Multi-Objective Optimization of Container Loading: Balancing Space Utilization, Unloading Obstacles, and Cargo Stability

Mahboob Elahi
Tampere University: Tampereen Yliopisto
Wael Mohammed
Tampere University: Tampereen Yliopisto
Samuel Olaiya Afolaranmi
samuel.afolaranmi@tuni.fi
Tampere University: Tampereen Yliopisto https://orcid.org/0000-0002-8132-3017
Jose Luis Martinez Lastra
Tampere University: Tampereen Yliopisto
Jose Antonio Perez Garcia
University of Vigo: Universidade de Vigo

Research Article

Keywords: container loading problem, cargo balance, metaheuristic optimization, genetic algorithm, multi-drop deliveries, stability constraints

Posted Date: October 27th, 2025

DOI: https://doi.org/10.21203/rs.3.rs-7712709/v1

Multi-Objective Optimization of Container Loading: Balancing Space Utilization, Unloading Obstacles, and Cargo Stability

Mahboob Elahi[1], Wael M. Mohammed[1],
Samuel Olaiya Afolaranmi[1*], Jose Luis Martinez Lastra[1],
José Antonio Pérez García[2]

[1]FAST-Lab Faculty of Engineering and Natural Sciences, Tampere University, Tampere, Finland.
[2]Design and Fabrication in Industrial Engineering, University of Vigo, Vigo, Spain.

*Corresponding author(s). E-mail(s): samuel.afolaranmi@tuni.fi;
Contributing authors: mahboob.elahi@tuni.fi; wael.mohammed@tuni.fi;
jose.martinezlastra@tuni.fi; japerez@uvigo.gal;

Abstract

The Container Loading Problem (CLP) involves arranging rectangular boxes within a container while satisfying practical constraints such as dimensional fit, weight capacity, stackability, delivery order, customer priorities, and cargo stability. In real-world multi-drop delivery scenarios, additional considerations include minimizing unloading obstacles (ULOs) and maintaining cargo balance to ensure transport safety. This study formally defines a tri-objective CLP through a mathematical model that (i) maximizes space utilization, (ii) minimizes ULOs, and (iii) constrains the center of gravity (CG) of the loaded cargo within specified limits along each axis. The model integrates diverse practical constraints into a unified formulation. However, due to its complexity, it is used here for problem definition rather than being solved directly to optimality.

A constraint-aware variant of the Non-dominated Sorting Genetic Algorithm II (NSGA-II) is developed to generate high-quality feasible solutions. Computational results on benchmark instances from Bischoff and Ratcliff, adapted to include CG balance requirements, demonstrate that the algorithm can efficiently capture trade-offs between volume utilization, unloading efficiency, and cargo balance. The approach addresses current limitations of CLP studies by jointly considering multiple operational objectives and stability constraints, providing realistic insight into how different priorities affect container space use.

Keywords: container loading problem, cargo balance, metaheuristic optimization, genetic algorithm, multi-drop deliveries, stability constraints

1 Introduction

Globalization, product customization, and intense market competition have increased the complexity of logistics, transportation, and distribution networks. Meeting customer expectations now requires efficient, cost-effective operations that balance high product quality with rapid delivery. Within this context, packaging, logistics, and distribution play a central role in controlling costs and ensuring timely access to goods [1].

A pivotal challenge in logistics is determining the optimal arrangement of cargo items within containers. Although placing boxes into containers appears straightforward, inefficient arrangements can lead to reduced delivery efficiency, increased logistics costs, or the need for additional containers. This challenge is known as the container loading problem (CLP). CLP instances vary across sectors, often requiring compliance with multiple operational and regulatory constraints, including weight limits, unloading order, loading priorities, load balancing, stackability, and cargo stability.

The CLP is a specialized type of three-dimensional packing problem that extends the concept of the three-dimensional ($3D$) knapsack problem [2]. In CLP, the goal is to pack a set of items—typically rectangular boxes—into a larger container while adhering to specific operational and regulatory constraints. Since it is often impossible

to fit all items into the container, the challenge is to select a subset that maximizes total value, measured by volume, profit, or other performance metrics, while satisfying these constraints. Being NP-hard [2, 3], CLP remains an active research area [4–14].

Different studies have emphasized different objectives, such as reducing the number of containers shipped [4, 5], increasing the total loaded weight [6], maximizing space utilization [7–12], or maximizing the total value of loaded cargo [12].

Despite the breadth of prior research, some gaps exists as provided below:

1. Practical operational constraints are often simplified or omitted.
2. Most research addresses a single objective, typically maximizing volume utilization.
3. Few methods integrate multiple real-life constraints into a unified solution framework.

To address these gaps, we formulate a tri-objective CLP that: (i) maximizes container space utilization, (ii) minimizes unloading obstacles (ULOs), and (iii) constrains the center of gravity (CG) of the loaded cargo within specified bounds along each axis to improve stability and transport safety. The first two objectives align with prior works such as [15, 16], which also address unloading order considerations, but our approach differs in how unloading difficulty is quantified. Instead of applying a penalty function, we evaluate ULOs by simulating the unloading process and counting cases where boxes for later deliveries block access to boxes for earlier deliveries. This direct measurement captures the operational impact of box placement without imposing arbitrary weights. The third objective extends the bi-objective framework of [15, 16] by introducing a CG balance constraint, addressing load stability and safety considerations that are critical in real-world container transport but not explicitly optimized in their approach.

