Q-Learning for Outbound Container Stacking at Container Terminals

Aaron Lim, Seokchan Lee, Jeongyoon Hong, Younghoo Noh, Sung Won Cho and Wonhee Lee

July 12, 2024

Q-learning for outbound container stacking at container terminals

Aaron Lim[1], Seokchan Lee[1], Jeongyoon Hong[1], Younghoo Noh[1], Sung Won Cho2[2][0000-0003-1470-5768],*, Wonhee Lee[3],*

1Department of Industrial Engineering, Dankook University, Cheonan, Korea;

2Department of Management Engineering, Dankook University, Cheonan, Korea;

3Maritime Digital Transformation Research Center, Korea Research Institute of Ships and Ocean Engineering, Daejeon, Korea;

*Corresponding author: sungwon.cho@dankook.ac.kr, weelon@kriso.re.kr

Abstract. The efficient stacking of outbound containers presents a significant challenge within container terminal operations. It's crucial to minimize the anticipated need for rehandling, as this directly impacts yard productivity and overall terminal efficiency. To address this challenge, we introduce a novel approach based on reinforcement learning. Our method employs Q-learning, incorporating Monte Carlo techniques to identify optimal storage locations by maximizing reward values. Furthermore, we've developed effective strategies for determining storage placements through extensive training iterations. Through numerical experimentation using real-world container terminal data, we've compared our model with existing algorithms. Numerical results highlight the robustness of our approach in navigating uncertain operational environments, its ability to support real-time decision-making, and its effectiveness in minimizing rehandling requirements.

Keywords: Container terminal; Q-learning; Yard management; Container stacking problem.

1 Introduction

Global trade growth leads to increased demand for container shipping worldwide (Woo et al., 2024). Terminals are vital in supply chain networks, with new ones built and existing ones expanded (Park et al., 2021; Cho et al., 2021). Efficient yard management connects various container movements, and container storage is key to reducing turnaround time. The study aims to determining storage locations for outbound containers.

Outbound container allocation at the yard block level is pre-scheduled; however, the specific storage location within the yard bay, referred to as the container stacking problem (CSP), is determined upon arrival (Zhang et al., 2003). CSP is complex due to uncertainties in the arrival sequence, which subsequently impact loading operations. Therefore, for efficient loading operations, the storage location of the container should be determined to minimize the expected rehandling.

Previous research on the container stacking problem (CSP) for outbound containers has categorized solutions into offline and online optimization techniques. The solution quality of existing offline optimization approaches decreases dramatically due to fluctuations in information, and a scalability issue is induced as the problem size is larger. In addition, studies that have proposed an online optimization approach do not calculate rehandling whenever the storage location of incoming containers is determined. And thus, the solution quality is not good.

In this study, we introduce a new approach using Q-learning in reinforcement learning. We designed a reinforcement learning framework by utilizing the state and novel reward function that reduces the possibility of rehandling events in the future by considering the free space in each stack. The proposed approach supports real-time decision making, such as the online optimization approach, and also reduces the optimality gap with the offline optimization approach compared to existing methodologies. In addition, we demonstrate the superiority of the proposed approach through comparison with existing online and offline algorithms according to various numerical experiments.

2 Literature review

The container stacking problem (CSP) is typically addressed through offline or online optimization strategies, chosen based on data availability. Offline optimization assumes complete data and aims to reduce uncertainty by generating nearly optimal solutions, while online optimization adapts to changing data availability in uncertain settings, striving to develop robust policies for diverse scenarios.

The offline optimization approach in CSP for outbound containers relies on weight information's completeness impacting rehandling. Kim, Park, and Ryu (2000) introduced a dynamic programming (DP) model and decision tree algorithm for scenarios with fixed container numbers per weight class. Zhang et al. (2014a) extended this model to handle larger instances, while Gharehgozli et al. (2014) devised a decision tree-based heuristic for practical pile problems using DP results. Despite these advancements, studies suggest the need for stacking strategies that account for uncertainty in container weights. Kang, Ryu, and Kim (2006) utilized simulated annealing and machine learning to mitigate rehandling due to weight uncertainties, while Zhang et al. (2014b) proposed a two-stage heuristic within a reasonable computation time.

