Transportation Research Part C 183 (2026) 105496

Contents lists available at ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

Learning to retrieve containers: A scale-diverse deep reinforcement learning approach for the container retrieval problem ⋆

Woo-Jin Shin, Inguk Choi, Sang-Hyun Cho, Hyun-Jung Kim*

Department of Industrial and Systems Engineering, Korea Advanced Institute of Science and Technology, 34141, Daejeon, Republic of Korea,

ARTICLE INFO

Keywords:
Container retrieval problem
Container relocation problem
Automated container terminals
Maritime logistics
Deep reinforcement learning
Combinatorial optimization

ABSTRACT

This study addresses the container retrieval problem (CRP), a key challenge in the storage yards of automated container terminals where operational efficiency directly affects vessel turnaround time and yard congestion. In storage yards, containers are stacked vertically to maximize space utilization; however, accessing one located below others requires relocating the blocking containers, leading to additional crane movements and delays. The CRP involves retrieving containers from multiple bays in a specified order while minimizing the total working time of the yard crane, with relocation position decisions being critical. The CRP poses several practical challenges: despite being $\mathcal{NP}$-hard, real-world instances often involve hundreds of containers, requiring high-quality solutions in real time; yard configurations also vary widely and change frequently, demanding methods that adapt effectively to arbitrary layouts. We propose a novel deep reinforcement learning approach incorporating (1) a size-agnostic network architecture, enabling a single trained network to handle diverse yard configurations, and (2) a scale-diverse learning framework, which trains on a various yard scales using a normalized loss to improve generalization and scalability. Experiments on well-known benchmarks with several hundred containers show that the proposed method substantially outperforms existing baselines across a wide range of yard sizes. It also scales to instances with thousands of containers and maintains strong performance in dynamic settings where retrieval orders are revealed online. Solutions are produced within a second for realistic instances, confirming its effectiveness and practical applicability in real-world automated container terminals. The implementation and datasets used in this study are publicly available in the GitHub repository: https://github.com/operagang/CRP_RL.

1. Introduction

Container terminals play a pivotal role in the global supply chain by enabling the seamless transfer of goods between oceangoing vessels and land-based transportation modes such as trucks and trains. As maritime logistics accounts for over 90 % of global trade (Wen et al., 2024), the steady growth of international shipping over recent decades has led to a continuous increase in container volumes handled at ports worldwide (Kim, 2024). This persistent surge in throughput has heightened the operational complexity of

⋆ This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) [RS-2024-00334171, RS-2025-02216640].

* Corresponding author.
E-mail address: hyunjungkim@kaist.ac.kr (H.-J. Kim).

https://doi.org/10.1016/j.trc.2025.105496
Received 26 August 2025; Received in revised form 7 December 2025; Accepted 7 December 2025
Available online 17 December 2025

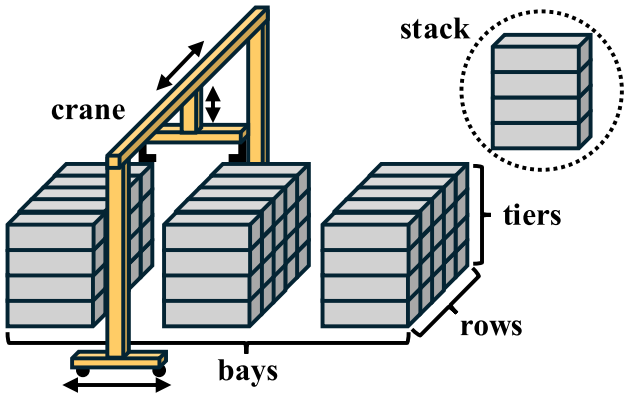


Fig. 1. Illustration of a container storage yard.

terminal activities, demanding smarter planning and automation to maintain smooth cargo flows and meet the growing demand for faster and more reliable logistics services.

In this context, container terminal operations can be broadly categorized into seaside and landside activities. Seaside activities include the allocation of quay cranes to vessels, the loading of export containers, and the discharging of import containers onto internal transport vehicles (Guo et al., 2024; Dragović et al., 2025). Landside operations–commonly referred to as yard operations–encompass the routing of internal trucks, the stacking of containers for storage, and the delivery of import containers to external trucks or trains for onward transport (Yue et al., 2023; Larsen et al., 2023).

This study focuses on one of the most critical landside operations–the container retrieval process conducted in the storage yard. Export and import containers are temporarily stored in the yard until they are retrieved by yard cranes according to the vessel stowage plan or the arrival schedule of external trucks (Hu et al., 2024). In order to maximize space utilization, containers in storage yards are stacked vertically. However, this arrangement often leads to situations in which a target container is blocked by others positioned above it. Because yard cranes can access only the topmost container in a stack, retrieving a buried container requires relocating the blocking containers to other stacks, a process that is both time-consuming and resource-intensive (Kim and Hong, 2006).

Several strategies have been proposed to alleviate this issue. One approach is to pre-plan storage locations so as to minimize future relocations (Tanaka et al., 2024; Wu and Jiang, 2025), while another is pre-marshalling, in which containers are reorganized prior to retrieval (Tanaka et al., 2019; Wu et al., 2023). Although such methods can reduce the number of relocations, eliminating them entirely is rarely feasible. Containers may arrive at the yard weeks in advance, complicating precise location planning, and limited yard space often forces stacking patterns that do not match the desired retrieval order (Lin et al., 2015). Consequently, some relocations remain unavoidable, underscoring the importance of effective container handling during the retrieval phase.

At the core of this challenge lies the container retrieval problem (CRP), first introduced by Lee and Lee (2010). The CRP begins with an initial yard configuration. As illustrated in Fig. 1, the yard consists of multiple bays, each containing several container stacks, and every container is assigned a predetermined retrieval order. The CRP entails determining a retrieval plan that serves all containers across multiple bays while strictly adhering to this order. The objective is to minimize the total working time of the yard crane, which includes travel time as well as the pick-up and set-down times for both retrieval and relocation moves. Efficiently solving this problem can substantially reduce critical operational costs in container terminals, including vessel turnaround time, truck waiting time, and yard congestion.

The CRP can be regarded as a practical extension of the more extensively studied blocks relocation problem (BRP). The BRP aims to minimize the number of relocations in single-bay scenarios (Kim and Hong, 2006), under the assumptions that retrieval is performed bay by bay, that crane travel time within a bay is negligible, and that only a few dozen containers are handled. In reality, however, container yards consist of multiple bays containing hundreds of containers, and retrieval orders may span across bays. In addition, intra-bay moves involve only trolley travel, whereas inter-bay moves require moving the entire crane, which takes significantly longer. This observation identifies crane working time as a more realistic and important performance measure, highlighting the significance of the CRP in addressing large-scale multi-bay scenarios. Furthermore, empirical evidence shows that minimizing crane working time generally reduces relocations, whereas the reverse does not necessarily hold (Voß and Schwarze, 2019).

The CRP presents several key challenges:

- Real-world yards may contain hundreds of containers, resulting in large-scale problem instances.
- Retrieval plans of superior quality need to be produced within real-time constraints.
- Yard configurations vary widely in terms of the number of bays, rows, tiers, and containers, requiring approaches that can adapt to any given layout.

Most prior studies have addressed the CRP through expert-designed heuristics; in this work, we aim to substantially enhance both solution quality and practical applicability.

To tackle these challenges, we propose a novel deep reinforcement learning (DRL) approach for the CRP. The method leverages long short-term memory (LSTM) units (Hochreiter and Schmidhuber, 1997) combined with an attention mechanism (Vaswani et al., 2017) to construct a size-agnostic network architecture capable of handling arbitrary yard configurations. Furthermore, we introduce a *scale-diverse learning framework* in which the network is trained on yard instances of varying sizes, using a normalized loss function to account for differences in objective scale. This framework allows the network to acquire robust policies and achieve scalability through generalization across varying instance sizes, including those not encountered during training. In addition, our approach eliminates the need for handcrafted, expert-designed features for container retrieval order representation.

Our experimental results show that the proposed DRL approach achieves excellent generalization across problem sizes, from tens to hundreds of containers, and consistently outperforms established baseline methods on well-known benchmarks. Even for problem instances with thousands of containers our method maintains high scalability and superior solution quality. The approach generates solutions in under one second for instances with fewer than one thousand containers and within one minute for instances involving several thousand containers, demonstrating strong real-world applicability. Furthermore, the method performs effectively in online settings where the retrieval order is revealed dynamically. In this context, the DRL model's rapid inference capability becomes particularly valuable, enabling real-time decision-making and reflecting its strong potential for deployment in dynamic and time-sensitive terminal environments.

The main contributions of this study are summarized as follows:

- **Complexity analysis and lower bound:** We formally analyze the computational complexity of the CRP and provide a lower bound for the crane working time, establishing a theoretical foundation for evaluating solution quality.
- **Model architecture and design:** We develop a DRL approach that integrates LSTM units and an attention mechanism to construct a size-agnostic network architecture. This design enables a single trained model to handle arbitrary yard configurations without retraining or feature engineering.
- **Learning framework for scalability:** We propose a novel scale-diverse learning framework that trains the model on yard instances of varying sizes using a normalized loss function. This framework allows the model to generalize effectively across different problem scales, including those unseen during training.
- **Comprehensive empirical validation:** Extensive experiments on standard benchmarks demonstrate that the proposed approach consistently outperforms existing methods in terms of both solution quality and computation time, across problem scales–including those beyond the training range.
- **Practical relevance and real-time applicability:** The proposed DRL model performs effectively in online scenarios where retrieval orders are revealed dynamically, demonstrating rapid inference capability and real-world applicability to dynamic, time-sensitive terminal environments.

The remainder of this paper is organized as follows: Section 2 reviews the relevant literature, Section 3 presents the problem description along with its complexity analysis and a lower bound, Section 4 details the proposed DRL approach, Section 5 reports the experimental results, and Section 6 concludes the paper.

2. Related works

In this section, we review the relevant literature. We first examine prior studies that have addressed the CRP, which is the focus of this study, followed by research on the related BRP that explicitly incorporates crane working time. We then summarize studies that have applied reinforcement learning (RL) techniques to similar problems. Finally, we highlight the limitations of these existing approaches and discuss how the present study aims to address them.

The CRP was first introduced by Lee and Lee (2010), who also proposed a benchmark set containing instances with 70 to 720 containers. They developed a mathematical programming-based three-phase heuristic: the first phase generates an initial solution; the second solves a binary integer program to minimize the number of relocations; and the third applies a mixed integer program to minimize crane working time. Forster and Bortfeldt (2012) proposed a tree search heuristic aimed at minimizing crane working time. Compared to Lee and Lee (2010), their method significantly reduced computation time from several hours to only a few seconds. Bian and Jin (2013) developed a three-phase heuristic combining local search and dynamic programming, optimizing both crane working time and the number of relocations, and producing solutions for benchmark instances within several minutes.

Lin et al. (2015) proposed a fast heuristic that selects appropriate relocation positions using expert-designed features to address the CRP. They conducted sensitivity analyses on crane working time by experimenting with different weights in the position priority evaluation. Kim et al. (2016) developed a heuristic rule capable of producing solutions very quickly, evaluating both crane working time and the number of relocations. Cifuentes and Riff (2020) applied a greedy randomized adaptive search procedure (GRASP) to minimize crane working time, obtaining solutions within several minutes. More recently, Đurasević and Đumić (2024) introduced a genetic programming (GP) approach for the automatic design of heuristics, targeting both crane working time and relocation minimization. Đurasević et al. (2025) extended this work to develop a GP framework capable of solving various CRP variants. Once trained, GP-based decision rules can generate solutions within seconds. Although the CRP has been widely studied, most existing approaches still depend on expert-designed features, and substantial potential remains for improving solution quality.

A well-known problem related to the CRP is the BRP, which considers only a single bay. Among the BRP studies, a few have taken crane working time into account; these works are briefly reviewed below. Ünlüyurt and Aydın (2012) addressed this variant by proposing a branch-and-bound algorithm, guaranteeing optimality but requiring long computation times, and a fast expert-designed rule that outperformed a crane working time-adapted version of Kim and Hong (2006)'s heuristic. Their experiments were limited to

50 containers in a single bay. Voß and Schwarze (2019) analyzed the relationship between crane working time and the number of relocations. Using a binary linear programming model, they showed that minimizing only relocations may lead to larger crane working times, whereas minimizing crane working time often yields solutions that also minimize relocations, highlighting the importance of considering crane working time directly.

Recently, RL approaches have been applied to related relocation problems. Only one study has applied DRL to the CRP that explicitly considers crane working time (Shin et al., 2025). They proposed a network incorporating expert-designed features combined with an attention mechanism to address the CRP. Although this represents the first DRL-based study for the CRP, the approach relies heavily on handcrafted features and has not fully addressed the scalability issue, despite being validated on instances involving several hundred containers.

Apart from that, all other RL studies on related problems have focused solely on the single-bay BRP, which aims to minimize the number of relocations rather than crane working time. Liu et al. (2023) was the first to apply DRL to the BRP, employing a dynamic attention model. However, its performance was inferior to that of existing algorithms. Yan et al. (2024) proposed a DRL approach that integrates a relocation-prediction network with beam search. While achieving fast solution times, their network architecture was designed for a fixed yard size (i.e., specific numbers of rows and tiers), thus limiting its applicability to larger configurations. Liu et al. (2025) addressed the BRP using a Q-learning framework, but their method depended on a non-approximated Q-table, which cannot handle unseen state-action pairs during training. Ma et al. (2025) extended the study by Liu et al. (2023) through a multi-decoder network to improve solution diversity; however, their approach still performed worse than conventional algorithms. Wang et al. (2025) developed a DRL-based method for the BRP with a stowage plan, combining it with imitation learning to enhance training efficiency. This method, however, required generating optimal solutions via the A* algorithm at each step, resulting in high computational costs. All of these studies were confined to minimizing the number of relocations, considered only single-bay layouts, and tested on relatively small instances (up to 30, 100, 64, 30, and 30 containers, respectively). In addition, none of them proposed mechanisms to improve generalization or scalability across varying yard sizes.

In summary, although numerous studies have been conducted on the CRP and its BRP variant, including RL-based approaches, they still suffer from the limitations discussed above. To address these gaps, this study proposes a novel DRL approach for the multi-bay CRP involving hundreds to thousands of containers, capable of producing high-quality solutions in real time. Our approach does not require handcrafted feature design for container retrieval orders and, through its size-agnostic architecture, enables a single trained network to handle any yard size. Furthermore, the proposed scale-diverse learning framework effectively solves a wide range of yard sizes and maintains strong performance even on problem instances significantly larger than those encountered during training. The effectiveness of the proposed method is validated through extensive comparisons with established CRP baselines on diverse datasets.

