OmniaScience

Journal of Industrial Engineering and Management
JIEM, 2026 – 19(1): 99-119 – Online ISSN: 2013-0953 – Print ISSN: 2013-8423
https://doi.org/10.3926/jiem.8741

Clustering–Based Column Generation and Heuristic Methods for the Container Loading Problem with Practical Constraints: A Case Study

Sezgi Tekil-Ergün[1,2*], Ferhan Çebi[1]

[1]*Department of Management Engineering, Istanbul Technical University, Istanbul (Turkey)*
[2]*INFORM GmbH, Aachen (Germany)*

[*]*Corresponding author: tekil17@itu.edu.tr*
cebife@itu.edu.tr

Received: February 2025
Accepted: January 2026

Abstract:

Purpose: This study addresses a real-world container loading problem (CLP) encountered in a logistics company in Turkey, filling a gap in the literature by solving practical constraints using a state-of-the-art algorithm. The problem involves constraints such as rotations, stackability, loading priorities, and mixed loading constraints.

Design/methodology/approach: To overcome the computational challenges posed by large-scale instances, a novel three-step approach is proposed. First, the K-Means clustering algorithm is applied to group objects with similar dimensions. Then, each group is allocated to containers using a Column Generation (CG) method combined with a 3D-Best Fit Decreasing with Orientation (3D-BFDO) algorithm. Additionally, the CG process is enhanced by integrating a machine learning (ML) model to predict reduced-cost columns, improving computational efficiency and solution quality.

Findings: Extensive experiments demonstrate that the proposed approach significantly improves container space utilization while reducing operational costs. The results highlight the effectiveness of ML and K-Means in enhancing traditional optimization techniques.

Research limitations/implications: The study focuses on a specific set of practical constraints relevant to real-world logistics applications. Further research could explore additional constraints and scalability to different logistics environments.

Practical implications: The approach offers a practical solution for logistics companies dealing with a large-scale CLP by optimizing space utilization and reducing operational costs. The integration of ML into CG presents a viable method for improving decision-making in logistics.

Originality/value: The study bridges the gap between theoretical models and real-world logistics challenges by introducing a data-driven enhancement to traditional optimization techniques. The proposed integration of K-Means clustering and ML into CG represents an innovative contribution to container loading optimization.

Keywords: container loading problem, column generation, case study, machine learning, k-means clustering

To cite this article:

Tekil-Ergün, S., & Çebi, F. (2026). Clustering–based column generation and heuristic methods for the container loading problem with practical constraints: A case study. *Journal of Industrial Engineering and Management*, 19(1), 99-119. https://doi.org/10.3926/jiem.8741

1. Introduction

Nowadays, managing logistics operations requires higher analytical capabilities and a contextual understanding of logistics systems because of the significant increase in transportation demand. As logistical challenges become greater, managing this complexity is becoming more intricate. At the forefront of these considerations lies the problem of CLP, an area that attracts the authors' attention due to its central role in promoting efficiency through cooperative and collaborative logistics operations. The CLP is concerned with optimizing capacity utilization while minimizing the number of containers used, thereby increasing operational efficiency.

Within this framework, geometric allocation models are the cornerstone to ensure a reasonable allocation of all rectangular objects to the corresponding containers. This approach not only rationalizes logistical processes but also has a significant impact on environmental sustainability. Hence, the optimization of container loading is of outstanding importance in the environmental context, as it has a direct impact on the use of resources and carbon emissions in transport logistics. By maximizing capacity utilization and reducing the number of containers required, companies can reduce the environmental footprint associated with transport activities. This phenomenon is notably evident in European Union (EU) countries with major container ports acting as primary entry points for goods into the EU. For instance, countries like Greece, Germany, the Netherlands, and Belgium exemplify this trend (European Commission, 2018).

The CLP is a variant of the bin packing problem, which is inherently NP-hard (Garey & Johnson, 1979). Early research in this domain focused on heuristic and exact algorithms for one-dimensional (1D) and two-dimensional (2D) bin packing problems (Gilmore & Gomory, 1961; Chung, Garey & Johnson, 1982). However, the three-dimensional (3D) variant poses additional challenges due to practical constraints, such as weight distribution, stability, and loading priorities (Junqueira, Oliveira, Carravilla & Morabito, 2013). These challenges make the problem highly relevant in real-world logistics operations, where the goal is to ensure an efficient arrangement of objects of varying dimensions within a container while minimizing unused space and satisfying stacking restrictions. A recent study by Zhu, Chen, Dai and Tao (2024) addressed the 3D bin packing problem (3D-BPP) with stacking constraints, proposing various optimization techniques. However, the study highlighted several limitations, including the absence of extensive real-world testing and the lack of consideration for additional logistics constraints, such as transportation-induced vibrations, weight distribution, and multi-container utilization. Furthermore, the generalizability of the proposed methods across different container types and industry-specific packaging regulations, such as cold storage or hazardous materials handling, requires further investigation and adaptation. Unlike previous studies, our proposed approach addresses not only the stacking constraints but also incorporates real-world practical constraints. Our method considers additional logistics factors, such as mixed loading patterns, mixed containers, stackability, customized rotation rules, weight distribution, and multi-container utilization, which have not been fully explored in prior works. This holistic approach enhances the practicality and generalizability of our solution, bridging the gap between theoretical models and real-world applications.

From a theoretical standpoint, the proposed approach relies on a decomposition framework that addresses the structural complexity of large, heterogeneous 3D-CLP. In our industrial dataset, nearly 4,000 items with widely varying dimensions create substantial geometric variance; therefore, K-Means clustering is used not only as a decomposition tool but also as a way to impose geometric structure by grouping volumetrically similar items, reducing variance, and enabling stack-based subproblems that are more tractable than the original instance. CG is then employed to explore the global pattern space efficiently because it can theoretically generate high-quality loading patterns without enumerating all feasible configurations; however, due to its well-known convergence issues in highly heterogeneous settings, a ML module is integrated after the initial iterations to predict columns with negative reduced cost and restrict the search to promising regions, thereby accelerating CG and preventing degeneracy. Finally, a 3D-BFDO heuristic ensures that the generated patterns translate into physically feasible placements under stackability, stability, and rotation constraints, which pure optimization models alone cannot guarantee. This combination of clustering, ML-assisted CG, and constructive heuristics thus forms a coherent theoretical framework that balances geometric feasibility, computational tractability, and operational realism for real-world CLP applications.

This study addresses a real-world CLP faced by a filter factory company in Turkey. The problem involves specific operational constraints, such as mixed loading patterns, stackability, and customized rotation rules. To solve this problem efficiently, we propose a novel three-step approach:

- Step 1: The K-Means clustering algorithm is applied to group packages with similar dimensions, reducing the complexity of the packing process.
- Step 2: The resulting clusters are allocated to containers using two state-of-the-art methods: the CG method and the 3D-BFDO algorithm.
- Step 3: A ML algorithm is applied to a large-scale real-world dataset to decrease the run time of the CG method.

K-Means clustering, introduced by Forgy (1965), has been widely applied in optimization problems as a preprocessing step to group similar items (Sheng, Xiuqin, Changjian, Hongxia, Dayong & Feiyue, 2017). Its integration with advanced packing methods such as CG and 3D-BFDO is a novel aspect of this study. CG, in particular, has demonstrated significant success in solving large-scale combinatorial problems (Pisinger & Sigurd, 2005), making it a suitable choice for this context.

The primary contributions of this study include:

- The introduction of a hybrid approach combining K-Means clustering with CG and 3D-BFDO algorithms to address practical constraints in CLP.
- A first attempt to use a ML algorithm to find promising columns in CG.
- A case study demonstrating the real-world applicability of the proposed method, with improvements in container utilization and operational efficiency.
- Insights into the impact of operational constraints, such as weight and stackability, on the performance of the proposed solution.

The rest of this paper is organized as follows: Section 2 provides a review of the literature on bin packing algorithms and their applications in CLP. Sections 3 and 4 outline the proposed methodology, covering K-Means clustering and packing algorithms. Section 5 introduces the case study, followed by Section 6, which presents the case study results. Section 7 discusses the results, and Section 8 concludes with recommendations for future research.

2. Literature Review

The bin packing (BP) problem aims to place a set of rectangular items into bins while avoiding overlap and keeping items axis-aligned, with the objective of minimizing the number of bins required—an NP-hard task (Garey & Johnson, 1979). BP variants are typically categorized by dimensionality, including 1D, 2D and 3D extensions. Early contributions such as Gilmore and Gomory (1961) introduced decomposition-based approaches for 2D-BP, while Chung et al. (1982) developed classical constructive heuristics including Next-Fit Decreasing Height (NFDH). For more complex instances, Jakobs (1996) applied genetic algorithms, and Bischoff and Marriott (1990) compared multiple constructive heuristics. Pisinger and Sigurd (2005) further demonstrated that delayed CG can produce strong bounds for packing problems.

