



Data-driven dynamic stacking strategy for export containers in container terminals

Hyun Ji Park¹ · Sung Won Cho¹ · Abhilasha Nanda¹ · Jin Hyoung Park¹

Accepted: 5 June 2022 / Published online: 27 June 2022
© The Author(s) 2022

Abstract

This study investigates a method for improving real-time decisions regarding the storage location of export containers while the containers are arriving. To manage the decision-making process, we propose a two module-based data-driven dynamic stacking strategy that facilitates stowage planning. Module 1 generates the Gaussian mixture model (GMM) specific to each container group for container weight classification. Module 2 implements the data-driven dynamic stacking strategy as an online algorithm to determine the storage location of an arriving container in real time. Numerical experiments were conducted using real-life data to validate the effectiveness of the proposed method compared to other alternative stacking strategies. These experiments revealed that the performance of the proposed method is robust, and therefore it can improve yard operations and container terminal competitiveness.

Keywords Container terminals · Container stacking problem (CSP) · Machine learning · Gaussian mixture model (GMM)

1 Introduction

With the development of container transportation, the task of efficiently managing scarce storage space resources in container terminals has become an important role for marine transport hubs. Containers arrive at the storage area randomly and are

✉ Sung Won Cho
gungnir@kriso.re.kr

Hyun Ji Park
qgw159@kriso.re.kr

Abhilasha Nanda
ananda@kriso.re.kr

Jin Hyoung Park
jin.h.park@kriso.re.kr

¹ Maritime Safety and Environmental Research Division, Korea Research Institute of Ships and Ocean Engineering, Daejeon, Republic of Korea

stacked on the ground in the arrangement of a yard block as shown in Fig. 1. Yard cranes (YCs) must first handle the containers located at the top tier, and the containers already stacked are rehandled to access the target container buried beneath them. Thus, inefficient handling can result in excessive operational delays, which can lead to bottlenecks in container flows. Therefore, to improve the productivity of container terminals, effective methods for determining the most efficient storage location of the arriving containers must be employed (Zhang et al. 2003).

Container terminals handle various types of incoming containers, which can be classified as import or export containers depending on the vehicles that carry them. Import containers are discharged from vessels and loaded onto external trucks (ETs), whereas export containers are transported to the terminal by ETs and are loaded onto vessels. All of these containers are temporarily stored in the yard, and the goal of the storage strategy is to minimize the amount of time the vehicles that are to be loaded stay in the terminal. For this reason, the way the containers are stored depends on the characteristics of the vehicles that are to be loaded, such as the vehicles' capacities and their arrival times. For example, ETs have small carrying capacities and large uncertainties in arrival times depending on the traffic conditions. Therefore, import containers are stacked at higher tiers, as the estimated retrieval times are shorter. In contrast, vessels carry large quantities of containers, and their arrival times are expected via berth plans. Thus, export containers are stored through decisions at two different levels: planning and operational (Chen and Lu 2012; Jiang and Jin 2017; Zhou et al. 2020; He et al. 2020a, b; Feng et al. 2021). At the planning level, containers are assigned to sub-blocks based on the container group, which is defined as a group of containers having the same departing vessel, port of destination (POD), size, and type (Kim et al. 2004; Zhen 2013, 2014; Jiang and Jin 2017; He et al. 2020a). At the operational level, containers are stacked within the range of a single yard bay in consideration of the loading operation (Kim et al. 2000; Zhang et al. 2010, 2014a, b). In the loading operation, planners schedule the loading sequence with two objectives: minimizing the handling effort of quay cranes (QCs) and YCs, and ensuring the stability of the vessel (Kim et al. 2000). Therefore, the storage configuration of export containers must be in a good shape to generate an efficient loading sequence. In this study, we focus on the container

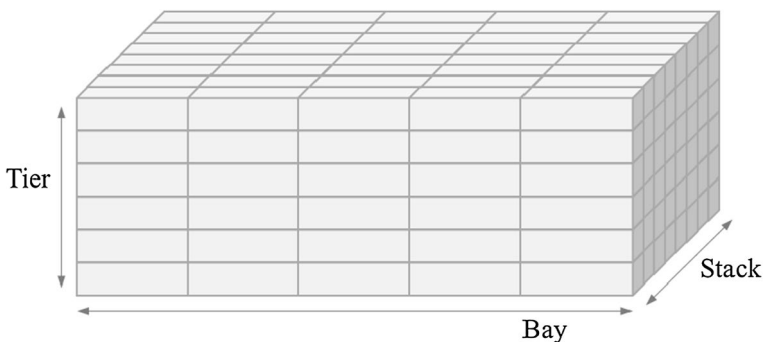


Fig. 1 Yard block configuration

In this study, we propose a stacking strategy based on an online algorithm in which decisions are made with incomplete knowledge of the future (Karp 1992). Unlike an offline algorithm that yields optimal solutions with extensive computations, an online algorithm facilitates operations in dynamic environments that do not have ample time to compute before performing tasks. Because containers cannot be held after they arrive and the order in which they arrive cannot be controlled, an online algorithm that allocates storage locations when containers arrive at the block is appropriate (Murty et al. 2005).

A number of studies have employed online algorithms for the CSP (Dekker et al. 2006; Borgman et al. 2010; Park et al. 2011; Chen and Lu 2012; Ambrosino et al. 2013; Güven and Eliiyi 2014, 2019; He et al. 2020b), but they do not take into account many practical considerations such as uncertainty of the weight, type and arrival timing of containers, uncertainty about the stowage instruction of the vessel, and the dynamic nature of YC workload according to other interconnected operations. Therefore, this study proposes a data-driven dynamic stacking strategy (DSS) based on an online algorithm.

The data-driven DSS consists of two modules. The first is the Gaussian mixture model (GMM) generation module, which clusters the container weight into several weight classes. The second is the DSS module based on the online algorithm to determine the storage location of an arriving container in real time. This module adjusts itself to dynamically respond to the environment.

The remainder of this paper is organized as follows. In Sect. 2, relevant literature on the CSP is reviewed. A detailed description of the proposed method is provided in Sect. 3. In Sect. 4, computational experiments are conducted and interpreted. Finally, the conclusions are drawn in Sect. 5.

2 Literature review

In this section, we review previous studies related to the CSP in container terminals. Many researchers have studied on the related problem, and we refer to Vis and De Koster (2003), Steenken et al. (2004), Stahlbock and Voß (2008), and Carlo et al. (2014) which conducted comprehensive reviews of numerous studies on the efficient operation of container terminals. The solutions to the CSP have been classified into two types: one for import containers and one for export containers. For import containers, De Castillo and Daganzo (1993) proposed two stacking strategies: a non-segregation strategy that includes all stacks of the same size from all vessels, and a segregation strategy in which containers from different vessels are segregated. Kim and Kim (1999) implemented a segregation strategy for import containers that estimated the expected total number of rehandles. They presented a mathematical model for the relationship between the height of the stack and the number of rehandles. They also considered the uncertainty in the arrival times of import containers for constant, periodic, and dynamic arrival rates. Kim and Kim (2002) proposed a cost model that determines the optimal storage space and number of transfer cranes for import containers. The cost model included the costs of space, transfer cranes, and ETs, and they illustrated the effectiveness of their deterministic and stochastic

models using numerical examples. Ting et al. (2010) proposed a category stacking strategy (also called a clustering stacking strategy) for import containers by analyzing historical data. They presented a pick-up booking system that categorizes the containers into several groups according to the pick-up priority predicted by historical data. Sauri and Martin (2011) extended the work of De Castillo and Daganzo (1993) and developed a segregation strategy that generates the fewest number of rehandles by mixing containers from different vessels. In addition, they considered the different probabilities of the time that elapsed before each container left the terminal as a function of the time each container arrived. Ambrosino et al. (2013) modeled import containers being loaded onto a train by comparing train loading policies (sequential, non-sequential, and partially sequential) for different stacking strategies (random, based on container weight, and based on container weight and commercial priority) in a container terminal. Maldonado et al. (2019) proposed three different stacking strategies based on the prediction (nominal, numerical, and nominal and numerical) of expected dwell times using the random forest method. They assessed their proposed method by applying it to two strategies (horizontal and vertical) in two scenarios (average and stressed).