3. Container retrieval problem

In this section, we first describe the CRP and analyze its computational complexity. We then propose a lower bound for the crane working time in the CRP.

3.1. Problem description

We consider the CRP that arises in the storage yard of an automated container terminal. The yard consists of B bays and R rows, forming $B \times R$ stacks in total. Each stack can accommodate up to T containers vertically. Without loss of generality, bay indices are numbered from 1 to B based on their relative location; row indices are from 1 to R; and tier indices increase from the bottom to the top, ranging from 1 to T. The initial yard configuration contains N containers.

Each container is assigned a unique retrieval order and must be retrieved by a single crane in ascending order. At each decision point, the container with the minimum retrieval order is called the *target container*, and the stack containing it is referred to as the *target stack*. If the target container is buried beneath other containers, the blocking containers must be relocated to other stacks. The decision point lies in selecting the relocation destination for each blocking container.

Based on the above description of the CRP, we make the following assumptions, which are commonly adopted in the literature:

A1. Each bay has a dedicated retrieval point located at row 0, modeling a truck that accesses the yard from one side.
A2. Containers are retrieved only through the retrieval point of their current bay, since trucks wait in the bay where the target container is located.
A3. Containers are only relocated if they block the target container. This is a common practice in container yards and is widely studied to reduce problem dimensionality without significantly compromising optimality.
A4. If the target container is on the top of its stack, it must be retrieved immediately. This condition can be directly derived from Assumption A3.
A5. No new containers arrive during the retrieval process.

Fig. 2 illustrates a CRP instance. The crane is currently positioned at bay 1, and the container labels denote their retrieval orders. A truck (i.e., the retrieval point) is located at row 0 of each bay. Container 1 is the current target and can be retrieved immediately. Afterward, container 2 becomes the next target, but it is blocked by containers 18 and 25. Therefore, containers 18 and 25 must be relocated before container 2 can be retrieved. These relocations can be performed to other bays. After retrieving containers 2, 3, and 4, the crane must move to bay 2 to retrieve container 5.

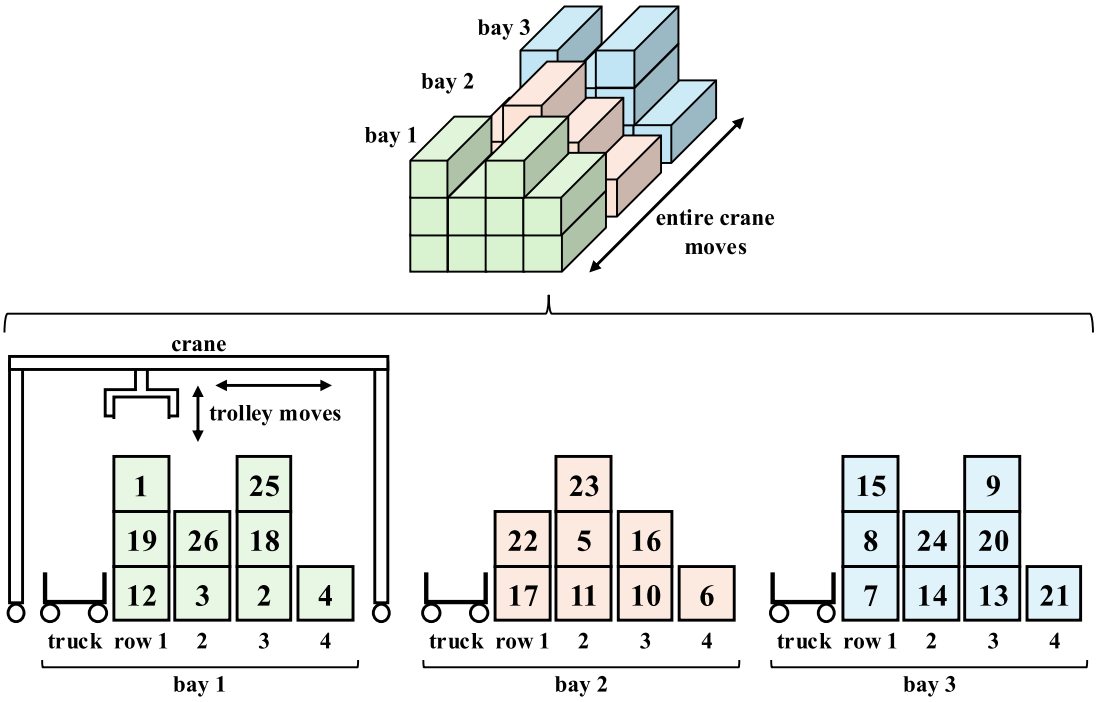


Fig. 2. Illustration of a CRP instance with 3 bays, 4 rows, and 26 containers.

The objective of the CRP is to retrieve all containers in the given order while minimizing the total working time of the crane, which includes both travel time and pick-up/set-down time. To describe the crane working time precisely, we define the following parameters:

- γ^{row}: unit time to move one row,
- γ^{bay}: unit time to move one bay,
- γ^{acc}: acceleration/deceleration time when switching bays,
- γ^{pd}: total handling time, including both pick-up and set-down operations.

Note that moving across bays requires the crane to reposition entirely, making γ^{bay} generally greater than γ^{row}, while γ^{acc} is typically much larger than γ^{bay}.

Each stack is identified by a bay index and a row index, denoted as (i, j) where i represents the bay and j the row. The travel time from stack (i, j) to another stack (i', j') is defined as:

$$\delta((i,j),(i',j')) = \gamma^{\text{acc}} \cdot \mathbb{I}(i \neq i') + \gamma^{\text{bay}} \cdot |i - i'| + \gamma^{\text{row}} \cdot |j - j'|, \tag{1}$$

where $\mathbb{I}(\cdot)$ is the indicator function that returns 1 if the condition is true and 0 otherwise.

The travel time required to retrieve the top container from stack (i, j) is denoted as $\delta^{\text{ret}}(i, j)$, where $(i, 0)$ represents the retrieval point of bay i:

$$\delta^{\text{ret}}(i,j) = \delta((i,j),(i,0)). \tag{2}$$

The travel time required to relocate the topmost blocking container from the target stack $(\bar{b}, \bar{r})$ to stack (i, j), denoted as $\delta^{\text{rel}}(i, j)$, is given by:

$$\delta^{\text{rel}}(i,j) = \delta((\bar{b},\bar{r}),(i,j)). \tag{3}$$

To obtain the total time required for each retrieval or relocation operation, we add the time for pick-up and set-down operations γ^{pd} to the respective travel times in (2) and (3).

The CRP typically involves hundreds of containers, making it difficult to obtain solutions through mathematical programming. For instance, Voß and Schwarze (2019) reports that even with a two-hour time limit, some instances with only 25 containers could not be solved. Therefore, as in prior CRP studies, we do not present such a formulation in this work. For readers interested in the mathematical formulation, Voß and Schwarze (2019) provides a model for the BRP that minimizes crane working time. Although it assumes a single-bay case, the formulation can be straightforwardly extended to the CRP by incorporating an additional cost term for bay movements.

3.2. Complexity and lower bound

While it has been shown that the BRP, which aims to minimize the number of relocations in a single bay, is $\mathcal{NP}$-hard (Caserta et al., 2012), no formal complexity analysis has yet been provided for the CRP. Following the approach of Caserta et al. (2012), we first establish that the CRP is also $\mathcal{NP}$-hard.

Theorem 1. *The CRP with finite B, R, and T is $\mathcal{NP}$-hard.*

Proof. We prove $\mathcal{NP}$-hardness via a polynomial-time many-one reduction from the BRP, which is known to be $\mathcal{NP}$-hard under finite stack and tier capacities (Caserta et al., 2012).

Consider the decision version of BRP:

BRP-DEC : Given a yard with S stacks and tier limit T, and a bound L,
does there exist a feasible plan that retrieves all containers with at most L relocations?

Similarly, define the decision version of CRP:

CRP-DEC : Given a yard with B bays, R rows per bay (so $S = BR$ stacks), tier limit T,
time coefficients $(\gamma^{\text{bay}}, \gamma^{\text{row}}, \gamma^{\text{acc}}, \gamma^{\text{pd}})$, and a bound K,
does there exist a feasible plan with total crane working time at most K?

Given an instance of BRP-DEC with S stacks, tier limit T, and relocation bound L, construct in polynomial time a corresponding CRP-DEC instance as follows: set $B = 1$ and $R = S$ (hence the same number of stacks $S = BR$) and keep the same tier limit T and container priorities (retrieval order). Choose the CRP time coefficients

$$\gamma^{\text{bay}} = \gamma^{\text{row}} = \gamma^{\text{acc}} = 0, \qquad \gamma^{\text{pd}} = 1,$$

so that travel- and acceleration-related times are null and only pick-up and set-down operations contribute to the working time.

Under these coefficients, for any feasible CRP plan τ on the constructed instance, the total working time equals the number of pick-up and set-down operations. Each container must be retrieved exactly once, so the number of retrieval operations is a constant N (the number of containers), independent of relocation decisions. Let $\text{Reloc}(\tau)$ denote the number of relocations in τ. Then the total working time is

$$W(\tau) \;=\; N \;+\; \text{Reloc}(\tau).$$

Therefore, the BRP instance admits a plan with at most L relocations if and only if the constructed CRP instance admits a plan with

$$W(\tau) \;\le\; K \;:=\; N + L.$$

Hence, setting $K = N + L$ yields a yes-instance of CRP-DEC exactly when the original is a yes-instance of BRP-DEC.

The transformation preserves feasibility (stacks, tiers, and retrieval order are unchanged), and it is clearly computable in polynomial time. Thus BRP-DEC $\le_m^p$ CRP-DEC. Since BRP-DEC is $\mathcal{NP}$-hard, CRP-DEC is $\mathcal{NP}$-hard as well. Consequently, optimizing CRP is $\mathcal{NP}$-hard.

The above argument shows hardness even in the special case $B = 1$ with zero travel/acceleration times. Therefore CRP with finite B, R, and T is $\mathcal{NP}$-hard. □

Next, we propose a lower bound on the total crane working time for the CRP. To the best of our knowledge, this is the first such bound, and it is actively used in subsequent experiments to evaluate the solution quality of various methods. We begin by introducing some notation. Let b_n and r_n denote the bay and row indices, respectively, of the container with retrieval order n. Let $x_{i,j,k}$ represent the retrieval order of the container located at bay i, row j, and tier k. If the position is empty, $x_{i,j,k}$ is defined to be zero.

Theorem 2. *The lower bound on the total crane working time is given by*

$$\text{LB} = \text{LB}^{\text{ret}} + \text{LB}^{\text{rel}},$$

where

$$\text{LB}^{\text{ret}} = \sum_{n=1}^{N-1} \left[\delta^{\text{ret}}(b_n, r_n) + \delta((b_n, 0), (b_{n+1}, r_{n+1}))\right] + \delta^{\text{ret}}(b_N, r_N) + N \cdot \gamma^{\text{pd}},$$

$$\text{LB}^{\text{rel}} = (2 \cdot \gamma^{\text{row}} + \gamma^{\text{pd}}) \cdot \sum_{i=1}^{B} \sum_{j=1}^{R} \sum_{k=2}^{T} \mathbb{I}\left(\min_{k'<k} x_{i,j,k'} < x_{i,j,k}\right).$$

Proof. The proposed lower bound LB consists of two components: LB^{ret} and LB^{rel}. The first component, LB^{ret}, accounts for the minimal time required to perform retrieval operations, while the second, LB^{rel}, estimates the minimal time required for relocation operations.

To retrieve each container n $(1 \le n \le N)$, the crane must move from the stack (b_n, r_n) to the retrieval point $(b_n, 0)$ of the same bay. After retrieving container n, the crane moves to the next target stack (b_{n+1}, r_{n+1}). This time is computed as $\delta^{\text{ret}}(b_n, r_n) +$

$\delta((b_n, 0), (b_{n+1}, r_{n+1}))$. Summing this from container 1 to $N-1$, and adding the time to retrieve the final container N, $\delta^{\text{ret}}(b_N, r_N)$, along with the pick-up and set-down time for all N containers, $N \cdot \gamma^{\text{pd}}$, yields LB^{ret}.

Next, we compute the relocation-related lower bound LB^{rel}. Under Assumption A3, the crane must return to the target stack after relocating a blocking container. Assuming relocation to the nearest stack, the minimum time per relocation is at least $2 \cdot \gamma^{\text{row}} + \gamma^{\text{pd}}$. Now, we estimate the minimum number of relocations. For each container $x_{i,j,k}$ located at bay i, row j, and tier k, if there exists a container beneath it with a smaller retrieval order, then the container must be relocated at least once. The number of such mandatory relocations is given by the indicator $\mathbb{I}\left(\min_{k'<k} x_{i,j,k'} < x_{i,j,k}\right)$. Summing over all containers and multiplying by the minimum relocation time yields LB^{rel}.

Combining both components, the total lower bound is LB. □

4. Deep reinforcement learning approach

In this section, we present our novel DRL approach for solving the CRP. Given the $\mathcal{NP}$-hardness of the problem, obtaining optimal solutions within short computation times is challenging, and heuristic approaches have thus been widely adopted. We leverage DRL to substantially improve upon the unsatisfactory solution quality of existing heuristics while enabling real-time decision making. We first formulate the CRP as a Markov decision process (MDP) and then introduce a size-agnostic network architecture capable of handling diverse yard sizes. Finally, we propose a scale-diverse learning framework that achieves high-quality, robust performance on yard sizes far larger and more diverse than those used for training. The overall framework of the proposed DRL approach is illustrated in Fig. 3.

4.1. Markov decision process

We model the CRP as an MDP, defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, where $\mathcal{S}$, $\mathcal{A}$, $\mathcal{P}$, and $\mathcal{R}$ denote the state space, action space, state transition function, and reward function, respectively. The decision-making procedure is described as follows.

Decision step. Each decision step is defined as the moment immediately after the crane completes a retrieval or relocation operation, provided that the current target container is blocked. At the beginning of the schedule, the first step is defined as the moment when all containers that can be immediately retrieved have been retrieved. This definition follows from Assumption A4 in the problem description, which states that if the target container is topmost, it must be retrieved immediately.