Online optimization approaches for outbound container stacking vary depending on the optimization objective, encompassing vertical, horizontal, and category strategies. Duinkerken, Evers, and Ottjes (2001) proposed a strategy to minimize reductions in stack capacity based on load category information, while Dekker, Voogd, and Asperen (2007) introduced a category stacking strategy to reduce rehandlings with stacking location preferences. Chen and Lu (2012) devised a hybrid sequence stacking method for enhanced optimization, and He, Wang, and Su (2020) developed heuristic algorithms combining reshuffle, lowest stack, and nearest stack rules. Moreover, dynamic factors such as yard configurations and stowage instructions are considered in stacking strategies. Park et al. (2011) proposed a dynamic policy adjustment algorithm, while Guven and Eliiyi (2014) suggested stacking strategies based on container attributes. Park et al.

(2022) and Cho et al. (2022) introduced data-driven and Gaussian mixture model (GMM)-based online stacking strategies, respectively.

Few studies focus on machine learning techniques for CSP despite recent computing advancements. Our study proposes the application of reinforcement learning to practical container terminal operations, enhancing decision-making and solution quality by considering state and reward design. We have developed a novel framework that improves upon existing models by refining the reward function to reduce rehandling events and enhance training efficiency through the consideration of free space in each stack.

3 Methodology

3.1 Meanings and vocabulary

MC Q-learning for CSP problem can be expressed with w_t, weight class of current container, as follows:

$$Q(s_t, w_t, a_t)_{\text{new}} = Q(s_t, w_t, a_t) + \alpha[G_t - Q(s_t, w_t, a_t)], \tag{1}$$

$$where\ G_t = R_{t+1}(s_t, w_t, a_t) + \gamma G_{t+1}, \quad a_t \in A(s_t).$$

The Q value is related to the rehandling value that is acquired when the container is piled in the current state, $s_t = (\boldsymbol{n_t}, \boldsymbol{h_t})$. When the reinforcement learning agent pile the container, the reward $R_{t+1}(s_t, a_t, w_t)$, which is a function of rehandling, is observed. The definitions associated with the Q-function and incentive are described in Table 1.

Table 1. Notations of the Q-function and reward

Variable	Definition
w	The current container weight class
s	The current state in Q-learning
$\mathbf{n}$	The vector with a size equal to the number of containers in tiers of the current state
$\mathbf{h}$	The vector with a set of the highest weight class in each tier of the current state
$[w_{min}, w_{max}]$	The weight range of the containers
$[t_{min}, t_{max}]$	The tier range of the bay
$(\mathbf{n}, \mathbf{h})$	The state representation with **n** and **h**
$A(s)$	The possible actions in the current state
a	Stack selection of current container
$Q(s, w, a)$	Estimated Q value when the agent place a current container due to an action *a*
$R(s, w, a)$	Reward value when the agent place a current container after an action *a*
α	Learning rate

γ	Discount factor

Fig. 1. Reinforcement learning agent for container stacking problem.

