov decision process

To further explore the optimal solution for the scheduling problem within the graph structure, this paper abstracts the decision-making process as a Markov Decision Process (MDP). Through this abstraction, the state transitions and decision-making behaviors in yard truck scheduling can be systematically described, with corresponding reward values assigned to each state and action. On this basis, a reinforcement learning-based approach is employed to solve the MDP, thereby effectively learning the optimal scheduling policy. This method dynamically adapts to changes in the complex operational environment and continuously optimizes the yard truck scheduling under conditions of uncertainty.

In the process of reinforcement learning, the agent learns the optimal scheduling policy through repeated interactions with the environment. The interaction between the agent and the environment is illustrated in Fig. 3. The agent observes the environment to obtain an observation, which represents the current state S . Based on the current state, the agent selects an action A which alters the environment. The environment then provides a corresponding reward R , and the agent, after receiving the reward R and the new state S_{t+1} , will determine the next action for the following time step.

The decision-making tasks in the dynamic container terminal environment are abstracted as a Markov Decision Process (MDP), represented by a six-tuple $(S, A, P, \gamma, R, \pi)$. S represents the set of states, encompassing all possible states that may arise during the yard truck scheduling process. A refers to the set of actions available for the yard truck to choose from at any given time. P denotes the probability of transitioning from the current state s to the next state s' when an action a from the action space is taken. γ is the discount factor, which balances the importance of immediate versus future rewards. R represents the reward function, which provides the agent with an immediate reward or penalty after performing an action. The policy function $\pi(s, a)$ represents the probability of executing action a in a given state s . The detailed construction methods of this MDP tuple are explained in Sect. 4.1–4.3.

State space

A data-driven container terminal simulation model was established based on graph theory, enabling the real-time retrieval of necessary state information. In this study, the system's state information is described using three node attributes: task node processing time, task node transition time, and the mapping entropy of task nodes. The formula for calculating the mapping entropy is presented in Eq. (12).

$$H_i = -\frac{K_i}{|V|-1} \sum_{j \in N_i} \log\left(\frac{K_i}{|V|-1}\right) \quad (12)$$

Here, K_i represents the degree of node V_i , V denotes the set of nodes, and N_i is the neighbor set of node V_i .

For each currently available yard truck task node, its processing time feature vector, transition time feature vector, and mapping entropy feature vector are calculated. These three feature vectors are concatenated into a single feature vector as the state input. The available task nodes change dynamically as yard trucks enter and exit working states. However, since the input dimensions of a neural network are typically fixed, this paper adopts

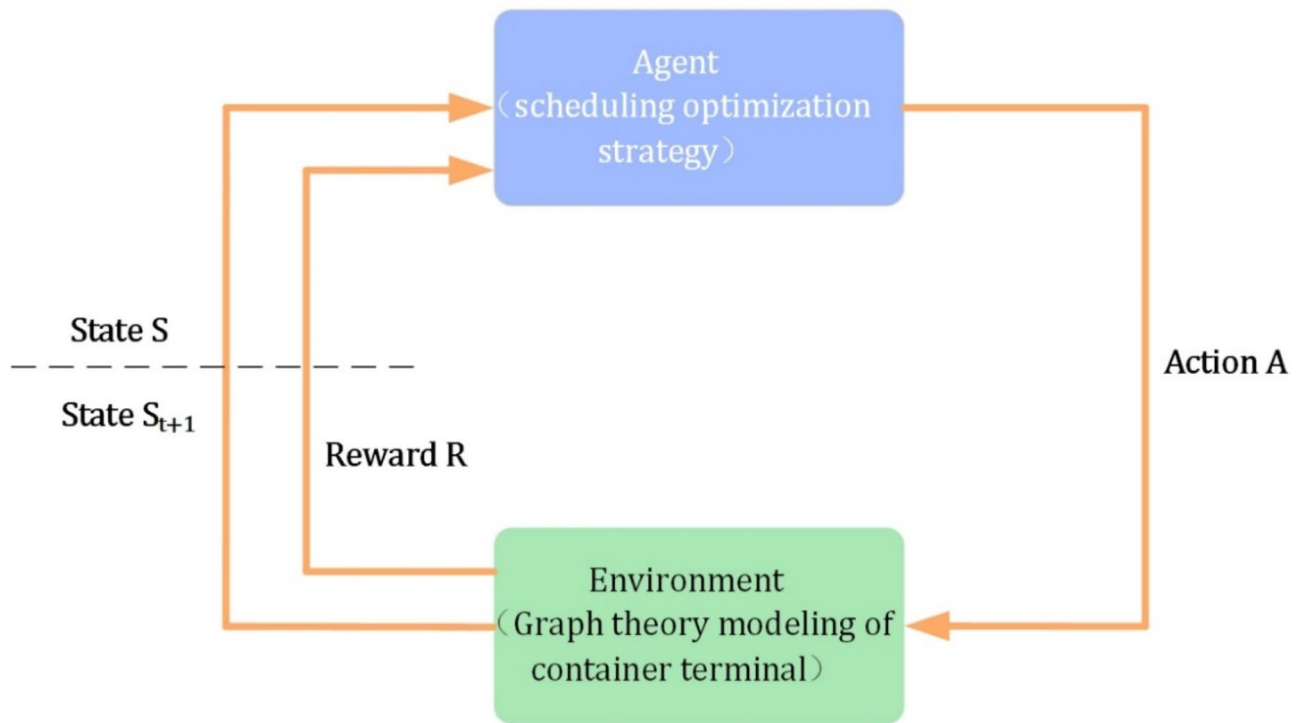


Fig. 3. Agent-environment interaction diagram.

a fixed input state dimension. Assuming the number of yard trucks is set to 5, the state dimension is fixed at 15. Similarly, if the number of yard trucks is set to 8, the state dimension is fixed at 24. When the number of task nodes decreases, any missing state dimensions are padded with -1 to maintain consistency. The fixed state dimension of 15 is determined by the three feature attributes of each truck and the specified number of trucks, which is set to 5. Consequently, the total dimension is $3 \times 5 = 15$. This fixed dimension ensures consistency in the neural network input, facilitating the handling of varying truck numbers while maintaining a uniform input format. Additionally, the use of -1 as a placeholder allows for adaptation to changes in available trucks.

The following are the specific steps for calculating the system state of the container terminal presented in this paper. The state dimension is determined based on the dimension set in Experiment 1 for small task loads.

By calculating the correlations between task nodes, the state space can capture more system information. This enhanced state representation provides the agent with richer decision-making data. It also helps the system better handle uncertainties in the terminal environment, such as equipment failures or task delays, thereby improving the overall system's robustness and responsiveness.