As the problem evolved toward real-world applications, additional constraints —orientation, stackability, weight balancing, stability and unloading priorities— became central in both research and practice. Junqueira et al. (2013) incorporated these constraints into a mixed-integer programming (MIP) and CG framework for 3D-CLP, while Nascimento, de-Queiroz and Junqueira (2021) analyzed the computational impact of 12 practical loading constraints. More recently, Gajda, Trivella, Mansini and Pisinger (2022) proposed a randomized constructive heuristic that considers unloading sequences and dynamic stability, highlighting the increasing focus on operationally realistic settings.

In recent years, the literature has shown a marked shift toward hybrid and advanced heuristic methodologies for 3D container loading. Zhang, Gu, Fang, Ji and Zhang (2022) introduced a multi-strategy hybrid heuristic for the single-container 3D-CLP, demonstrating notable improvements in packing efficiency. Şafak and Erdoğan (2023) developed a Large Neighbourhood Search (LNS) algorithm to address multi-container loading, emphasizing the importance of neighbourhood diversification for large-scale heterogeneous datasets. Krebs, Ehmke and Koch (2023) integrated container loading with vehicle routing decisions, showing that spatial feasibility strongly affects distribution planning. These studies highlight the increasing relevance of large-scale, multi-container, and operationally integrated loading problems.

Automation-oriented approaches have also emerged. Jiao, Huang, Song, Li and Wang (2024) proposed a robotic loader–based formulation (CLP-RLS), incorporating mechanical and operational constraints arising in automated warehouses. Additionally, multi-objective metaheuristics have received increasing attention; Truong and Chien (2024) introduced a multi-population swarm method capable of simultaneously optimizing utilization, load balance and packing quality. Parallel to these heuristic and robotic developments, learning-enhanced optimization methods have gained traction. Zhao, Zhu, Xu, Huang and Xu (2022) proposed a deep reinforcement learning (DRL) framework for online 3D-BPP, while Murdivien and Um (2023) introduced BoxStacker, a DRL mechanism optimized for stability and density in virtual logistics systems. Wong, Tsai and Ou (2024) presented a hybrid heuristic–DRL strategy for online 3D-BPP, and Tsang, Mo, Chung and Lee (2025) released DeepPack3D, integrating heuristic search with learned spatial representations for large-scale online scenarios. Within this hybrid and learning-enhanced line of research, Montes-Franco, Martinez-Franco, Tabares and Álvarez-Martínez (2025) proposed a hybrid container loading algorithm that incorporates a dynamic stability representation to ensure mechanically feasible load patterns, further emphasizing the growing trend toward physically realistic hybrid models.

CG remains an effective decomposition technique for constrained loading, both in classical 2D settings and recent 3D extensions. CG-based formulations have been shown to improve tractability, especially when combined with constructive heuristics or domain-specific decomposition strategies.

Despite these advances, several limitations persist in the literature. First, recent heuristic and hybrid metaheuristic studies (Zhang et al., 2022; Şafak & Erdoğan, 2023; Truong et al., 2024) focus on optimization performance but do not incorporate clustering-based decomposition, which is essential for large-scale heterogeneous real-world datasets. Second, while CG-based models remain powerful, existing work does not combine K-Means clustering, CG and 3D constructive heuristics within a unified framework capable of handling practical CLP constraints such as customized rotation rules, weight balancing, stackability, mixed loading patterns and multi-container utilization. Third, although DRL and learning-supported methods (Zhao et al., 2022; Murdivien & Um, 2023; Wong et al., 2024; Tsang et al., 2025; Montes-Franco et al., 2025) have shown promise, they primarily address simplified or online settings and do not integrate machine-learning-aided column selection to accelerate CG in industrial-scale CLP environments.

To address these gaps, this study introduces a hybrid three-step methodology that integrates K-Means clustering, CG and a 3D-BFDO heuristic, complemented by a ML model for predicting promising columns. By combining clustering-based decomposition with CG and constructive 3D placement, the proposed approach enhances scalability, reduces computational effort and improves feasibility under real-world operational constraints.

Synthesizing across these prior contributions, three main gaps become evident: (i) heuristic and DRL-based methods excel in flexibility but lack physically realistic constraint modeling, (ii) CG formulations offer strong optimization capability but scale poorly with heterogeneous, industry-sized datasets, and (iii) clustering methods are rarely used to reduce item-level variance prior to packing, despite their potential to improve geometric consistency. The proposed hybrid methodology unifies these strands by applying clustering for structural decomposition, CG for optimized pattern generation, and 3D-BFDO heuristics for feasible placement under practical constraints. This integration directly addresses the scalability, realism, and tractability limitations consistently identified in the literature.

3. K-Means Clustering

K-Means clustering is a widely used algorithm for partitioning a dataset into groups based on similarity. It partitions data into K clusters, where each data point belongs to the cluster with the nearest centroid. The algorithm minimizes the sum of squared distances between data points and their respective cluster centroids.

Let $X = \{x_1,\dots, x_n\}$ be a dataset residing in a d-dimensional Euclidean space, and let $A = \{a_1,\dots, a_K\}$ represent the cluster centers. The objective function of K-Means is given by:

$$J(z, A) = \sum_{i=1}^{n}\sum_{k=1}^{K} z_{ik} \parallel x_i - a_k \parallel^2,$$

Where z_{ik} is a binary variable indicating whether data point x_i belongs to the k-th cluster ($z_{ik} = 1$) or not ($z_{ik} = 0$). The algorithm iteratively updates cluster centers a_k and cluster memberships z_{ik} until convergence:

$$a_k = \frac{\sum_{i=1}^{n} z_{ik}\, x_i}{\sum_{i=1}^{n} z_{ik}}, \quad z_{ik} = \begin{cases} 1 & \text{if } \parallel x_i - a_k \parallel^2 = \min\limits_{1\le k\le K} \parallel x_i - a_k \parallel^2, \\ 0 & \text{otherwise.} \end{cases}$$

Here, $||x_i - a_k||^2$ represents the Euclidean distance between data point x_i and cluster center a_k.

K-Means clustering is particularly beneficial as a preprocessing step for complex optimization problems like the CLP. By clustering objects with similar dimensions (e.g., length, width, height), K-Means reduces problem complexity and facilitates better packing efficiency. This method enables the optimization of container space utilization by grouping objects that fit well together. For instance, Sheng et al. (2017) demonstrated that clustering improves computational performance and space utilization in packing algorithms, making it a valuable addition to logistics operations.

This study utilizes K-Means clustering to group objects based on their dimensions, serving as a preprocessing step before applying advanced packing algorithms. By forming clusters of similar objects, the proposed approach simplifies the packing problem, reduces computational time, and ensures that practical constraints such as stackability and stability are better addressed. The integration of K-Means with CG and 3D-Best Fit algorithms represents a novel contribution to solving large-scale CLP instances efficiently.

4. Problem Definition and Solution Approach

This study proposes an integrated approach to address the CLP, combining a 3D-BFDO and CG packing model. The 3D-BFDO algorithm builds on the work by Dube, Kanavathy and Woodview (2006), while the CG model draws on the principles of Dantzig and Wolfe (1960). The proposed methodology consists of two phases. In the first phase, objects are packed into artificial stacks using the 3D-BFDO algorithm. In the second phase, these stacks are assigned to containers, adhering to practical constraints such as stackability, where heavy objects cannot be placed atop light objects.

Let $I = \{1,\dots, i\}$ represent the set of heavy objects, where each heavy object i has dimensions l_i, w_i, h_i, and a weight q_i. $K = \{1,\dots, k\}$ is the set of available containers, each characterized by dimensions L_k, W_k, and H_k (length, width, and height) and weight Q_k. Let $U = \{1,\dots, u\}$ denote the set of light objects, where each light object u has dimensions l_u, w_u, h_u, and a weight q_u. The objective is to minimize wasted space in containers after placing all objects.

To facilitate efficient packing, we utilize K-Means clustering for both heavy and light objects based on their three-dimensional properties. Heavy objects (I) are grouped into clusters $C_I = \{c_1,\dots, c_c\}$, where c represents the number of clusters. Similarly, light objects (U) are clustered into $C_U = \{c_1,\dots, c_g\}$, with g denoting the number of clusters.

4.1. Model Formulation

In this section, we adapt stackability constraints to the model formulation initially proposed by Chen, Lee and Shen (1995). The parameters and decision variables are shown as follows:

Parameters:

- l_i, w_i, h_i, q_i: Length, width, height, and weight of heavy object $i \in I$.
- l_u, w_u, h_u, q_u: Length, width, height, and weight of light object $u \in U$.

- L_k, W_k, H_k, Q_k: Length, width, height, and weight capacity of container $k \in K$.
- M: A sufficiently large constant for big-M constraints.