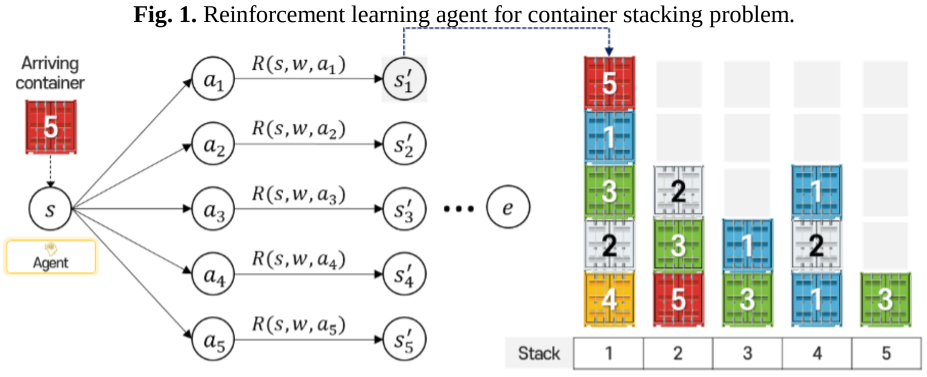

Figure 1 illustrates the CSP, showing the positioning of certain container sets and the action of the agent transferring the current container to the bay. $A(s) = \{a_1, a_2, a_3, a_4, a_5\}$ is the feasible action set for the current state. If the agent positions the container in stack 1, the reward $R(s, w, a_1)$ is obtained in the current state.

If the sequence of the container is settled within a constant ratio, the optimal Q value can be directly generated by solving the Bellman equation. Nevertheless, if the proportion and the sequence of container are ambiguous, the optimal Q value alters in each scenario, and it is challenging to discover an optimal solution in each scenario. Therefore, we enhance the Q value toward the optimal Q value according to the payoff perceived while the agent stacks the container.

3.2. State design

States in CSP are determined by bay configuration. Multiple states are needed to show bay configuration, requiring many episodes for agent training. The CSP aims to position containers with minimal rehandling, maintaining their position once placed. Identifying key components in the stacking problem is crucial for effectively defining a state: (1) Containers' locations remain fixed once placed. (2) Recognizing potential bay arrangements is essential for stacking containers efficiently. (3) Rehandling calculations are performed after containers are stacked.

For potential arrangement, the quantity of containers in each stack should be indicative of the state. Additionally, the heaviest container in each stack should be indicative of rehandling. Therefore, state signifies the quantity of containers and the heaviest container in each stack, which can be denoted as $(\mathbf{n}, \mathbf{h})$.

3.3. Reward design

Reward function should be formulated with rehandling. The proposed reward function aims to reduce current and future rehandling instances. In this study, the reward is used as a penalty mechanism to minimize rehandling. The penalty function for rehandling is articulated as follows:

$$R_1(s,w,a) = \begin{cases} -1, & w < h \\ 0, & otherwise \end{cases} \tag{2}$$

When considering only R_1 related to rehandling, a trial and error method will be uti lized for learning. The stacking problem's attributes were utilized to minimize the nee d for trial and error. Positioning lighter containers lower in CSP may reduce rehandlin g. In accordance with these attributes, the heuristic function can be defined as

$$R_2(s,w,a) = \frac{t_{\max} - t}{t_{\max} - t_{\min}} \times \frac{w_{\max} - w}{w_{\max} - w_{\min}} \tag{3}$$

where the variable t represents the current tier of the container.

The increase of R_2 is related to the weight increase and low positioning of the container. This aspect reduces rehandling in stacking. Therefore, the reward function combines both R_1 and R_2. The payoff function is articulated as follows:

$$R(s,w,a) = R_1(s,w,a) + R_2(s,w,a) \tag{4}$$

Algorithm 1 is MC Q-learning for the CSP with port terminal instances.

Algorithm 1 Q-learning algorithm for CSP

```
1:  Initialize $Q(s,w,a), \forall s$
2:  for episode = 1 to $N$ do
3:      Select a specific weight distribution from the instance
4:      Generate a container set $M$ based on the specific weight distribution
5:      for $t$ = 1 to length($M$) do
6:          $w_t \leftarrow M[t]$
7:          if $X > \epsilon // X \in (0,1)$ then
8:              $a_t \leftarrow argmax_a Q(s_t, w_t, a_t)$
9:          else
10:             $a_t \leftarrow a'$
11:         end if
12:         $R_{t+1} \leftarrow R_1(s_t, w_t, a_t) + R_2(s_t, w_t, a_t)$
13:         update $\boldsymbol{n}_{t+1}, \boldsymbol{h}_{t+1}$
14:         $S_{t+1} \leftarrow (\boldsymbol{n}_{t+1}, \boldsymbol{h}_{t+1})$
15:     end for
16:     for t = 1 to $M$ do
17:         $G_t \leftarrow R_{t+1} + \gamma G_t$
18:         $Q(s_t, w_t, a_t) \leftarrow Q(s_t, w_t, a_t) + \alpha(G_t - Q(s_t, w_t, a_t))$
19:     end for
20: end for
```

Acknowledgments. This research was supported by Korea Institute of Marine Science & Technology Promotion(KIMST) funded by the Ministry of Oceans and Fisheries, Korea(RS-2024-00407024).