Algorithm 1 Calculate State from Available Nodes and Attributes

Input: available_nodes, attributes, fixed_dim = 15
Output: fixed_state

- 1 available_nodes_attrs ← []
- 2 for each attribute in attributes do
- 3 cal_attributes ← eval('self.sp' + attribute)
- 4 available_nodes_attrs ← append(cal_attributes(available_nodes))
- 5 end for
- 6 available_nodes_attrs ← to_array(available_nodes_attrs)
- 7 concatenated_state ← concatenate(available_nodes_attrs, axis=1)
- 8 current_length ← len(concatenated_state)
- 9 if current_length < fixed_dim then
- 10 padding_size ← fixed_dim − current_length
- 11 padding ← array of size padding_size filled with − 1
- 12 fixed_state ← concatenate(concatenated_state, padding)
- 13 else
- 14 fixed_state ← slice(concatenated_state, 0, fixed_dim)
- 15 end if
- 16 return fixed_state

Action space

In the process of yard truck scheduling, the scheduling system needs to coordinate globally to select the most suitable yard truck to execute the current container task. A single heuristic rule may prevent the system from making optimal decisions in certain scenarios. Therefore, incorporating different heuristic rules as part of the action set for reinforcement learning provides the agent with a richer range of action choices. This allows the algorithm to select the most appropriate action in various states, enhancing decision-making flexibility and robustness. Heuristic rules are often based on human expert experience, and integrating them into reinforcement learning algorithms effectively combines human intelligence with machine learning, improving the intelligence of the scheduling system. In this paper, five heuristic rules—SST, LTT, SPT, LPT, and LME—are selected to form the action space of the reinforcement learning algorithm. The descriptions of these heuristic rules are shown in Table 4. The agent, based on the current state, selects an action to execute according to Eq. (13), and the algorithm then identifies the optimal yard truck task node to execute the current container task based on the chosen action.

$$a^* = \underset{a_i}{\operatorname{argmax}} Q(a_i) \quad (13)$$

Here, a_i represents one of the five heuristic rules selected in this paper, and $Q(a_i)$ is the score of action a_i . a^* is the action with the highest score.

Reward function

The reward function plays a guiding and motivating role in reinforcement learning by helping the agent learn the optimal strategy. After the agent takes an action based on the current state, the algorithm determines the yard truck node that will execute the container task according to the selected heuristic rule. Once the yard truck completes the task, the reward function evaluates the effectiveness of the agent's action in a given state,

Heuristic Rule	Rule Descriptions	Advantages
STT	Shortest task switching time	Reducing idle time and waiting time between tasks, thereby improving overall operational efficiency
LTT	Longest task switching time	Preventing system bottlenecks, reducing task backlogs, and improving system stability and robustness
SPT	Shortest processing time	Reducing overall task waiting time, increasing system processing speed, and improving overall operational efficiency
LPT	Longest processing time	Reducing task backlogs and balancing system load to avoid prolonged tasks delaying the overall system progress
LME	Maximum mapping entropy	Able to quantify the importance and complexity of nodes, ensuring the efficient operation of key nodes, thereby optimizing resource allocation and scheduling efficiency across the entire system

Table 4. Heuristic rules for action space.

helping the agent distinguish between the quality of different actions. By using rewards as guidance, the agent is incentivized to take actions that yield the highest rewards. In alignment with the goal of the yard truck scheduling problem, this paper defines the reward function R using the objective function M from the yard truck scheduling mathematical model. The reward function is formulated as shown in Eq. (14).

$$R = \begin{cases} 0, & \text{Scheduling task incomplete} \\ L \frac{1}{M}, & \text{Scheduling task complete} \end{cases} \quad (14)$$

In the formula, M represents the objective function from the previously mentioned yard truck scheduling mathematical model, which aims to minimize the maximum completion time of each loading and unloading equipment. L is a very small integer that serves as the gain of the reward. The shorter the completion time after scheduling, the greater the reward assigned to the agent. This reward mechanism encourages the agent to select and optimize strategies that minimize completion time, thereby enhancing the overall efficiency of the scheduling system.

DDQN algorithm for solving the yard truck scheduling problem

DQN

While the Q-learning algorithm performs well in solving simple scheduling tasks, it has notable limitations. Traditional Q-learning uses a tabular form to represent the value function, storing all Q-values in a structure called a Q-table. However, due to the finite size of the Q-table, real-world problems often have multi-dimensional state and action spaces, leading to the “curse of dimensionality,” which significantly impacts the convergence speed of Q-learning. As a result, Q-learning struggles to handle complex control problems involving high-dimensional states and actions³³. To solve this problem, the DeepMind team developed an innovative technique that cleverly combines reinforcement learning with deep neural networks³⁴. Instead of using a Q-table, a neural network is used to approximate the optimal action-value function $Q_*(s, a)$, referred to as a Deep Q-Network (DQN), denoted as $Q(s, a; \omega)$. Here, ω represents the parameters of the neural network, which are randomly initialized and subsequently updated to make the predicted values $Q(s, a; \omega)$ converge towards the optimal action-value function $Q_*(s, a)$. Temporal Difference (TD) learning is commonly used in reinforcement learning to update the network parameters, and the TD target in the DQN algorithm can be expressed by Eq. (15).

$$\hat{y}_t = r_t + \gamma \max_{a \in A} Q(s_{t+1}, a; \omega) \quad (15)$$

In the equation, r_t represents the current reward, γ is the discount factor, and s_{t+1} is the next state following the current state s_t . The TD target \hat{y}_t is the prediction made by the neural network at time step $t + 1$, which is partially based on the observed reward. Replacing the optimal action-value function $Q_*(s, a)$ with the neural network $Q(s, a; \omega)$ gives the following expression, as shown in Eq. (16).

$$\underbrace{Q(s_t, a_t; \omega)}_{\text{predicted } \hat{q}_t} \approx r_t + \underbrace{\gamma \max_{a \in A} Q(s_{t+1}, a; \omega)}_{\text{TD target } \hat{y}_t} \quad (16)$$

The predicted value \hat{q}_t in the equation is the prediction made by the neural network at time step t . Both \hat{q}_t and \hat{y}_t are approximations of the optimal action-value function $Q_*(s, a)$, but the TD target \hat{y}_t is partially based on the actual observed reward, making it more reliable than \hat{q}_t . Therefore, the loss function is defined as shown in Eq. (17) to ensure that the output of the DQN more closely approximates the TD target value. The loss function is used to optimize and update the network parameters.

$$L = \frac{1}{2} (Q(s_t, a_t; \omega) - \hat{y}_t)^2 \quad (17)$$

DDQN

The Q-learning algorithm has a notable drawback: the DQN trained using it tends to overestimate the true value, and this overestimation is often non-uniform, which can lead to suboptimal performance of the DQN. To address this issue, Van Hasselt³⁵ proposed the Double Deep Q-Network (DDQN) algorithm. DDQN mitigates the overestimation problem by introducing two separate Q-networks. One is the Current Q-Network, used for action selection, and the other is the Target Q-Network, used for action evaluation. These two networks separate action selection from action evaluation, thereby reducing overestimation.

Figure 4 illustrates the DDQN-based collaborative scheduling training framework for truck dispatching developed in this study. The agent observes the current terminal environment to obtain a set of specific state information. Based on the current state, the agent uses the DDQN model to predict the Q-values for each possible action and selects the action with the highest Q-value, aiming to support the formation of an optimal scheduling strategy. Once an action is selected, the agent assigns the appropriate truck to begin the task. The truck's operation updates the overall terminal environment, modeled through a graph-based approach, and the environment provides feedback on the updated state and rewards or penalties. Through continuous interaction with the environment, the agent accumulates training data, which is used to update the DDQN network weights and improve Q-value estimation accuracy. As training progresses, the agent learns to identify and execute optimal scheduling strategies within the dynamic and complex port environment, thereby achieving high resource utilization and operational efficiency in truck scheduling tasks.