State. At each decision step t, the state $\mathcal{S}_t \in \mathcal{S}$ is defined by the yard configuration and the crane position. The yard configuration consists of the number of bays B, rows R, tiers T, and the container priorities (retrieval orders) $x^t_{i,j,k}$, where $x^t_{i,j,k}$ denotes the priority of the container located at bay i, row j, and tier k at step t (set to zero if the position is empty). The crane position is represented by its bay and row indices. Note that, by the definition of a decision step, the crane is never in the middle of an operation in any state.

Action. An action $\mathcal{A}_t \in \mathcal{A}$ is to select a stack to which the topmost blocking container will be relocated. The action space $\mathcal{A}$ consists of all stacks, with certain prohibited actions defined as follows: (i) selecting the current target stack, and (ii) selecting a stack that has already reached its maximum capacity. These restrictions prevent meaningless and invalid actions.

State transition. The state transition function $\mathcal{P}(\mathcal{S}_{t+1} \mid \mathcal{S}_t, \mathcal{A}_t)$ describes the deterministic evolution of the yard configuration. Given a state $\mathcal{S}_t$, after taking an action $\mathcal{A}_t$–i.e., relocating the topmost blocking container–if the target container subsequently appears on top, it is retrieved immediately. This process is repeated until the next target container is no longer topmost and further relocations are required, at which point the resulting configuration is defined as the next state $\mathcal{S}_{t+1}$. The decision process terminates when all containers have been retrieved.

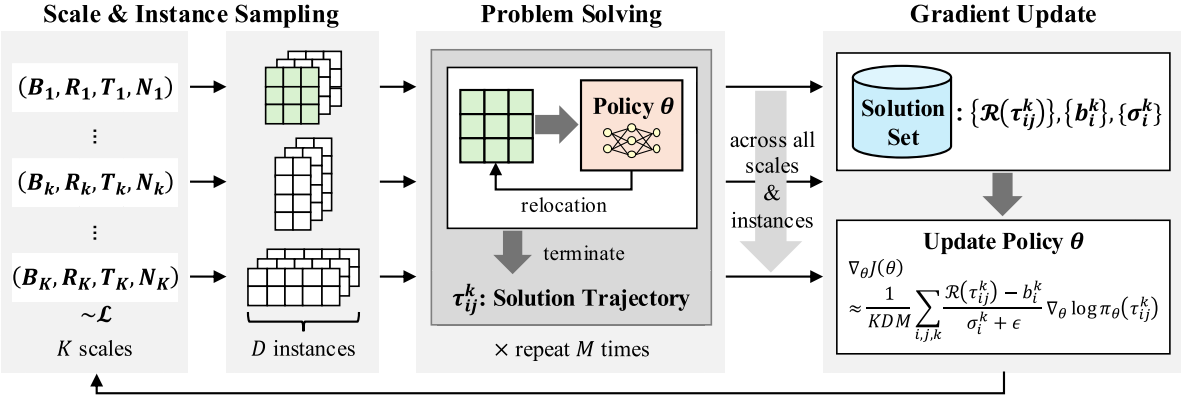


Fig. 3. The overall framework of the proposed DRL approach.

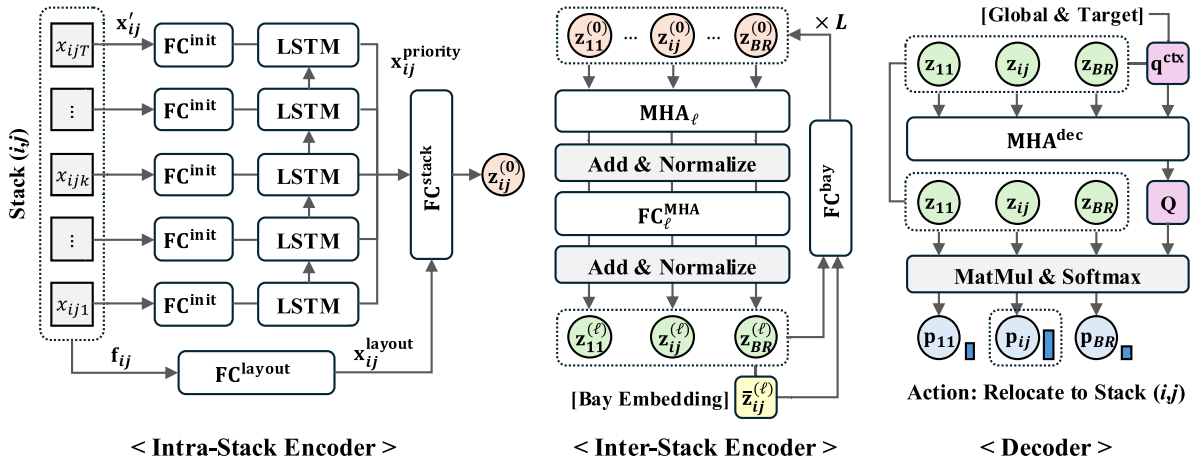


Fig. 4. Network architecture based on an encoder-decoder framework.

Reward. The reward function $\mathcal{R}(S_t, \mathcal{A}_t)$ is defined such that the agent receives a reward of 0 until termination and is assigned the negative value of the total crane working time at the terminal state. Since the training algorithm updates the parameters on an episode basis (i.e., after a complete solution is obtained), this sparse reward design does not pose any issues. Details regarding the choice of the training algorithm are provided in Section 4.3.

4.2. Network architecture

In our DRL approach, the policy network adopts an encoder-decoder architecture (see Fig. 4). The encoder is composed of two modules: an *intra-stack encoder*, which uses an LSTM (Hochreiter and Schmidhuber, 1997) to capture the sequential arrangement of containers within each stack, and an *inter-stack encoder*, which employs an attention mechanism (Vaswani et al., 2017) to model interactions and contextual relationships among stacks. Given the yard configuration, the encoder generates a latent embedding for each stack, integrating both local (within-stack) and global (across-stack) information. These embeddings are then passed to the decoder, which converts them into action-selection probabilities. This design allows the network to effectively process yard configurations of arbitrary size while preserving the structural dependencies inherent in the CRP.

4.2.1. Intra-stack encoder

The intra-stack encoder derives an initial embedding for each stack (i, j) based on its container arrangement and spatial location. By employing an LSTM, the encoder can effectively process stacks of arbitrary heights, enabling the model to handle variable-tier configurations without requiring fixed-size inputs. For notational simplicity, we omit the decision step index t in the following.

First, we normalize the priority values $x_{i,j,k}$ (retrieval order) to the range $[0, 1]$ such that the target container's value is maximized and well separated from that of empty positions. Here, we assume that the priorities $x_{i,j,k}$ are adjusted so that the target container always has a value of 1, and they range from 1 to N, where N is the current number of containers:

$$\hat{x}_{i,j,k} = \begin{cases} 1 - \dfrac{x_{i,j,k} - 1}{N}, & \text{if } x_{i,j,k} > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

To incorporate positional information, we also define a tier vector

$$\mathbf{p}^{\text{asc}} = \left[\frac{0}{T-1}, \frac{1}{T-1}, \ldots, \frac{T-1}{T-1}\right] \in \mathbb{R}^T, \tag{5}$$

where T is the maximum number of tiers. Finally, the two features are concatenated to form the stack feature matrix

$$\mathbf{x}'_{i,j} = \left[\hat{\mathbf{x}}_{i,j},\ \mathbf{p}^{\text{asc}}\right] \in \mathbb{R}^{T\times 2}, \tag{6}$$

which serves as the input to the LSTM layer for sequential feature extraction.

The concatenated priority and tier vectors $\mathbf{x}'_{i,j}$ are first projected into a d-dimensional space through an initial fully connected layer:

$$\mathbf{x}''_{i,j} = \text{FC}^{\text{init}}(\mathbf{x}'_{i,j}) \in \mathbb{R}^{T\times d}. \tag{7}$$

We then apply an LSTM to effectively capture the sequential information along the tier dimension:

$$\bar{\mathbf{o}}_{i,j},\ \mathbf{o}_{i,j} = \text{LSTM}\left(\mathbf{x}''_{i,j}\right) \in \mathbb{R}^d, \tag{8}$$

where $\bar{\mathbf{o}}_{i,j}$ denotes the average of the LSTM outputs over all tiers, and $\mathbf{o}_{i,j}$ is the output corresponding to the top tier (i.e., the last element in the sequence). Finally, these two representations are concatenated and linearly transformed to obtain the priority feature embedding for stack (i, j):

$$\mathbf{x}_{i,j}^{\text{priority}} = \mathbf{W}^{\text{LSTM}} \cdot \left[\bar{\mathbf{o}}_{i,j},\ \mathbf{o}_{i,j}\right] \in \mathbb{R}^d. \tag{9}$$

In addition to the priority feature, we embed the spatial location and structural attributes of each stack. For stack (i, j), we construct a feature vector:

$$\mathbf{f}_{i,j} = \left[n_i^{\text{bay}}, n_j^{\text{row}}, d_i^{\text{bay}}, d_j^{\text{row}}, h_{i,j}, \bar{h}_{i,j}, n_{i,j}^{\text{ret}}, n_{i,j}^{\text{rel}}, \alpha, \beta\right] \in \mathbb{R}^{10}, \tag{10}$$

where:

- n_i^{bay} and n_j^{row} are the normalized bay and row indices:

$$n_i^{\text{bay}} = \frac{|i-1|}{B-1}, \quad n_j^{\text{row}} = \frac{|j-1|}{R-1}.$$

- d_i^{bay} and d_j^{row} are the normalized distances from the target stack $(\bar{b}, \bar{r})$:

$$d_i^{\text{bay}} = \frac{|i-\bar{b}|}{B-1}, \quad d_j^{\text{row}} = \frac{|j-\bar{r}|}{R-1}.$$

- $h_{i,j}$ is the current stack height, and $\bar{h}_{i,j} = T - h_{i,j}$ is the number of remaining empty tiers.
- $n_{i,j}^{\text{ret}}$ and $n_{i,j}^{\text{rel}}$ are the normalized times for retrieving the top container from (i, j) and relocating the top container of the target stack to (i, j), respectively:

$$\delta^{\max} = \max_{i,j} \left\{\max\left(\delta^{\text{ret}}(i,j),\ \delta^{\text{rel}}(i,j)\right)\right\},$$

$$n_{i,j}^{\text{ret}} = \frac{\delta^{\text{ret}}(i,j)}{\delta^{\max}}, \quad n_{i,j}^{\text{rel}} = \frac{\delta^{\text{rel}}(i,j)}{\delta^{\max}}.$$

- $\alpha = \frac{N}{B \cdot R \cdot T}$ is the yard utilization ratio.
- $\beta = \frac{B}{R}$ is the bay-to-row ratio.

Then, the layout feature embedding for stack (i, j) is obtained via a fully connected transformation:

$$\mathbf{x}_{i,j}^{\text{layout}} = \text{FC}^{\text{layout}}\left(\mathbf{f}_{i,j}\right) \in \mathbb{R}^d. \tag{11}$$

Finally, the priority feature embedding $\mathbf{x}_{i,j}^{\text{priority}}$ and the layout feature embedding $\mathbf{x}_{i,j}^{\text{layout}}$ are concatenated and passed through a fully connected layer to produce the initial stack embedding:

$$\mathbf{z}_{i,j}^{(0)} = \text{FC}^{\text{stack}}\left(\left[\mathbf{x}_{i,j}^{\text{priority}},\ \mathbf{x}_{i,j}^{\text{layout}}\right]\right) \in \mathbb{R}^d. \tag{12}$$

In the intra-stack encoder, all features related to container priority, bay, row, and tier indices are normalized. This normalization is crucial for achieving scalability, ensuring that the trained model remains robust when applied to yard configurations significantly larger than those used during training. This prevents the model from failing when faced with input scales it has never encountered during training, enabling it to operate based on the relative magnitudes of the values rather than their absolute scales.

In addition, while existing CRP studies often rely on various handcrafted features to represent the container priority composition within a stack–such as the minimum and maximum priorities, whether the stack is the target stack, whether the target container is blocked, and the number of containers blocking higher-priority containers–our approach processes raw container priority features directly using an LSTM. This enables the model to effectively capture the sequential arrangement of containers within each stack without the need for manual feature engineering.

4.2.2. Inter-stack encoder

The inter-stack encoder facilitates information exchange across stacks, enabling the model to capture both priority arrangement patterns and spatial relationships among stacks. By incorporating a multi-head attention (MHA) mechanism, the encoder can flexibly process an arbitrary number of stacks without requiring any modification to the input dimension. Let $\mathbf{Z}^{(0)} \in \mathbb{R}^{(B \cdot R) \times d}$ denote the initial stack embeddings produced by the intra-stack encoder, where each embedding corresponds to a stack located at bay-row coordinates (i, j).

At each encoder layer ℓ, an MHA module processes the stack embeddings to exchange information between stacks, followed by a feed-forward network for feature refinement:

$$\mathbf{Z}'^{(\ell)} = \text{IN}\left(\text{MHA}_\ell\left(\mathbf{Z}^{(\ell-1)}\right) + \mathbf{Z}^{(\ell-1)}\right) \in \mathbb{R}^{(B \cdot R) \times d}, \tag{13}$$

$$\hat{\mathbf{Z}}^{(\ell)} = \text{IN}\left(\text{FC}_\ell^{\text{MHA}}\left(\mathbf{Z}'^{(\ell)}\right) + \mathbf{Z}'^{(\ell)}\right) \in \mathbb{R}^{(B \cdot R) \times d}, \tag{14}$$

where MHA_ℓ denotes a self-attention layer in which the query, key, and value are identical. $\text{IN}(\cdot)$ denotes instance normalization, and the residual connections preserve the original stack-level representations.

To facilitate both stack-level and bay-level interactions, we compute a bay-level aggregated feature for each bay i by averaging over its R rows, and then concatenate it with the corresponding stack-level feature before passing through a feed-forward layer:

$$\bar{\mathbf{z}}_{i,j}^{(\ell)} = \frac{1}{R} \sum_{j'} \hat{\mathbf{z}}_{i,j'}^{(\ell)} \in \mathbb{R}^d, \tag{15}$$

$$\mathbf{Z}^{(\ell)} = \mathrm{FC}^{\mathsf{bay}}\left(\left[\hat{\mathbf{Z}}^{(\ell)},\ \bar{\mathbf{Z}}^{(\ell)}\right]\right) \in \mathbb{R}^{(B\cdot R)\times d}. \tag{16}$$

This design choice reflects the CRP characteristic that inter-bay travel times are relatively long, thereby allowing the model to more explicitly incorporate information from stacks within the same bay.

By stacking multiple such inter-stack encoder layers, the model is able to propagate information effectively across the entire yard, while preserving both local and global spatial context.