Given the combinatorial complexity of jointly optimizing these objectives under practical constraints, we employ a constraint-aware hybrid metaheuristic that combines the Non-dominated Sorting Genetic Algorithm II (NSGA-II) with an extreme-point placement strategy based on the Deepest-Bottom-Left Fill (DBLF) heuristic [17]. This hybrid NSGA-II plus DBLF approach produces high-quality feasible load plans that reveal trade-offs among the three objectives. Computational experiments are conducted on benchmark instances from Bischoff and Ratcliff, adapted to incorporate multi-delivery scenarios and CG balance constraints, to demonstrate how different operational priorities influence container layouts.

The remainder of this paper is organized as follows: Section 2 reviews practical CLP constraints and solution methods. Section 3 details the problem definition, constraints, mathematical formulation, and placement heuristic. Section 4 describes the metaheuristic algorithm. Section 5 presents the experimental setup and results. Section 6 concludes and outlines future research directions. Additional pseudocode and parameter-tuning results are provided in Appendix A.

2 Literature review

George and Robinson were the pioneers in introducing the Container Loading Problem (CLP) [18], which seeks the most efficient arrangement of a set of boxes within a container [6]. In practice, numerous constraints must be considered, including dimensional fit, orientation restrictions, weight distribution, and spatial limitations [19]. Early research often addressed loading problems in one or two dimensions; however, real-world logistics typically requires a three-dimensional (3D) formulation.

Over time, the CLP has evolved into multiple variants distinguished by their objectives and constraint sets. Regardless of the variant, the core requirements remain: boxes must not overlap, and the arrangement must respect freight regulations such as maximum container weight. Additional operational constraints encountered in practice

include box orientation rules, stackability, load stability, delivery sequence requirements, and customer-specific restrictions. These constraints can interact in complex ways, and while many studies have explored them individually, far fewer have sought to integrate multiple real-life constraints into a single optimization framework—a gap this study aims to address [20].

2.1 Practical constraints in CLP

In container loading, constraints can be broadly classified into *hard constraints* and *soft constraints*. Hard constraints are non-negotiable and must be strictly satisfied; any loading pattern that violates them is infeasible. Soft constraints are desirable but allow limited violations within predefined tolerances. Following and extending the classification proposed in [20, 21], Figure 1 summarizes the main categories considered in this study. In contrast to prior works that address these constraints individually, our approach integrates them into a unified framework, with particular focus on multi-drop unloading obstacles (ULOs) and center-of-gravity (CG) balance—two constraints rarely optimized jointly in CLP literature.

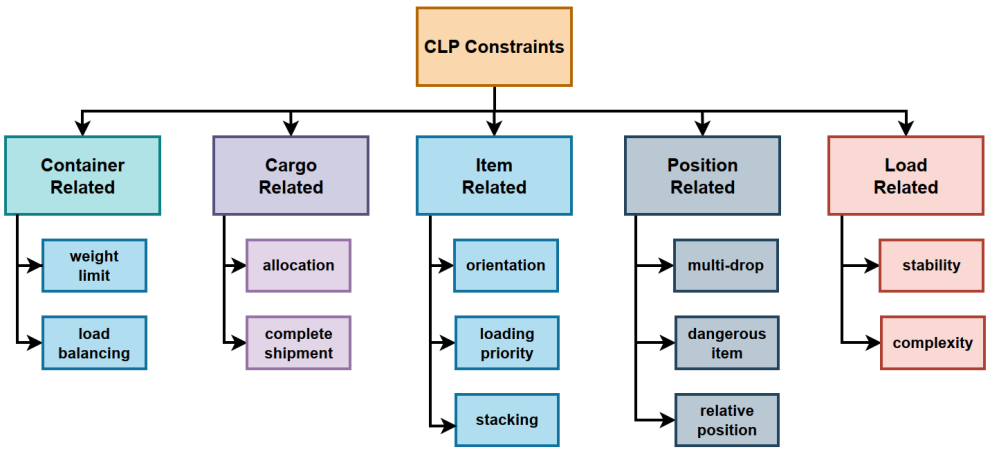


Fig. 1: CLP constraint classification

1. Constraints related to cargo containers or vehicles:
   (a) *Weight limit constraints* define the maximum allowable weight for the container or vehicle, determined by structural properties and enforced as strict requirements [22, 23].
   (b) *Load balancing* strategies ensure that cargo weight is evenly distributed. Some methods define a "stable region" where the center of mass must be located [22, 24], while others aim to position it close to an optimal point [25]. Balancing may be enforced along the longitudinal axis [26], both longitudinal and transverse axes [27, 28], or all three axes (longitudinal, transverse, vertical) [24, 25, 29].
2. Constraints related to cargo composition:
   (a) *Full shipment constraints* require certain groups of items to be loaded in their entirety or not at all [13].
   (b) *Allocation constraints* prohibit certain items from coexisting in the same container (e.g., food with perfumes or petroleum products) [30, 31].
3. Constraints related to individual items:
   (a) *Loading priorities* dictate item selection under limited space, mixing rigid and flexible rules [32, 33].
   (b) *Stacking constraints* limit the load-bearing capacity of boxes, classifying items as fragile or non-fragile [34, 35].
   (c) *Orientation constraints* limit item rotations to six orthogonal orientations (Figure 2) [36, 37].