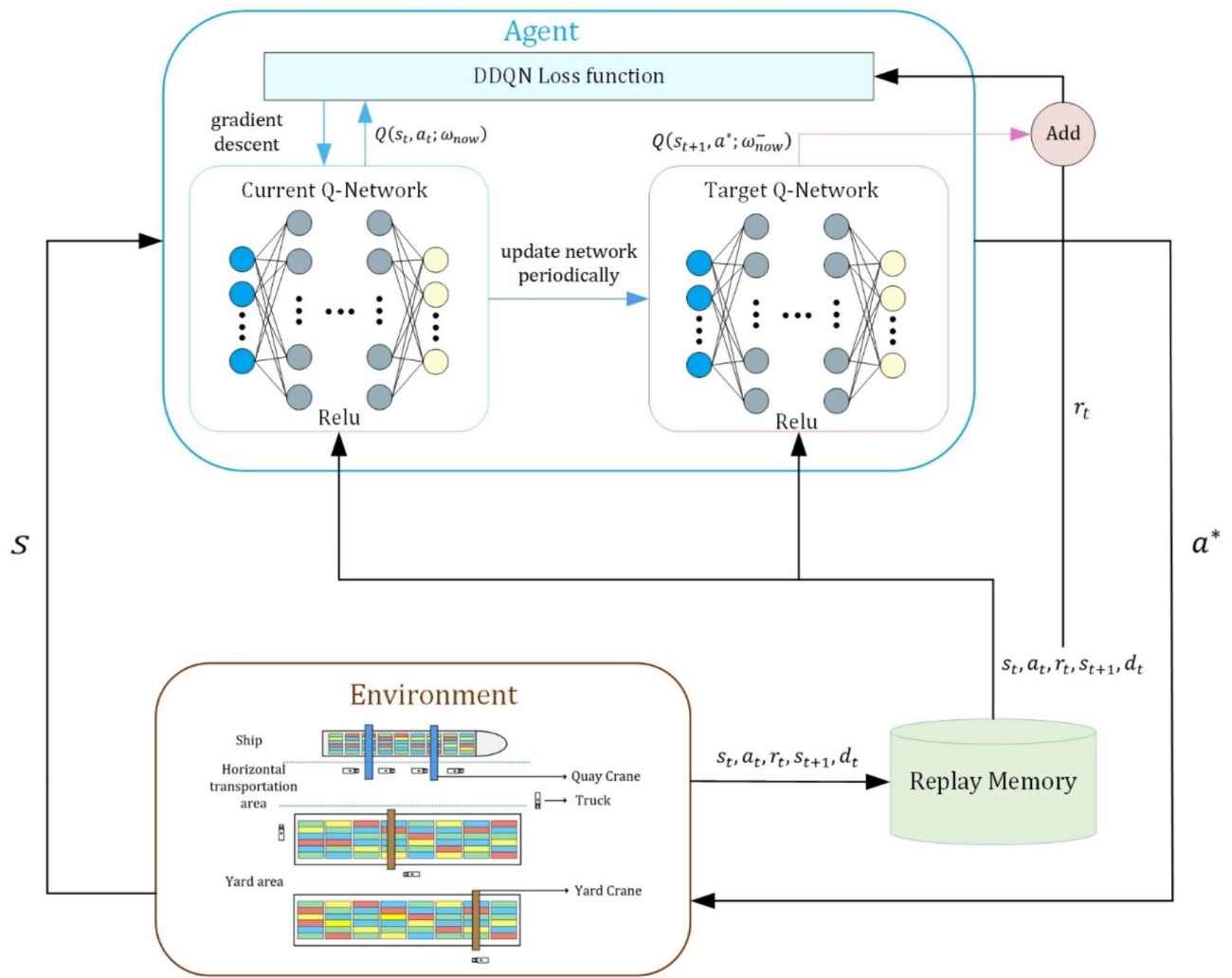


Fig. 4. DDQN-Based yard truck collaborative scheduling training framework diagram.

In deep reinforcement learning, the experience replay buffer plays a crucial role. It is primarily used to store the experiences generated by the agent’s interactions with the environment, aiding the training process of the DQN algorithm. The main advantages of the experience buffer are its ability to break the correlation between data, reduce overfitting, and enhance the training performance of the algorithm. Additionally, it increases data utilization, improving sample efficiency. Each experience sample in this paper consists of five components: the current state (s_t), the selected action (a_t), the obtained reward (r_t), the next state after the action is executed (s_{t+1}), and a termination flag (d_t), which indicates whether the scheduling has finished. These experience samples are stored for the agent to use during future learning phases. Before training the network, the performance can be fine-tuned by adjusting the size of the experience replay buffer (buffer_size), which is set to 4000 in this paper. After determining the size of the replay buffer, the agent randomly samples from the experience buffer to perform training.

This paper adopts a dynamic epsilon-greedy decay strategy. In the early stages of training, the exploration rate is relatively high, allowing the agent to make more random action selections. As training progresses, the exploration rate gradually decreases, enabling the agent to rely more on its acquired knowledge for decision-making. This approach helps prevent model overfitting due to excessive strategy adjustments. A smaller learning rate during the later stages of training helps enhance the model’s generalization capability, allowing it to better adapt to unseen data.

As seen in Fig. 4, at each step, a quintuple ($s_t, a_t, r_t, s_{t+1}, d_t$) is randomly sampled from the experience replay buffer. Let the current Q-network and target Q-network have parameters ω_{now} and ω_{now}^- , respectively. A forward pass is performed through the current Q-network to obtain the predicted value \hat{q}_t , as shown in Eq. (18).

$$\hat{q}_t = Q(s_t, a_t; \omega_{now}) \tag{18}$$

Based on state s_{t+1} , the optimal action a^* is selected using the current Q-network, as shown in Eq. (19).

$$a^* = \underset{a \in A}{\operatorname{argmax}} Q(s_{t+1}, a; \omega_{now}) \quad (19)$$

The target Q-network is then used to evaluate the Q-value of this optimal action, as shown in Eq. (20).

$$\widehat{q}_{t+1} = Q(s_{t+1}, a^*; \omega_{now}^-) \quad (20)$$

The TD target and TD error are calculated, as shown in Eqs. (21) and (22), respectively.

$$\widehat{y}_t = r_t + \gamma \widehat{q}_{t+1} \quad (21)$$

$$\delta_t = \widehat{q}_t - \widehat{y}_t \quad (22)$$

These two neural networks have the same architecture but employ different parameter update strategies. The parameters ω of the current Q-network model are updated during each training iteration, primarily using the gradient descent algorithm. The gradient descent update for the current value network model parameters is performed as shown in Eq. (23).

$$\omega_{new} \leftarrow \omega_{now} - \alpha \cdot \delta_t \cdot \nabla_{\omega} Q(s_t, a_t; \omega_{now}) \quad (23)$$

In the equation, ω_{new} represents the new parameters of the current Q-network, and α is the learning rate. The parameters of the target Q-network are updated by directly copying the parameters from the current Q-network after a fixed time interval. This delayed update method enhances the stability of the algorithm's training. This paper designs a port environment simulation model based on graph theory and, on this foundation, proposes the Graph-DDQN Scheduler algorithm to achieve coordinated scheduling optimization of container trucks using the DDQN algorithm. The specific steps of this method are detailed in Algorithm 2.