4.2.3. Decoder

The decoder is responsible for selecting the relocation destination stack for the topmost blocking container. It first aggregates the encoder outputs to form a global yard representation:

$$\mathbf{z}^{\mathsf{global}} = \frac{1}{B \cdot R} \sum_{i,j} \mathbf{z}_{i,j} \in \mathbb{R}^d, \tag{17}$$

where $\mathbf{z}_{i,j}$ denotes the final embedding of stack (i, j) from the encoder. Next, the decoder constructs a context query vector by concatenating the embedding of the current target stack $\mathbf{z}_{\bar{b},\bar{r}}$ and the global embedding $\mathbf{z}^{\mathsf{global}}$, followed by a linear transformation:

$$\mathbf{q}^{\mathsf{ctx}} = \mathbf{W}^{\mathsf{dec}} \cdot \left[\mathbf{z}_{\bar{b},\bar{r}},\ \mathbf{z}^{\mathsf{global}}\right] \in \mathbb{R}^d. \tag{18}$$

The context vector $\mathbf{q}^{\mathsf{ctx}}$ first serves as the query in a cross-attention operation, where the keys and values are the stack embeddings $\mathbf{Z}$ obtained from the encoder. Through this process, the cross-attention module $\mathrm{MHA}^{\mathsf{dec}}$ produces an enhanced context representation $\mathbf{Q}$ that integrates the target-stack and global-yard information with relevant details from all other stacks. Using this enhanced context vector, the decoder then computes a score $\mathbf{u}$ for each stack by comparing it with the corresponding stack embedding, ultimately yielding the relocation probability distribution $\mathbf{p}$ over all stacks. The resulting computation is expressed as follows:

$$\mathbf{Q} = \mathbf{W}^{Q} \cdot \mathrm{MHA}^{\mathsf{dec}}\left(\mathbf{q}^{\mathsf{ctx}},\ \mathbf{Z},\ \mathbf{Z}\right) \in \mathbb{R}^d, \tag{19}$$

$$\mathbf{K} = \mathbf{W}^{K} \cdot \mathbf{Z} \in \mathbb{R}^{(B\cdot R)\times d}, \tag{20}$$

$$\mathbf{u} = C \cdot \tanh\left(\frac{\mathbf{Q}\cdot\mathbf{K}^{\top}}{\sqrt{d}}\right) - \mathrm{mask} \in \mathbb{R}^{B\cdot R}, \tag{21}$$

$$\mathbf{p} = \mathrm{softmax}(\mathbf{u}), \tag{22}$$

$$(i, j) \sim \mathrm{Categorical}(\mathbf{p}). \tag{23}$$

Here, C is a scaling constant, and mask assigns large negative values to prohibited actions. The vector $\mathbf{p}$ represents the action probabilities over all stacks. During training, the relocation destination (i, j) is sampled from $\mathrm{Categorical}(\mathbf{p})$ to promote exploration. Through this process, each stack embedding is explicitly compared with the enhanced context vector, allowing the model to evaluate both its absolute favorability as a relocation destination and its relative suitability compared to other stacks, while preserving size-agnostic generalization capability.

4.3. Scale-diverse learning

In this section, we propose a scale-diverse learning framework to enhance scalability in variable yard sizes and large-scale environments. The proposed training algorithm is based on REINFORCE (Williams, 1992). While deep Q-network (DQN) (Mnih et al., 2015) and proximal policy optimization (PPO) (Schulman et al., 2017) have been popular for solving combinatorial optimization (CO) problems, DQN suffers from action-value approximation errors in large discrete action spaces, and PPO can be unstable in such settings due to its sensitivity to advantage estimation. In recent years, numerous studies have shown that REINFORCE, with its simple gradient estimator and direct applicability to discrete action spaces, achieves strong performance in CO tasks and has been increasingly adopted in this field.

Beyond performance considerations, REINFORCE offers an additional advantage in that it learns from complete trajectories, eliminating the need to design rewards for every state-action pair. This property is particularly beneficial for the CRP, where each action corresponds to a relocation, while retrievals are automatically performed whenever possible after a relocation. As a result, a single action may not only involve one crane operation but also trigger multiple subsequent retrievals, making it difficult to accurately assess its exact contribution to the total crane working time. Consequently, reward design at the per-action level is not straightforward in the CRP setting.

Our REINFORCE-based training algorithm is summarized in Algorithm 1. The approximated gradient ascent for maximizing the expected return $J(\theta)$ is given by

$$\nabla_\theta J(\theta) \approx \frac{1}{KDM} \sum_{k=1}^{K} \sum_{i=1}^{D} \sum_{j=1}^{M} \frac{\mathcal{R}(\tau_{i,j}^k) - b_i^k}{\sigma_i^k + \epsilon} \cdot \nabla_\theta \log \pi_\theta(\tau_{i,j}^k), \tag{24}$$

Algorithm 1 Scale-Diverse Training Algorithm with Normalized Return.

Require: Layout set $\mathcal{L}$, epochs E, steps per epoch P, yard scales per step K, instances per scale D, rollouts per instance M

Initialize policy parameters θ
for epoch = 1 to E **do**
  for step = 1 to P **do**
    $\{\ell_k\}_{k=1}^{K} \leftarrow \text{SampleYardScale}(\mathcal{L})$
    for each $k = 1$ to K **do**
      $\{a_i^k\}_{i=1}^{D} \leftarrow \text{RandomInstance}(\ell_k)$
      $\{\tau_{i,j}^k\}_{j=1}^{M} \leftarrow \text{SampleRollout}(a_i^k, \pi_\theta) \quad \forall i \in \{1,\dots,D\}$
      $b_i^k \leftarrow \frac{1}{M}\sum_{j=1}^{M} \mathcal{R}(\tau_{i,j}^k) \quad \forall i \in \{1,\dots,D\}$
      $\sigma_i^k \leftarrow \text{StdDev}(\{\mathcal{R}(\tau_{i,j}^k)\}_{j=1}^{M}) \quad \forall i \in \{1,\dots,D\}$
      $g_{i,j}^k \leftarrow \frac{\mathcal{R}(\tau_{i,j}^k)-b_i^k}{\sigma_i^k+\epsilon} \cdot \nabla_\theta \log \pi_\theta(\tau_{i,j}^k) \quad \forall i \in \{1,\dots,D\}, \quad j \in \{1,\dots,M\}$
    end for
    $\nabla_\theta J(\theta) \leftarrow \frac{1}{KDM}\sum_{k=1}^{K}\sum_{i=1}^{D}\sum_{j=1}^{M} g_{i,j}^k$
    $\theta \leftarrow \text{Adam}(\theta, \nabla_\theta J(\theta))$
  end for
end for

where the baseline b_i^k and the standard deviation σ_i^k are computed as

$$b_i^k = \frac{1}{M}\sum_{j=1}^{M} \mathcal{R}(\tau_{i,j}^k), \tag{25}$$

$$\sigma_i^k = \text{StdDev}\left(\left\{\mathcal{R}(\tau_{i,j}^k)\right\}_{j=1}^{M}\right). \tag{26}$$

Here, K is the number of yard scales evaluated in each training step, D is the number of instances per scale, and M is the number of sampled solutions per instance. The notation τ denotes a solution trajectory, $\mathcal{R}(\cdot)$ is the return for a given solution trajectory, defined as the negative of the total crane working time, and $\pi_\theta(\cdot)$ represents the solution probability under policy parameters θ. ϵ is a small positive constant to prevent division by zero.

To improve generalization, the agent is simultaneously evaluated and updated on K different yard scales in each training step rather than being trained on a single scale. Each yard scale is defined by its number of bays, rows, tiers, and containers, and for each scale we generate D random instances (see Section 5.1.2 for the detailed procedure). The policy is then evaluated on all $K \times D$ instances. This allows the policy to learn to handle diverse yard scales within a single update.

The baseline b_i^k serves to reduce the variance of the advantage ($\mathcal{R} - b$) in REINFORCE, thereby accelerating learning (Kool et al., 2019b). Following the observation that baselines computed from multiple sampled solutions are more stable than those based on a single solution (Kool et al., 2019a; Kwon et al., 2020), we set b_i^k to the average return of the M solutions generated for each instance via sampling inference.

Since a single gradient update covers multiple yard scales, the absolute objective values can differ significantly across scales, which may bias the update toward certain scales. To mitigate this effect, we normalize the advantage by dividing it by the standard deviation σ_i^k of the sampled solutions' returns for each instance. This variance normalization balances the influence of each scale during training.

In our training procedure, the yard scale is randomly sampled within a predefined set. One might consider that adaptively sampling scales based on the agent's current performance (Zhou et al., 2023) could yield better results. However, our objective is not to improve performance on a fixed and finite validation set, but rather to achieve robust performance across an unbounded variety of yard configurations. This makes such an adaptive approach less applicable. Similarly, as in multi-task learning, designing an architecture that explicitly indicates the current target task (Liu et al., 2024) is also challenging in this setting.

By evaluating the agent on multiple yard configurations and updating it in a single step, the proposed framework enables stable performance improvement across instances of varying scales. Although the memory limitations imposed by long trajectories restrict training to relatively small-scale cases with only a few dozen containers, the trained agent demonstrates strong scalability, achieving high performance even on much larger instances involving thousands of containers that were never encountered during training. This scalability arises from exposing the agent to diverse yard scales during training, allowing it to learn the relative importance among stacks within a yard configuration rather than overfitting to absolute scale differences. As discussed in Section 4.2, this is further facilitated by normalizing all input features in a scale-independent manner.

5. Experimental results

This section presents the evaluation of the proposed DRL approach. We first describe the experimental settings, including the benchmark datasets, training procedure, and comparison algorithms. Subsequently, we report the experimental results and provide

a detailed analysis. All codes and datasets used in the experiments are publicly available in the GitHub repository: https://github.com/operagang/CRP_RL.

5.1. Settings

5.1.1. Benchmark dataset

We utilize the benchmark dataset proposed by Lee and Lee (2010), which is widely used in CRP research. The dataset contains problem instances with 70 to 720 containers and includes two configuration types: R and U. The R type indicates that containers are randomly distributed, whereas the U type signifies that all containers in each stack are arranged in an upside-down retrieval order. Each instance name encodes its configuration, where the first character is either R or U, followed by two characters representing the number of bays, two characters for the number of rows per bay, and two characters for the maximum stack height, with the value after the underscore denoting the total number of containers. For example, R021606_0140 denotes an instance in which containers are randomly distributed, with 2 bays, 16 rows, 6 tiers, and a total of 140 containers. For each scale, the dataset contains five R-type instances and two U-type instances.

To evaluate performance at extremely large scales, we generated an additional benchmark by extending the scale of Lee and Lee (2010)'s instances at the bay level, while keeping the number of rows and tiers unchanged, as these dimensions are generally constrained by crane size. The new instances comprise layouts with 20 or 30 bays, containing 1440, 1920, 2160, or 2880 containers. Both R-type and U-type configurations were generated, with 20 instances provided for each.

For all benchmarks, $\gamma^{\text{row}} = 1.2$, $\gamma^{\text{bay}} = 3.5$, $\gamma^{\text{acc}} = 40$, and $\gamma^{\text{pd}} = 30$, where γ^{row} is the unit time to move one row, γ^{bay} is the unit time to move one bay, γ^{acc} is the acceleration/deceleration time when switching bays, and γ^{pd} is the total handling time including both pick-up and set-down operations. These parameter settings follow the convention used in previous studies.

5.1.2. Training procedure

The training was conducted using randomly generated instances, each containing 35 to 70 containers. The number of bays, rows, and tiers was randomly determined to achieve a yard occupancy rate between 60 % and 80 %. The number of rows was limited to at most 16, while the number of tiers was restricted to either 6 or 8. In scale-diverse training, yard scales were sampled by randomly selecting bay, row, tier, and container counts that satisfied the above conditions. Container placement within the yard was randomized, and only R-type configurations were used for training.

The model was trained for 100 epochs (E), with 100 update steps per epoch (P). Each step used $K = 4$ yard scales, with $D = 32$ instances per scale, and each instance was repeated $M = 16$ times. The learning rate was set to 3×10^{-4}. The embedding dimension was $d = 128$. A single LSTM layer was employed. The MHA module comprised $\ell = 3$ layers, each with 8 heads, and a scaling constant of $C = 10$. FC^{init} consisted of dimensions $2 \to d/2 \to d$. $\text{FC}^{\text{layout}}$ had dimensions $10 \to d/2 \to d$. FC^{stack} had dimensions $2d \to 4d \to d$. FC^{MHA} had dimensions $d \to 4d \to d$. FC^{bay} had dimensions $2d \to 4d \to d$.

5.1.3. Comparison algorithms

For comparison, we selected representative approaches from the CRP literature. The approaches of Lee and Lee (2010) and Bian and Jin (2013) were excluded from comparison due to their long computational times and relatively poor solution quality. The selected algorithms are as follows:

- **TS**: The tree search algorithm proposed by Forster and Bortfeldt (2012).
- **Lin**: The expert-designed heuristic rule proposed by Lin et al. (2015). In this heuristic, parameters P_r and P_b are set to 30 and 300, respectively, which minimize the crane working time in their experiments.
- **Kim**: The expert-designed heuristic rule proposed by Kim et al. (2016).
- **GRASP**: The GRASP algorithm proposed by Cifuentes and Riff (2020).
- **GP**: The GP approach proposed by Đurasević et al. (2025), using the best-driven heuristic rule from their Fig. 19(b).
- **Leveling**: The heuristic proposed by Zehendner et al. (2017), developed for the *online CRP*. This method was selected to enable comparison in the online setting. A detailed description of the online setting is provided later in Section 5.2.5.
- **Shin**: The DRL approach proposed by Shin et al. (2025), which employs a network based on expert-designed features.

For the Lee and Lee (2010) benchmark experiments, the results for Kim, TS, and GRASP, including both objective value and computational time, were taken directly from their respective papers. For the remaining algorithms, benchmark results were not available in the literature; therefore, we implemented these methods and report their performance in this study.

Together with Shin, we developed two additional DRL approaches inspired by Liu et al. (2023) and Ma et al. (2025) for a more comprehensive comparison among DRL-based methods. Since both Liu et al. (2023) and Ma et al. (2025) addressed the single-bay BRP with the objective of minimizing the number of relocations, their approaches could not be directly applied to the CRP. Therefore, we adapted their learning algorithms to our network model to ensure a fair comparison. The detailed implementation and training procedures are described in Section 5.2.2.

5.1.4. Evaluation metric and implementation details

To compare the performance of algorithms, we use the gap with respect to the lower bound of the total crane working time, calculated as $\text{Gap} = 100 \times \frac{\text{obj}-\text{LB}}{\text{LB}}$, where LB denotes the lower bound proposed in Theorem 2. For each scale, we report the average

Table 1
Performance comparison on the random benchmark proposed by Lee and Lee (2010).