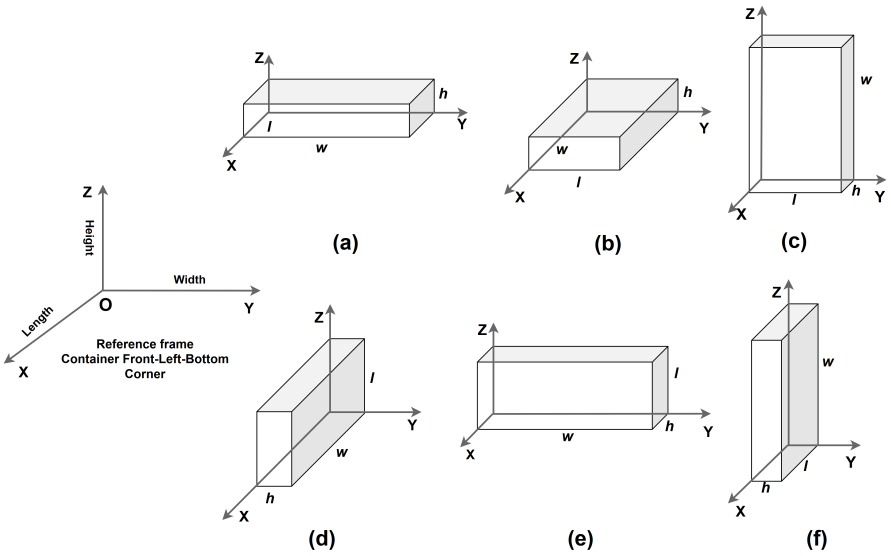


Fig. 2: Box Orientations

Item rotations affect stacking stability. While much literature focuses on rectangular items, some industrial applications consider irregular shapes in both bin packing [38] and container loading [39].

4. Constraints related to positioning:
   (a) *Absolute positioning constraints* require specific items to be placed at designated positions [40–43].
   (b) *Relative positioning constraints* require certain items to be loaded close together to facilitate unloading [39].
   (c) *Multi-drop constraints* involve rebalancing cargo after partial unloading [44]. While unloading obstacles have rarely been studied in CLP, [16] proposed a bi-objective model that considers them with space utilization.
5. Load-related constraints:
   (a) *Cargo stability* is critical to avoid damage during transport [45, 46]. Approaches include static stability [47] and vertical stability via support ratio [20, 33]. Many

studies assume compact packing ensures stability, but practical operations often require filler materials [48, 49].

(b) *Complexity constraints* ensure load plans are easy to execute. Guillotine-cutting constraints are one example [50–52].

While studies [20, 53–55], have started to include practical constraints in CLP models—such as stacking rules, weight limits, and customer priorities—these factors are often considered separately or in combinations that overlook real-world trade-offs. In particular, unloading efficiency and balance across all three CG axes are rarely addressed together. The work by [16] is the closest in spirit to ours, but it does not explicitly optimize CG balance. To the best of our knowledge, this is the first approach to simultaneously optimize space utilization, unloading efficiency, and multi-axis CG stability in a single, integrated metaheuristic framework

2.2 Methods for Solving CLP

Researchers have addressed the CLP using a wide range of approaches, including exact methods, heuristics, and metaheuristics. The preference for heuristic and metaheuristic algorithms stems from several practical realities. First, the CLP is NP-hard, and solving large-scale, constraint-rich instances exactly is computationally prohibitive. Second, while the basic CLP can be formulated as a mixed integer program (MIP), incorporating multiple practical constraints—such as stability, delivery sequence, or multi-axis load balance—significantly increases model complexity. Third, standard CLP formulations introduce symmetries and big-M constraints that weaken linear programming relaxations, making branch-and-bound approaches inefficient. Finally, logistics operations demand solutions within minutes, which exact methods cannot typically deliver [49, 56, 57].

2.2.1 Exact solutions for CLP

Exact methods for CLP are relatively scarce in the literature. Early work in [58] introduced a general formulation for multidimensional packing, and [59] presented a detailed branch-and-bound method for 3D-CLP. A comparative study in [60] reviewed multiple exact formulations for cutting and packing problems [59, 61–65], noting that even modest instances (fewer than 20 boxes) often exceeded six hours of computation.

Despite advances in commercial solvers and recent efforts to incorporate constraints like load balancing and stability into CLP models [20, 25, 66], exact methods remain impractical for solving the type of large, constraint-rich problems addressed in this study. Real-world scenarios typically involve dozens or even hundreds of boxes, multiple delivery destinations, and strict center-of-gravity (CG) balance requirements across all three axes. These complexities push computation times well beyond what is feasible for time-sensitive logistics operations.

Most existing exact CLP studies are limited to small instances—often fewer than 20 boxes—and rely on long solver runtimes (typically capped at 3600 seconds) to obtain feasible solutions [20, 53, 54]. Additionally, such models often exclude unloading obstacle constraints and simplify CG balance to one or two dimensions, making them incompatible with the full tri-objective formulation considered here. For these reasons, relying solely on exact methods like MIP would neither scale to realistic instances nor provide a fair basis for evaluating the proposed approach.