Decision Variables:

- $x_{ik} \in \{0,1\}$: Binary variable indicating whether stack $i \in S$ is packed into container $k \in K$.
- $x_{ij}^+, x_{ij}^- \in \{0,1\}$: Binary variables where $x_{ij}^+ = 1$ and $x_{ij}^- = 1$ if stacks i and j are assigned to the same container and stack i is located to the left and right of stack j, respectively.
- $y_{ij}^+, y_{ij}^- \in \{0,1\}$: Binary variables where $y_{ij}^+ = 1$ and $y_{ij}^- = 1$ if stacks i and j are assigned to the same container and stack i is located behind and in front of stack j, respectively.
- $z_{ij}^+, z_{ij}^- \in \{0,1\}$: Binary variables where $z_{ij}^+ = 1$ and $z_{ij}^- = 1$ if stacks i and j are assigned to the same container and stack i is located below and above stack j, respectively.
- $n_k \in \{0,1\}$: Binary variable indicating if container $k \in K$ is selected.
- $x_i, y_i, z_i, \geq 0$: Continuous variables representing the coordinates of stack $i \in S$.
- $l_i^x, l_i^y, l_i^z, \in \{0,1\}$: Binary variables representing the orientation of stack i concerning its length, width, and height parallel to the X-axis, Y-axis, or Z-axis, respectively.
- $w_i^x, w_i^y, w_i^z, \in \{0,1\}$: Binary variables representing the orientation of stack i concerning its width, width, and height parallel to the X-axis, Y-axis, or Z-axis, respectively.
- $h_i^x, h_i^y, h_i^z, \in \{0,1\}$: Binary variables representing the orientation of stack i concerning its height, width, and height parallel to the X-axis, Y-axis, or Z-axis, respectively.

The objective function and constraints are formulated as follows:

$$\min \quad \sum_{k \in K} n_k \tag{1}$$

- Packing Constraints:

$$\sum_{k \in K} x_{ik} = 1, \quad \forall i \in S \tag{2}$$

$$x_{ik} \leq n_k, \quad \forall i \in S, \forall k \in K \tag{3}$$

- Dimensional Constraints:

$$x_i + l_i l_i^x + w_i w_i^x + h_i h_i^x \leq L_k + (1 - x_{ik})M, \quad \forall i \in S, k \in K \tag{4}$$

$$y_i + l_i l_i^y + w_i w_i^y + h_i h_i^y \leq W_k + (1 - x_{ik})M, \quad \forall i \in S, k \in K \tag{5}$$

- Height Constraint:

$$z_i + l_i l_i^z + w_i w_i^z + h_i h_i^z \leq H_k + (1 - x_{ik})M, \quad \forall i \in S, k \in K \tag{6}$$

- Weight Constraint:

$$\sum_{i \in I} q_i \, x_{ik} + \sum_{j \in U} q_j \, x_{jk} \leq Q_k n_k, \quad \forall k \in K \tag{7}$$

- Relative Positioning Constraints:

$$x_i + l_i l_i^x + w_i w_i^x + h_i h_i^x \le x_j + (1 - x_{ij}^+)M, \quad \forall i, j \in S, i < j \tag{8}$$

$$x_j + l_j l_j^x + w_j w_j^x + h_j h_j^x \le x_i + (1 - x_{ij}^-)M, \quad \forall i, j \in S, i < j \tag{9}$$

$$y_i + l_i l_i^y + w_i w_i^y + h_i h_i^y \le y_j + (1 - y_{ij}^+)M, \quad \forall i, j \in S, i < j \tag{10}$$

$$y_j + l_j l_j^y + w_j w_j^y + h_j h_j^y \le y_i + (1 - y_{ij}^-)M, \quad \forall i, j \in S, i < j \tag{11}$$

$$z_i + l_i l_i^z + w_i w_i^z + h_i h_i^z \le z_j + (1 - z_{ij}^+)M, \quad \forall i, j \in S, i < j \tag{12}$$

$$z_j + l_j l_j^z + w_j w_j^z + h_j h_j^z \le z_i + (1 - z_{ij}^-)M, \quad \forall i, j \in S, i < j \tag{13}$$

- Orientation and Validity Constraints:

$$l_i^x + l_i^y + l_i^z = 1, \quad \forall i \in S \tag{14}$$

$$w_i^x + w_i^y + w_i^z = 1, \forall i \in S \tag{15}$$

$$h_i^x + h_i^y + h_i^z = 1, \quad \forall i \in S \tag{16}$$

$$l_i^x + w_i^x + h_i^x = 1, \quad \forall i \in S \tag{17}$$

$$l_i^y + w_i^y + h_i^y = 1, \quad \forall i \in S \tag{18}$$

$$l_i^z + w_i^z + h_i^z = 1, \quad \forall i \in S \tag{19}$$

$$x_i, y_i, z_i \ge 0, \quad \forall i \in S \tag{20}$$

- Stacking Constraint:

$$z_{ij}^+ = 0, z_{ij}^- = 0, \quad \forall i \in I, j \in U \tag{21}$$

Objective function (1) minimizes the number of containers used. Constraint (2) ensures that all stacks are packed into a container, while Constraint (3) guarantees that a container is considered used only if it contains at least one stack. Constraints (4)–(6) ensure that all stacks are placed within the container's physical dimensions (length, width, and height). Constraint (7) imposes the weight limit for each container, ensuring the total weight of packed heavy and light objects does not exceed the container's capacity. Constraints (8)–(13) enforce relative positions of stacks (left, right, behind, in front, below, and above). Constraints (14)–(19) define the proper orientation of a stack, while Constraint (20) imposes non-negativity on the coordinates. Finally, Constraint (21) ensures that heavy-weight stacks are not placed above lightweight stacks.

Additionally, we define stacks as groups of objects placed together, with the following relationships: a container contains multiple stacks, and each stack consists of one or several objects. In the next section, we introduce how the 3D-BFDO algorithm generates these stacks. Subsequently, we assign these stacks to containers using the CG algorithm introduced in Section 4.2.2.

4.2. K-Means Integrated Column Generation and 3D-BFDO Algorithm

The K-Means Integrated CG and 3D-BFDO (KM-3DCGBFO) algorithm combines K-Means clustering with the CG and 3D-BFDO algorithm to optimize packing efficiency. The KM-3DCGBFO algorithm operates in two main phases: (1) the 3D-BFDO method generates stacks by assigning objects based on spatial and weight constraints, and (2) the CG method assigns these stacks to containers. This approach minimizes wasted space while adhering to clustering results and constraints defined in the problem definition section.

4.2.1. 3D-BFDO Algorithm

The 3D-BFDO algorithm is employed to assign objects to stacks, considering spatial and weight constraints. By iteratively placing objects in stacks, the algorithm minimizes unused space by selecting the stack configuration that leaves the least remaining space after placement. The method also accounts for six possible rotations of each object to maximize packing efficiency.

To facilitate stacking, the set of feasible stacks for heavy-weight clusters (*SI*) and light-weight clusters (*SU*) are generated, forming $S = SI + SU$. Each stack is initialized with dimensions matching the minimum container size to ensure compatibility with all containers in *K*. The KM-3DCGBFO algorithm begins by initializing clusters for heavy and light objects using K-Means clustering. The heavy objects (C_I) and light objects (C_U) are processed independently to ensure compatibility with weight and spatial constraints. Once clusters are defined, stacks are initialized with dimensions matching the smallest container, ensuring compatibility across all available containers in the system. During the stack creation process, objects are added to stacks only if the total weight of the stack, including the object to be added, does not exceed the weight capacity of the container (*Qk*). After adding an object, the stack dimensions are dynamically updated to reflect the maximum coordinates of the contained objects. These updates ensure accurate placement for subsequent objects in the stack. To optimize the utilization of container space, the algorithm considers all six possible orientations for each object and selects the most efficient configuration based on spatial constraints. This rotation flexibility enhances packing efficiency by minimizing unused space. Finally, the algorithm returns updated stacks ($S = SI + SU$) as the output, which are subsequently assigned to containers in the next phase using the CG method. The algorithm processes objects in each cluster, ensuring weight and spatial constraints are respected:

3D-BFDO Algorithm Approach

```
0:  Initialize $CI$ (heavy − weight clusters)and $CU$ (light − weight clusters).
1:  Initialize empty sets of stacks $SI$ for heavyweight clusters and $SU$ for lightweight clusters..
2:  Set stack dimensions to the minimum container size in $K$.
3:  Initialize coordinates of objects $i \in I$ as $x_{is_c} = 0, y_{is_c} = 0, z_{is_c} = 0$.
4:  Initialize coordinates of objects $j \in U$ as $x_{js_g} = 0, y_{js_g} = 0, z_{js_g} = 0$.
5:  for all cluster $c$ in $CI$ do
6:      for all object $i$ in $c$ do
7:          if $qsc + qi \leq Qk$ then
8:              Assign object $i$ to stack $s_c$ using 3D − BFDO.
9:              Update stack dimensions $l_{s_c} = \max(x_{is_c}), w_{s_c} = \max(y_{is_c}), h_{s_c} = \max(z_{is_c})$.
10:             Add $s_c$ to $SI$ with updated weight $q_{s_c}$.
11: for all cluster $g$ in $CU$ do
12:     for all object $j$ in $g$ do
13:         if $qsg + qj \leq Qk$ then
14:             Assign object $j$ to stack $s_g$ using 3D − BFDO.
15:             Update stack dimensions $l_{s_g} = \max\left(x_{js_g}\right), w_{s_g} = \max\left(y_{js_g}\right), h_{s_g} = \max\left(z_{js_g}\right)$.
16:             Add $s_g$ to $SU$ with updated weight $q_{s_g}$ .
17: return updated stacks $S = SI + SU$
```

By integrating clustering with 3D-BFDO, the algorithm achieves efficient packing while respecting practical constraints. The generated stacks are subsequently assigned to containers using the CG method, detailed in Section 4.2.2.