4 References

1. Chen, L., & Lu, Z. (2012). The storage location assignment problem for outbound containers in a maritime terminal. International Journal of Production Economics, 135(1), 73-80.
2. Cho, S. W., Park, H. J., & Lee, C. (2021). An integrated method for berth allocation and quay crane assignment to allow for reassignment of vessels to other terminals. Maritime Economics & Logistics, 23, 123-153.
3. Cho, S. W., Park, H. J., Kim, A., & Park, J. H. (2022). GMM-based online optimization for container stacking in port container terminals. Computers & Industrial Engineering, 173, 108671.
4. Dekker, R., Voogd, P., & Van Asperen, E. (2007). Advanced methods for container stacking. Container Terminals and Cargo Systems: Design, Operations Management, and Logistics Control Issues, 131-154.
5. Duinkerken, M. B., Evers, J. J., & Ottjes, J. A. (2001, June). A simulation model for integrating quay transport and stacking policies on automated container terminals. In Proceedings of the 15th european simulation multiconference (pp. 909-916). San Diego, CA, USA.
6. Gharehgozli, A. H., Yu, Y., de Koster, R., & Udding, J. T. (2014). A decision-tree stacking heuristic minimising the expected number of reshuffles at a container terminal. International Journal of Production Research, 52(9), 2592-2611.
7. Güven, C., & Eliiyi, D. T. (2014). Trip allocation and stacking policies at a container terminal. Transportation Research Procedia, 3, 565-573.
8. He, Y., Wang, A., & Su, H. (2020). The impact of incomplete vessel arrival information on container stacking. International Journal of Production Research, 58(22), 6934-6948.
9. Kang, J., Ryu, K. R., & Kim, K. H. (2006). Deriving stacking strategies for export containers with uncertain weight information. Journal of Intelligent Manufacturing, 17, 399-410.
10. Kim, K. H., Park, Y. M., & Ryu, K. R. (2000). Deriving decision rules to locate export containers in container yards. European journal of operational research, 124(1), 89-101..
11. Park, H. J., Cho, S. W., & Lee, C. (2021). Particle swarm optimization algorithm with time buffer insertion for robust berth scheduling. Computers & Industrial Engineering, 160, 107585.
12. Park, H. J., Cho, S. W., Nanda, A., & Park, J. H. (2023). Data-driven dynamic stacking strategy for export containers in container terminals. Flexible Services and Manufacturing Journal, 35(1), 170-195.
13. Park, T., Choe, R., Kim, Y. H., & Ryu, K. R. (2011). Dynamic adjustment of container stacking policy in an automated container terminal. International Journal of Production Economics, 133(1), 385-392.
14. Woo, S. H., Park, H. J., Cho, S. W., & Kim, K. H. (2024). Proactive berth scheduling with data-driven buffer time in container terminals. International Transactions in Operational Research, 31(5), 2875-2902.
15. Zhang, C., Liu, J., Wan, Y. W., Murty, K. G., & Linn, R. J. (2003). Storage space allocation in container terminals. Transportation Research Part B: Methodological, 37(10), 883-903.
16. Zhang, C., Wu, T., Kim, K. H., & Miao, L. (2014a). Conservative allocation models for outbound containers in container terminals. European Journal of Operational Research, 238(1), 155-165.
17. Zhang, C., Wu, T., Zhong, M., Zheng, L., & Miao, L. (2014b). Location assignment for outbound containers with adjusted weight proportion. Computers & Operations Research, 52, 84-93.