Algorithm 2: Graph-DDQN Scheduler Algorithm

Input: Graph-based port environment model G , replay memory D , learning rate α , discount factor γ , exploration rate ϵ , number of episodes N , mini-batch size B , target network update frequency T

Output: Trained Q-network, optimal truck dispatching policy π^*

- 1 Initialize current Q-network parameters ω_{now}
- 2 Initialize target Q-network parameters $\omega_{\text{now}}^- \leftarrow \omega_{\text{now}}$
- 3 **for** $episode = 1$ to N **do**
- 4 Reset port environment G
- 5 Initialize state s (Ref: Algorithm 1)
- 6 Initialize episode return $episode_return = 0$
- 7 Initialize $d_t = \text{False}$
- 8 **while** d_t is not true **do**
- 9 **if** random number $< \epsilon$ **then**
- 10 Select a random action a
- 11 **else**
- 12 Select action $a^* = \arg \max_{a \in \mathcal{A}} Q(s_{t+1}, a; \omega_{\text{now}})$
- 13 **end**
- 14 Execute action a^* , observe reward r_t , next state s_{t+1} , and termination flag d_t
- 15 Update episode return $episode_return \leftarrow episode_return + r_t$
- 16 Store transition $(s_t, a_t, r_t, s_{t+1}, d_t)$ in replay memory D
- 17 Sample a random min-batch of size B transitions $(s_t, a_t, r_t, s_{t+1}, d_t)$ from D
- 18 **if** d_t is true **then**
- 19 Set $y_t = r_t$
- 20 **else**
- 21 Select action $a^* = \arg \max_{a \in \mathcal{A}} Q(s_{t+1}, a; \omega_{\text{now}})$
- 22 Set $y_t = r_t + \gamma Q(s_{t+1}, a^*; \omega_{\text{now}}^-)$
- 23 **end**
- 24 Perform a gradient descent step on $(y_t - Q(s_t, a_t; \omega_{\text{now}}))^2$ with respect to ω_{now}
- 25 **if** $episode \bmod T == 0$ **then**
- 26 Update target network: $\omega_{\text{now}}^- \leftarrow \omega_{\text{now}}$
- 27 **end**
- 28 $s_t \leftarrow s_{t+1}$
- 29 **end**
- 30 Decay exploration rate ϵ
- 31 **end**

Experimental design and results analysis**Instance settings**

This paper designs two sets of instance scales to verify the effectiveness and superiority of the DDQN-based collaborative yard truck scheduling optimization algorithm. The container terminal generates various task instruction queues based on the vessel's arrival times and navigation plans. This study selects two instance configurations reflecting commonly encountered task quantities in actual port operations: a small and a large set of container loading and unloading task instruction queues, representing typical cases within each category. The two instance configurations selected in this study are designed based on the different instruction queues encountered in actual port operations, representing typical small and large container loading and unloading

Instance name	Number of loading tasks	Number of unloading tasks		Number of quay cranes	Number of yard trucks	Number of yard cranes
Instance 1-20-5	10	10	2	5		2
Instance 2-80-8	40	40	2	8		2

Table 5. Instance configuration. Note: Instance name 1-20-5 refers to the first training instance, which includes 20 container tasks and 5 yard trucks.

hyperparameter	Instance 1-20-5 value		Instance 2-80-8 value
Learning rate	0.00001	0.000001	
Number of epochs	20,000	20,000	
Hidden layers	128	256	
Discount factor	0.96	0.96	
Experience replay buffer size	4000	4000	
Minimum fill size	1000	1000	
Batch size	128	128	
Target network update frequency	100	100	

Table 6. Hyperparameter value table.

task queues. The configuration of these instances is shown in Table 5. In all instances, the task transition times and operating times for quay cranes, yard trucks, and yard cranes at the container terminal are modeled to follow a normal distribution. This is based on an investigation conducted at a traditional container terminal at Ningbo-Zhoushan Port, allowing for a more accurate simulation of the complexity and variability inherent in real terminal operations.

To facilitate simulation experiments, this study assumes a known quay crane operation sequence. In the small-task scenario, the first 10 container tasks are designated for unloading operations by Quay Crane 1, while the subsequent 10 container tasks are designated for loading operations by Quay Crane 2. In the large-task scenario, the first 40 container tasks are assigned to Quay Crane 1 for unloading, and the subsequent 40 container tasks are assigned to Quay Crane 2 for loading.

Experimental hyperparameter settings

In this study, the DDQN algorithm designed is consistently used for training. The specific hyperparameters for the experiment are shown in Table 6. During the training process for both instances, the parameter L in the reward function is set to 1 and 3, respectively.

Benchmark

The DQN algorithm can learn optimal allocation strategies across multiple tasks is used as a benchmark for comparative analysis. In addition, this study will analyze and evaluate the heuristic scheduling rules based on five priority factors: LTT, LME, MPT, SPT, and STT, as comparison benchmarks.

Evaluation metrics

In the optimization of internal truck scheduling at container terminals, this study selects the makespan as the key evaluation metric. The makespan directly impacts vessel berthing time, which is a critical indicator for assessing terminal performance⁶. By minimizing the makespan, vessel turnaround time can be accelerated, terminal operational efficiency can be improved, and overall resource scheduling performance can be optimized. Compared to other metrics, the makespan more accurately reveals efficiency bottlenecks within the scheduling system, making it highly valuable for practical applications.

In addition, to comprehensively evaluate the optimization performance of the scheduling algorithm, this study introduces quay crane waiting time and truck utilization rate as auxiliary metrics. Quay crane waiting time measures the utilization efficiency of high-cost equipment, with lower waiting times indicating effective reduction of resource idleness. Truck utilization rate reflects the rationality of resource allocation and the efficiency of its usage. By integrating these metrics, a multidimensional evaluation framework is constructed to comprehensively assess the efficiency of the scheduling system and its adaptability in dynamic environments.

Experimental results analysis

The training of the DDQN and DQN algorithms was conducted on a host machine equipped with an i7-13700 H@2.4 GHz processor and 16GB of RAM. In the small-task scenario, the training process took approximately 2 h, while in the large-task scenario, it required around 23 h. To clearly illustrate the convergence process of the algorithms, the training results were smoothed using a sliding window of size 500. The cumulative reward convergence trends for the small-task and large-task scenarios are shown in Fig. 5(a) and 5(b), respectively.