2.2.2 Heuristics-based solutions for CLP

Heuristics remain the primary practical approach for producing optimal or near-optimal load plans in the 3D-CLP, especially when multiple operational constraints must be satisfied. A wide range of heuristic strategies have been proposed, which [37] classify by packing logic and method type:

1. **Block-building:** The container is populated with cuboid blocks, each comprising multiple boxes. Examples include a tree-search method [67], a tabu-search variant [68], and a hybrid simulated annealing–tabu search method [69].
2. **Stack-building:** The container is filled with vertical stacks arranged on the floor to maximize space utilization. Examples include the heuristic of [70] and a genetic algorithm approach by [8].
3. **Guillotine-cutting:** A slicing tree model divides the container into sections via guillotine cuts, with each leaf corresponding to a single box. An example is the graph-search method of [71].
4. **Horizontal layer-building:** The container is loaded bottom-up with horizontal layers that maximize coverage of the load surface below [70, 72].
5. **Wall-building:** Vertical "walls" of boxes are packed side by side to fill the container volume [18, 27, 49, 73].

2.3 Metaheuristics-based solutions for CLP

Metaheuristics provide a practical means of producing high-quality solutions for complex optimization problems without guaranteeing optimality. Compared to exact methods, they offer reduced computational demands and greater flexibility for incorporating problem-specific constraints. Common metaheuristic paradigms include Genetic Algorithms (GA) [74], Particle Swarm Optimization (PSO) [75], Tabu Search (TS) [76], Artificial Bee Colony Optimization [77], Bat Algorithm [78], Symbiotic Organisms Search [79], Gradient Evolution [80], Cuckoo Algorithm [81], and others [82, 83].

For CLP with practical constraints, heuristics are often tailored to specific scenarios, while metaheuristics provide a general search framework that can be adapted across diverse instances. Numerous metaheuristic approaches have been applied to CLP, including GA [8, 11, 84], PSO [85–88], TS [89], Artificial Bee Colony Optimization [90], and Simulated Annealing [91–93]. Researchers have also enhanced base

algorithms to improve search performance, such as combining memetic algorithms with tabu search [94] or integrating K-nearest neighbor operators into the Cuckoo algorithm [95].

Among multi-objective metaheuristics, the Non-dominated Sorting Genetic Algorithm II (NSGA-II) is widely used for CLP and related packing problems due to its ability to generate diverse Pareto-optimal trade-offs. For example, [16] proposed a bi-objective CLP model maximizing space utilization and reducing unloading obstacles. Our work builds on this direction but extends it to a tri-objective framework by adding a center-of-gravity (CG) balance constraint, addressing load stability in addition to utilization and unloading efficiency. Furthermore, instead of penalizing unloading obstacles, we directly simulate the unloading process to count them, ensuring that the measure of unloading difficulty reflects real operational impact.

3 Methodology

To formalize the CLP with real-world constraints, we present a mixed-integer programming (MIP) formulation. This model captures the full 3D geometry, delivery order, cargo stability, and center-of-gravity (CG) balance within a unified mathematical framework. However, due to the problem's NP-hardness and the added complexity of multi-objective modeling—including unloading obstacles and full 3D CG control—this formulation becomes computationally intractable for realistic instance sizes (see also [20, 53, 54]). Consequently, while we do not solve this model directly, it serves as the formal foundation for our hybrid metaheuristic approach described in Section 4.

3.1 Problem definition

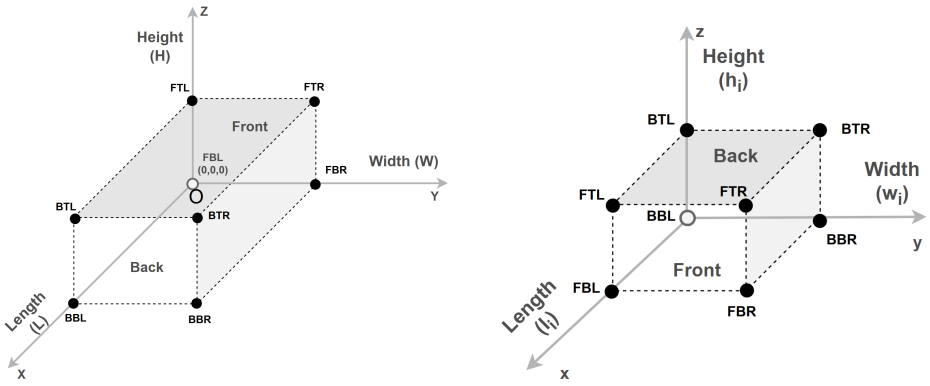
This study addresses the single-container loading problem (CLP) with a comprehensive set of practical constraints. We define the CLP here as a *tri-objective* problem with

the following goals: (i) maximize container volume utilization, (ii) minimize unnecessary box movements during unloading—termed *unloading obstacles* (ULOs)—which increase handling time and cost, and (iii) maintain the loaded cargo's center of gravity (CG) within specified bounds along each axis to ensure stability and transport safety.

Formally, let there be a set of N axis-aligned rectangular boxes, where box i has dimensions (l_i, w_i, h_i) and weight q_i for $i = 1, 2, \ldots, N$, and a rectangular container C of dimensions (L, W, H). The task is to select and place a subset of boxes in C such that:

- All boxes are placed without overlap and within container boundaries.
- The objectives (i)–(iii) are optimized jointly.
- All operational and regulatory constraints are satisfied, including geometry, orientation, maximum weight, load balancing, weight distribution, special cargo handling, cargo stability, stackability, multi-drop delivery, and item loading priority.