Because the storage periods for import containers vary depending on the arrival time of ETs and trains, the CSP for import containers has been studied in a way that enables the arrival times to be predicted probabilistically. In contrast, the CSP for export containers considers vessel characteristics. Kim et al. (2000) proposed a dynamic programming (DP) model to determine the storage locations of export containers based on their weights. They assumed that heavier containers should be loaded onto the lower tiers of a vessel to guarantee its stability. Therefore, heavier containers are stacked at the higher tiers of the yard block to reduce the expected number of rehandles. Furthermore, they developed a decision tree to support real-time decisions. Duinkerken et al. (2001) evaluated the performance of the remaining stack capacity (RSC) strategy, which considers the stack height and container category using various stacking strategies (random, levelling, and closest position). Dekker et al. (2006) proposed a category stacking strategy that allows online optimization to facilitate loading operations. They used a simulation method to compare random stacking with category stacking based on the number of rehandles. Kang et al. (2006) presented a stacking strategy for export containers with uncertain weight information using a simulated annealing approach to minimize the number of rehandles. Furthermore, they proposed an advanced stacking strategy that overcomes the uncertainty in container weight through machine learning techniques. Park et al. (2011) proposed stacking strategies to dynamically determine the stacking location as the operational environment changes. The proposed strategies, which were based on an online algorithm, were generated by evaluating the weights of the decision criteria during the evaluation period. Simulations were conducted for a variety of stacking strategies, which were demonstrated to be effective in reducing QC delays. Chen and Lu (2012) proposed a hybrid sequence stacking algorithm (HSSA) based on an online algorithm to make decisions in real time. They observed that the HSSA outperformed the random and vertical stacking strategies in terms of the number of rehandles. Zhang et al. (2010) analyzed the error of a key model transformation in Kim et al. (2000) and presented the correct form. Zhang et al. (2014a), which was

an extension of the studies by Kim et al. (2000) and Zhang et al. (2010), proposed two conservative models by reinterpreting the punishment coefficient for stacking light containers on top of stacks loaded with heavy containers. The proposed models outperformed the previous optimized models in terms of static and dynamic indicators. Zhang et al. (2014b) considered adjusting the proportion of unarrived containers in each weight class to a non-constant proportion in the constant proportion DP model proposed by Kim et al. (2000) and Zhang et al. (2010). In numerical experiments, they demonstrated that the proposed models with the adjusted weight class proportions for the remaining containers improved the stacking quality. Hu et al. (2014) proposed a branch-and-bound method based on the least-cost priority queue (LCBB) to obtain an optimal solution in which the number of rehandles is minimized, using the HSSA proposed by Chen and Lu (2012) to calculate the upper bound for the LCBB. Güven and Eliiyi (2014) studied two stacking strategies (random stacking and category stacking) for export containers. They considered container weight as another category attribute, and grouped containers with a weight of less than three tons into the same category. Güven and Eliiyi (2019) extended Güven and Eliiyi (2014) and expanded the stacking strategies to include all types of containers (export, transit, import, and empty containers). They compared three stacking strategies (random stacking, attribute-based stacking, and weight-relaxed stacking) through simulations. He et al. (2020b) studied stacking strategies that consider the uncertainty in the arrival sequence of vessels, assuming that the weight information and arrival order of the containers are known. Based on the three stacking rules (least reshuffle rule, lowest stack rule, and nearest stack rule), five heuristic algorithms were proposed according to a set of rules.

The contributions of our study in the context of the aforementioned studies are summarized as follows. First, this is the first study that applies predictive analytics for container weight classification to prescribe optimal decisions for the CSP. Most previous studies have simplified the problem by assuming three classes (light, medium, and heavy), and certain studies (Kang et al. 2006; Zhang et al. 2014b) have considered the weight uncertainty for each class. However, in practice, these assumptions are not practical because the weight is classified according to the size of the vessel and the range of weights of the containers to be loaded. Therefore, this study analyzes the historical data for container weight, and estimates the classification model for container weight class as a GMM using a machine learning technique. Second, we propose a dynamic stacking strategy that considers multiple bays. To the best of our knowledge, most studies have focused on the CSP for a single bay, and the stacking strategy was applied homogeneously to each bay. However, in a real-world environment, the proportion of containers in each weight class assigned to each bay is not constant. Hence, the remaining containers must respond dynamically to the containers that are already stacked. Therefore, this study presents a stacking strategy that responds to the configurations of multiple bays. Third, we develop a category stacking strategy to present practical alternatives that reflect real-world considerations. In most studies on the CSP for export containers (the exceptions being Dekker et al. 2006; Güven and Eliiyi 2019), the problem is defined as minimizing the expected number of rehandles based on the weight class. However, heavier containers are normally loaded onto lower tiers, depending on the configuration

of the stowage and the precise weight of the containers. The category stacking that clusters containers with similar weights into the same stack can facilitate stowage planning by providing containers of various weights on the top tier in the yard.

3 Problem description

This study aims to optimize real-time decisions regarding the precise storage location of export containers for a given storage area while the containers are arriving. The storage area consists of multiple stacks in a bay, as shown in Fig. 3. In this study, the decision on the storage location refers to the selection of one stack in the storage area. In practice, the storage areas of export containers are not shared with those of import containers (Kim and Kim 1999; He et al. 2020b; Hu et al. 2021). Furthermore, each storage area designates areas for containers belonging to the same container group (Kim et al. 2004; Zhen 2013, 2014; Jiang and Jin 2017; He et al. 2020a). For these reasons, different stacking decisions are made depending on the container group. Therefore, we focus on stacking decisions that apply to the containers in a single container group.

The goal is to assign an arriving container to a stack in a way that conforms to the category stacking strategy. Category stacking for export containers aims to cluster containers with similar weights in the same stack. However, it is difficult to achieve this outcome due to the randomness of the arriving containers. The following two aspects need to be considered to improve the stacking quality according to the category stacking strategy: (1) *how* to define the similarity in container weights and (2) *how* to define the specific location assignment rules for each storage area. For aspect (1), researchers have manually classified container weights into several weight classes with approximate weight ranges. For aspect (2), the location assignment rules have been executed in the same way for all storage areas. However, predictive analytics for container weight classification and container assignment rules specific to dynamic storage areas can enhance the stacking strategy. In this respect, we propose a data-driven DSS that outperforms existing stacking strategies.

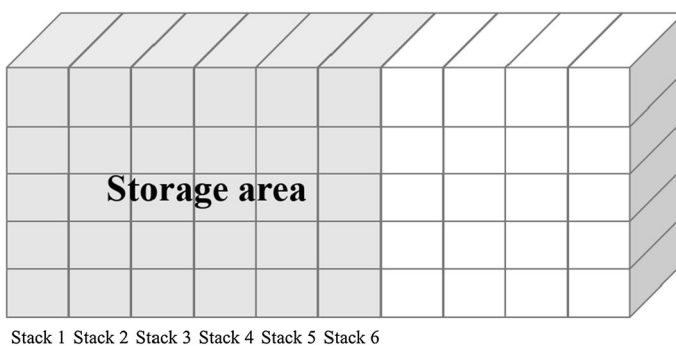


Fig. 3 Yard bay configuration

The proposed container stacking system is divided into two modules, as shown in Fig. 4. Module 1 is executed in advance, and module 2 is triggered by a container arriving at the port. In module 1, GMM-based predictive models, which cluster the container weights into several weight classes for each container group, are generated. Next, once a container has arrived, the pre-selected storage area is given, and the weight class for the container is predicted by the GMM generated in module 1. In module 2, the precise storage location (i.e., stack) for the container is selected using the proposed dynamic stacking strategy for the given storage area. The proposed stacking strategy is intended to dynamically adjust itself depending on the stack configuration in the storage area. A detailed description of the modules is provided in the following subsections.