Problem	Kim		TS		GRASP		GP		Lin		Ours	
	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time
R011606_0070	140.6	0.01	107.8	8.2	37.0	0.2	17.2	0.1	9.5	0.1	**7.7**	0.1
R021606_0140	72.8	0.01	58.3	9.1	46.0	1.3	52.5	0.1	20.0	0.1	**7.1**	0.2
R041606_0280	55.6	0.01	44.4	9.5	55.8	13.8	50.8	0.4	22.5	0.3	**5.3**	0.3
R061606_0430	53.9	0.01	46.5	10.2	58.9	26.8	48.3	0.6	25.4	0.6	**5.9**	0.4
R081606_0570	57.6	0.01	45.9	10.7	61.1	51.4	46.4	1.1	23.5	1.0	**5.8**	0.5
R101606_0720	64.4	0.01	47.7	11.9	61.7	126.0	43.4	1.5	21.6	1.7	**5.9**	0.7
R011608_0090	161.9	0.01	101.4	10.1	44.2	1.0	22.0	0.1	11.4	0.1	**10.2**	0.1
R021608_0190	90.3	0.01	57.6	10.1	59.0	6.1	70.8	0.2	26.0	0.2	**9.4**	0.2
R041608_0380	81.9	0.01	49.8	10.8	65.0	44.1	60.7	0.5	32.0	0.5	**10.0**	0.4
R061608_0570	88.0	0.01	47.3	11.9	66.7	189.5	55.2	1.0	32.2	1.0	**10.2**	0.6

Gap: Percentage gap with lower bound. Time: Computation time in seconds.

Table 2
Performance comparison on the upside-down benchmark proposed by Lee and Lee (2010).

Problem	Kim		TS		GP		Lin		Ours	
	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time
U011606_0070	125.7	0.01	93.4	1.2	14.1	0.1	8.9	0.1	**4.9**	0.1
U021606_0140	80.2	0.01	54.4	2.9	75.8	0.1	15.6	0.1	**5.0**	0.1
U041606_0280	68.7	0.01	45.6	6.1	69.2	0.4	28.2	0.4	**3.7**	0.3
U061606_0430	67.1	0.01	43.8	5.6	65.8	0.9	28.4	0.8	**4.1**	0.4
U081606_0570	69.0	0.01	44.4	3.8	64.6	1.4	27.7	1.3	**3.0**	0.5
U101606_0720	69.9	0.01	44.4	6.9	62.7	2.0	26.3	2.2	**4.5**	0.7
U011608_0090	127.7	0.01	89.8	10.0	14.4	0.1	**10.1**	0.1	10.8	0.1
U021608_0190	81.3	0.01	55.5	4.5	73.6	0.2	26.1	0.2	**8.2**	0.2
U041608_0380	64.2	0.01	41.3	5.3	69.3	0.6	24.2	0.6	**5.5**	0.4
U061608_0570	64.4	0.01	43.5	4.6	70.4	1.2	28.2	1.2	**7.3**	0.6

Gap: Percentage gap with lower bound. Time: Computation time in seconds.

gap across all instances. In addition, the average computational time in seconds is also reported. The lower bound values and the results of our DRL approach for each instance are provided in Tables A.7 and A.8 in Appendix A. During testing, the DRL approach employed a greedy inference strategy, where the agent deterministically selects the action with the highest predicted probability at each decision step rather than sampling from the action distribution.

All implementations were developed in Python. All experiments were conducted on a personal computer equipped with an Intel Core i9-14900K CPU and an NVIDIA RTX 4090 GPU. The training of the DRL model required approximately 40 hours.

5.1.5. Experimental analysis overview

The following research questions will be addressed in the subsequent sections to evaluate the proposed DRL approach:

- **Q1:** How superior and robust is the proposed DRL approach across different yard scales, including large-scale instances with hundreds of containers that were unseen during training?
- **Q2:** How much better does the proposed DRL approach perform compared to existing DRL-based methods?
- **Q3:** How much does each component of the framework–the LSTM module, attention mechanism, and scale-diverse learning strategy–contribute to performance improvement?
- **Q4:** How well does the proposed DRL approach scale to extremely large problem instances with thousands of containers?
- **Q5:** How effective is the proposed DRL approach in a realistic online setting that incorporates operational constraints?
- **Q6:** How does the DRL agent behave in different scenarios, evaluate candidate stacks, and differ from expert-designed heuristic rules, and what insights can be gained from its behavior?

5.2. Results

5.2.1. Q1. Superiority and robustness across yard scales

We evaluate the performance of the proposed DRL approach on the benchmark dataset of Lee and Lee (2010), which covers a wide range of scales from 70 to 720 containers and from 1 to 10 bays. Table 1 reports the results for the R-type instances, while Table 2 presents the results for the U-type instances. In addition, Fig. 5 provides a visual comparison against the best-performing baseline, Lin, for both R- and U-type instances.

From Table 1, it can be observed that our DRL approach significantly outperforms all baselines across all scales for the R-type instances. In particular, it achieves an average optimality gap of 7.8 % and a maximum gap of approximately 10 %, demonstrating robust performance while producing solutions for all instances within one second, which highlights its strong applicability. Even

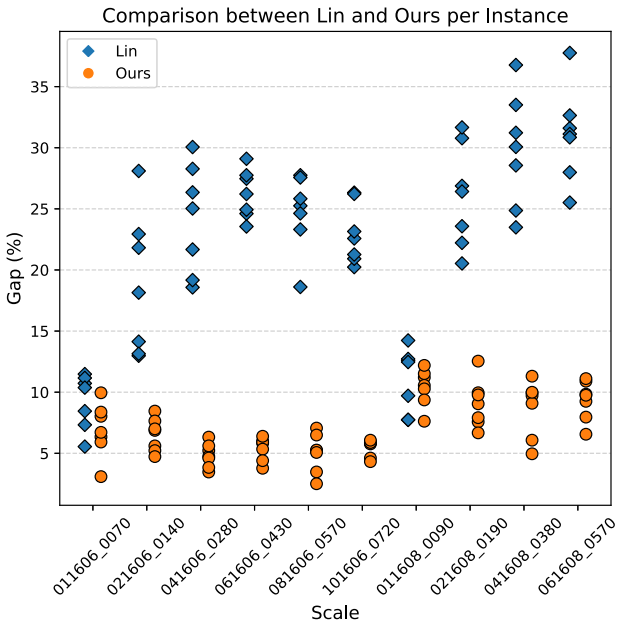


Fig. 5. Per-instance gap comparison with Lin on the benchmark proposed by Lee and Lee (2010).

compared to the strongest baseline, Lin, our approach improves the gap by more than 1 % for the smallest scale, with the improvement widening as the scale increases, reaching up to 22 %. Importantly, although the DRL agent was trained on instances with at most 70 containers, it successfully learned an efficient and robust policy that generalizes to diverse yard configurations containing hundreds of containers.

Similar trends are observed in Table 2 for the U-type instances. GRASP was excluded from the U-type comparison as no results were reported for these instances in the original study. Our approach achieves an average gap of 5.7 % and a maximum gap of 10.8 %, maintaining high-quality solutions robustly across different scales. Against Lin, the gap widens to over 20 % for larger scales. It is worth noting that the DRL agent was trained solely on R-type instances without any specialized training for the U-type. This demonstrates the model's ability to generalize well regardless of the container arrangement.

Fig. 5 presents a clear visual comparison with Lin, plotting the results for five R-type and two U-type instances at each scale. Except for the two smallest scales (011606_0070 and 011608_0090), where the performance difference is less pronounced, our DRL approach consistently outperforms Lin. For these two smallest scales, the performance decrease is limited to an average gap of 1.4 % for only three instances, whereas the remaining 11 instances exhibit clear improvements, averaging 2.3 %. Another noteworthy observation is that Lin's gap increases with problem scale, whereas our DRL approach maintains consistently low gaps, underscoring its robustness.

5.2.2. *Q2. Comparison with existing DRL-based approaches*

We next compare our approach with other DRL-based methods to demonstrate the effectiveness of the proposed learning framework. The first comparison is made with Shin, which was recently published to solve the CRP (Shin et al., 2025). Shin proposed a network model that relies on expert-designed features and was trained using the REINFORCE algorithm with a greedy baseline (Kool et al., 2019b) on a single yard configuration.

To provide a more comprehensive set of comparisons, we additionally implemented two DRL approaches based on Liu et al. (2023) and Ma et al. (2025). Both studies addressed the single-bay BRP with the objective of minimizing the number of relocations and therefore could not be directly applied to the CRP. In particular, their network architectures do not incorporate stack position information, making it impossible to learn or evaluate crane working time. To ensure fairness, we adapted their learning algorithms to our network model. Both methods are based on the REINFORCE algorithm: Liu et al. (2023) used the batch-average performance of the current policy as the baseline, whereas Ma et al. (2025) adopted a greedy baseline (Kool et al., 2019b).

All approaches were trained under the same experimental conditions, with the only difference being that they were trained on a fixed yard configuration consisting of two bays, six rows, six tiers, and fifty containers–chosen in consideration of GPU memory constraints. Although Ma et al. (2025) proposed a multi-decoder architecture for the BRP, this structure was not adopted here, as it would require substantial modifications to our network and replicating multiple decoders for the CRP would lead to excessive memory usage. We refer to the implementations based on Liu et al. (2023) and Ma et al. (2025) as DRL1 and DRL2, respectively. Other DRL studies for the BRP, such as those incorporating beam search with a relocation-prediction network or A* algorithms, were excluded from comparison because their architectures are fundamentally different from ours and cannot be adapted to the CRP simply by replacing the network model.

Table 3
Performance comparison of different DRL-based approaches on the benchmark proposed by Lee and Lee (2010).

Problem	DRL1	DRL2	Shin	Ours	Problem	DRL1	DRL2	Shin	Ours
R011606_0070	34.4	16.6	12.3	**7.7**	U011606_0070	32.0	20.8	14.6	**4.9**
R021606_0140	49.9	14.8	11.2	**7.1**	U021606_0140	80.6	11.3	8.9	**5.0**
R041606_0280	90.0	38.6	10.7	**5.3**	U041606_0280	66.4	28.1	7.7	**3.7**
R061606_0430	80.0	18.0	15.0	**5.9**	U061606_0430	85.4	50.2	9.5	**4.1**
R081606_0570	91.6	29.2	16.1	**5.8**	U081606_0570	94.5	56.7	8.9	**3.0**
R101606_0720	82.6	24.4	17.0	**5.9**	U101606_0720	94.5	58.3	10.2	**4.5**
R011608_0090	49.8	21.8	19.2	**10.2**	U011608_0090	55.3	29.3	33.6	**10.8**
R021608_0190	90.3	22.1	16.8	**9.4**	U021608_0190	116.0	15.7	16.6	**8.2**
R041608_0380	117.4	35.8	20.3	**10.0**	U041608_0380	128.0	42.1	11.6	**5.5**
R061608_0570	126.9	41.8	24.2	**10.2**	U061608_0570	129.5	55.4	15.0	**7.3**

Values represent the percentage gap from the lower bound.

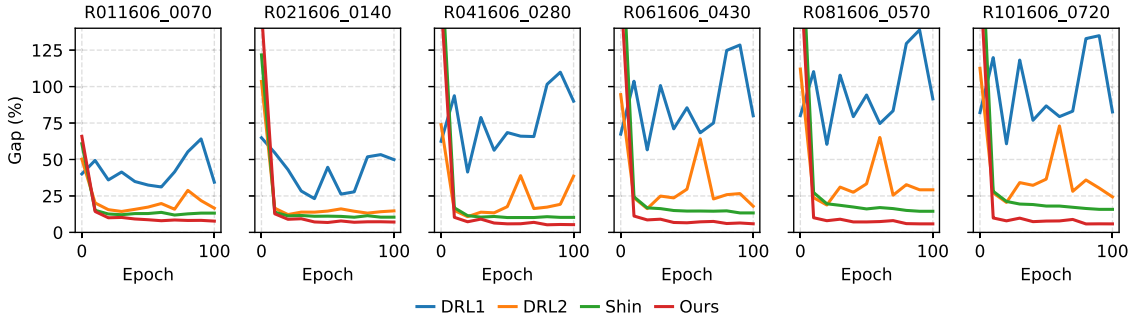


Fig. 6. Performance comparison of the DRL approaches on the random benchmark proposed by Lee and Lee (2010) during training. Each curve indicates the percentage gap from the lower bound measured every 10 epochs.

Table 3 summarizes the performance of all DRL approaches on the full Lee and Lee (2010) benchmark, while Fig. 6 illustrates the training progress for six representative scales. A clear performance hierarchy can be observed among the methods–namely, DRL1, DRL2, Shin, and Ours–in which our proposed approach demonstrates consistently superior performance across all instances. DRL1 employs a baseline computed as the average score of the current policy over different instances within a batch; however, because this baseline value depends on the composition of each batch, the same instance may be evaluated differently across batches, which likely caused unstable training. Note that the implementation of DRL1 was directly reproduced using its publicly available source code, ensuring faithful replication of the original method. DRL2 exhibits relatively smoother convergence, yet stable learning is observed mainly in instance R021606_0140, whose scale most closely matches the training configuration, revealing limited generalization capability and underscoring the importance of our scale-diverse learning strategy.

The Shin model shows stable learning behavior across scales despite being trained on a single yard configuration, which can be attributed to its use of expert-designed features that encode preprocessed yard information. Nevertheless, our proposed approach consistently outperforms all others throughout training, confirming the effectiveness of both our network design and the learning framework. All methods produced solutions within one second, so computation times were not reported.

5.2.3. Q3. Contribution of framework components

Table 4 and Fig. 7 present the results of the ablation study conducted to evaluate the effect of each component of the proposed approach. For the LSTM ablation, instead of using Eqs. (8) and (9), the output of Eq. (7) was averaged over tiers and passed through a fully connected layer. For the attention ablation, the result of Eq. (12) was concatenated with the target stack embedding and the global embedding computed as in Eq. (17), and then passed through a fully connected layer to compute the selection probability for each stack. For the scale-diverse learning ablation, training was performed solely on random instances with a yard scale of two bays, six rows, six tiers, and 50 containers.