The study utilizes a Cartesian coordinate system. The container (Fig. 3a) has its origin $O(0, 0, 0)$ located at the front–bottom–left (FBL) corner, which is the reference point from which the box placement process begins. Fig. 3b shows a local frame affixed to box i, with its origin $O(0, 0, 0)$ at the back–bottom–left (BBL) corner of the box. In both illustrations, the corner points are identified and calculated as follows: **FTL:** Front top left $(x + l_i, y, z + h_i)$, **FTR:** Front top right $(x + l_i, y + w_i, z + h_i)$, **FBL:** Front bottom left $(x + l_i, y, z)$, **FBR:** Front bottom right $(x + l_i, y + w_i, z)$, **BTL:** Back top left $(x, y, z + h_i)$, **BTR:** Back top right $(x, y + w_i, z + h_i)$, **BBL:** Back bottom left (x, y, z), **BBR:** Back bottom right $(x, y + w_i, z)$.



(a) Container (b) Box

Fig. 3: Cartesian coordinate system used in this study

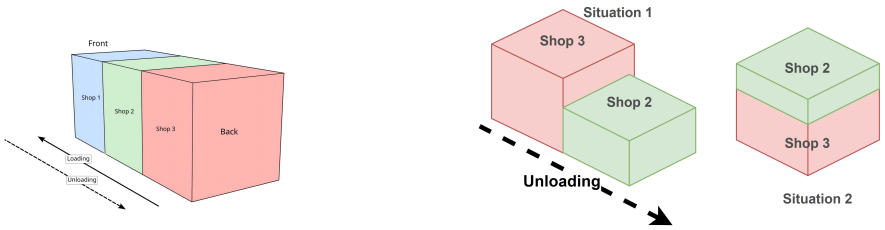
3.2 The constraints

This study incorporates the following practical constraints, some enforced strictly within the GA, others evaluated as soft objectives or implicitly satisfied through compact packing:

1. **Geometry (C1):** All boxes must be placed within the container boundaries without overlapping. This is enforced as a hard feasibility check during placement.
2. **Orientation (C2):** Boxes are restricted to six orthogonal orientations (Fig. 2). The default orientation (**a**) keeps (l_i, w_i, h_i) unchanged; other orientations are obtained by swapping dimensions as in Fig. 2(b)–(f).
3. **Multi-drop and unloading obstacles (C3):** Goods belong to multiple customers with a predefined delivery sequence. To help avoid burying later-delivery items behind earlier ones, we follow a *last-in, first-out* (LIFO) principle as a soft preference—ideally, boxes for later deliveries are placed deeper inside the container, and those for earlier deliveries are nearer to the door.

Strict LIFO and rigid physical partitioning are *not* enforced. The conceptual grouping shown in Fig. 4a, is used only to guide initial placement and to simulate the unloading process; boxes may be placed anywhere if it improves packing efficiency.

When a later-delivery box ends up on top of or in front of an earlier-delivery box, the placement is permitted, but each such case is counted as an *unloading obstacle* (ULO). A ULO occurs when a box for customer i blocks direct access to a box for customer j with $j < i$ (see Fig. 4b). The total ULO count is treated as a *soft objective* in the NSGA-II optimization, allowing trade-offs with space utilization and CG balance.



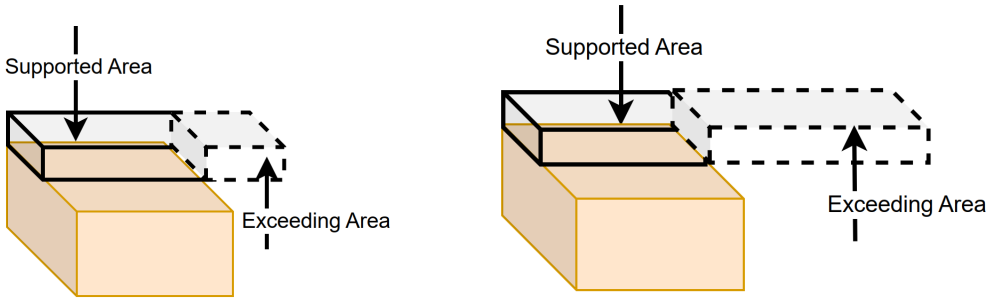
(a) Conceptual grouping of container space according to delivery sequence.

(b) Illustrative examples of ULO situations.

4. **Item stackability (C4):** Each item has a binary stackability value. A value of 0 (fragile) means no box can be placed above it; a value of 1 means the box is fully stackable with no explicit weight limit. Unlike studies that model load-bearing strength or maximum pressure [96], we adopt this simplified binary assumption to avoid additional complexity. Extending the model to graded load-bearing capacities is left as future work.
5. **Item allocation (C5):** Certain items must be loaded separately to prevent quality degradation or meet safety regulations—for example, food products must not be stored with chemicals or petroleum-based goods. Additionally, goods classified as dangerous under the ADR Agreement (European Agreement concerning the International Carriage of Dangerous Goods by Road) must be positioned near an

unloading point (typically the container door) to allow for rapid removal in case of an accident or emergency. ADR classes cover a wide range of hazardous materials, including explosives, gases, flammable liquids and solids, oxidizing substances, toxic or infectious materials, radioactive material, corrosives, and other hazardous goods. This placement rule for ADR cargo is treated as a strict requirement in our model.