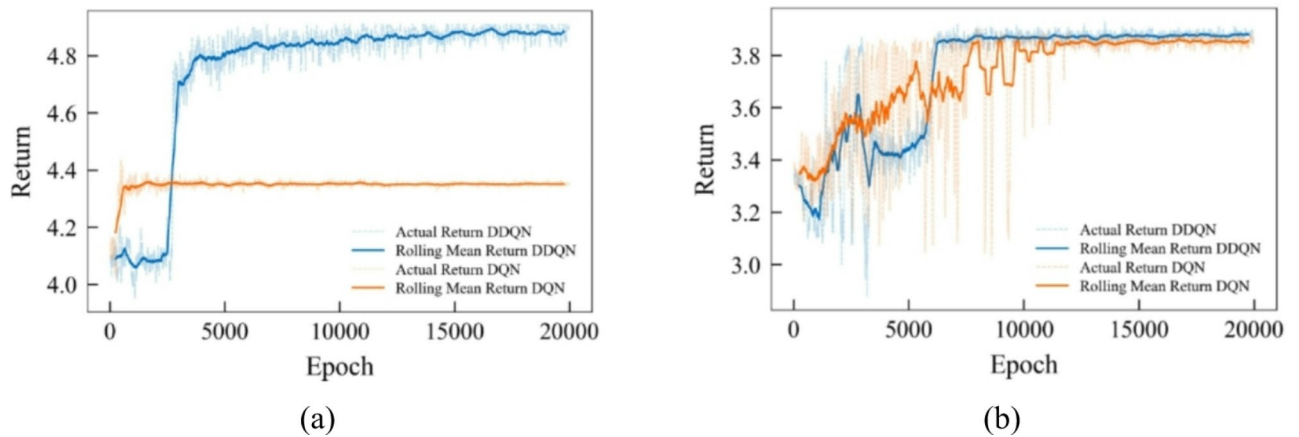


Fig. 5. Cumulative Reward Convergence Plot. **(a)** Small task scenario **(b)** Large task scenario.

From Fig. 5(a), it can be observed that the DDQN algorithm exhibits low reward values during the initial training phase. However, between 2500 and 5000 episodes, the reward values increase rapidly to above 4.8, eventually stabilizing at 4.9. In contrast, the DQN algorithm converges more quickly during the initial phase but ultimately stabilizes at a reward value of only 4.35. The rapid early convergence of DQN is attributed to its tendency for Q-value overestimation during evaluation, resulting in inflated initial rewards. However, due to insufficient strategy reliability, the final reward values are lower and exhibit poorer stability. In comparison, DDQN effectively mitigates the Q-value overestimation issue by decoupling action selection from value evaluation, enabling the model to identify more reliable strategies in later stages and enhancing both performance and stability.

From Fig. 5(b), it can be observed that in the more complex state space of the large-task scenario, both DDQN and DQN undergo extended exploration phases. DDQN achieves rapid convergence after approximately 6500 episodes, whereas DQN requires up to 12,000 episodes to converge, with its final reward value still lower than that of DDQN. The superiority of DDQN in complex environments can be attributed to its higher sample efficiency and strategy optimization capability. By leveraging the experience replay mechanism and dual-network architecture, DDQN effectively utilizes historical data, significantly mitigating the negative impact of complex states on strategy stability.

In summary, the proposed DDQN algorithm not only effectively mitigates the overestimation bias inherent in DQN but also adapts more efficiently to complex and dynamic environments. It achieves higher convergence efficiency and strategy reliability, demonstrating significant performance advantages in scheduling optimization tasks.

Figure 6 presents the Gantt charts illustrating the scheduling results of the DDQN and DQN algorithms under both small-task and large-task scenarios, providing a visual representation of the temporal arrangement of container tasks and the utilization of equipment resources. Taking Fig. 6(a) as an example, the horizontal axis represents the time, while the vertical axis denotes the list of equipment. Specifically, M1-1 and M1-2 correspond to the first and second quay cranes in the quay crane stage, respectively; M2-1 to M2-5 represent the five container trucks in the container truck stage; and M3-1 and M3-2 correspond to the first and second yard cranes in the yard crane stage, respectively.

From Fig. 6(a) and 6(b), it can be observed that under the DDQN algorithm, the task distribution among the five trucks is more balanced, with fewer task switches and reduced waiting times. In contrast, under the DQN algorithm, the first and second trucks are overburdened with tasks, while the fourth truck experiences significant idle time. Similarly, in the large-task scenario, the comparison between Fig. 6(c) and 6(d) further demonstrates that the scheduling strategy of DDQN results in more reasonable task allocation, effectively improving the utilization efficiency of truck resources.

This phenomenon further validates the superiority of the DDQN algorithm proposed in this study. By utilizing a dual-network architecture to mitigate the overestimation of Q-values and incorporating an experience replay mechanism to fully leverage historical data, the DDQN algorithm can more accurately capture the characteristics of task distribution. This enables the dynamic adjustment of scheduling strategies, thereby achieving more efficient resource allocation and overall scheduling performance in complex scenarios.

To more comprehensively validate the superiority of the proposed DDQN algorithm, the algorithm was executed 10 times under different random seeds based on the instance configuration shown in Table 5, and the average of the resulting outcomes was taken as the experimental result. The experimental results were analyzed using the three evaluation metrics defined in Sect. 6.4, with the detailed comparisons presented in Tables 7 and 8.

From the experimental results presented in Tables 7 and 8, it can be observed that in both small-task and large-task scenarios, the performance of different algorithms exhibits significant differences in terms of QC waiting time, makespan, and container truck utilization. Among these metrics, makespan, as the most critical evaluation criterion, directly reflects the overall efficiency of the scheduling system and serves as an essential standard for assessing the effectiveness of scheduling strategies.