3.1 Module 1: generation of GMM

In module 1, we aggregate the historical data for the container weights over the entire voyage for each container group, and then generate the GMM specific to each container group for the container weight classification. The GMM is a model that represents a population as a linear superposition of subpopulations, and it assumes that each subpopulation follows a Gaussian distribution. Because the class label (i.e., subpopulation) of the data point is unknown, the GMM is an unsupervised learning method (Figueiredo and Jain 2002). In addition, the GMM is a soft clustering method that uses probabilistic inference to explain how much a given data point is associated with a certain cluster (i.e., subpopulation). Due to these characteristics, the GMM has been widely employed in unsupervised classification applications in which data tend to follow multimodal and complex distributions. In this study, the container weight class was defined for each cluster of the GMM. Furthermore,

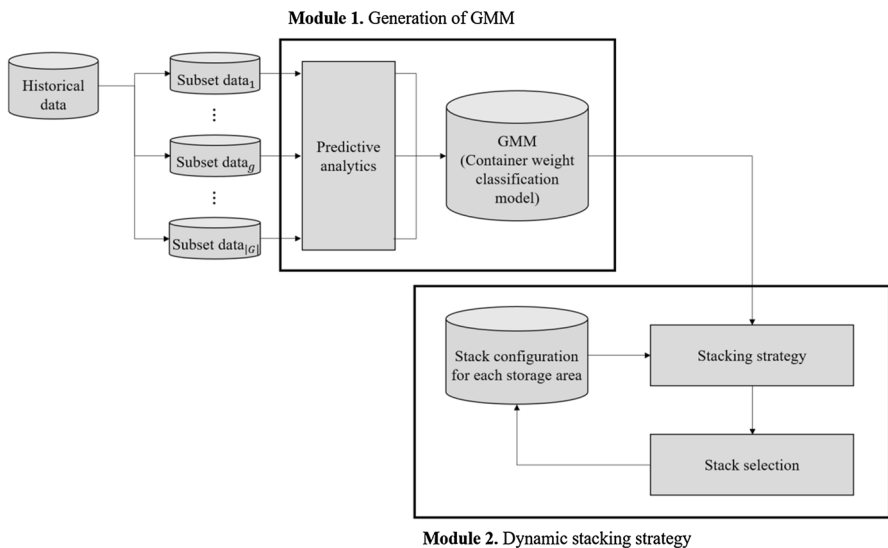


Fig. 4 Overview of the proposed container stacking system

the probable weight class was predicted for new container input. For the detailed description, the following notations are introduced:

Notation	
G	Set of container groups, indexed by g
K_g	Set of clusters (i.e., weight class) for container group g , indexed by k
$x(i)$	Container weight of container i
$g(i)$	Container group of container i
$k(i)$	Probable weight class of container i
μ_k^g	Mean of cluster k for container group g
Σ_k^g	Covariance of cluster k for container group g
π_k^g	Mixture weight of cluster k for container group g ; $0 \leq \pi_k^g \leq 1$ and $\sum_{k \in K_g} \pi_k^g = 1$
z_k^g	Latent indicator variable; defined as 1 if the observation data belongs to cluster k and 0 otherwise; $\sum_{k \in K_g} z_k^g = 1, p(z_k^g = 1) = \pi_k^g$
$p(x g)$	Marginal probability distribution of container weight x for container group g
$\gamma(z_k^g g)$	Posterior probability (i.e., ‘‘responsibility’’) that container weight x is observed from cluster k for container group g ; $\gamma(z_k^g g) = p(z_k^g = 1 x, g)$

The GMM is parameterized by the mean μ_k^g , covariance Σ_k^g , and mixture weight π_k^g . The assignment of the unknown class label z_k^g is considered a latent variable instead of a parameter. It enables the joint distribution $p(x, z_k^g)$ to marginalize the variable z_k^g out to define the cost function independently of z_k^g . The resulting standard form of the GMM is written as

$$p(x|g) = \sum_{k \in K_g} p(z_k^g) p(x|z_k^g) = \sum_{k \in K_g} \pi_k^g N(x|\mu_k^g, \Sigma_k^g). \tag{1}$$

To estimate the three parameters of the GMM, the objective function is to maximize the marginal likelihood of the data ($p(x|g)$). Because there is no analytical solution, a numerical method was employed for the maximum likelihood estimation. The most widely used method is expectation maximization (EM). The EM algorithm estimates the model parameters through iterations of the expectation step (E step) and maximization step (M step). Given the initialized model parameters and the log-likelihood estimate, the E step uses the model parameters to evaluate the responsibility $\gamma(z_k|g)$ via

$$\gamma(z_k^g|g) = C_g \times \pi_k^g N(x|\mu_k^g, \Sigma_k^g), \tag{2}$$

where $C_g = 1 / \sum_{k \in K_g} \pi_k^g N(x|\mu_k^g, \Sigma_k^g)$. Then the M step re-estimates the model parameters using this responsibility value. These iterations lead to the convergence of the model parameters, and the resulting trained GMM provides the probable weight class of the new container input via

$$k(i) = \operatorname{argmax}_{k \in K_g} \gamma(z_k^g|g) \tag{3}$$

3.2 Module 2: dynamic stacking strategy

In module 2, we executed the DSS as an online algorithm to determine the storage location of an arriving container in real time. The overall framework of the proposed DSS is shown in Fig. 5. The algorithm inputs include the GMM-based prediction results for the weight class as well as the yard stack configuration of the pre-selected storage area for the arriving container. The storage location for the container is determined after the weight class-to-stack assignment. Both methods accommodate the GMM-based prediction results for the weight class obtained from module 1. Furthermore, the weight class-to-stack assignment method adapts the adjustment mechanism according to the dynamic change in the stack configuration, which leads to the generation of the stacking strategy that is specific to the storage area.

For the weight class-to-stack assignment, conventional approaches of category stacking apply the assignment method homogeneously for all storage areas, and they remain consistent during the period in which containers are arriving. However, in real-world environments, the stack configuration cannot be the same for all storage areas, and thus the stacking strategy must be adjusted according to the different stack configurations. Taking into account the limitations of conventional methods, we propose an improved approach to accommodate the dynamic change in the stack configuration of the storage areas. For the detailed description, the following notations are introduced:

Notation	
A_g	Set of storage areas for container group g , indexed by a
S_a	Set of stacks in storage area a , indexed by s
K_{as}	Set of weight classes designated for stack s in storage area a , indexed by k
R_k^g	Range of cumulative mixture weight values for weight class k
R_{as}	Range of cumulative mixture weight values for stack s in storage area a
$a(i)$	Pre-selected storage area for container i
$s(i a)$	Stack to be selected for container i given storage area a
t_{as}	Remaining slot capacity (in number of containers) of stack s in storage area a
f_{as}^k	Contribution of weight class k to R_{as} for $k \in K_{as}^g$
X_{as}^g	List of container weights for stack s in storage area a of container group g

First, we describe the procedure for the weight class-to-stack assignment. This assignment satisfies a many-to-many relationship. Regarding the GMM-based prediction results for the weight class, we focus on the mixture weight π_k^g , which

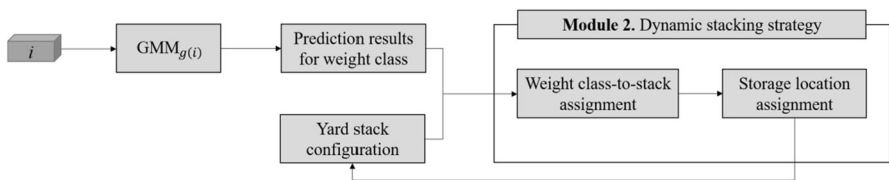


Fig. 5 Framework of the proposed dynamic stacking strategy

represents the estimated size of the weight class k . Given that the indices of the GMM are sorted in ascending order according to weight, the cumulative range of π_k^g for each weight class can be represented by

$$R_k^g = \left\{ r : \sum_{\substack{k' \in K_g \\ k' < k}} \pi_{k'}^g < r \leq \sum_{\substack{k' \in K_g \\ k' \leq k}} \pi_{k'}^g, r \in \mathbb{R} \right\}. \tag{4}$$

Likewise, given that the indices of the stacks are sorted, the cumulative range of π_k^g for each stack can be represented by

$$R_{as} = \left\{ r : \sum_{\substack{s' \in S_a \\ s' < s}} C_a \times t_{as'} < r \leq \sum_{\substack{s' \in S_a \\ s' \leq s}} C_a \times t_{as'}, r \in \mathbb{R} \right\}, \tag{5}$$

where $C_a = 1 / \sum_{s \in S_a} t_{as}$. Equation (5) calculates the coverage range for each stack based on the number of remaining slots, t_{as} . For example, if the storage area consists of four empty stacks, the cumulative ranges R_{as} are set to (0, 0.25], (0.25, 0.5], (0.5, 0.75], and (0.75, 1]. Then,

$$K_{as}^g = \{ k : R_k^g \cap R_{as} \neq \emptyset, k \in K_g \} \tag{6}$$

defines the weight class-to-stack assignment K_{as}^g such that the membership of the weight classes in a stack is determined by whether the elements of R_k^g lie in the specified R_{as} . The variable K_{as}^g is continuously adjusted because dynamic container placements change t_{as} , which impact R_{as} .