4.2.2. Column Generation Approach

In this section, we introduce a CG model for assigning stacks to containers. CG is an efficient method for solving variants of bin packing problems, as the problem is split into a set partitioning (master problem)(MP) and a sub-problem (SP). In this approach, CG is applied to assign stacks to the relevant containers.

Master Problem:

The initial step to solving the 3D-BPP involves expressing it as a set-partitioning formulation using Dantzig-Wolfe decomposition, which enables the use of the CG technique. The central concept of this set-partitioning formulation for the 3D-BPP is to systematically list all feasible packing arrangements. Let P be the set of all feasible packing configurations for containers. For each packing configuration $p \in P$, a binary variable d_p is defined, which equals 1 if the packing configuration is selected and 0 otherwise. A binary parameter $\theta_{sp} \in \{0,1\}$ indicates whether stack $s \in S$ is included in configuration p. The master problem is formulated as follows:

$$\min \sum_{p \in P} d_p \tag{22}$$

$$\text{s.t.} \qquad \sum_{p \in P} \theta_{sp}\, d_p = 1 \qquad \forall s \in S \tag{23}$$

$$d_p \in \{0,1\} \qquad \forall p \in P \tag{24}$$

The objective function (22) minimizes the number of containers used. Constraints (23) ensure that each stack is allocated to exactly one packing pattern, while constraints (24) enforce the binary nature of the decision variables. Initially, the MP can start with a single packing pattern P' containing all stacks. This set P' is iteratively extended by generating promising packing patterns using the SP. Based on the relaxed MP (allowing $d_p \geq 0$) and restricted master problem (RMP) (limited to P'), we compute dual variables $\beta_s \geq 0$. The dual of the MP is given by:

$$\max \sum_{s \in S} \beta_s \tag{25}$$

$$\text{s.t.} \qquad \sum_{s \in S} \theta_{ps}\, \beta_s \leq 1 \qquad \forall p \in P' \tag{26}$$

$$\theta_{ps} \in \mathbb{R} \tag{27}$$

The reduced cost of a packing pattern p is computed as:

$$1 - \sum_{s \in S} \theta_{ps}\, \beta_s \tag{28}$$

Sub-Problem:

In the SP, we solve the 3D-BPP to generate new packing patterns with minimum reduced costs. This involves solving the following optimization problem, incorporating the constraints introduced in Section 4.1:

$$\min \sum_{k \in K} n_k - \sum_{i \in S} \theta_i\, \bar{\beta}_i \tag{29}$$

s.t. Constraints (3-21) and constraints (2) are modified as:

$$\sum_{i \in K} x_{ik} = \theta_i \qquad \forall i \in S. \tag{30}$$

The dual parameters $\overline{\beta_i}$ are derived from the MP. For each iteration of the CG process, we solve the SP to find a feasible packing pattern s with the minimum negative reduced cost. New columns (variables) with negative reduced costs are added to the RMP, and the RMP is re-solved to generate updated dual values. This iterative process continues until no further variables with negative reduced costs are found. The overall algorithmic steps of the KM-3DCGBFO approach are visually represented in Figure 1.

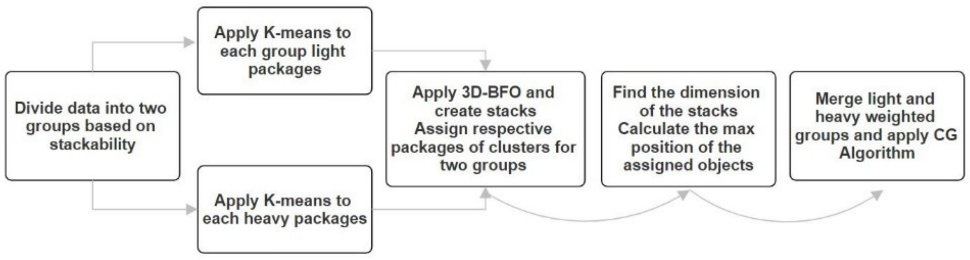
Apply K-means to each group light packages
Divide data into two groups based on stackability
Apply K-means to each heavy packages
Apply 3D-BFO and create stacks Assign respective packages of clusters for two groups
Find the dimension of the stacks Calculate the max position of the assigned objects
Merge light and heavy weighted groups and apply CG Algorithm

Figure 1. Algorithmic steps of KM-3DCGBFO

4.2.3. Machine Learning Integration in Column Generation Approach

To enhance the efficiency of the CG process for large-scale real-world datasets, we propose integrating a ML model to predict the likelihood of a packing pattern having a negative reduced cost. This integration significantly reduces computational overhead by prioritizing columns that are more likely to contribute to an improved solution, thereby accelerating convergence.

Master Problem:

The MP formulation remains the same as defined in Section 4.2.2, where dual variables $\beta_s \geq 0$ are computed for each stack $s \in S$. These dual values, which indicate the marginal contribution of each stack to the objective function, serve as critical features in the ML training phase. Instead of allowing the SP to generate columns in an unrestricted manner, we employ ML to filter and prioritize the most promising candidates.

Training the Machine Learning Model:

To effectively guide the CG process, a supervised learning approach is implemented using an XGBoost model, which has been chosen due to its robustness in handling non-linear relationships and preventing overfitting. The training process follows a progressive time-series-based learning strategy:

a. *Feature Engineering*: Historical data from previously solved CG instances is used to create the training dataset. Features include stack dimensions (length, width, height), stack weight, dual values $\beta_s \geq 0$ from the MP, packing pattern characteristics (e.g., total weight, volume utilization, feasibility checks) and each column is labelled based on whether it had a negative reduced cost (i.e., was included in the MP).

b. *Target Variable*: The target variable is a binary classification label formulated in equation (31) and each column is assigned a binary label:

$$Label = \begin{cases} 1 & if\ reduced\ cost\ < 0; \\ 0 & otherwise. \end{cases} \tag{31}$$

c. *Training Algorithm:* XGBoost is selected due to its ability to handle non-linearity and importance weighting and the dataset is updated dynamically using a rolling window strategy, ensuring that only the most recent iterations contribute to training.

Integration into the Column Generation Process:

The algorithm begins with an empty column pool and an untrained XGBoost model. In the first four iterations, standard CG is executed, where the SP generates candidate columns, and all feasible columns are tested without ML assistance. From the tenth iteration onward (based on the data-set this can be higher), the ML model is integrated to predict which columns are most likely to have a negative reduced cost, significantly reducing the number of columns that need to be evaluated. Once these predicted columns are selected, they undergo a feasibility check to ensure they satisfy all constraints before being added to the column pool. The RMP is then updated and solved with the new columns, refining the solution iteratively. After at least five iterations, the ML model is retrained using the updated dataset, which includes newly generated columns and their reduced cost evaluations. The iteration counter is then incremented, and the process continues until the stopping criterion is met, ensuring an efficient balance between computational effort and solution quality.The integration of the trained ML model into the CG process involves the following steps:

Integration into the CG Process Steps

```
Initialize:  Set iteration counter $t = 1$
             Define empty column pool $P = \{ \}$
             Train dataset $D = \{ \}$(Initially empty)
             Set $ML = XGBoost( )$
             While reduced cost is non − negative:
Step 1:         Solve RMP
                Compute dual variables $\beta s$ for all $s \in S$
Step 2:         Generate new columns using Sub − Problem (SP)
                 If $t < 10$: # Standard CG process in initial iterations
                   Generate feasible columns $C_t$ using standard SP
                   Store {column features, reduced cost outcome} in dataset D
                Else: # ML − guided CG process after iteration
                   Generate a candidate set $C_t$ using standard SP
                   Apply ML model: Predict reduced cost likelihood for each column $c \in C_t$
                   Select a subset $C_t^{ML}$with high likelihood of negative reduced cost
                   Store {column features, reduced cost outcome} in dataset D
                   Retrain XGBoost model with the updated dataset
Step 3:         Feasibility Check
                Validate all columns in $C_t^{ML}$against problem constraints with SP
                If feasible, add them to the column pool P and solve RMP and update solution
Step 5:         Update iteration counter $t = t + 1$
```

By filtering irrelevant columns with ML, the CG process converges with fewer iterations and only promising columns are evaluated in the SP so it reduced the computational load. The approach scales well to larger problem instances with diverse characteristics.