Table 4 reports the performance degradation on the Lee and Lee (2010) benchmark when each component is removed during training. Across all scales, each component shows a substantial impact on performance. The LSTM module has a more pronounced effect when the tier (stack height) is larger (8 tiers vs. 6 tiers), suggesting its necessity for effectively processing tier sequences; the performance difference averages 2.0 % vs. 0.6 % for R-type and 2.8 % vs. 1.4 % for U-type instances. The attention mechanism yields an even larger effect than the LSTM, with the R-type performance gap ranging from 3.7 % to 11.3 % and the U-type from 2.1 % to 10.0 %. Its impact is particularly evident in smaller-scale cases, likely because with fewer stacks, clearer inter-stack information exchange allows for more detailed action selection. In contrast, the scale-diverse learning has the greatest impact on larger-scale cases, with performance differences reaching 15.6 % for R-type and 26.7 % for U-type instances. These results indicate that scale-diverse learning is essential for achieving scalability and for ensuring the robustness of the DRL approach.

Table 4
Ablation study results on the benchmark proposed by Lee and Lee (2010).

Problem	w/o LSTM	w/o Attn.	w/o SD learning	Problem	w/o LSTM	w/o Attn.	w/o SD learning
R011606_0070	1.5 (↑)	7.8 (↑)	2.0 (↑)	U011606_0070	1.5 (↑)	10.0 (↑)	0.9 (↑)
R021606_0140	0.2 (↑)	5.4 (↑)	0.8 (↑)	U021606_0140	0.3 (↑)	3.4 (↑)	1.5 (↑)
R041606_0280	0.7 (↑)	5.1 (↑)	7.1 (↑)	U041606_0280	1.5 (↑)	3.7 (↑)	7.6 (↑)
R061606_0430	0.6 (↑)	5.2 (↑)	12.5 (↑)	U061606_0430	1.6 (↑)	3.7 (↑)	14.4 (↑)
R081606_0570	0.3 (↑)	3.9 (↑)	12.4 (↑)	U081606_0570	1.8 (↑)	2.7 (↑)	22.5 (↑)
R101606_0720	0.3 (↑)	3.7 (↑)	14.7 (↑)	U101606_0720	1.9 (↑)	2.1 (↑)	26.7 (↑)
R011608_0090	2.1 (↑)	11.3 (↑)	4.5 (↑)	U011608_0090	1.5 (↑)	6.2 (↑)	4.2 (↑)
R021608_0190	2.8 (↑)	9.4 (↑)	2.3 (↑)	U021608_0190	4.2 (↑)	4.0 (↑)	1.7 (↑)
R041608_0380	1.6 (↑)	9.1 (↑)	11.3 (↑)	U041608_0380	3.5 (↑)	6.6 (↑)	13.3 (↑)
R061608_0570	1.7 (↑)	6.9 (↑)	15.6 (↑)	U061608_0570	2.0 (↑)	4.2 (↑)	20.4 (↑)

w/o LSTM: Without LSTM layers. w/o Attn.: Without attention mechanism. w/o SD learning: Without scale-diverse learning. Values represent the gap difference(%) from the proposed model. (↑) indicates an increase in gap (i.e., degradation).

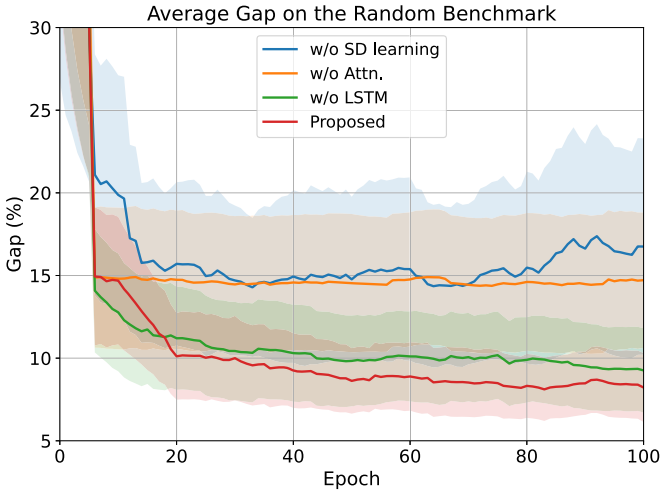


Fig. 7. Average gap on the random benchmark proposed by Lee and Lee (2010) during training. Each curve shows the moving average of average gap values, and shaded regions represent one standard deviation.

Fig. 7 illustrates the average gap over all R-type instances of the Lee and Lee (2010) benchmark during training for each model variant. The results again highlight the clear contribution of each component. For the LSTM ablation, the model initially shows a faster convergence rate than the proposed model; however, it ultimately converges to consistently worse performance. In the case of the attention ablation, the model quickly converges to a poor performance level and exhibits no further improvement throughout training. For the scale-diverse learning ablation, stable training could not be achieved, and the model ultimately fails to converge.

5.2.4. Q4. Scalability to extremely large instances

Table 5 and Fig. 8 present the evaluation results on the extremely large-scale benchmark that we generated. Following the procedure of Lee and Lee (2010), we created instances with 20 and 30 bays containing several thousand containers to further assess the scalability of our DRL approach. For comparison, we selected GP and Lin, which showed relatively strong performance among the existing baselines. For each scale, 20 instances were generated for both R- and U-type configurations. Table 5 reports the average gap and computation time, while Fig. 8 provides box plots comparing our DRL approach with the best-performing baseline, Lin.

As shown in Table 5, our proposed DRL approach consistently outperforms the baselines across all scales and produces high-quality solutions within 40 seconds even for extremely large-scale instances. When compared with Lin, our approach achieves a performance advantage ranging from 10.6 % to 23.9 %. Furthermore, as illustrated in Fig. 8, the performance gap between our approach and Lin is clearly distinguishable across all evaluated scales.

While the above experiments provide clear evidence of the proposed model's scalability, it is acknowledged that real container yards with dozens of bays are typically operated by multiple cranes. Therefore, the results presented here may not fully represent such multi-crane environments. Nonetheless, verifying that the DRL approach remains robust even for instances with several thousand containers–far exceeding the scale of typical problems–demonstrates its strong stability and reliability in terms of both solution quality and computation time. This scalability provides confidence that the proposed method can reliably and effectively handle practical-scale problems (involving several hundred containers), even when confronted with unseen and more complex problem distributions. A more detailed discussion on extending the proposed approach to multi-crane environments is provided in Section 6.

Table 5
Performance comparison on the extremely large-scale benchmark.

Problem	GP		Lin		Ours	
	Gap	Time	Gap	Time	Gap	Time
R201606_1440	40.8	23.8	25.0	21.4	**8.2**	9.8
R301606_2160	37.4	52.9	26.3	47.5	**13.9**	16.7
R201608_1920	46.9	38.0	31.4	35.1	**12.3**	20.8
R301608_2880	43.1	83.2	31.7	76.3	**21.1**	32.4
U201606_1440	58.8	31.9	30.7	29.9	**6.8**	16.1
U301606_2160	55.4	69.7	33.2	65.5	**15.0**	24.9
U201608_1920	62.6	46.6	32.7	44.2	**10.2**	24.4
U301608_2880	59.3	103.6	35.2	97.4	**24.3**	39.3

Gap: Percentage gap with lower bound. Time: Computation time in seconds.

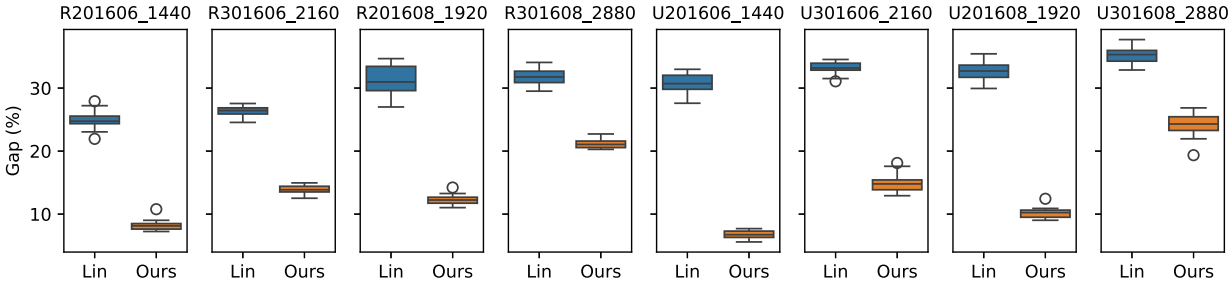


Fig. 8. Percentage gap with lower bound comparison between Lin and Ours on the extremely large-scale benchmark.

5.2.5. Q5. Effectiveness in online setting

In the standard CRP, it is assumed that the complete retrieval order of all containers is known in advance. However, in practice, this assumption is often unrealistic, which has motivated studies on the online CRP, where only the retrieval orders of the next H containers are known, and one additional retrieval order is revealed after each retrieval (Zehendner et al., 2017). To evaluate our DRL approach in such an online setting, we set $H \in \{1, 5, 10, 20\}$ and tested it on the Lee and Lee (2010) benchmark. For comparison, we considered the leveling heuristic proposed by Zehendner et al. (2017), the Lin heuristic adapted to the online setting, and the Lin heuristic in the standard CRP (full information).

When applying Lin to the online setting, the retrieval order of all unknown containers was set to $H + 1$, assuming that the known order is re-indexed from 1 at each retrieval. In the online setting, Lin may encounter unintended ties during decision making. To address this, we incorporated an additional tie-breaking rule that selects the stack with the shortest travel time. For the DRL approach, the retrieval order of unknown containers was represented by a learnable single parameter that can be optimized during training. All the online results of the DRL approach were obtained using the model trained under the $H = 20$ setting, demonstrating its generalization capability across different lookahead horizons.

The leveling heuristic relocates blocking containers to the stack with the lowest height in order to equalize stack heights. It was originally designed for the single-bay case; to extend it to the multi-bay problem, we restricted relocations to stacks within the same bay whenever possible, as bay movements are considerably longer than row movements, and broke ties by selecting the stack with the shortest travel time. Notably, without restricting relocations to the same bay, the gap increases to almost four times its original value.

Table 6 reports the performance of our DRL approach in the online setting relative to each baseline. Note that the leveling heuristic operates independently of the lookahead horizon H, and the results of Lin and Ours under the full-information setting are duplicated from Table 1 for reference. For each H, our method outperforms both the leveling heuristic and Lin across all tested scales. Compared with the leveling heuristic, the performance gap is more pronounced for smaller-scale instances, whereas compared with Lin, the largest gap is observed for larger-scale instances. Notably, even when compared with Lin in the full-information CRP, our DRL approach achieves superior performance in the online setting, particularly for large-scale problems.

Fig. 9 visualizes the results reported in Table 6. It can be observed that our DRL approach consistently outperforms the leveling heuristic across all problem scales in the online setting. When compared with Lin under the full-information setting, Lin performs better on smaller-scale instances; however, our approach achieves superior performance on larger-scale instances even under the online setting. As expected, the performance of our method improves with increasing lookahead horizon H. Moreover, as the problem scale grows (i.e., the number of containers increases), the performance differences across different H values become less pronounced. Although the performance is inferior to that under the full-information setting, these results demonstrate the capability of our DRL approach to learn effective decision-making rules based solely on container configurations, even when the retrieval order information is incomplete.

Table 6
Performance comparison in the online setting for the benchmark proposed by Lee and Lee (2010).

Problem		Online ($H=1$)		Online ($H=5$)		Online ($H=10$)		Online ($H=20$)		Full information	
	Leveling	Lin	Ours	Lin	Ours	Lin	Ours	Lin	Ours	Lin	Ours
R011606_0070	31.1	31.7	**26.8**	25.0	**18.8**	17.7	**13.8**	13.2	**10.2**	9.5	7.7
R021606_0140	21.0	35.2	**16.5**	31.8	**15.4**	28.4	**14.4**	25.8	**12.1**	20.0	7.1
R041606_0280	18.2	35.2	**14.3**	34.9	**13.9**	32.5	**12.9**	29.8	**11.5**	22.5	5.3
R061606_0430	18.8	37.1	**16.8**	36.0	**15.1**	35.3	**15.3**	33.6	**14.1**	25.4	5.9
R081606_0570	17.5	37.3	**15.0**	35.8	**14.4**	35.6	**14.3**	33.6	**13.2**	23.5	5.8
R101606_0720	16.5	36.4	**15.8**	36.3	**16.0**	34.6	**16.1**	33.2	**15.2**	21.6	5.9
R011608_0090	38.5	39.9	**33.3**	30.8	**28.9**	24.5	**22.1**	17.4	**15.7**	11.4	10.2
R021608_0190	34.0	46.1	**29.3**	43.0	**27.8**	36.6	**23.4**	32.1	**19.1**	26.0	9.4
R041608_0380	31.6	47.3	**27.5**	44.1	**25.2**	43.2	**24.2**	38.6	**22.0**	32.0	10.0
R061608_0570	28.4	47.2	**26.1**	45.1	**26.6**	44.1	**25.5**	39.9	**24.0**	32.2	10.2

Values represent the percentage gap from the lower bound. “Full information" indicates complete knowledge of future retrieval orders. Bold values indicate the best performance among the methods under the same lookahead horizon (H), as well as the leveling heuristic.

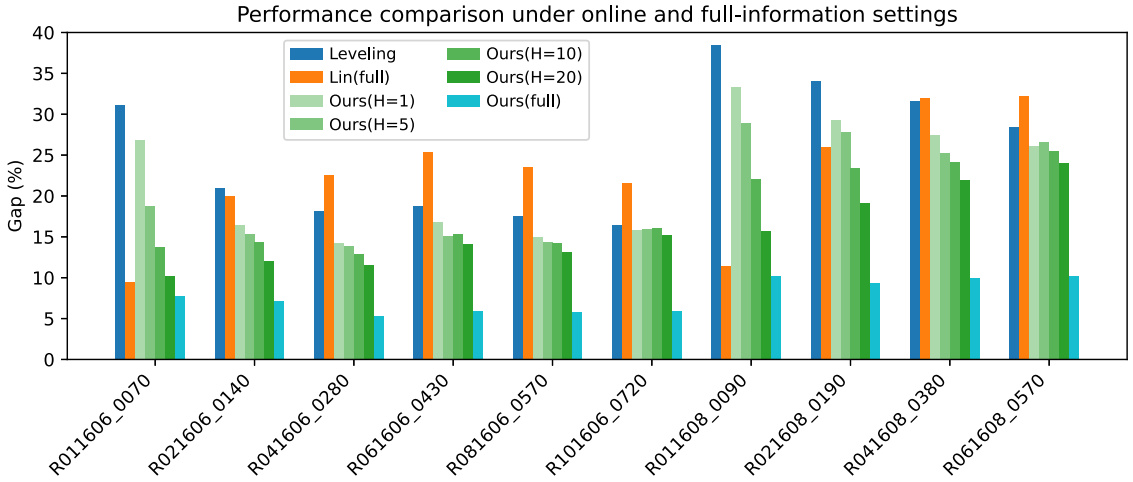


Fig. 9. Performance comparison under online and full-information settings on the benchmark proposed by Lee and Lee (2010).