Figures 6 and 7 provide an example of the adjustment mechanism in the weight class-to-stack assignment procedure. It is supposed that two storage areas are assigned containers that belong to the same container group. The number of weight classes is set to four for the container group, and the number of stacks is set to four for both storage areas. The mixture weights π_k^g are set to 0.25 for all weight classes. Figure 6 shows the initialized K_{as}^g in which there are no differences between the storage areas.

In contrast, Fig. 7 shows the adjusted K_{as}^g after a total of 22 containers are stacked. The adjustment varies depending on the storage area. We now elaborate the steps to derive K_{as}^g for storage area ‘‘A’’ using Eqs. (4)–(6). First, R_k^g is always (0, 0.25], (0.25, 0.5], (0.5, 0.75], and (0.75, 1] for the weight classes. Second, R_{as} is initialized as (0, 0.25], (0.25, 0.5], (0.5, 0.75], and (0.75, 1] for the empty storage area. After three, three, one, and three containers are stacked in each stack, R_{as} is updated as (0, 0.2], (0.2, 0.4], (0.4, 0.8], and (0.8, 1] for the stacks. Therefore, it is necessary to check whether there is an overlap between R_k^g and R_{as} . For weight class 1, the infimum and supremum of R_k^g lie in R_{as} for stacks 1 and 2, respectively. Accordingly, K_{as}^g of

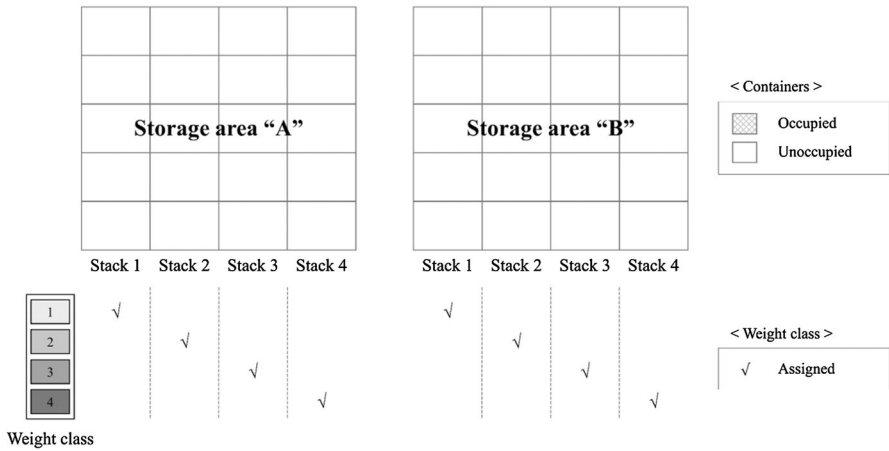


Fig. 6 Illustration of an initialized weight class-to-stack assignment ($\pi_k^g = 0.25$ for $k = 1, 2, 3, 4$)

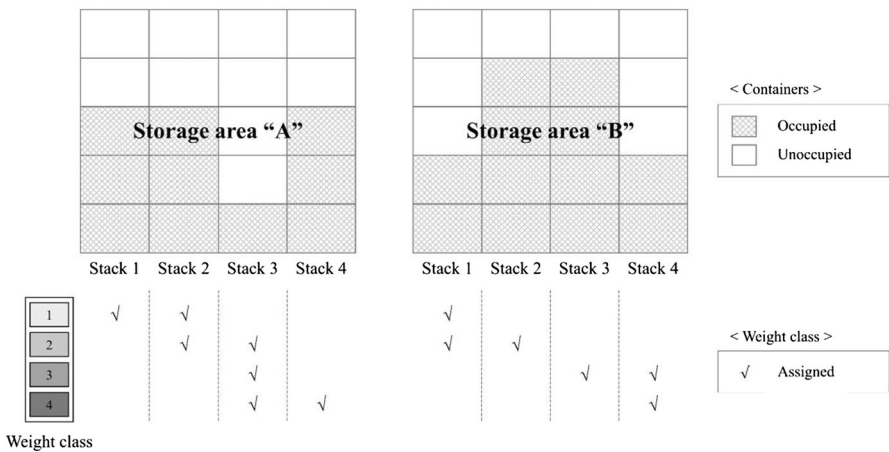


Fig. 7 Illustration of an adjusted weight class-to-stack assignment ($\pi_k^g = 0.25$ for $k = 1, 2, 3, 4$)

weight class 1 includes stacks 1 and 2. In this way, the K_{as}^g of the weight classes are dynamically updated when the stack configuration is changed, establishing the stacking strategy specific to the storage area.

Next, we describe the procedure for the storage location assignment for a container, given the weight class-to-stack assignment. Regarding the GMM-based prediction results for the weight class, we focus on the responsibility value $\gamma(z_k^g | g)$, which represents the probability that a container belongs to a certain weight class. If a single weight class is assigned to each stack (e.g., the weight class-to-stack assignment is 1–1, 2–2, 3–3, 4–4), as shown in Fig. 6, an effective strategy is to select a stack with the largest $\gamma(z_k^g | g)$ for the corresponding weight

class k . However, because multiple weight classes can be assigned to each stack, as shown in Fig. 7, a more sophisticated strategy that considers the contributions of multiple weight classes should be employed. In this context,

$$f_{as}^k = \min\{\sup R_k^g, \sup R_{as}\} - \max\{\inf R_k^g, \inf R_{as}\} \tag{7}$$

defines the contribution of weight class k to R_{as} . Then,

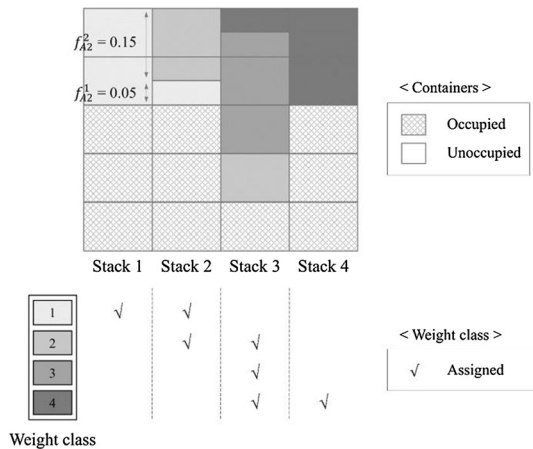
$$s(i|a) = \operatorname{argmax}_{s \in S_a} \sum_{k \in K_{as}^{g(i)}} C_{as} \times f_{as}^k \times \gamma(z_k^g | g(i)), \tag{8}$$

where $C_{as} = 1 / \sum_{k \in K_{as}^{g(i)}} f_{as}^k$, indicates the proposed strategy that ensures the selection of a stack with the largest weighted average of $\gamma(z_k^g | g)$ according to the normalized f_{as}^k .

Figure 8 provides an example of the storage location assignment procedure, extending the example for storage area ‘‘A’’ in Fig. 7. In this example, f_{A1}^k of stack 1 is derived as $f_{A1}^1 = \min\{0.25, 0.2\} - \max\{0, 0\} = 0.2 - 0 = 0.2$ for weight class 1. Similarly, the resulting f_{A2}^k of stack 2 for weight classes 1 and 2 are $f_{A2}^1 = 0.25 - 0.2 = 0.05$ and $f_{A2}^2 = 0.4 - 0.25 = 0.15$, respectively. Then the weighted averages of $\gamma(z_k^g | g)$ are derived as $f_{A1}^1 / f_{A1}^1 \times \gamma(z_1^g | g)$ for stack 1 and $f_{A2}^1 / (f_{A2}^1 + f_{A2}^2) \times \gamma(z_1^g | g) + f_{A2}^2 / (f_{A2}^1 + f_{A2}^2) \times \gamma(z_2^g | g)$ for stack 2. Thus, for a new container input, the stack that yields the largest weighted average of $\gamma(z_k^g | g)$ is selected among all the stacks in the given storage area.

The overall procedure of the DSS is described by Algorithm 1.