6. **Cargo stability (C6):** We enforce vertical stability by requiring that each stackable box placed on other items has at least 75% of its base area supported. This check ensures that stacked cargo remains stable even under gravity or vehicle movement. Boxes placed directly on the floor are exempt. Horizontal (lateral) stability is not modeled explicitly, but is typically achieved due to the dense packing structure of the DBLF heuristic (Section 4.3).



(a) Accepted (b) Not accepted

Fig. 5: Illustrations of item stability via supported area

7. **Weight limit (C7):** The total loaded weight must not exceed the container or vehicle maximum capacity Q^{MAX}. This is enforced as a hard constraint.
8. **Center-of-gravity (CG) balance (C8$^{\mathbf{soft}}$):** The CG is evaluated along both the longitudinal (front–rear) and transverse (left–right) axes for the fully loaded container and after each delivery. Deviations beyond allowable bounds are not forbidden but are minimized as the third NSGA-II objective. This balances stability with space utilization and ULO minimization.

3.3 Mathematical modeling for CLP

This section presents a Mixed-Integer Programming (MIP) formulation for the container loading problem (CLP), including the practical constraints in Subsection 3.2, with three objectives: (i) maximize container volume utilization, equation (1); (ii) minimize unloading obstacles, equation (2); and (iii) minimize CG deviation, equation (3).

The geometric-constraint modeling follows [9, 97]. The container has dimensions $L \times W \times H$. For each box i, the dimensions are l_i, w_i, h_i, and the coordinate of a box is defined at its lower–back–left corner. We place the origin at the *front–bottom–left* corner $(0, 0, 0)$; $+x$ points toward the *rear (door)*, $+y$ to the right, and $+z$ upward. Length, width, and height align with x, y, z, respectively.

Sets and indices

I Set of boxes, indexed by i, k ($|I| = n$)

O Set of allowable orientations, indexed by o ($|O| = 6$)

P Set of relative positions between two boxes, indexed by p ($|P| = 6$)

T Set of priority levels, indexed by t

$D = \{x, y, z\}$ Set of coordinate axes

Parameters

L, W, H Container length, width, and height

l_i, w_i, h_i Length, width, and height of box i

m_i Weight of box i

$Q^{\max}$ Maximum container weight capacity

R_d Axis length along d ($R_x = L$, $R_y = W$, $R_z = H$)

$\sigma_i \in \{0, 1\}$ 1 if box i is stackable (requires support), 0 otherwise

θ Minimum support ratio required for vertical stability (default $\theta = 0.75$)

g_x, g_y, g_z Ideal CG coordinates (e.g., container midpoints)

$\alpha_x = \alpha_y = \alpha_z = 0.10$ Default CG deviation bound (10% of R_d on each axis)

M_d Big-M for axis d ($M_x = L$, $M_y = W$, $M_z = H$)

$P_t \subseteq I$ Subset of boxes with priority level t

$\pi(i)$ Delivery rank (smaller = earlier), e.g., $\pi(i) = t$ if $i \in P_t$

$s_{i,o,d}$ Size of box i along axis d when in orientation o

Decision variables

$u_i \in \{0,1\}$ 1 if box i is loaded, 0 otherwise

$r_{i,o} \in \{0,1\}$ 1 if box i is loaded in orientation o

$c_{i,d} \geq 0$ Coordinate of the lower–back–left corner of box i along axis d

$\phi_i \in \{0,1\}$ 1 if box i is placed directly on another box, 0 otherwise

$b_{i,k,p} \in \{0,1\}$ Relative position p active between *ordered* pairs (i,k)

$t_d^+, t_d^- \geq 0$ Positive/negative CG moment deviations along axis d

$z_d \geq 0$ Absolute CG deviation along axis d (linearized from $|t_d^+ - t_d^-|$)

$\mathrm{ULO}_{ik} \in \{0,1\}$ Unloading-obstacle indicator for precedence pair (earlier i, later k)

3.4 Objectives

$$\max Z_1 = \frac{\sum_{i \in I} (l_i w_i h_i)\, u_i}{LWH} \tag{1}$$

$$\min Z_2 = \sum_{i \in I} \sum_{k \in I\,:\,\pi(i) < \pi(k)} \mathrm{ULO}_{ik} \tag{2}$$

$$\min Z_3 = \frac{1}{|D|} \sum_{d \in D} z_d \tag{3}$$

3.5 Constraint modeling

1. **Box orientations:**

These constraints define how box orientations are modeled. Equation (4) ensures that if a box is selected for loading ($u_i = 1$), exactly one of its allowable orientations

must be chosen. Equations (5)–(7) compute the box's effective dimensions (l'_i, $w'i$, $h'i$) along the x, y, and z axes, respectively, based on the selected orientation. The values $s_{i,o,d}$ specify the length of box i along axis d in orientation o, and the binary variables $r_{i,o}$ activate the corresponding dimensions.