4.3. K-Means Clustering Based 3D-BFDO Algorithm

In this sub-section, the proposed approach is adapted to the 3D-BFDO algorithm. This adaptation is necessary as the KM-3DCGBFO may exhibit slower performance with large instances or scenarios where the number of clusters is low, requiring heuristic methods for efficiency. The algorithm aims to maximize volume utilization (i.e., minimizing the number of containers used). Similar to the KM-3DCGBFO, we apply K-Means clustering to the packages and implement the solution approach using the 3D-BFDO algorithm (Dube et al., 2006), named KM-3DBFO.

Let S be the set of clustered objects, consisting of light-weighted objects SI and heavy-weighted objects SU, such that $S = SI + SU$. Each stack also has length (l), height (h), width (w), volume (v), and weight (q). We obtain S_d by sorting the set S in descending order based on the weight of items in the clusters to prioritize heavy objects and prevent placing them on top of lighter ones. Initially, we set the parameters for the number of containers k, volume V, length L, and weight Q of the containers to zero. The set of assigned items is denoted by β, which starts as an empty set, and γ indicates the set of unassigned items. The general concept of the algorithm is shown below.

```
K-Means Clustering Based 3D-BFDO
Step 0: Initialize $k \leftarrow 1, V_k \leftarrow 0, L_k \leftarrow 0, Q_k \leftarrow 0, \beta \leftarrow \emptyset, \gamma \leftarrow \{1, \dots, \theta\}$ .
Step 1: Apply $\mathrm{K-Means}$ and get set of clusters $S$.
Step 2: Sort set of cluster $S$ in descending order and get $S_d$.
Step 3: Choose the top − most cluster $s' \in S_d$
Step 4: Run 3D − BFDO algorithm to assign $s' \in S_d$ to $k$, where $k \in K$
Step 5: Update assigned set $\beta$ and unassigned set $\gamma$ .
Step 6: Update length $L_k$, total weight $Q_k$, and total volume $V_k$ of the container $k$.
Step 7: Update $S_d \leftarrow S_d \backslash s'$ $k \leftarrow k + 1$ $V_k \leftarrow 0, Q_k \leftarrow 0, L_k \leftarrow 0$.
```

5. Case Study and Data Collection

In the filter manufacturing facility under investigation, various types of filters are produced, including air filters, oil filters, fuel filters, and air dryer filters. Each type plays a crucial role in the operation of automotive and industrial machinery. The company implementing our algorithm specializes in producing these diverse filters. Loading these filter boxes into containers or trucks presents a logistical challenge. While air filters are not particularly heavy, they occupy considerable volume. Thus, it is crucial to avoid placing heavier boxes on top of these lightweight filters. Conversely, fuel filters, primarily made of iron, are significantly heavier and must be positioned at the bottom of the container to ensure stability during transport. Proper load distribution within the container is vital, as imbalances could lead to axle damage. Given the variety of products, load distribution was meticulously planned, accounting for differences in volume and weight. Reflecting the problem's complexity, we assume all packages must be delivered to a single customer. The standardized container lengths of 20' and 40' were key parameters influencing our loading strategies and container usage efficiency.

Filters are systematically packaged into containers based on their types, dimensions, and package sizes, as detailed in Table 1. In this table, "Filter No." indicates the unique identification number assigned to each filter type, "Inner Box" quantifies the number of small filter boxes within the package, "w" signifies the width of the small filter boxes, "l" represents their length, and "h" denotes their height. Additionally, "W" indicates the width of the large package, "L" specifies the length, and "H" denotes the height. "Total Small Boxes" represents the total count of small boxes within the package, "Total Big Boxes" quantifies the number of large boxes, and "Total Volume" denotes the collective volume occupied by the large boxes.

Filter No.	Inner Box	w	l	h	W	L	H	Tot. Small Boxes	Tot. Big Boxes	Tot. Vol
AD2000	8	145	145	175	305	305	365	248	31	1.053
AD2006	8	145	145	175	305	305	365	144	18	0.611
AF4303A	1	335	335	265	-	-	-	175	175	5.204
AF4340	1	305	305	395	-	-	-	200	200	7.349
AF4347	1	360	360	380	-	-	-	25	25	1.231
AF4412	1	200	200	460	-	-	-	200	200	3.680
AF4532	1	210	210	460	-	-	-	75	75	1.521
EF1015	12	138	138	158	567	429	173	60	5	0.210
EF1018	12	125	125	315	515	390	330	60	5	0.331

Table 1. Real Case Data

6. Results of the proposed method

In this section, we aim to examine the performance of the proposed approaches KM-3DBFO and KM-3DCGBFO. The model has been implemented using Python, running on a desktop with an Intel® Core™ i7-8550U processor with 16GB of RAM and a 64-bit platform, using a Windows 10 operating system. The run-time limit for each instance is fixed to the 600s. We first adapted the data set generated by and later on we compared our results by adapting 3D-BFDO Algorithm with orientation to handle unstackable constraints and IBM ILOG CPLEX optimizer (CPLEX) solver. Consequently, we test our approach on the large data set provided by the company and compare our results with the company's loading results. As the company has two types of filters, heavy and light, we first divide them into two groups. For each group, we apply K-Means considering the three dimensions of the packages (see Figure 2).

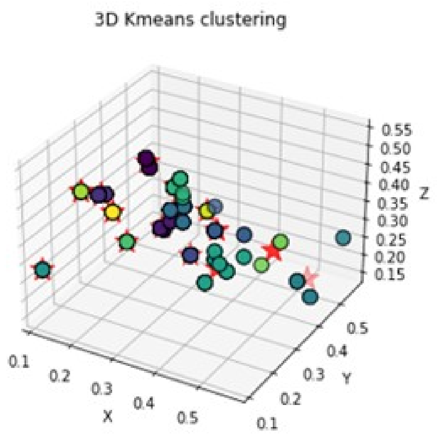

Figure 2. K-Means Clustering for 15 clusters for real case data

Each object within the clusters is assigned to the stacks (see Figure 3), and the maximum length, depth, and height of these stacks are determined. To achieve more distributed stacks, the upper bound of the stacks' dimensions is set to half the dimensions of a 20-foot container. It is important to note that when calculating the maximum dimensions of the stacks, the gap resulting from this assumption is considered in the calculation of the container's filling rate.

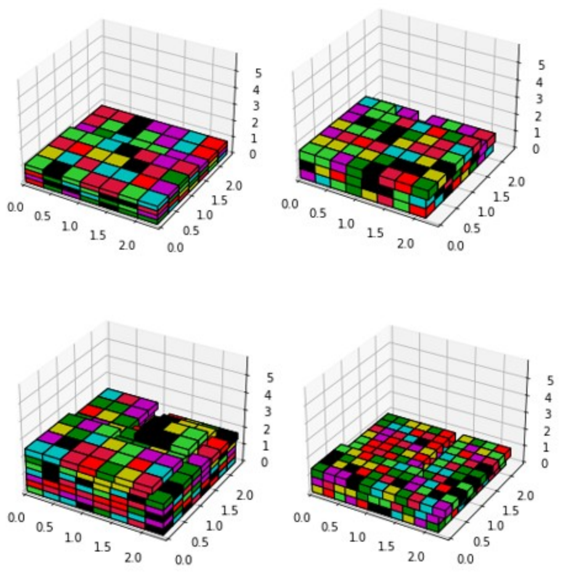

Figure 3. An example of stacks

Subsequently, we assign these stacks to containers using the proposed KM-3DBFO and KM-3DCGBFO methods. Typically, CG can exhibit poor performance for large-sized data. However, by creating stacks and reducing the sample size through clustering beforehand, we mitigate this issue. We observe that when the number of clusters is low, KM-3DCGBFO performs better, whereas when the number of clusters is high, the KM-3DBFO algorithm performs better. In other words, when the packages have various dimensions, creating fewer clusters facilitates efficient stack creation and CG execution. Conversely, when the packages exhibit high diversity and the number of items is high, it is advantageous to implement the KM-3DBFO method. As shown in the tables, we first apply our methods to the dataset by varying the cluster numbers. Subsequently, we implement these methods with real-case data. In Tables 2 and 3, the column U+I indicates the number of packages. For the CG and KM-3DCGBFO methods reported in Table 2, and for the 3D-BFDO and KM-3DBFO approaches reported in Tables 3, respectively, the columns R, CPU, and n denote the filling rate of the first container, the computational time, and the number of containers found by the corresponding algorithms. $|CI|$ and $|CU|$ indicate the number of clusters for light and heavy weight clusters, respectively. In the table, we formulate the data set wtpack1-25-F-10P as follows: the first (F) 25 packages have been selected by taking 10 of each type, and (L) indicates the last objects. In each of the data sets, half of the packages are designated as stackable (heavy) objects and the other half as unstackable (light) objects. For example, for wtpack1-25-F-10P, as shown in Table 3, the first 125 out of 250 packages are assumed to be heavy, while the remaining packages are light. The table demonstrates that when the diversity of the packages is high in the data set, setting a high number of clusters is more efficient for achieving a high filling rate. Note that the effect of orientation is not included in the tables, as it was observed to have no significant impact on this data model. Therefore, we only show results when the orientation parameter is set to 1. We also observe that K-Means achieves better performance when the diversity of the packages is high. We also set time limit to 600 for data set.