Fig. 8 Illustration of the storage location assignment procedure ($\pi_k^g = 0.25$ for $k = 1, 2, 3, 4$)



Algorithm 1 Data-driven DSS**Input:** Container i

- 1: Let $g = g(i)$, $x = x(i)$, $a = a(i)$
- 2: Select an element s from S_a with the largest value according to Equation (8)
- 3: **if** more than one element lead to the largest value **then**
- 4: Randomly choose an element s from the set with the largest value
- 5: **end if**
- 6: Update the yard stack configuration so that $t_{as} = t_{as} - 1$ and x is appended to X_{as}^g
- 7: Update the weight class-to-stack assignment according to Equations (4)–(6)

4 Numerical experiments

We conducted numerical experiments to validate the data-driven DSS (also called a GMM-DSS), which features the GMM-based weight clustering and dynamic adjustment mechanism for storage location assignment. We illustrate the improvements achieved by our proposed method through the experiments comparing with the stacking strategies that do not employ GMMs and/or dynamic adjustment mechanisms. First, the input data in the numerical experiment are described. Second, as a result of module 1, the generated GMM to define the weight class for each test instance is reported. Third, the impacts of the unit number of stacks on the algorithm performance are analyzed. Finally, an analysis comparing the stacking performances of the container stacking strategies is presented. This analysis shows how the wealth of data can be applied in the CSP to provide valuable decision support. All the algorithms were coded in Python and executed on a PC with an i5-6600H 3.3 GHz Intel Core processor and 8.0 GB of RAM.

4.1 Input data

The input data used in the numerical experiments were collected for 10 months in 2018 from a typical container terminal in Busan, Republic of Korea. The original data included detailed container information, such as the time of arrival and departure, departing vessel, POD, size, type, and weight of each container. Because the stacking decisions are made according to the container group, the original data were classified into container groups with the same attributes (e.g., departing vessel, POD, size, and type). In addition, only 20-foot containers were used in this study. Subsequently, a dataset of 12 container groups was used for the analysis. Table 1 reports the details of the selected 12 instances.

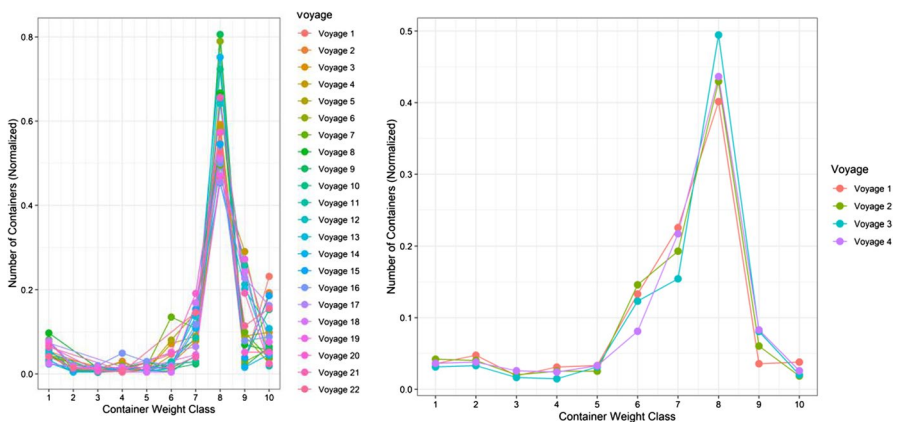
Next, we present the input data distributions over the entire voyage to check the justification for converting the container weights. This analysis comes from our assumption that the probability distribution of the container weights can be

Table 1 Test instances for numerical experiments

Instance no	Average number of containers for each voyage	Number of voyages	Average container weights (kg)
1	45	12	19,243
2	85	5	20,388
3	106	11	21,204
4	147	22	22,247
5	198	6	20,729
6	322	5	19,735
7	390	4	20,229
8	419	4	21,648
9	493	4	19,801
10	526	5	19,828
11	526	7	20,032
12	559	6	19,879

made available from the historical data due to the repeated tendencies over the voyage. It is rarely studied in literature to handle the information on container weights. Only a few studies, such as Kang et al. (2006) and Zhang et al. (2014b), utilized the true probability distribution or portion of the weight groups assuming such information is given or estimated by analysis of historical data in advance.

Figure 9 shows the distribution of the container weights over the entire voyage for instances 4 and 7. The x -axis indicates the range of container weight uniformly divided by 10 classes, and the y -axis indicates the number of containers for each voyage which is normalized to a value between 0 and 1. In this case, the distribution of container weights was similar over the entire voyage of the same instance. This is reasonable because the composition of the export cargo tends to

**Fig. 9** Distribution of container weights over the entire voyage (Instances 4 and 7)

be similar for each voyage. Due to the extensive data, GMM-based predictions for weight classes can be a powerful tool for effective category stacking in yards.

For container groups of a general type, the terminal had 19 blocks consisting of 50 bays each, as well as 10 stacks and six tiers. The storage area for the same container group was reserved in a unit of stacks. Because the unit number of stacks is usually set to 10 stacks (i.e., one bay) or five stacks, we conducted the experiments for both cases where the experimental results are presented in Sect. 4.5. The total number of storage areas was set to the minimum value required for the corresponding container group. Considering the buffer storage space for rehandling operations that exists in practice, the maximum allowable tier for containers was limited to the fifth tier.

4.2 Design of experiments

For each instance in this study, we divided a training dataset for generating the GMM in module 1 and a test dataset for simulation. The test dataset is constructed by randomly selecting a voyage and collecting a corresponding list of loaded containers from the historical data of the test ship-lanes (test instances). The training dataset is the remainder of historical data except for the test dataset. The results of GMM-based clustering with the training dataset are provided in Sect. 4.3, followed by sensitivity and comparative analysis results.

In simulation experiments, two kinds of randomness are considered: the container arrival sequence and assignment to a storage area. The detailed list of the to-be-stacked containers is unknown in advance and even unpredictable in practice. Further, a pre-selected storage area is given because the designation of the storage area for an arriving container is dynamically assigned depending on the workload of the YC. Therefore, the container arrival sequence was made by randomly selecting a container list in the test dataset, and a storage area was randomly assigned to a container in the simulations.

The stacking performance was evaluated based on category stacking, which aims to cluster containers with similar weights into the same stack. Therefore, we introduced an evaluation function

$$E(g) = \frac{\sum_{a \in A_g} \sum_{s \in S_a} SD(X_{as}^g)}{\sum_{a \in A_g} |S_a|}, \quad (9)$$

where the standard deviation of the container weights in the stack was measured. The evaluation function $E(g)$ indirectly minimizes the makespan during future loading operations for a given category stacking strategy.

4.3 GMM-based clustering for container weights

Using 12 test instances, we employed the GMM method to cluster the container weights into K clusters. For the training data, a set of preliminary experiments was conducted to investigate the appropriate value of K for each test instance, varying K

from 1 to 5 in steps of 1. The value of K was assigned the smallest value based on the Bayesian information criterion (BIC). Figure 10 shows the resulting GMM for the two test instances. The histogram in Fig. 10 indicates that the container weights exhibited a multi-peak distribution, and in both instances, it can be seen that five clusters are most appropriate. Figure 10 shows that GMM obtained via statistical modeling reasonably represents a distribution that is difficult to express as a single normal distribution. The results of module 1 for all the instances are presented in Appendix A.

4.4 Sensitivity analysis of the unit number of stacks

This section validates the performance of GMM-DSS according to the unit number of stacks. The unit number of stacks within a storage area is usually set to 10 stacks for export containers, but it varies from terminal to terminal. For the sensitivity analysis of the unit number of stacks, $|S_a|$ was set to 5, 10, 15, and 20. The GMM was generated using the training dataset, and then GMM-DSS was implemented using the test dataset. Table 2 reports the change in the average of $E(g)$ according to the variation in the unit number of stacks $|S_a|$. In the case of instance 1, it was excluded from experiments on more than 15 stacks as the number of tested containers was less than 50. In most instances, it is observed that $E(g)$ decreases as the unit number of stacks increases. These results imply that as the number of unit stacks increases, the weight distribution of containers may have been relatively stable and homogeneous over storage areas since the containers are less scattered into multiple storage areas. Meanwhile, Instances 3, 5, and 6 yield the minimum $E(g)$ when the unit number of stacks is equivalent to 10 and 15, although the differences of $E(g)$ in instances are not significant. The GMM-DSS works reliably to comply with the category stacking strategy, including the case of 10 stacks widely used in container terminals. Thus it is believed to be worth being introduced into the terminal operating system.