Although the computation times are not explicitly reported in Table 6, they are equivalent to those in Table 1, as the same network architecture was used for all experiments. These results collectively highlight one of the key advantages of the proposed DRL framework–its ability to make rapid, high-quality decisions in dynamic and uncertain environments. Once trained, the model infers actions within milliseconds, enabling real-time decision making without the need for re-optimization. From this perspective, the online experiments demonstrate how this decision-making capability, coupled with fast inference, provides a practical and scalable solution for real-world container retrieval operations.

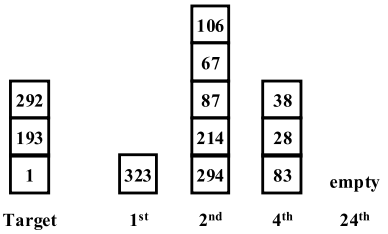
5.2.6. Q6. Behavioral analysis of the DRL agent

In this section, we provide a detailed analysis of the behavior of the trained DRL agent. When relocating a blocking container, it is generally considered a rational decision to place it onto a stack composed exclusively of containers with retrieval orders larger than that of the blocking container itself. Such stacks are referred to as *well-located*, as this relocation does not cause any further blocking. Expert-designed heuristics often adopt this decision rule aggressively; in particular, Lin applies a strict rule that, whenever a well-located stack is available, the blocking container must be relocated there.

However, our analysis shows that only 80 % of the actions taken by the DRL agent are well-located. This indicates that always selecting a well-located stack, even when available, does not necessarily yield the best overall outcome. Moreover, since bay travel requires significantly more time than row travel, bay changes are generally chosen more conservatively. Consistent with this intuition, we observe that among relocations involving a bay change, the proportion of well-located decisions is higher, at 95 %.

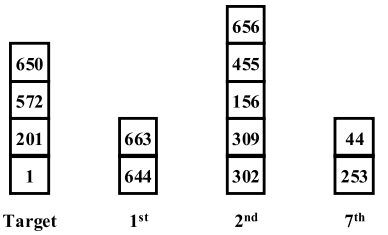
To obtain a more granular view of the agent’s decision process, we analyze three representative decision scenarios and examine how the agent prioritizes candidate stacks. Fig. 10 illustrates one example for each scenario. In every subfigure, the left panel is a table that lists the bay-row coordinates of the target stack and a subset of relocation candidates selected for their characteristic features. For each candidate, we report its bay-row location, its action rank within the feasible action space, the selection probability output by the policy, and whether it is well-located. The right panel provides a schematic of the yard highlighting the target stack and the selected candidates.

Stack	Bay	Row	Probability	Well
Target	2	4	-	-
1st	4	4	32.9%	True
2nd	2	6	32.3%	False
4th	2	3	19.3%	False
24th	1	1	0.0%	True



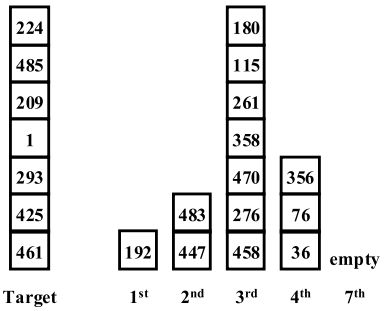
(a) Scenario 1 – Well-located: Container placed without future blocking.

Stack	Bay	Row	Probability	Well
Target	10	14	-	-
1st	9	7	17.37%	False
2nd	10	13	17.36%	False
7th	10	3	0.02%	False



(b) Scenario 2 – No well-located option: All stacks lead to future blocking.

Stack	Bay	Row	Probability	Well
Target	5	2	-	-
1st	4	3	18%	False
2nd	5	1	16%	True
3rd	5	13	13%	False
4th	5	11	9%	False
7th	5	5	7%	True



(c) Scenario 3 – Well-located not selected: Well-located option existed but was not chosen.

Fig. 10. Three representative decision scenarios illustrating the DRL agent's stack selection behavior.

The first scenario, illustrated in Fig. 10(a), shows the agent relocating a blocking container to a different bay in order to achieve a well-located placement. In this case, there was no stack within the same bay (bay 2) that could provide a well-located placement for the blocking container 292. Consequently, the highest-probability choice was a well-located stack in bay 4, despite the need for a bay change. The second-ranked choice was a stack in row 6 of the same bay. Notably, despite not enabling a well-located relocation, the second-ranked stack had a selection probability comparable to that of the top-ranked stack, reflecting the agent's conservative tendency to avoid bay changes.

The second-ranked stack also had the largest minimum priority order among all candidate stacks in bay 2. Compared to the fourth-ranked candidate in row 3, the second-ranked stack had a minimum priority order of 67 vs. 28, meaning that relocating container 292 there would delay the point at which it must be moved again to retrieve container 67 or 28. This explains the large difference in their selection probabilities. The third-ranked stack is omitted from the discussion, as it shares the same characteristics as the second-ranked one.

Additionally, there was an empty stack in bay 1, which, like the top-ranked stack, could have provided a well-located placement. However, its selection probability was considerably lower. This is because the empty stack in bay 1 had a higher likelihood of being used to well-locate other containers in future moves. Given the availability of a better well-located option (the 1st-ranked stack), the empty stack was entirely excluded from consideration, despite being in a closer bay. These decision patterns are similar to the deterministic logic used in traditional expert-designed heuristics.

The second scenario, shown in Fig. 10(b), represents a case where no stack was available to well-locate container 650. The highest-ranked candidate was the stack in bay 9, row 7, which required a bay change yet still received the highest selection probability. Compared to the second-ranked stack in bay 10, row 13, the highest-ranked stack contained containers 663 and 644, whose retrieval times would be very close to that of container 650. This likely explains why it was considered a favorable action despite the cost of a bay move. Within the same bay (bay 10), the selection pattern followed the same reasoning as in Scenario 1: stacks with a larger minimum priority order received higher selection probabilities. For instance, the second-ranked stack had a selection probability of

17.36 %, whereas the seventh-ranked stack's probability was only 0.02 %, due to a large gap in their minimum priority orders (156 vs. 44). Stacks ranked 3rd to 6th are omitted from the analysis, as their characteristics were similar to those of the 2nd-ranked stack.

The last case, shown in Fig. 10(c), involves relocating container 224 to a stack in bay 4, even though there was a stack within the same bay (bay 5) where relocating 224 would have resulted in a well-located relocation. While the 2nd-ranked stack could have yielded a well-located relocation, placing container 224 there would eliminate the opportunity to subsequently well-locate containers 225–446 in that stack. Hence, the agent appeared to judge that relocating to a stack containing only container 192–which had a retrieval order close to that of 224–was also valuable despite the bay movement. For other stacks in bay 5 where a well-located relocation was not possible, stacks with a larger minimum priority order again received higher selection probabilities. Bay 5 also contained an empty stack that could have yielded a well-located relocation; however, similar to the analysis in Scenario 1, it was relatively disfavored due to its higher likelihood of accommodating other containers in a well-located manner compared to the 2nd-ranked stack.

In summary, the DRL agent demonstrates a clear understanding of the importance of stacks that allow for well-located relocations and those with a large minimum priority order, and it makes decisions accordingly. Furthermore, when multiple well-located relocation options exist, the agent accurately assesses how much each option would limit future opportunities for well-located relocations. It also considers the cost of bay changes, often favoring options that minimize unnecessary inter-bay movements unless a significant future benefit is expected. These types of knowledge are all exploited in expert-designed heuristics, where they are applied through deterministic logic. However, the DRL agent leverages these principles while accounting for complex, intertwined conditions, enabling it to select actions that ultimately yield higher-quality solutions.

6. Conclusion

We presented a novel DRL approach to the CRP that combines a size-agnostic network architecture with a scale-diverse learning framework. The resulting policy generalizes across heterogeneous yard layouts and scales, including those unseen during training, substantially outperforming all existing baselines on established benchmarks while producing solutions within a second for realistic instances. Behaviorally, the learned policy internalizes principles used by human-designed heuristics yet adapts them flexibly by weighing future opportunity costs and travel-time asymmetries, which contributes to superior solution quality. Moreover, the proposed method demonstrates strong robustness and adaptability in dynamic and online environments, and, when combined with its rapid decision-making capability, proves highly practical for real-world deployment. Its consistent performance under time-critical and continuously changing yard conditions further highlights its potential for real-time control and decision support in automated container terminals.

A key future extension involves operating multiple cranes within a yard block composed of multiple bays. Efficiency considerations in large blocks can motivate the use of multiple cranes, which inherently introduces physical interference among cranes. Conceptually related to this challenge is the multi-crane scheduling problem (MCSP), where a predefined set of transportation tasks must be assigned and sequenced across cranes while obtaining a non-interfering schedule (Shin and Kim, 2025). An integrated perspective combining CRP with MCSP would simultaneously optimize both relocation destinations and crane schedules, naturally leading to a bi-level optimization framework, where the upper level defines CRP-related decisions and the lower level represents non-interference MCSP coordination. Extending our DRL approach to such a unified setting represents a practically significant research direction with considerable operational value for multi-crane yard blocks.

In addition, the proposed methodology can be extended to more general retrieval-order structures. Rather than assuming a fully ordered retrieval sequence, future work may consider arbitrary precedence relations derived from vessel stowage plans. Accounting for the arrival of new containers would further broaden applicability, enabling the joint treatment of retrieval, relocation, and storage decisions. Moreover, different online-revelation patterns for retrieval orders present a rich spectrum of dynamic settings that align well with the adaptive and real-time nature of the proposed learning framework.

Data availability

All data and code will be publicly accessible through the link included in the manuscript.

CRediT authorship contribution statement

Woo-Jin Shin: Writing - review & editing, Writing - original draft, Visualization, Validation, Software, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization; **Inguk Choi:** Writing - review & editing, Validation, Software, Methodology, Investigation, Data curation; **Sang-Hyun Cho:** Writing - review & editing, Software, Methodology, Investigation, Data curation, Conceptualization; **Hyun-Jung Kim:** Writing - review & editing, Supervision, Resources, Project administration, Methodology, Funding acquisition, Conceptualization.

Appendix A. Comprehensive results by instance

For completeness, this appendix provides the total crane working time lower bound values derived from Theorem 2, along with the corresponding results obtained by the proposed DRL approach, for every individual instance used in this study. The results for the Lee and Lee (2010) benchmark are presented in Table A.7, while the results for the extremely large-scale benchmark generated in this study are reported in Table A.8.

Table A.7
Lower bounds and our DRL results on crane working time for the Lee and Lee (2010)'s benchmark.

Problem	LB	Ours	Problem	LB	Ours	Problem	LB	Ours
R011606_0070_001	4520.4	4807.2	R101606_0720_001	83976.4	88874.2	U011606_0070_001	5239.2	5401.2
R011606_0070_002	4544.4	4909.2	R101606_0720_002	82196.0	86938.6	U011606_0070_002	5182.8	5530.8
R011606_0070_003	4599.6	4872.0	R101606_0720_003	82932.9	87841.3	U021606_0140_001	13449.0	14151.0
R011606_0070_004	4714.8	5184.0	R101606_0720_004	83099.2	87990.8	U021606_0140_002	13620.3	14264.7
R011606_0070_005	4558.8	4940.4	R101606_0720_005	82410.5	87415.7	U041606_0280_001	30541.8	31601.4
R021606_0140_001	12373.5	13067.1	R011608_0090_001	6146.4	6835.2	U041606_0280_002	30470.2	31641.4
R021606_0140_002	11612.7	12594.3	R011608_0090_002	6094.8	6796.8	U061606_0430_001	49200.8	51058.8
R021606_0140_003	12305.4	13151.4	R011608_0090_003	6386.4	7060.8	U061606_0430_002	49392.4	51565.0
R021606_0140_004	12949.8	13939.8	R011608_0090_004	6234.0	6709.2	U081606_0570_001	68488.9	70868.5
R021606_0140_005	12068.4	12913.2	R011608_0090_005	6180.0	6814.8	U081606_0570_002	66918.6	68598.2
R041606_0280_001	29393.2	31252.0	R021608_0190_001	17308.2	19479.0	U101606_0720_001	88442.3	92520.9
R041606_0280_002	28005.7	29469.1	R021608_0190_002	17039.7	18581.7	U101606_0720_002	89142.2	92991.0
R041606_0280_003	28164.8	29502.8	R021608_0190_003	17779.8	19549.8	U011608_0090_001	6931.2	7580.4
R041606_0280_004	28063.6	29638.0	R021608_0190_004	17115.3	18411.3	U011608_0090_002	7027.2	7884.0
R041606_0280_005	28696.6	30020.2	R021608_0190_005	17554.8	18944.4	U021608_0190_001	18648.3	19892.7
R061606_0430_001	46704.5	49427.3	R041608_0380_001	39623.0	43565.0	U021608_0190_002	18241.5	20024.7
R061606_0430_002	45977.8	48741.4	R041608_0380_002	39272.6	43713.8	U041608_0380_001	43109.9	45728.3
R061606_0430_003	45840.0	48545.2	R041608_0380_003	38556.6	42304.2	U041608_0380_002	42509.7	44618.1
R061606_0430_004	45533.4	47965.0	R041608_0380_004	39542.9	43138.7	U061608_0570_001	66255.8	71530.6
R061606_0430_005	46105.3	49053.1	R041608_0380_005	39399.0	43336.2	U061608_0570_002	66732.8	71110.8
R081606_0570_001	63535.5	66863.9	R061608_0570_001	62023.6	68784.2			
R081606_0570_002	63783.9	67141.5	R061608_0570_002	62080.0	68978.6			
R081606_0570_003	64565.3	69125.9	R061608_0570_003	62551.8	68339.8			
R081606_0570_004	63768.9	67910.7	R061608_0570_004	62960.8	69155.2			
R081606_0570_005	64902.9	68192.9	R061608_0570_005	63951.0	70186.4			

Table A.8
Lower bounds and our DRL results on crane working time for the extremely large-scale benchmark.