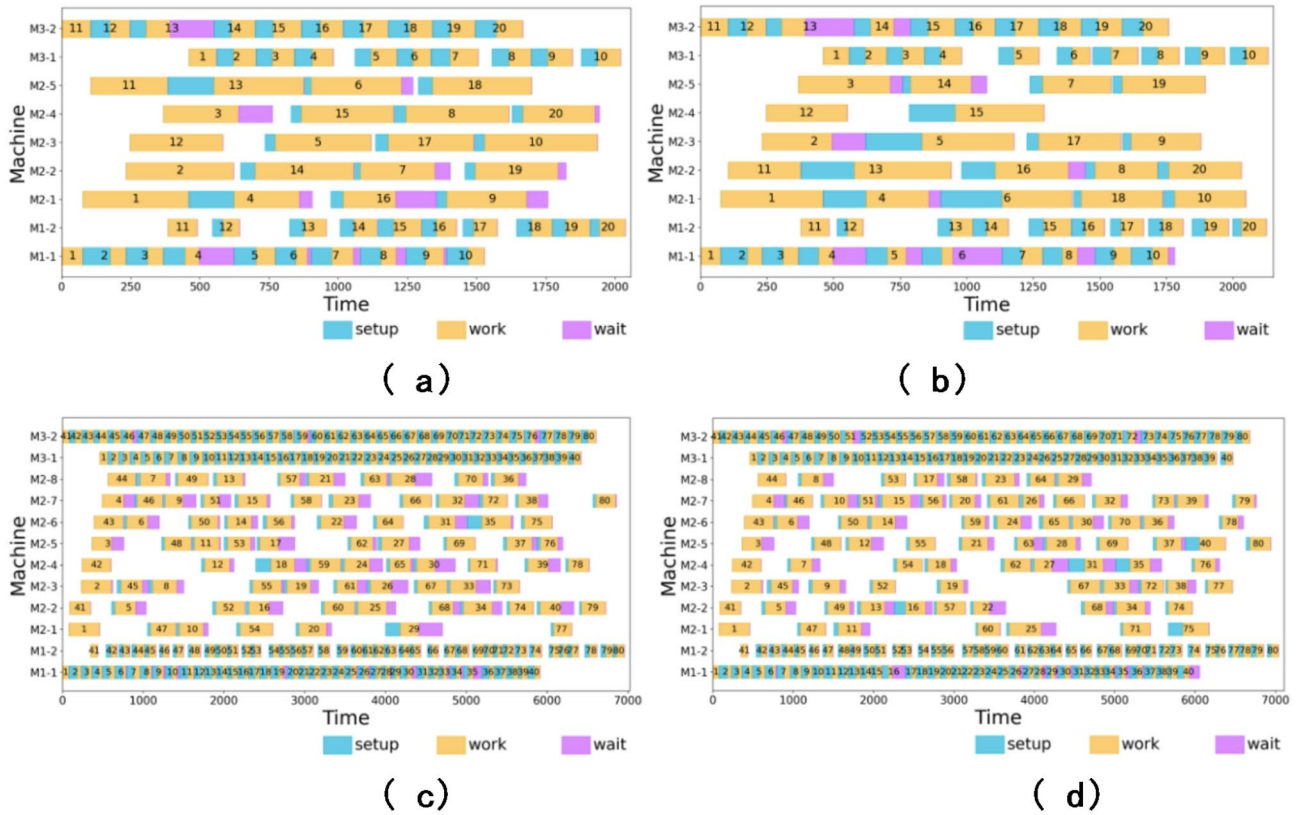


Fig. 6. Truck scheduling Gantt chart (a)DDQN - Small task scenario (b)DQN- Small task scenario (b)DDQN- Large task scenario (d) DQN- Large task scenario.

Method	Waiting Time of QCs (s)	Makespan (s)	Truck utilization
LTT	782	2490	76%
LME	614	2411	70%
MPT	480	2649	67%
SPT	551	2265	68%
STT	208	2174	68%
DQN	463	2131	71%
DDQN	212	2038	76%

Table 7. Comparison of experimental results for small task scenarios.

Method	Waiting Time of QCs (s)	Makespan (s)	Truck utilization
LTT	3033	8994	59%
LME	1901	8240	52%
MPT	1228	7894	62%
SPT	980	7800	48%
STT	420	7132	55%
DQN	547	7036	56%
DDQN	404	6957	60%

Table 8. Comparison of experimental results for large task scenarios.

The scheduling scheme based on the STT rule outperforms other heuristic rules in minimizing QC waiting time and makespan. This superiority arises from the STT rule's core strategy of reducing task switching times, which effectively shortens task completion times by lowering the frequency of equipment transitions. However, due to its insufficient consideration of balancing container truck task load distribution, the utilization rate of container trucks remains relatively low. This indicates that while the STT rule demonstrates certain advantages in local optimization, it exhibits shortcomings in achieving balanced utilization of global resources.

In comparison, the DQN algorithm demonstrates superior performance over the STT rule in terms of makespan and container truck utilization; however, its QC waiting time remains relatively long. This is primarily attributed to the issue of Q-value overestimation in the DQN decision-making process, which leads to an imbalanced resource allocation during the initial phase of tasks, thereby affecting overall scheduling efficiency. Although DQN can achieve some degree of dynamic strategy adjustment through reinforcement learning, its capability for global optimization in complex task scenarios remains limited.

The proposed DDQN algorithm exhibits optimal performance in both scenarios, particularly excelling in the most critical metric, makespan. In the small-task scenario, the QC waiting time under DDQN is only 4 s longer than that of the STT rule, yet it achieves the shortest makespan and the highest container truck utilization. In the large-task scenario, DDQN significantly outperforms other algorithms in terms of makespan and QC waiting time, although its container truck utilization is slightly lower than that of the MPT rule. Nevertheless, its overall performance remains superior. These results demonstrate that DDQN effectively balances multiple scheduling objectives and optimizes resource allocation, with a particularly notable advantage in reducing makespan.

The bar charts in Figs. 7 and 8 visually demonstrate that, compared to the least effective LTT method, the DDQN algorithm achieves a 72.9% and 23.1% reduction in QC waiting time and makespan, respectively, in the small-task scenario. In the large-task scenario, these improvements increase to 86.7% and 22.6%, respectively. These visualized results further substantiate the significant advantages of the DDQN algorithm in optimizing scheduling efficiency.

In summary, the experimental results validate the superiority of the proposed DDQN algorithm. By employing a dual-network structure to mitigate the issue of Q-value overestimation and leveraging the experience replay mechanism to effectively learn from historical data, DDQN achieves dynamic strategy optimization. This enables the algorithm to not only significantly reduce makespan and QC waiting time but also improve the utilization efficiency of container truck resources. Consequently, DDQN provides a more efficient solution for scheduling operations in complex port environments.

Conclusion

This study proposes a Deep Reinforcement Learning method based on Deep Double Q-Networks (DDQN) to address the collaborative scheduling problem of container trucks at traditional container terminals. The system's state space, action space, and reward function were designed, modeling the container truck scheduling problem as a Markov Decision Process. A data-driven container terminal environment was constructed for interactive training with the agent. The trained model was tested using two sets of instances, and the experimental results demonstrated that the proposed algorithm effectively identifies optimal scheduling strategies, outperforming five other heuristic-based scheduling methods and the DQN algorithm overall. In the small task scenario, the DDQN algorithm reduced QCs waiting time and maximum completion time by 72.9% and 23.1%, respectively, compared to the least effective method among the fixed heuristic rules. In the large task scenario, the DDQN algorithm achieved reductions of 86.7% in QCs waiting time and 22.6% in maximum completion time compared to the same baseline. The testing results validate the effectiveness and superiority of the proposed algorithm. The