4.5 Comparative analysis of container stacking strategies

For comparison purposes, we implemented four alternative stacking strategies: a GMM-based static stacking strategy (GMM-SS), current practice, hybrid sequence

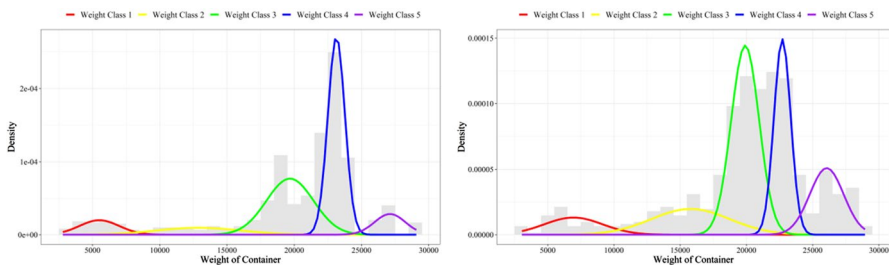


Fig. 10 Histogram and GMM-based density curve of the container weights (Instances 7 and 11)

Table 2 Impact of the unit number of stacks on average $E(g)$

Instance no	Stack \times Tier (Container), Average $E(g)$			
	5 \times 5 (25)	10 \times 5 (50)	15 \times 5(75)	20 \times 5(100)
1	647.3	426.9	–	–
2	676.7	650.8	644.7	643.7
3	806.4	662.9	584.8	601.4
4	552.1	420.2	344.0	319.3
5	461.7	283.6	320.7	301.1
6	466.7	296.9	315.7	304.5
7	666.5	571.5	530.6	515.1
8	551.0	422.4	378.8	366.9
9	478.9	323.4	285.4	265.4
10	502.5	356.1	316.4	299.8
11	486.4	354.7	303.3	279.2
12	449.5	326.3	282.1	261.9

stacking strategy (HSSS), and a random stacking strategy (RSS). For the GMM-SS, the weight classes and storage locations of the containers were determined by our GMM-based dynamic stacking strategy (GMM-DSS), while the dynamic adjustment mechanism in the weight class-to-stack assignment was ignored. For the current practice, the weight classes were defined by dividing the container weights into weight classes by ‘number of stacks in a bay,’ where each weight class has equal size of weight range according to the historical data. In addition, the storage locations of the containers were determined in such a way that the stack and weight classes were allocated on a one-to-one basis. For the HSSS proposed by Chen and Lu (2012), the weight classes were defined by dividing the container weights weight classes by ‘(number of stacks in a bay)+(number of tiers in a stack)–3.’ Then, HSSS induces heavier containers to be stacked in the left upper locations and lighter containers to be stacked in the right lower locations. In the RSS, the containers were stacked in a way that filled the stack sequentially according to the arrival order without considering the container weights. All five strategies, including our proposed method, were executed in seconds; thus, they are suitable for an online algorithm.

We conducted a simulation to compare the proposed method to the four alternative strategies for two different cases where the unit of storage area is five stacks or 10 stacks. The resulting performance for the stacking strategies is presented in Tables 3 and 4 for the cases in terms of the average $E(g)$ for 100 repetitions. For both cases, the proposed GMM-DSS method obtained the minimum $E(g)$ on average for all test instances. Specifically, for the case of five stacks in Table 3, the GMM-DSS performed better than the GMM-SS by 6.0%. This is the result of the dynamic adjustment mechanism in the weight class-to-stack assignment. In addition, the performance of the GMM-DSS was better than the current practice, HSSS, and RSS by 33.1, 41.6 and 44.1%, respectively. This result implies that the proposed method was more effective as the data-driven approach as well as the dynamic adjustment mechanism was incorporated into the stacking strategy. For

Table 3 Experimental results for the test instances (the unit of storage area = five stacks)

Instance no	Stacking strategy, Average $E(g)$				
	GMM-DSS	GMM-SS	Current Practice	HSSS	RSS
1	647.3	880.5	1043.5	1503.1	1726.4
2	676.7	680.1	862.9	884.7	886.9
3	806.4	870.2	1091.4	1291.0	1348.8
4	552.1	573.8	919.2	1002.5	1030.6
5	461.7	467.6	522.4	533.4	540.2
6	466.7	472.7	519.3	536.7	537.8
7	666.5	676.0	859.1	946.3	981.2
8	551.0	555.4	1089.3	1206.1	1256.4
9	478.9	504.2	813.9	940.1	975.0
10	502.5	528.2	867.5	984.9	1023.1
11	486.4	506.8	790.3	904.8	935.4
12	449.5	458.5	707.2	813.0	832.6

Table 4 Experimental results for the test instances (the unit of storage area = 10 stacks)

Instance no	Stacking strategy, Average $E(g)$				
	GMM-DSS	GMM-SS	Current practice	HSSS	RSS
1	426.9	447.1	584.0	1040.8	2059.9
2	650.8	669.6	855.0	876.9	948.0
3	662.9	721.7	1060.9	1240.2	1346.3
4	420.2	566.1	765.1	977.1	1029.1
5	283.6	310.1	493.0	513.8	588.8
6	296.9	315.2	493.3	519.8	586.2
7	571.5	598.5	775.7	910.5	984.6
8	422.4	452.9	973.4	1146.1	1293.5
9	323.4	346.4	733.0	898.0	975.0
10	356.1	386.6	792.1	948.8	1021.8
11	354.7	381.0	784.3	894.9	939.8
12	326.3	347.5	729.8	829.7	852.0

the case of 10 stacks in Table 4, the results show that proposed method is superior to the other methods as in the case of five stacks. The performance of the GMM-DSS was better than the GMM-SS, current practice, HSSS, and RSS by 8.1, 43.6, 52.8, and 59.6%, respectively. Moreover, Fig. 11 demonstrates the robust performance of our proposed method for the experimental repetitions. The current practice performed better on average than the HSSS and RSS, and the proposed GMM-DSS and GMM-SS are dominant over the current practice. Therefore, we

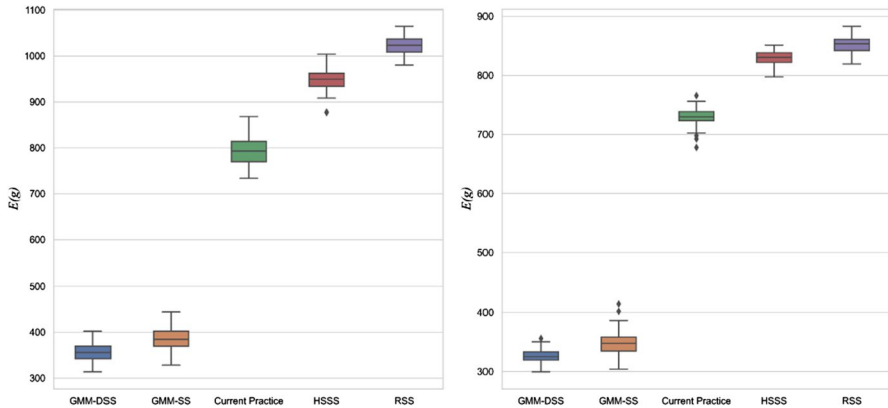


Fig. 11 Box plots of experimental results for $E(g)$ after 100 repetitions (the unit of storage area = 10 stacks, Instances 10 and 12)

can conclude that the proposed GMM-DSS method is worth introducing in practice because it has the most reliable and best performance.

4.6 Comparative analysis according to variability

This section validates the algorithm performances according to variability in container weights. The proposed GMM-DSS utilizes the information on container weights from historical data and creates a GMM to approximate the distribution of container weights; thus it should be verified whether the GMM-DSS and GMM-SS can yield robust performance under a certain degree of variability. To this end, container weights $x(i)'$ for the test were randomly generated as follows: $x(i)' = \min(\max(x(i) + 500 \times \sigma \times N(0, 1), LB(i)), UB(i))$ (kg), where $x(i)$ is a randomly selected sample data from the training dataset, σ is related to the variability, $N(0, 1)$ is the random generator from a standard normal distribution, and $LB(i)$ and $UB(i)$ are the lower bound and upper bound on weights of container i introduced to avoid extreme values. σ was set to 0, 1, 2, 3, and 4. Figure 12 shows the performance of five stacking algorithms for the experimental repetitions under variability on container weights. For all the cases of σ , the proposed GMM-DSS and GMM-SS dominantly outperform the other algorithms in terms of $E(g)$. GMM-DSS slightly outperforms GMM-SS. Furthermore, although performance degrades as variability increases, there is no significant difference in the change compared to other algorithms. Thus, it can be concluded that the proposed algorithms are robust to variabilities by taking advantage of predictive analytics.