		CG				KM-3DCGBFO		
Sample	U+I	R	CPU	n		R	CPU	n
wtpack1-25-F-1P	25	2.31	10.2	1		2.31	0.06	1
wtpack1-50-F-1P	50	2.92	600	3		4.38	0.23	1
wtpack1-25-F-2P	50	1.76	600	3		3.26	0.03	1
wtpack1-50-F-2P	100	2.62	600	3		10.41	0.52	1
wtpack1-25-L-1P	25	3.67	6.78	1		3.67	0.67	1
wtpack1-50-L-1P	50	1.71	600	3		10.28	0.92	1
wtpack1-25-L-2P	100	2.70	600	3		2.70	0.67	1
wtpack1-50-L-2P	200	2.71	600	3		14.82	0.92	1

Table 2. Numerical results of KM-3DCGBFO Approach in Bischoff and Ratcliff (1995)

Sample	U+I	3D-BFDO				KM-3DBFO[a]				KM-3DBFO[b]				KM-3DBFO[c]		
		R	CPU	n		R	CPU	n		R	CPU	n		R	CPU	n
wtpack1-25-F-10P	250	0.49	14.6	2		0.76	2.4	2		0.75	10.1	2		0.76	9.1	3
wtpack1-25-F-30P	750	0.56	40.9	4		0.95	10.4	4		0.85	8.6	4		0.92	9.1	5
wtpack1-50-F-10P	500	0.57	5.32	4		0.99	4.3	3		0.8	9.1	4		0.85	4.4	4
wtpack1-50-F-15P	750	0.62	15.1	4		0.93	9.5	3		0.84	9.1	4		0.92	9.1	4
wtpack1-100-F-1P	250	0.49	10.7	2		0.7	8.6	1		0.69	3.8	1		0.45	3.8	2
wtpack1-100-F-2P	750	0.51	52.1	3		0.5	0.5	3		0.6	0.4	3		0.61	0.4	3
wtpack1-50-L-10P	500	0.55	55.9	3		0.76	8.4	3		0.74	3.4	3		0.93	4.5	3
wtpack1-50-L-15P	750	0.65	102.3	4		0.91	0.2	4		0.88	2.7	4		0.85	2.7	4

[a]Clusters set to (|CI| = 2, |CU| = 3); [b]Clusters set to (|CI| = 5, |CU| = 5); [c]Clusters set to (|CI| = 7, |CU| = 8).

Table 3. Numerical results of KM-3DBFO Approach in Bischoff and Ratcliff (1995)

Table 4-6 present the comparative results of different container loading approaches, including manual assignment by the company (Manual), and our proposed algorithms: KM-3DBFO, KM-3DCGBFO, and MLKM-3DCGBFO. In the tables, in addition to the Table 3, column R_o shows rotation. We have also tested our results with the commercial loading solver CargoWiz. According to the results from the company's manual loading process, the filling rates achieved were 0.75 for the 20-foot container in the first dataset (1A_20) and 0.72 for the second (2A_20) dataset for the first container. In contrast, the commercial CargoWiz solver, tested with all relevant problem constraints activated, provided filling rates of 0.88 and 0.85, respectively. Our proposed algorithms significantly outperformed both manual assignment and the CargoWiz solver, achieving superior filling rates. As shown in Table 4, the KM-3DBFO algorithm achieved a filling rate of 0.93 for the 1A_20 dataset with the rotation parameter set to 1, representing a 6% improvement over CargoWiz and a 24% improvement compared to the company's manual approach when the numbers of clusters is set to 100.

However, as the number of clusters increased, the filling rates exhibited a decreasing trend, suggesting that while clustering improves space utilization, an optimal number of clusters is crucial to maintain efficiency. Table 5 further compares the CG-based methods, showing that KM-3DCGBFO achieved even better filling rates than KM-3DBFO but at the cost of higher computational time. The MLKM-3DCGBFO method provided the highest filling rates across all datasets while significantly reducing computational time compared to traditional CG methods. Notably, the MLKM-3DCGBFO algorithm achieved a filling rate of 0.96 for the 2A_20 dataset with a considerable reduction in CPU time compared to standard KM-3DCGBFO.

Sample	R_O	U+I	3D-BFDO			KM-3DBFO[a]				KM-3DBFO[b]				Manual
			R	CPU	n	R	CPU	n		R	CPU	n		R
1A_20	1	3775	0.81	>1h	>1	0.93	100.2	3		0.85	98.4	3		0.75
1A_20	2	3775	0.80	>1h	>1	0.92	121.9	3		0.85	103.5	3		0.75
1A_20	3	3775	0.80	>1h	>1	0.92	119.4	3		0.85	102.9	3		0.75
2A_20	1	4148	0.85	>1h	>1	0.95	156.2	3		0.82	212.1	3		0.72
2A_20	2	4148	0.85	>1h	>1	0.95	156.8	3		0.82	256.3	3		0.72
2A_20	3	4148	0.85	>1h	>1	0.94	121.1	3		0.82	278.9	3		0.72

[a]In this scenario, the number of clusters has been set to (|CI| = 100, |CU| = 100).
[b]In this scenario, the number of clusters has been set to (|CI| = 150, |CU| = 150).

Table 4. Comparison of results for 3D-BFDO, KM-3DBFO and Manual Assignment of the Company

Sample	U+I	CG				KM-3DCGBFO[a]				MLKM-3DCGBFO[a]			Manual
		R	CPU	n		R	CPU	n		R	CPU	n	R
1A_20	3775	0.87	>1h	>=1		0.94	2600.3	2		0.94	1142.7	2	0.75
2A_20	4148	0.89	>1h	>=1		0.96	2778.2	2		0.96	1032.7	2	0.72

[a]In this scenario, the number of clusters has been set to (|CI| = 100, |CU| = 100).

Table 5. Comparison of results for CG, KM-3DCGBFO, MLKM-3DCGBFO and Manual Assignment of the Company

Sample	U+I	CG				KM-3DCGBFOb				MLKM-3DCGBFOb			Manual
		R	CPU	n		R	CPU	n		R	CPU	n	R
1A_20	3775	0.87	>1h	>=1		0.90	2621.9	2		0.90	1331.7	2	0.75
2A_20	4148	0.89	>1h	>=1		0.91	2856.2	2		0.91	986.3	2	0.72

[b]In this scenario, the number of clusters has been set to (|CI| = 150, |CU| = 150).

Table 6. Comparison of results for CG, KM-3DCGBFO, MLKM-3DCGBFO and Manual Assignment of the Company

When integrating ML into CG, the standard CG framework is applied for the first ten iterations, where the SP generates a large number of candidate columns, selecting only those with negative reduced costs into the RMP. After the tenth iteration, ML is introduced to predict the likelihood of new columns having a negative reduced cost. This enables the SP to focus on a smaller, more promising subset of columns rather than exhaustively generating a large number of potential candidates. In the |CI|=150 stack case, while the standard KM-3DCGBFO continues to produce an increasing number of columns—reaching 170 in the final iteration—MLKM-3DCGBFO begins with 70 suggested columns in iteration ten and gradually refines its output over time. Despite generating fewer columns, MLKM-3DCGBFO effectively identifies feasible solutions, with the number of selected columns in the RMP closely aligning with those from the standard KM-3DCGBFO approach. By the final iteration, MLKM-3DCGBFO significantly reduces computational overhead, proposing only 48 candidate columns compared to 170 in the standard approach, while maintaining a similar level of solution quality. This demonstrates that ML-assisted column selection not only accelerates convergence but also minimizes the number of SP evaluations, making the approach highly scalable for large-scale optimization problems.

In summary, while KM-3DBFO offers a balance between quality and computational efficiency, KM-3DCGBFO provides superior solution quality, and MLKM-3DCGBFO enhances computational performance without sacrificing solution accuracy. Thus, KM-3DBFO is recommended for scenarios requiring adaptability to high cluster counts, whereas MLKM-3DCGBFO is ideal for larger-scale problems with computational constraints. The comprehensive results of all algorithms for the real dataset are presented in Figure 4.