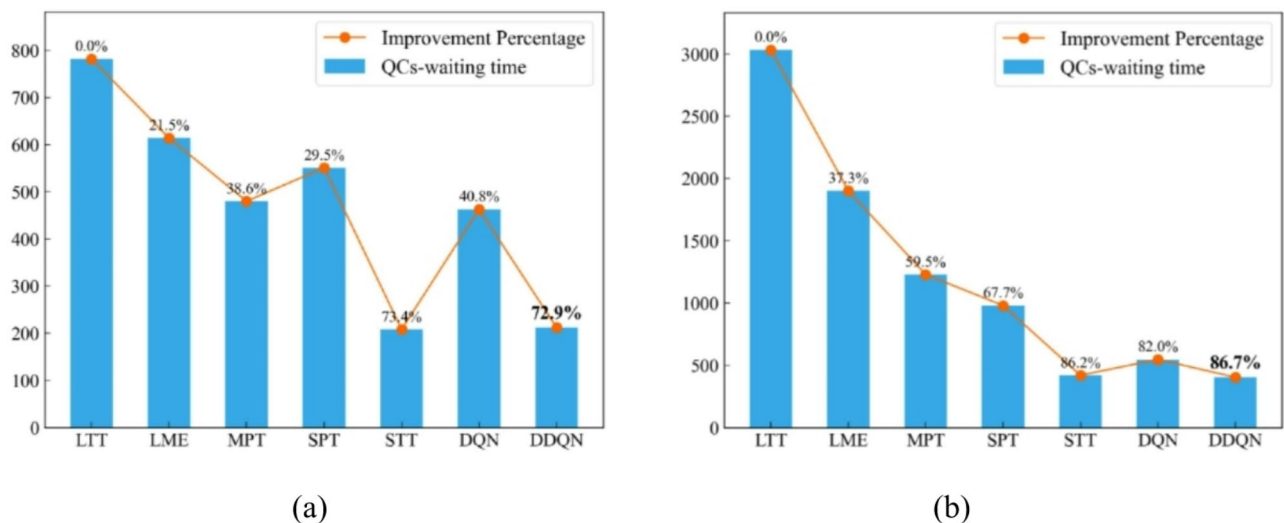


Fig. 7. Chart of QC Waiting Time Improvement (a) Small task scenario (b) Large task scenario.

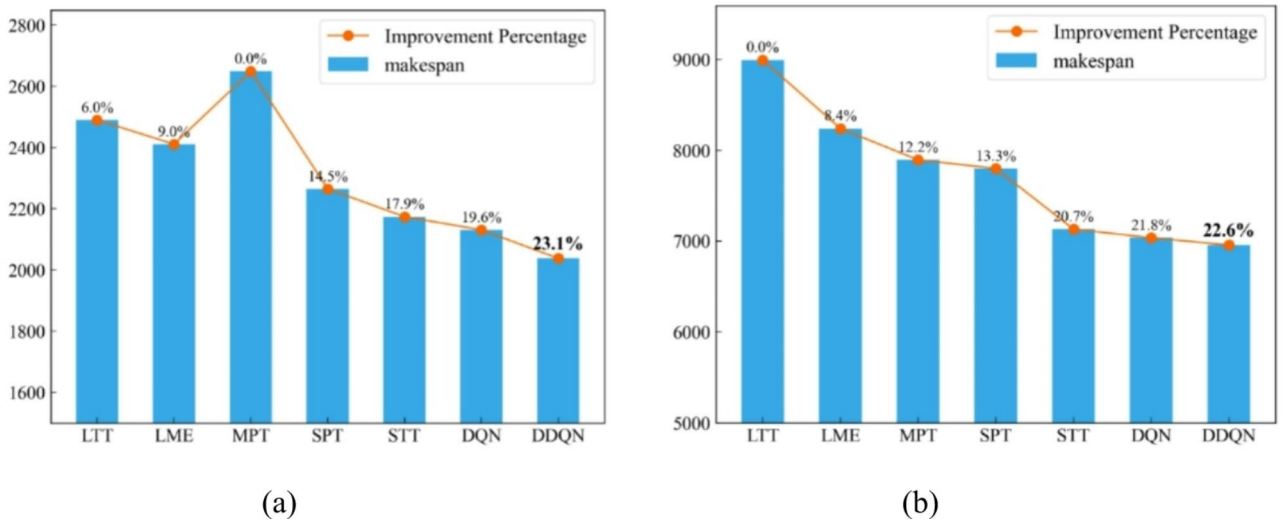


Fig. 8. Chart of Makespan Improvement (a) Small task scenario (b) Large task scenario.

DDQN-based scheduling algorithm presented in this study offers valuable insights for the digital and intelligent transformation of traditional container terminals.

Future research can consider the following two aspects: first, the current state space employs a fixed dimension, which can be addressed by utilizing Graph Neural Networks (GNNs) to handle varying dimensionality issues; second, collision avoidance and dynamic path planning for container trucks can be integrated into real-time scheduling to further enhance the effectiveness and safety of the scheduling process.

Data availability

The data that support the findings of this study are available from Ningbo Daxie Container Terminal Company Ltd but restrictions apply to the availability of these data, which were used under license for the current study, and so are not publicly available. Data are however available from the corresponding author upon reasonable request and with permission of Ningbo Daxie Container Terminal Company Ltd.

Received: 8 November 2024; Accepted: 18 February 2025

Published online: 26 February 2025

References

- Yan, R., Wang, S., Zhen, L. & Laporte, G. Emerging approaches applied to maritime transport research: Past and future. *Commun. Transp. Res.* **1**, 100011. <https://doi.org/10.1016/j.commtr.2021.100011> (2021).
- Verschuur, J., Koks, E. E. & Hall, J. W. Ports' criticality in international trade and global supply-chains. *Nat. Commun.* **13**, 4351. <https://doi.org/10.1038/s41467-022-32070-0> (2022).
- Yu, H., Deng, Y., Zhang, L., Xiao, X. & Tan, C. Yard operations and management in automated container terminals: A review. *Sustainability* **14**, 3419. <https://doi.org/10.3390/su14063419> (2022).
- Kizilay, D., Hentenryck, P. V. & Eliiyi, D. T. Constraint programming models for integrated container terminal operations. *Eur. J. Oper. Res.* **286**, 945–962. <https://doi.org/10.1016/j.ejor.2020.04.025> (2020).
- Zou, W. Q., Pan, Q. K. & Wang, L. An effective multi-objective evolutionary algorithm for solving the AGV scheduling problem with pickup and delivery. *Knowl. Based Syst.* **218**, 106881. <https://doi.org/10.1016/j.knosys.2021.106881> (2021).
- Naeem, D., Gheith, M. & Eltawil, A. A comprehensive review and directions for future research on the integrated scheduling of quay cranes and automated guided vehicles and yard cranes in automated container terminals. *Comput. Ind. Eng.* **179**, 109149. <https://doi.org/10.1016/j.cie.2023.109149> (2023).
- Li, W., Cai, L., He, L. & Guo, W. Scheduling techniques for addressing uncertainties in container ports: A systematic literature review. *Appl. Soft Comput.* **162**, 111820. <https://doi.org/10.1016/j.asoc.2024.111820> (2024).
- Gu, Q., Wang, Q., Li, X. & Li, X. A surrogate-assisted multi-objective particle swarm optimization of expensive constrained combinatorial optimization problems. *Knowl. Based Syst.* **223**, 107049. <https://doi.org/10.1016/j.knosys.2021.107049> (2021).
- Shao, Y. et al. Multi-Objective neural evolutionary algorithm for combinatorial optimization problems. *IEEE Trans. Neural Netw. Learn. Syst.* **34**, 2133–2143. <https://doi.org/10.1109/TNNLS.2021.3105937> (2023).
- Zhao, B. et al. A Privacy-Preserving genetic algorithm for combinatorial optimization. *IEEE Trans. Cybern.* **54**, 3638–3651. <https://doi.org/10.1109/TCYB.2023.3346863> (2024).
- Ding, H. & Gu, X. Improved particle swarm optimization algorithm based novel encoding and decoding schemes for flexible job shop scheduling problem. *Comput. Oper. Res.* **121**, 104951. <https://doi.org/10.1016/j.cor.2020.104951> (2020).
- Zhang, F., Mei, Y., Nguyen, S. & Zhang, M. Evolving scheduling heuristics via genetic programming with feature selection in dynamic flexible Job-Shop scheduling. *IEEE Trans. Cybernetics* **51**, 1797–1811. <https://doi.org/10.1109/tyb.2020.3024849> (2021).
- Wang, G. G., Gao, D. & Pedrycz, W. Solving multiobjective fuzzy Job-Shop scheduling problem by a hybrid adaptive differential evolution algorithm. *IEEE Trans. Industr. Inf.* **18**, 8519–8528. <https://doi.org/10.1109/tii.2022.3165636> (2022).
- Miao, C., Chen, G., Yan, C. & Wu, Y. Path planning optimization of indoor mobile robot based on adaptive ant colony algorithm. *Comput. Ind. Eng.* **156**, 107230. <https://doi.org/10.1016/j.cie.2021.107230> (2021).
- Pehlivanoglu, Y. V. & Pehlivanoglu, P. An enhanced genetic algorithm for path planning of autonomous UAV in target coverage problems. *Appl. Soft Comput.* **112**, 107796. <https://doi.org/10.1016/j.asoc.2021.107796> (2021).