Problem	LB	Ours	Problem	LB	Ours	Problem	LB	Ours
R201606_1440_001	185654.7	201141.1	R301608_2880_001	412618.3	497210.1	U201608_1920_001	266991.6	293230.6
R201606_1440_002	185017.4	199287.2	R301608_2880_002	412466.5	502117.0	U201608_1920_002	265067.1	289895.9
R201606_1440_003	184467.5	201070.1	R301608_2880_003	413674.7	502627.5	U201608_1920_003	265811.9	290388.5
R201606_1440_004	186344.6	200553.8	R301608_2880_004	410666.0	503958.8	U201608_1920_004	265728.0	294706.4
R201606_1440_005	186561.5	203384.8	R301608_2880_005	412071.9	499865.1	U201608_1920_005	264772.7	292591.4
R201606_1440_006	184526.2	198026.2	R301608_2880_006	411946.9	499226.9	U201608_1920_006	265889.6	289885.4
R201606_1440_007	185334.0	198757.6	R301608_2880_007	412898.6	496905.9	U201608_1920_007	264972.8	291671.2
R201606_1440_008	185210.1	201308.9	R301608_2880_008	410088.6	493372.1	U201608_1920_008	267162.3	294065.9
R201606_1440_009	184670.8	204576.2	R301608_2880_009	412112.4	501641.0	U201608_1920_009	267328.1	291699.9
R201606_1440_010	184788.0	198762.0	R301608_2880_010	413041.2	499005.4	U201608_1920_010	266219.2	291002.8
R201606_1440_011	185435.9	200369.9	R301608_2880_011	413977.5	499603.5	U201608_1920_011	266528.1	294988.5
R201606_1440_012	184427.2	199992.0	R301608_2880_012	410509.9	496623.7	U201608_1920_012	266693.3	299857.1
R201606_1440_013	185828.0	201455.0	R301608_2880_013	412386.9	497928.9	U201608_1920_013	266394.4	294127.6
R201606_1440_014	185457.3	200694.7	R301608_2880_014	410592.2	495009.8	U201608_1920_014	266188.7	294678.9
R201606_1440_015	184913.6	198982.4	R301608_2880_015	412599.9	502827.9	U201608_1920_015	266396.4	294450.8
R201606_1440_016	184844.7	200348.5	R301608_2880_016	414346.7	498359.3	U201608_1920_016	267346.2	295441.6
R201606_1440_017	182950.3	198843.7	R301608_2880_017	415515.1	500920.2	U201608_1920_017	266175.2	294439.6
R201606_1440_018	184145.3	197895.7	R301608_2880_018	412191.0	501014.1	U201608_1920_018	266326.1	292784.7
R201606_1440_019	185001.8	199251.2	R301608_2880_019	416190.0	507566.6	U201608_1920_019	264807.0	293353.0
R201606_1440_020	184487.1	199223.1	R301608_2880_020	416085.6	504054.0	U201608_1920_020	266590.9	292046.3
R301606_2160_001	303756.6	348499.2	U201606_1440_001	196601.6	210028.0	U301608_2880_001	435124.5	545376.1
R301606_2160_002	302575.9	344546.1	U201606_1440_002	198429.9	212167.3	U301608_2880_002	432316.6	515982.6
R301606_2160_003	304327.0	347307.2	U201606_1440_003	198878.7	210525.7	U301608_2880_003	434504.3	529878.9
R301606_2160_004	304172.2	344212.1	U201606_1440_004	198509.8	213603.4	U301608_2880_004	437012.7	544157.8
R301606_2160_005	304376.6	345883.3	U201606_1440_005	196719.0	209154.8	U301608_2880_005	435125.7	548411.6
R301606_2160_006	306153.6	350379.8	U201606_1440_006	196565.0	207543.2	U301608_2880_006	433713.7	534753.9
R301606_2160_007	306865.6	352627.8	U201606_1440_007	198668.2	212780.4	U301608_2880_007	435819.8	543598.1
R301606_2160_008	308330.9	348655.8	U201606_1440_008	197122.8	211115.2	U301608_2880_008	433442.9	535520.3
R301606_2160_009	303859.2	346141.6	U201606_1440_009	195770.8	210842.0	U301608_2880_009	435966.9	536529.9
R301606_2160_010	303545.6	345627.1	U201606_1440_010	198424.2	210750.0	U301608_2880_010	435100.3	540152.3
R301606_2160_011	302977.5	344565.3	U201606_1440_011	196984.0	210023.0	U301608_2880_011	435850.1	550684.9
R301606_2160_012	305787.9	348127.9	U201606_1440_012	198376.9	213467.3	U301608_2880_012	436730.8	549350.9
R301606_2160_013	303441.5	341426.1	U201606_1440_013	198410.6	213485.0	U301608_2880_013	436365.5	537656.6
R301606_2160_014	304242.9	348073.9	U201606_1440_014	198003.1	212327.5	U301608_2880_014	435833.8	552884.2
R301606_2160_015	305081.8	346795.6	U201606_1440_015	196592.2	209000.8	U301608_2880_015	433882.3	533617.3
R301606_2160_016	301912.8	343771.4	U201606_1440_016	197923.3	210096.9	U301608_2880_016	435474.3	544949.3
R301606_2160_017	306002.5	344402.9	U201606_1440_017	196671.1	209720.7	U301608_2880_017	436338.1	539444.1
R301606_2160_018	303727.0	348970.0	U201606_1440_018	197184.5	209553.3	U301608_2880_018	436054.6	542565.0
R301606_2160_019	304532.7	350078.7	U201606_1440_019	196441.6	211199.6	U301608_2880_019	433420.7	538233.9
R301606_2160_020	305639.6	345422.6	U201606_1440_020	196655.7	209642.1	U301608_2880_020	436492.1	553282.3
R201608_1920_001	252268.6	283004.8	U301606_2160_001	321510.6	370771.0			
R201608_1920_002	252728.9	284748.5	U301606_2160_002	323066.5	377384.5			
R201608_1920_003	250471.5	281013.5	U301606_2160_003	321963.6	369534.8			
R201608_1920_004	250255.2	281041.6	U301606_2160_004	321455.6	365493.6			
R201608_1920_005	252902.1	282638.5	U301606_2160_005	323929.1	371224.7			
R201608_1920_006	250628.7	279820.7	U301606_2160_006	322327.2	373090.8			
R201608_1920_007	251625.6	282530.6	U301606_2160_007	322978.4	364708.6			
R201608_1920_008	251145.5	282870.5	U301606_2160_008	322882.5	369928.2			
R201608_1920_009	251911.7	281140.1	U301606_2160_009	321077.3	379242.7			
R201608_1920_010	251710.9	280583.4	U301606_2160_010	322791.8	366297.2			
R201608_1920_011	250342.0	282164.5	U301606_2160_011	320925.4	368454.2			
R201608_1920_012	251499.6	284872.2	U301606_2160_012	323233.5	365738.5			
R201608_1920_013	251687.5	280572.5	U301606_2160_013	324866.4	369999.6			
R201608_1920_014	252718.8	288659.2	U301606_2160_014	322807.5	371046.7			
R201608_1920_015	249737.4	277281.6	U301606_2160_015	323714.7	380648.9			
R201608_1920_016	253647.1	286444.7	U301606_2160_016	324061.8	378249.4			
R201608_1920_017	252918.3	283668.3	U301606_2160_017	321524.1	367252.1			
R201608_1920_018	253161.6	284912.8	U301606_2160_018	318680.0	367026.0			
R201608_1920_019	252697.2	283099.6	U301606_2160_019	321757.3	370053.1			
R201608_1920_020	250768.9	283439.7	U301606_2160_020	323741.0	368363.6			

References

Bian, Z., Jin, Z.-h., 2013. Optimization on retrieving containers based on multi-phase hybrid dynamic programming. Procedia-Soc. Behav. Sci. 96, 844–855.
Caserta, M., Schwarze, S., Voß, S., 2012. A mathematical formulation and complexity considerations for the blocks relocation problem. Eur. J. Oper. Res. 219 (1), 96–104.
Cifuentes, C.D., Riff, M.C., 2020. G-CREM: A GRASP approach to solve the container relocation problem for multibays. Appl. Soft Comput. 97, 106721.

Dragović, B., Dragović, A., Dulebenets, M.A., 2025. The quay crane operation problem at marine container terminals: bibliometric analysis, emerging trends, and future research opportunities. Transp. Res. Part E: Logistics and Transportation Review 201, 104266.
Forster, F., Bortfeldt, A., 2012. A tree search procedure for the container relocation problem. Comput. Oper. Res. 39 (2), 299–309.
Guo, L., Zheng, J., Du, J., Gao, Z., Fagerholt, K., 2024. Integrated planning of berth allocation, quay crane assignment and yard assignment in multiple cooperative terminals. Transp. Res. Part E: Logist. Transp. Rev. 183, 103456.
Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9 (8), 1735–1780.
Hu, H., Mo, J., Zhen, L., 2024. Integrated optimization of container allocation and yard cranes dispatched under delayed transshipment. Transp. Res. Part C: Emerg. Technol. 158, 104429.
Kim, K.-H., 2024. Planning and Operation of Container Terminals. Elsevier.
Kim, K.H., Hong, G.-P., 2006. A heuristic rule for relocating blocks. Comput. Oper. Res. 33 (4), 940–954.
Kim, Y., Kim, T., Lee, H., 2016. Heuristic algorithm for retrieving containers. Comput. Ind. Eng. 101, 352–360.
Kool, W., van Hoof, H., Welling, M., 2019a. Buy 4 reinforce samples, get a baseline for free! In: ICLR 2019 Workshop: Deep RL Meets Structured Prediction. Workshop paper.
Kool, W., Van Hoof, H., Welling, M., 2019b. Attention, learn to solve routing problems! In: International Conference on Learning Representations.
Kwon, Y.-D., Choo, J., Kim, B., Yoon, I., Gwon, Y., Min, S., 2020. Pomo: policy optimization with multiple optima for reinforcement learning. Adv. Neural Inf. Process. Syst. 33, 21188–21198.
Larsen, R.B., Guo, W., Atasoy, B., 2023. A real-time synchromodal framework with co-planning for routing of containers and vehicles. Transp. Res. Part C: Emerg. Technol. 157, 104412.
Lee, Y., Lee, Y.-J., 2010. A heuristic for retrieving containers from a yard. Comput. Oper. Res. 37 (6), 1139–1147.
Lin, D.-Y., Lee, Y.-J., Lee, Y., 2015. The container retrieval problem with respect to relocation. Transp. Res. Part C: Emerg. Technol. 52, 132–143.
Liu, F., Lin, X., Wang, Z., Zhang, Q., Xialiang, T., Yuan, M., 2024. Multi-task learning for routing problem with cross-problem zero-shot generalization. In: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 1898–1908.
Liu, F., Ye, T., Zhang, Z., 2023. Dynamic attention model–a deep reinforcement learning approach for container relocation problem. In: International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems. Springer, pp. 273–285.
Liu, L., Feng, Y., Zeng, Q., Chen, Z., Li, Y., 2025. A q-learning-based algorithm for the block relocation problem. J. Heurist. 31 (1), 14.
Ma, J., Yang, Y., Xiang, R., Liu, X., Li, Y., 2025. Multi-decoder dynamic attention model for container relocation problem. Transp. Res. Record: J. Transp. Res. Board 2679 (6), 798–814.
Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al., 2015. Human-level control through deep reinforcement learning. Nature 518 (7540), 529–533.
Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O., 2017. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.
Shin, W.-J., Jung, J.-K., Cho, S.-H., Kim, H.-J., 2025. Deep reinforcement learning for the container retrieval problem. In: 2025 IEEE 21st International Conference on Automation Science and Engineering (CASE). IEEE, pp. 2990–2995.
Shin, W.-J., Kim, H.-J., 2025. Advances in multi-crane scheduling: extended non-interference modeling and temporal-constraint-based valid inequalities. Eur. J. Oper. Res. https://doi.org/10.1016/j.ejor.2025.10.032.
Tanaka, S., ElWakil, M., Eltawil, A., 2024. The parallel stack loading problem of minimizing the exact number of relocations. Comput. Oper. Res. 169, 106712.
Tanaka, S., Tierney, K., Parreño-Torres, C., Alvarez-Valdes, R., Ruiz, R., 2019. A branch and bound approach for large pre-marshalling problems. Eur. J. Oper. Res. 278 (1), 211–225.
Ünlüyurt, T., Aydın, C., 2012. Improved rehandling strategies for the container retrieval process. J. Adv. Transp. 46 (4), 378–393.
Đurasević, M., Đumić, M., 2024. Designing relocation rules with genetic programming for the container relocation problem with multiple bays and container groups. Appl. Soft Comput. 150, 111104.
Đurasević, M., Đumić, M., Gala, F. J.G., 2025. Multitask genetic programming for automated design of heuristics for the container relocation problem. Eng. Appl. Artif. Intell. 144, 110001.
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I., 2017. Attention is all you need. Adv. Neural Inf. Process. Syst. 30.
Voß, S., Schwarze, S., 2019. A note on alternative objectives for the blocks relocation problem. In: International Conference on Computational Logistics. Springer, pp. 101–121.
Wang, P., Xu, Q., Li, Y., Chen, Q., Tao, J., Qin, W., Huang, H., Zou, Y., 2025. Learning-based hybrid algorithms for container relocation problem with storage plan. Transp. Res. Part E: Logist. Transp. Rev. 197, 104048.
Wen, B., Kim, K.H., Feng, X., 2024. Sequencing ship operations considering the trajectory of the quay crane spreader. Transp. Res. Part B: Methodol. 186, 102987.
Williams, R.J., 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach. Learn. 8 (3), 229–256.
Wu, L., Jiang, Z., 2025. The parallel stack loading problem considering multiple batches of storage plates. Transp. Res. Part E: Logist. Transp. Rev. 193, 103873.
Wu, L., Jiang, Z., Wang, F., 2023. Integer programming model and branch-and-cut algorithm for the stack inbound and pre-marshalling problem. Comput. Oper. Res. 155, 106238.
Yan, Q., Song, R., Kim, K.-H., Wang, Y., Feng, X., 2024. Optimizing container relocation operations by using deep reinforcement learning. Maritime Policy Manag. 52 (8), 1288–1310.
Yue, L.-J., Fan, H.-M., Fan, H., 2023. Blocks allocation and handling equipment scheduling in automatic container terminals. Transp. Res. Part C: Emerg. Technol. 153, 104228.
Zehendner, E., Feillet, D., Jaillet, P., 2017. An algorithm with performance guarantee for the online container relocation problem. Eur. J. Oper. Res. 259 (1), 48–62.
Zhou, J., Wu, Y., Song, W., Cao, Z., Zhang, J., 2023. Towards omni-generalizable neural methods for vehicle routing problems. In: International Conference on Machine Learning. PMLR, pp. 42769–42789.