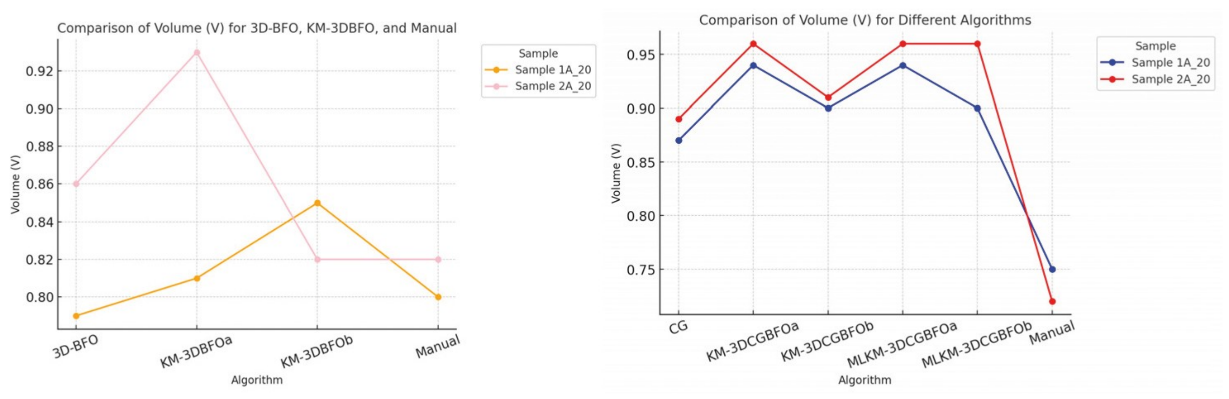

Figure 4. Graphical representation of algorithmic results

7. Discussion

The numerical results obtained in this study demonstrate that the proposed hybrid methods—KM-3DBFO, KM-3DCGBFO, and MLKM-3DCGBFO—achieve substantial improvements in container utilization, runtime efficiency, and scalability when compared both to baseline algorithms and to the performance characteristics reported in recent literature.

The comparative results show that our proposed algorithms deliver substantially higher improvements in container utilization than recent Large Neighborhood Search based (LNS) methods. While Yang, Zhou, Zhang, Zhang and Jin (2025) reports modest gains of 2.71% and 1.66% over Hybrid Adaptive Large Neighborhood Search (HALNS) —with average loading rates above 85%—these results are obtained on smaller benchmark datasets. In contrast, our industrial datasets contain 3775 and 4148 items, making the problem significantly more complex and heterogeneous. Furthermore, their approach does not incorporate physical stackability constraints, whereas our methodology explicitly models real 3D stackability, including vertical support, surface alignment, and load-balance requirements. On the real-world datasets (1A_20 and 2A_20), KM-3DBFO increases the company's manual loading rate from 75% to 93% (+24%) and improves upon CargoWiz from 88% to 94% (+6%). The MLKM-3DCGBFO algorithm further raises utilization to 96%, far exceeding the 1–3% improvement range reported by Yang et al. (2025). While the Large Neighborhood Search Algorithm based on Q-learning (Q-LNS) provides stable performance, our ML-assisted CG method reduces computation time by 40–60% (from 2778 s to 1032 s) without sacrificing container count or loading quality. In terms of computational efficiency, the three proposed methods show clear differences: the baseline 3D-BFDO requires more than one hour on both industrial datasets, whereas KM-3DBFO solves the same instances in roughly 100–150 seconds. KM-3DCGBFO achieves higher utilization but requires approximately 2600–2800 seconds, and the MLKM-3DCGBFO variant reduces this to 1032 seconds while maintaining identical solution quality. Overall, combining clustering-based decomposition with CG and ML yields not only superior loading efficiency but also improved scalability and physical feasibility for large, heterogeneous industrial datasets where traditional LNS-type methods provide only limited improvement.

The comparative evaluation indicates that the proposed KM-3DBFO, KM-3DCGBFO, and MLKM-3DCGBFO methods achieve substantially stronger performance than the LNS framework of Şafak and Erdoğan (2023), both in terms of utilization levels and scalability under real-world operational constraints. Their LNS obtains high-quality solutions on classical benchmark instances (100–250 items), reaching 80–92% utilization within 3–10 minutes time limits and achieving optimality in most LN instances when rotation around horizontal axes is restricted. However, these instances remain relatively small and structurally homogeneous compared to real industrial settings. In contrast, our experiments involve significantly larger and more heterogeneous datasets, with 3775 and 4148 items per instance, scales not addressed in Şafak and Erdoğan (2023). Despite this considerable increase in complexity, KM-3DBFO achieves 93% utilization in approximately 100–150 seconds, while MLKM-3DCGBFO reaches 96% utilization with a 1,032-second runtime. These results represent clear improvements over both manual loading (75–72%). In Şafak and Erdoğan (2023), improvements over state-of-the-art heuristics typically remain within the 1 –3% range; in comparison, our methods provide 6–24% improvements on real industrial data.

Junqueira et al. (2013) reported that pure CG becomes unsuitable for large heterogeneous 3D-CLP instances, often failing to converge within standard computation limits. This is consistent with our results: traditional CG exceeded the 600s threshold in all wtpack datasets. However, KM-3DCGBFO consistently solved instances within feasible time (0.03–0.92s for medium-sized datasets), and MLKM-3DCGBFO further reduced subproblem expansions by 3.5× (e.g., 170 → 48 columns). This indicates that the hybrid clustering + ML design effectively overcomes the convergence bottleneck historically associated with CG in CLP.

ML-assisted CG provides a scalable and physically realistic framework for large, heterogeneous CLPs. Unlike recent LNS- and DRL-based approaches, which typically report modest improvements on small benchmark instances, the proposed hybrid methods achieve substantially higher utilization and computationally feasible runtimes on real industrial datasets while strictly enforcing 3D stackability and stability constraints. The ML-enhanced CG module further mitigates classical convergence limitations in CG by reducing unnecessary column expansion and lowering overall computation time. Taken together, these findings indicate that the proposed approach offers a more robust, efficient, and operationally applicable solution than existing heuristic, learning-based, or exact frameworks in the current literature.

Beyond performance comparisons, the results reveal important theoretical and practical implications. Theoretically, they show that clustering-based decomposition is an effective mechanism for stabilizing CG under high heterogeneity, offering a scalable middle ground between purely heuristic strategies and purely exact formulations. The addition of ML-based column prediction also provides empirical evidence that hybrid learning–optimization architectures can systematically alleviate the degeneracy and slow convergence issues traditionally associated with CG in 3D-CLP. Practically, the method enables logistics companies to load highly diverse products—such as filters, oil containers, and mixed industrial packaging—more efficiently while rigorously satisfying real 3D stackability and stability requirements. This leads to reduced transportation costs, fewer containers, and more reliable loading plans, directly supporting operational efficiency in real-world logistics environments.

8. Conclusion

In this study, we proposed three innovative heuristic algorithms: KM-3DBFO**,** KM-3DCGBFO**,** and MLKM-3DCGBFO**,** to tackle the CLP in real-world logistics scenarios. These algorithms combine the strengths of K-Means clustering with advanced packing techniques such as the 3D-BFDO and CG approaches and incorporate ML to enhance solution quality and computational efficiency.

Our contributions are threefold:

1. We developed a novel integration of K-Means clustering with 3D packing algorithms, enabling improved space utilization and computational scalability in large-scale logistics problems.
2. We introduced ML into the CG framework (MLKM-3DCGBFO) to predict promising configurations, significantly reducing computational overhead and achieving superior filling rates.
3. We customized practical constraints including container stability, stackability, weight limits, and package rotation requirements into the algorithms, making them adaptable to real-world industrial requirements.

Through comparative analysis of real-case datasets (as detailed in Tables 3 and 4), we demonstrated the superiority of our proposed algorithms. The KM-3DBFO algorithm performed well in scenarios with lower cluster counts, leveraging K-Means clustering to optimize container space while adapting to diverse item distributions. The KM-3DCGBFO approach, which benefits from the CG framework, showed enhanced performance with higher cluster numbers, highlighting its ability to handle detailed packing configurations effectively.

The ML-enhanced MLKM-3DCGBFO algorithm consistently outperformed all other methods, achieving the highest filling rates while maintaining competitive runtimes. For example, in the 1A_20 dataset, MLKM-3DCGBFO achieved a filling rate of 0.94, improving upon CargoWiz's results by 6% and surpassing the company's manual assignments by 25%. This demonstrates the added value of ML in predicting promising columns for the CG process, reducing unnecessary computations, and enhancing runtime efficiency. Furthermore, the results underscored the importance of balancing cluster numbers for different algorithms. While higher cluster counts favored CG-based approaches, lower cluster counts resulted in better performance for the KM-3DBFO algorithm. This highlights the importance of considering package diversity, dimensions, and problem characteristics when selecting an appropriate loading strategy.