16. Shi, K., Wu, Z., Jiang, B. & Karimi, H. R. Dynamic path planning of mobile robot based on improved simulated annealing algorithm. *J. Franklin Inst.* **360**, 4378–4398. <https://doi.org/10.1016/j.jfranklin.2023.01.033> (2023).
17. Chen, X., Bai, R., Qu, R. & Dong, H. Cooperative double-layer genetic programming Hyper-Heuristic for online container terminal truck dispatching. *IEEE Trans. Evol. Comput.* **27**, 1220–1234. <https://doi.org/10.1109/tevc.2022.3209985> (2023).
18. Luo, J. & Wu, Y. Scheduling of container-handling equipment during the loading process at an automated container terminal. *Comput. Ind. Eng.* **149**, 106848. <https://doi.org/10.1016/j.cie.2020.106848> (2020).
19. Yue, L. J., Fan, H. M. & Fan, H. Blocks allocation and handling equipment scheduling in automatic container terminals. *Transp. Res. Part. C: Emerg. Technol.* **153**, 104228. <https://doi.org/10.1016/j.trc.2023.104228> (2023).
20. Lu, Y. & Le, M. The integrated optimization of container terminal scheduling with uncertain factors. *Comput. Ind. Eng.* **75**, 209–216. <https://doi.org/10.1016/j.cie.2014.06.018> (2014).
21. Castilla-Rodríguez, I., Expósito-Izquierdo, C., Melián-Batista, B., Aguilar, R. M. & Moreno-Vega, J. M. Simulation-optimization for the management of the transshipment operations at maritime container terminals. *Expert Syst. Appl.* **139**, 112852. <https://doi.org/10.1016/j.eswa.2019.112852> (2020).
22. Kastner, M., Nellen, N., Schwiwientek, A. & Jahn, C. Integrated simulation-based optimization of operational decisions at container terminals. *Algorithms* **14**, 42. <https://doi.org/10.3390/a14020042> (2021).
23. Heuillet, A., Couthouis, F., & Díaz-Rodríguez, N. Explainability in deep reinforcement learning. *Knowl. Based Syst.* **214**, 106685. <https://doi.org/10.1016/j.knsys.2020.106685> (2021).
24. Wang, X. et al. Deep reinforcement learning: A survey. *IEEE Trans. Neural Netw. Learn. Syst.* **35**, 5064–5078. <https://doi.org/10.1109/TNNLS.2022.3207346> (2024).
25. Zheng, X., Liang, C., Wang, Y., Shi, J. & Lim, G. Multi-AGV dynamic scheduling in an automated container terminal: A deep reinforcement learning approach. *Mathematics* **10**, 4575. <https://doi.org/10.3390/math10234575> (2022).
26. Zhong, R., Wen, K., Fang, C. & Liang, E. in *13th Asian Control Conference (ASCC)*. 110–115 (IEEE). (2022).
27. Zhang, Z. et al. in *IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)* 1182–1186 (2022). (2022).
28. Wei, Q., Yan, Y., Zhang, J., Xiao, J. & Wang, C. A Self-Attention-Based deep reinforcement learning approach for AGV dispatching systems. *IEEE Trans. Neural Netw. Learn. Syst.* **35**, 7911–7922. <https://doi.org/10.1109/TNNLS.2022.3222206> (2024).
29. Zhang, Y. C., Bai, R. B., Qu, R., Tu, C. F. & Jin, J. H. A deep reinforcement learning based hyper-heuristic for combinatorial optimisation with uncertainties. *Eur. J. Oper. Res.* **300**, 418–427. <https://doi.org/10.1016/j.ejor.2021.10.032> (2022).
30. Jin, J., Cui, T., Bai, R. & Qu, R. Container Port truck dispatching optimization using Real2Sim based deep reinforcement learning. *Eur. J. Oper. Res.* **315**, 161–175. <https://doi.org/10.1016/j.ejor.2023.11.038> (2024).
31. Zhang, R., Jin, Z., Ma, Y. & Luan, W. Optimization for two-stage double-cycle operations in container terminals. *Comput. Ind. Eng.* **83**, 316–326. <https://doi.org/10.1016/j.cie.2015.02.007> (2015).
32. Xu, Y., Zhu, J. & Li, Q. Scheduling optimization for twin ASC in an automated container terminal based on graph theory. *Adv. Multimedia*. **2022**, 1–12. <https://doi.org/10.1155/2022/7641084> (2022).
33. Tong, Z., Ye, F., Liu, B., Cai, J. & Mei, J. DDQN-TS: A novel bi-objective intelligent scheduling algorithm in the cloud environment. *Neurocomputing* **455**, 419–430. <https://doi.org/10.1016/j.neucom.2021.05.070> (2021).
34. Tong, Z., Chen, H., Deng, X., Li, K. & Li, K. A scheduling scheme in the cloud computing environment using deep Q-learning. *Inf. Sci.* **512**, 1170–1191. <https://doi.org/10.1016/j.ins.2019.10.035> (2020).
35. Van Hasselt, H., Guez, A. & Silver, D. in *Proceedings of the AAAI conference on artificial intelligence*.

Acknowledgements

This work is financially supported by Ningbo Science and Technology Innovation 2025 Major Project, project code 2022Z228 and the Ningbo Natural Science Programme, Young Doctor Innovative Research Project, project code 2023J401;

Author contributions

Port data provision and compilation, R.Z., J.H., model construction, S.C., Q.L, L.M. CF.K., code formulation and development, S.C., Q.L., algorithm training and testing, S.C., Q.L., manuscript writing, S.C., Q.L, All authors have read and agreed to the published version of the manuscript.

Declarations

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to Q.L.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>.

© The Author(s) 2025