5 Conclusion

The goal of this paper is to introduce the wealth of data generated by the container terminal into the methodology of determining the storage location of the arriving containers. We presented a two module-based DSS to manage the CSP

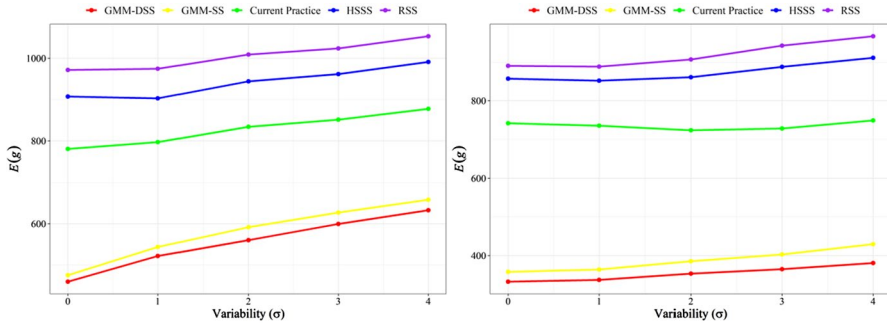


Fig. 12 Results according to variability on $E(g)$ (the unit of storage area = 10 stacks, Instances 5 and 9)

in container terminals. In our study, the CSP was solved by a category stacking strategy to facilitate stowage planning. In contrast to previous studies, the proposed method applies predictive analytics for container weight classification using historical data, and container assignment to a specific storage area is executed dynamically to outperform the existing stacking strategies. To the best of our knowledge, these two main features have not been introduced in the literature before. Specifically, in module 1, we generated a container group-specific GMM for container weight classification. In module 2, we implemented the DSS as an online algorithm to determine the storage location of an arriving container in real time. We conducted numerical experiments to validate the effectiveness of the proposed method using real-life data from a typical container terminal in Busan, Republic of Korea. For the generality of terminal environments, two different cases for the reservation unit in a storage area are considered in the experiments. The experimental results showed that the proposed method outperforms four other alternative stacking strategies (i.e., the GMM-SS, current practice, HSSS, and RSS). Therefore, our proposed method performs robustly and can further improve yard operations and terminal competitiveness.

The experimental results are encouraging in that it implies that the improvements in the stacking performance are related to the category stacking. The proposed method is expected to achieve greater improvements in practice when data-driven inferences can be derived from more historical data. In future research, our method could be extended to transshipment containers by considering the detailed characteristics of those operations. In addition, novel stacking strategies, such as the flexible space-sharing strategy and related operations such as stowage planning, could be integrated into our method to generate greater effectiveness in terminal operations.

Appendix A: The results of module 1 for the test instances

See Table 5.

Table 5 Parameters of GMM

Instance no	k	$\mu_k^g(\text{kg})$	Σ_k^g	π_k^g
1	1	5600	2.0	0.17
	2	14,500	21.5	0.22
	3	21,200	1.1	0.23
	4	26,800	2.3	0.38
2	1	6200	5.5	0.09
	2	18,500	10.8	0.21
	3	22,800	0.5	0.70
3	1	7900	16.3	0.17
	2	19,600	2.2	0.22
	3	23,200	0.6	0.26
	4	25,800	3.9	0.25
	5	29,800	0.0	0.11
4	1	3400	0.2	0.05
	2	16,500	35.4	0.09
	3	20,300	0.1	0.05
	4	22,700	0.3	0.60
	5	28,000	2.5	0.21
5	1	8000	14.3	0.13
	2	22,000	6.5	0.59
	3	23,500	0.0	0.24
	4	27,300	0.0	0.04
6	1	8000	14.3	0.13
	2	22,000	6.5	0.59
	3	23,500	0.0	0.24
	4	27,300	0.0	0.04
7	1	5500	2.2	0.07
	2	12,900	9.9	0.08
	3	19,700	3.1	0.34
	4	23,100	0.4	0.43
	5	27,100	1.3	0.08
8	1	5900	3.1	0.07
	2	11,200	3.1	0.04
	3	22,000	8.3	0.66
	4	27,300	0.0	0.23
9	1	7300	6.9	0.07
	2	16,400	7.5	0.18
	3	19,800	0.8	0.38
	4	22,600	0.4	0.24
	5	26,500	1.3	0.13

Table 5 (continued)

Instance no	k	μ_k^g (kg)	Σ_k^g	π_k^g
10	1	7700	7.0	0.08
	2	17,100	5.6	0.22
	3	20,100	0.9	0.37
	4	22,700	0.4	0.20
	5	26,400	1.4	0.13
11	1	6900	4.9	0.07
	2	15,800	8.8	0.15
	3	19,900	1.1	0.38
	4	22,700	0.4	0.24
	5	26,100	1.6	0.16
12	1	7400	6.3	0.07
	2	16,300	6.7	0.15
	3	19,800	1.3	0.42
	4	22,700	0.7	0.24
	5	26,600	2.2	0.12

All Σ_k^g are scaled by a factor of 10^{-6}

Acknowledgements The authors are grateful to the anonymous reviewers for reading the manuscript carefully and providing constructive comments which greatly helped to improve this paper.

Author contributions HJP: Conceptualization of this study, Methodology, Validation, Formal analysis, Writing—Original Draft, Writing—Review Editing. SWC: Conceptualization of this study, Methodology, Formal analysis, Writing—Original Draft, Writing—Review Editing, Project administration, Supervision. AN: Software, Data Curation, Visualization. JHP: Writing—Review Editing, Project administration, Supervision, Funding acquisition.

Funding This research is a part of the project titled “Development of Open Platform Technologies for Smart Maritime Safety and Industries” funded by the Korea Research Institute of Ships and Ocean Engineering (PES4450).

Availability of data and material Not applicable.

Code availability Not applicable.

Declarations

Conflict of interest The authors declare no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission

directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Ambrosino D, Caballini C, Siri S (2013) A mathematical model to evaluate different train loading and stacking policies in a container terminal. *Marit Econ Logist* 15(3):292–308. <https://doi.org/10.1057/mel.2013.7>
- Borgman B, van Asperen E, Dekker R (2010) Online rules for container stacking. *OR Spectr* 32(3):687–716. <https://doi.org/10.1007/s00291-010-0205-4>
- Carlo HJ, Vis IF, Roodbergen KJ (2014) Storage yard operations in container terminals: literature overview, trends, and research directions. *Eur J Oper Res* 235(2):412–430. <https://doi.org/10.1016/j.ejor.2013.10.054>
- Chen L, Lu Z (2012) The storage location assignment problem for outbound containers in a maritime terminal. *Int J Prod Econ* 135(1):73–80. <https://doi.org/10.1016/j.ijpe.2010.09.019>
- De Castillo B, Daganzo CF (1993) Handling strategies for import containers at marine terminals. *Transp Res B Methodol* 27(2):151–166. [https://doi.org/10.1016/0191-2615\(93\)90005-U](https://doi.org/10.1016/0191-2615(93)90005-U)
- Dekker R, Voogd P, Asperen EV (2006) Advanced methods for container stacking. *OR Spectr* 28(4). <https://doi.org/10.1007/s00291-006-0038-3>
- Duinkerken MB, Evers JJ, Ottjes JA (2001) 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.
- Feng Y, Song DP, Li D (2021) Smart stacking for import containers using customer information at automated container terminals. *Eur J Oper Res*. <https://doi.org/10.1016/j.ejor.2021.10.044>
- Figueiredo MAT, Jain AK (2002) Unsupervised learning of finite mixture models. *IEEE Trans Pattern Anal Mach Intell* 24(3):381–396. <https://doi.org/10.1109/34.990138>
- Güven C, Eliyi DT (2014) Trip allocation and stacking policies at container terminal. *Transp Res Proc* 3:565–573. <https://doi.org/10.1016/j.trpro.2014.10.035>
- Güven C, Eliyi DT (2019) Modelling and optimisation of online container stacking with operational constraints. *Marit Policy Manag* 46(2):201–216. <https://doi.org/10.1080/03088839.2018.1450529>
- He J, Tan C, Yan W, Huang W, Liu M, Yu H (2020a) Two-stage stochastic programming model for generating container yard template under uncertainty and traffic congestion. *Adv Eng Inform* 43:101032. <https://doi.org/10.1016/j.aei.2020.101032>
- He Y, Wang A, Su H (2020b) The impact of incomplete vessel arrival information on container stacking. *Int J Prod Res* 58(22):6934–6948. <https://doi.org/10.1080/00207543.2019.1686188>
- Hu W, Wang H, Min Z (2014) A storage allocation algorithm for outbound containers based on the outer-inner cellular automaton. *Inf Sci* 281:147–171. <https://doi.org/10.1016/j.ins.2014.05.022>
- Hu H, Mo J, Zhen L (2021) Improved Benders decomposition for stochastic yard template planning in container terminals. *Transp Res c Emerg Technol* 132:103365. <https://doi.org/10.1016/j.trc.2021.103365>
- Jiang XJ, Jin JG (2017) A branch-and-price method for integrated yard crane deployment and container allocation in transshipment yards. *Transp Res B Methodol* 98:62–75. <https://doi.org/10.1016/j.trb.2016.12.014>
- Kang J, Ryu KR, Kim KH (2006) Deriving stacking strategies for export containers with uncertain weight information. *J Intell Manuf* 17(4):399–410. <https://doi.org/10.1007/s10845-005-0013-x>
- Karp RM (1992) On-line algorithms versus off-line algorithms: How much is it worth knowing the future? In: *IFIP Congress (1)*, Vol. 12, pp. 416–429.
- Kim KH, Kim HB (1999) Segregating space allocation models for container inventories in port container terminals. *Int J Prod Econ* 59(1–3):415–423. [https://doi.org/10.1016/S0925-5273\(98\)00028-0](https://doi.org/10.1016/S0925-5273(98)00028-0)
- Kim KH, Kim HB (2002) The optimal sizing of storage space and handling facilities for import containers. *Transp Res B Methodol* 36(9):821–835. [https://doi.org/10.1016/S0191-2615\(01\)00033-9](https://doi.org/10.1016/S0191-2615(01)00033-9)

- Kim KH, Park YM, Ryu KR (2000) Deriving decision rules to locate export containers in container yards. *Eur J Oper Res* 124(1):89–101. [https://doi.org/10.1016/S0377-2217\(99\)00116-2](https://doi.org/10.1016/S0377-2217(99)00116-2)
- Kim KH, Kang JS, Ryu KR (2004) A beam search algorithm for load sequencing of outbound containers in port container terminals. *OR Spectr* 26(1):93–116. <https://doi.org/10.1007/s00291-003-0148-0>
- Maldonado S, González-Ramírez RG, Quijada F, Ramírez-Nafarrate A (2019) Analytics meets port logistics: a decision support system for container stacking operations. *Decis Support Syst* 121:84–93. <https://doi.org/10.1016/j.dss.2019.04.006>
- Murty KG, Liu J, Wan YW, Linn R (2005) A decision support system for operations in a container terminal. *Decis Support Syst* 39(3):309–332. <https://doi.org/10.1016/j.dss.2003.11.002>
- Park T, Choe R, Kim YH, Ryu KR (2011) Dynamic adjustment of container stacking policy in an automated container terminal. *Int J Prod Econ* 133(1):385–392. <https://doi.org/10.1016/j.ijpe.2010.03.024>
- Sauri S, Martin E (2011) Space allocating strategies for improving import yard performance at marine terminals. *Transp Res E Logist Transp Rev* 47(6):1038–1057. <https://doi.org/10.1016/j.tre.2011.04.005>
- Stahlbock R, Voß S (2008) Operations research at container terminals: a literature update. *OR Spectr* 30(1):1–52. <https://doi.org/10.1007/s00291-007-0100-9>
- Steenken D, Voß S, Stahlbock R (2004) Container terminal operation and operations research—a classification and literature review. *OR Spectr* 26(1):3–49. <https://doi.org/10.1007/s00291-003-0157-z>
- Ting SC, Wang JS, Kao SL, Pitty FM (2010) Categorized stacking models for import containers in port container terminals. *Marit Econ Logist* 12(2):162–177. <https://doi.org/10.1057/mel.2010.4>
- Vis IF, De Koster R (2003) Transshipment of containers at a container terminal: an overview. *Eur J Oper Res* 147(1):1–16. [https://doi.org/10.1016/S0377-2217\(02\)00293-X](https://doi.org/10.1016/S0377-2217(02)00293-X)
- Zhang C, Liu J, Wan YW, Murty KG, Linn RJ (2003) Storage-space allocation in container terminals. *Transp Res B Methodol* 37(10):883–903. [https://doi.org/10.1016/S0191-2615\(02\)00089-9](https://doi.org/10.1016/S0191-2615(02)00089-9)
- Zhang C, Chen W, Shi L, Zheng L (2010) A note on deriving decision rules to locate export containers in container yards. *Eur J Oper Res* 205(2):483–485. <https://doi.org/10.1016/j.ejor.2009.12.016>
- Zhang C, Wu T, Kim KH, Miao L (2014a) Conservative allocation models for outbound containers in container terminals. *Eur J Oper Res* 238(1):155–165. <https://doi.org/10.1016/j.ejor.2014.03.040>
- Zhang C, Wu T, Zhong M, Zheng L, Miao L (2014b) Location assignment for outbound containers with adjusted weight proportions. *Comput Oper Res* 52:84–93. <https://doi.org/10.1016/j.cor.2014.06.012>
- Zhen L (2013) Yard template planning in transshipment hubs under uncertain berthing time and position. *J Oper Res Soc* 64(9):1418–1428. <https://doi.org/10.1057/jors.2012.108>
- Zhen L (2014) Container yard template planning under uncertain maritime market. *Transp Res E Logist Transp Rev* 69:199–217. <https://doi.org/10.1016/j.tre.2014.06.011>
- Zhou C, Wang W, Li H (2020) Container reshuffling considered space allocation problem in container terminals. *Transp Res E Logist Transp Rev* 136:101869. <https://doi.org/10.1016/j.tre.2020.101869>

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

Hyun Ji Park received the B.S. and M.S. degrees in industrial management engineering from Korea University, Seoul, Republic of Korea, in 2019 and 2021, respectively. Since 2021, she has been a researcher with Maritime Safety and Environmental Research Division, Korea Research Institute of Ships and Ocean Engineering (KRISO), Daejeon, Republic of Korea. Her research interests include maritime logistics, optimization methods, and predictive analytics.

Sung Won Cho received the B.S. degree in air transport, transportation, and logistics from the Korea Aerospace University, Gyeonggi-do, Republic of Korea, in 2016, and the Ph.D. degree in industrial management engineering from Korea University, Seoul, Republic of Korea, in 2021. He is currently a senior researcher with Maritime Safety and Environmental Research Division, Korea Research Institute of Ships and Ocean Engineering (KRISO), Daejeon, Republic of Korea. His research interests include maritime logistics, intelligent transportation systems, optimization methods, and data analytics.

Abhilasha Nanda received the B.Tech. degree in information technology from Vellore Institute of Science and Technology (VIT University), Vellore, India, the M.S. degree in information technology from Hong Kong University of Science and Technology (HKUST), Hong Kong SAR, and the Ph.D. degree in computer science from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Republic of Korea. She is currently a post doctoral researcher with Maritime Safety and Environmental Research Division, Korea Research Institute of Ships and Ocean Engineering (KRISO), Daejeon, Republic of Korea. Her research interests include computer vision, deep learning based object detection, and data analytics.

Jin Hyoung Park received the B.S. and M.S. degrees in computer science from Kyungpook National University, Daegu, Republic of Korea, in 1994 and 1996, respectively, and the Ph.D. degree in computer science from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Republic of Korea, in 2013. He is currently the senior director with Maritime Safety and Environmental Research Division, Korea Research Institute of Ships and Ocean Engineering (KRISO), Daejeon, Republic of Korea. His research interests include AI-based maritime safety systems, maritime informatics and digital twin applications in maritime industries.