The proposed algorithms were validated in a real-world setting at a Turkish Filter Factory, where they successfully optimized container loading operations while adhering to specific industrial constraints. Metrics such as filling rate, runtime, and container utilization demonstrated the scalability and adaptability of the algorithms. Notably, the MLKM-3DCGBFO algorithm provided the most robust solution, balancing solution quality and computational efficiency across all tested scenarios. While the proposed approach has shown strong performance in structured logistics settings (e.g., filter manufacturing), its effectiveness in highly heterogeneous packaging environments remains an open question. Future work should explore adaptations for dynamic and real-time optimization scenarios.

As a result, this study introduces a comprehensive framework for optimizing container loading in logistics, combining clustering techniques, advanced packing methods, and ML. By doing so, we address practical constraints and provide scalable solutions for complex logistics problems. The contributions of this research extend beyond academic relevance, offering actionable insights for real-world logistics operations. Future research can further explore:

1. The integration of real-time optimization systems to dynamically adapt to changing logistics conditions.
2. Multi-objective optimization frameworks to balance conflicting goals such as cost, time, and environmental impact.
3. Enhancing ML models to predict loading configurations with greater accuracy and adaptability to diverse datasets.

Our findings demonstrate the potential of hybrid approaches to revolutionize container loading operations, paving the way for greener, more efficient, and cost-effective logistics solutions.

Acknowledgement

This paper is derived from the first author's dissertation at Istanbul Technical University. The authors extend their heartfelt gratitude to Filter Company for providing the data, and to the loading manager, whom we sadly lost during the study, for his continuous support. The authors also wish to express their sincere appreciation to the editor and anonymous reviewers for their valuable comments and constructive suggestions, which greatly improved the quality of this work.

Declaration of Conflicting Interests

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

References

Bischoff, E.E., & Marriott, M.D. (1990). A comparative evaluation of heuristics for container loading. *European Journal of Operational Research,* 44, 267-276. Available at: https://www.sciencedirect.com/science/article/abs/pii/037722179090362F

Bischoff, E.E., & Ratcliff, M. (1995). Issues in the development of approaches to container loading. *Omega,* 23, 377-390. Available at: https://www.sciencedirect.com/science/article/abs/pii/030504839500015G

Chen, C.S., Lee, S.M., & Shen, Q. (1995). An analytical model for the container loading problem. *European Journal of Operational Research,* 80, 68-76. Available at: https://www.sciencedirect.com/science/article/abs/pii/037722179400002T

Chung, F.R., Garey, M.R., & Johnson, D.S. (1982). On packing two-dimensional bins. *SIAM Journal on Algebraic Discrete Methods,* 3, 66-76. Available at: https://epubs.siam.org/doi/10.1137/0603007

Dantzig, G.B., & Wolfe, P. (1960). Decomposition principle for linear programs. *Operations Research,* 8, 101-111. Available at: https://pubsonline.informs.org/doi/10.1287/opre.8.1.101

Dube, E., Kanavathy, L.R., & Woodview, P. (2006). Optimizing three-dimensional bin packing through simulation. In *Sixth IASTED International Conference Modelling, Simulation, and Optimization.* Available at: https://www.semanticscholar.org/paper/OPTIMIZING-THREE-DIMENSIONAL-BIN-PACKING-THROUGH-Dube/bb9986af2f26f7726fcef1bc684eac8239c9b853

European Commission (2018). *Road freight transport statistics.* Available at: https://ec.europa.eu/eurostat/statistics-explained/pdfscache/9217.pdf

Forgy, E.W. (1965). Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *Biometrics,* 21, 768-769. Available at: https://www.scirp.org/reference/referencespapers?referenceid=2317605

Gajda, M., Trivella, A., Mansini, R., & Pisinger, D. (2022). An optimization approach for a complex real-life container loading problem. *Omega,* 107, 102559. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0305048321001687

Garey, M.R., & Johnson, D.S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company. Available at: https://perso.limos.fr/~palafour/PAPERS/PDF/Garey-Johnson79.pdf

Gilmore, P.C., & Gomory, R.E. (1961). A linear programming approach to the cutting-stock problem. *Operations Research,* 9, 849-859. Available at: https://pubsonline.informs.org/doi/10.1287/opre.9.6.849

Jakobs, S. (1996). On genetic algorithms for the packing of polygons. *European Journal of Operational Research,* 88, 165-181. Available at: https://www.sciencedirect.com/science/article/abs/pii/0377221794001669

Jiao, G., Huang, M., Song, Y., Li, H., & Wang, X. (2024). Container loading problem based on robotic loader system: An optimization approach. *Expert Systems with Applications,* 236, 121222. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0957417423017244

Junqueira, L., Oliveira, J.F., Carravilla, M.A., & Morabito, R. (2013). An optimization model for the vehicle routing problem with practical three dimensional loading constraints. *International Transactions in Operational Research,* 20, 645-666. Available at: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1475-3995.2012.00872.x

Krebs, C., Ehmke, J.F., & Koch, H. (2023). Effective loading in combined vehicle routing and container loading problems. *Computers & Operations Research,* 149, 105988. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0305054822002258

Montes-Franco, A.M., Martinez-Franco, J.C., Tabares, A., & Álvarez-Martínez, D. (2025). A Hybrid Approach for the Container Loading Problem for Enhancing the Dynamic Stability Representation. *Mathematics,* 13(5), 869. Available at: https://www.mdpi.com/2227-7390/13/5/869

Murdivien, S.A., & Um, J. (2023). BoxStacker: Deep Reinforcement Learning for 3D Bin Packing Problem in Virtual Environment of Logistics Systems. *Sensors,* 23(15), 6928. Available at: https://www.mdpi.com/1424-8220/23/15/6928

Nascimento, O.X., de-Queiroz, T.A., & Junqueira, L. (2021). Practical constraints in the container loading problem: Comprehensive formulations and exact algorithm. *Computers & Operations Research,* 128, 105186. Available at: https://www.sciencedirect.com/science/article/pii/S0305054820303038

Pisinger, D., & Sigurd, M. (2005). The two-dimensional bin packing problem with variable bin sizes and costs. *Discrete Optimization,* 2, 154-167. Available at: https://www.sciencedirect.com/science/article/pii/S1572528605000216

Şafak, Ö., & Erdoğan G. (2023). A large neighbourhood search algorithm for solving multi container loading problem. *Computers & Industrial Engineering,* 154, 106199. Available at: https://www.sciencedirect.com/science/article/pii/S0305054823000631?ssrnid=4122570&dgcid=SSRN_redirect_SD

Sheng, L., Xiuqin, S., Changjian, C., Hongxia, Z., Dayong, S., & Feiyue,W. (2017). Heuristic algorithm for the container loading problem with multiple constraints. *Computers & Industrial Engineering,* 108, 149-164. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0360835217301705

Truong, L.H., & Chien, C.F. (2024). Multi-objective multi-population simplified swarm optimization for container loading optimization with practical constraints. *Applied Soft Computing,* 165, 112038. Available at: https://www.sciencedirect.com/science/article/abs/pii/S1568494624008123

Tsang, Y.P., Mo, D.Y., Chung, K.T., & Lee, C.K.M. (2025). DeepPack3D: A Python package for online 3D bin packing optimization by deep reinforcement learning and constructive heuristics. *Software Impacts,* 23, 100732. Available at: https://www.sciencedirect.com/science/article/pii/S2665963824001209

Wong, C.C., Tsai, T.T., & Ou, C.K. (2024). Integrating Heuristic Methods with Deep Reinforcement Learning for Online 3D Bin-Packing Optimization. *Sensors,* 24(16), 5370. Available at: https://www.mdpi.com/1424-8220/24/16/5370

Yang, H., Zhou, W., Zhang, J., Zhang, X., & Jin, Y. (2025). A large neighborhood search algorithm based on Q-learning for multi-container loading problem. *Expert Systems with Applications,* 269, 126429. Available at: https://www.sciencedirect.com/science/article/abs/pii/S095741742500051X

Zhang, D., Gu, C., Fang, H., Ji, C., & Zhang, X. (2022). Multi-strategy hybrid heuristic algorithm for single container weakly heterogeneous loading problem. *Computers & Industrial Engineering,* 170, 108302. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0360835222003618

Zhao, H., Zhu, C., Xu, X., Huang, H., & Xu, K. (2022). Learning practically feasible policies for online 3D bin packing. *Science China Information Sciences,* 65(1), 112105. Available at: https://link.springer.com/article/10.1007/s11432-021-3348-6

Zhu, W., Chen, S., Dai, M., & Tao, J. (2024). Solving a 3d bin packing problem with stacking constraints. *Computers & Industrial Engineering,* 188, 09814. Available at: https://www.sciencedirect.com/science/article/pii/S0360835223008380