$$\sum_{o \in O} r_{i,o} = u_i \qquad \forall i \in I \tag{4}$$

$$l'_i = \sum_{o \in O} s_{i,o,x}\, r_{i,o} \qquad \forall i \in I \tag{5}$$

$$w'_i = \sum_{o \in O} s_{i,o,y}\, r_{i,o} \qquad \forall i \in I \tag{6}$$

$$h'_i = \sum_{o \in O} s_{i,o,z}\, r_{i,o} \qquad \forall i \in I \tag{7}$$

2. **Non-overlap among placed boxes:** To ensure that no two loaded boxes overlap inside the container, we define six binary variables $b_{i,k,p}$ for each pair of boxes (i, k), where each $p \in P = 1, \ldots, 6$ encodes a specific relative position: right, left, front, rear, below, or above. For example, $b_{i,k,1} = 1$ means box k is placed to the right of box i. Constraints (8)–(13) enforce these relationships using big-M formulations to maintain separation between boxes along the x, y, and z axes. At least one of these six conditions must hold for every pair of loaded boxes, enforced by constraint (14). If either box is not selected ($u_i = 0$ or $u_k = 0$), constraints (15)–(16) deactivate the position variables. Finally, constraints (17)–(22) ensure that the relative position encoding is symmetric and consistent between box pairs — for instance, if k is to the right of i, then i must be to the left of k. For any loaded pair, at least one relation in Fig.6 holds.

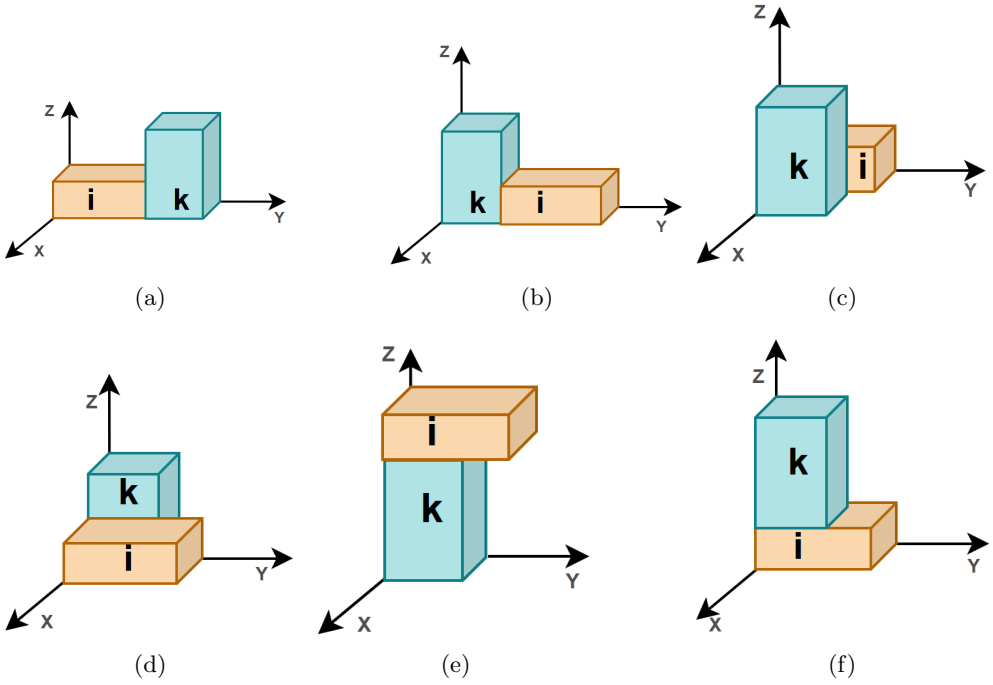


Fig. 6: Relative-position encoding $b_{i,k,p}$ for a pair of boxes i (orange) and k (blue). (a) $b_{i,k,1} = 1$ (k right of i), (b) $b_{i,k,2} = 1$ (k left of i), (c) $b_{i,k,3} = 1$ (k front of i), (d) $b_{i,k,4} = 1$ (k rear of i), (e) $b_{i,k,5} = 1$ (k below i), (f) $b_{i,k,6} = 1$ (k above i). Axes: $+x$ toward rear (door), $+y$ right, $+z$ up. See equations (8)–(13).

$$c_{i,y} + w'_i \le c_{k,y} + M_y(1 - b_{i,k,1}) \qquad \forall i \neq k \tag{8}$$

$$c_{k,y} + w'_k \le c_{i,y} + M_y(1 - b_{i,k,2}) \qquad \forall i \neq k \tag{9}$$

$$c_{i,x} + l'_i \le c_{k,x} + M_x(1 - b_{i,k,3}) \qquad \forall i \neq k \tag{10}$$

$$c_{k,x} + l'_k \le c_{i,x} + M_x(1 - b_{i,k,4}) \qquad \forall i \neq k \tag{11}$$

$$c_{i,z} + h'_i \le c_{k,z} + M_z(1 - b_{i,k,5}) \qquad \forall i \neq k \tag{12}$$

$$c_{k,z} + h'_k \le c_{i,z} + M_z(1 - b_{i,k,6}) \qquad \forall i \neq k \tag{13}$$

$$\sum_{p \in P} b_{i,k,p} \ge u_i + u_k - 1 \qquad \forall i < k \tag{14}$$

$$b_{i,k,p} \le u_i \qquad \forall i \neq k,\ p \in P \tag{15}$$

$$b_{i,k,p} \le u_k \qquad \forall i \neq k,\ p \in P \tag{16}$$

Symmetry of ordered binaries. For each unordered pair (i, k) the forward/reverse variables agree:

$$b_{i,k,1} = b_{k,i,2} \qquad \forall\, i < k \tag{17}$$

$$b_{i,k,2} = b_{k,i,1} \qquad \forall\, i < k \tag{18}$$

$$b_{i,k,3} = b_{k,i,4} \qquad \forall\, i < k \tag{19}$$

$$b_{i,k,4} = b_{k,i,3} \qquad \forall\, i < k \tag{20}$$

$$b_{i,k,5} = b_{k,i,6} \qquad \forall\, i < k \tag{21}$$

$$b_{i,k,6} = b_{k,i,5} \qquad \forall\, i < k \tag{22}$$

Numerical example (non-overlap constraint). Assume two loaded boxes, i and k, with the following effective coordinates and dimensions:

- Box i: origin $(c_{i,x}, c_{i,y}, c_{i,z}) = (0,\ 0,\ 0)$ and size $(l'_i, w'_i, h'_i) = (2,\ 2,\ 2)$
- Box k: origin $(c_{k,x}, c_{k,y}, c_{k,z}) = (0,\ 2.5,\ 0)$ and size $(l'_k, w'_k, h'_k) = (2,\ 2,\ 2)$

We test the relative placement to ensure no overlap.

Step 1: Determine relation direction.

Since box k starts at $y = 2.5$ and box i ends at $y = 2.0$, we conclude that k is strictly to the **right** of i:

$$c_{i,y} + w'_i = 0 + 2 = 2 \quad < \quad c_{k,y} = 2.5$$

So, we activate the corresponding binary variable:

$$b_{i,k,1} = 1 \quad \text{(box } k \text{ is right of } i\text{)}$$

Step 2: Validate using equation (8):

$$c_{i,y} + w'_i \leq c_{k,y} + M_y(1 - b_{i,k,1}) \Rightarrow 2 \leq 2.5 + M_y(0) = 2.5$$

All other $b_{i,k,p}$ are set to 0, and no other relative positions are active. The model therefore identifies a valid non-overlapping configuration using $b_{i,k,1}$.

Symmetry: The reverse relation must also hold via the symmetry constraints:

$$b_{k,i,2} = 1 \quad \text{(box } i \text{ is left of } k\text{)}$$

This confirms that the model captures directional non-overlap relationships consistently.

3. **Boundary constraints:** Each box must be placed fully within the container dimensions. The following constraints ensure that the box's extent along the x, y, and z axes does not exceed the container length L, width W, and height H, respectively:

$$c_{i,x} + l'_i \leq L \qquad \forall i \in I \tag{23}$$

$$c_{i,y} + w'_i \leq W \qquad \forall i \in I \tag{24}$$

$$c_{i,z} + h'_i \leq H \qquad \forall i \in I \tag{25}$$

Multi-drop unloading obstacles (ULO):

We define binary indicators ULO_{ik} for precedence pairs (i,k) with $\pi(i) < \pi(k)$. A later box k is an obstacle for earlier i if it sits deeper along x or stacked above i, which we encode via:

$$\mathrm{ULO}_{ik} \geq b_{i,k,3} \qquad \forall\, i,k \in I \text{ with } \pi(i) < \pi(k) \tag{26}$$

$$\mathrm{ULO}_{ik} \geq b_{i,k,5} \qquad \forall\, i,k \in I \text{ with } \pi(i) < \pi(k) \tag{27}$$

$$\mathrm{ULO}_{ik} \leq b_{i,k,3} + b_{i,k,5} \qquad \forall\, i,k \in I \text{ with } \pi(i) < \pi(k) \tag{28}$$

$$\mathrm{ULO}_{ik} \leq u_i, \quad \mathrm{ULO}_{ik} \leq u_k \qquad \forall\, i,k \in I \text{ with } \pi(i) < \pi(k) \tag{29}$$

Constraints (26)–(29) make ULO_{ik} an indicator (1 if k blocks i; 0 otherwise). We treat ULO *softly* by minimizing their sum:

$$\min Z_2 \;=\; \sum_{i \in I} \sum_{k \in I\,:\,\pi(i)<\pi(k)} \mathrm{ULO}_{ik} \tag{30}$$

So blockings are allowed but penalized in the objective.

4. **Cargo stability:** Each stackable box ($\sigma_i = 1$) must have at least a θ-fraction of its base supported by underlying boxes or the container floor. We use auxiliary logic (not shown in full here) to track the overlapping footprint and compute the support area.

$$\mathrm{SupportArea}_i \;\geq\; \theta \cdot (l'_i \cdot w'_i) \cdot \sigma_i \cdot \phi_i \qquad \forall i \in I \tag{31}$$

5. **Weight constraint:** The total weight of all boxes inside the container must not exceed the maximum weight capacity of the container.

$$\sum_{i \in I} m_i\, u_i \;\leq\; Q^{\max} \tag{32}$$

CG balance (linear, per axis):

We control the displacement of the loaded center of gravity (CG) from a geometric target point g_d along each axis $d \in D = \{x, y, z\}$. This target point corresponds to the midpoint of the container along that axis, i.e., $g_d = R_d/2$, where R_d is the full length of the container on axis d. As such, we express deviations in terms of percentage of $R_d/2$ — i.e., the container's half-length per axis — and enforce a bound of $\pm 10\%$ of that half-length, which aligns with industry norms for load balance.

Let $c_{i,d}$ be the *min* coordinate of box i along axis d. The centroid of box i on axis d is then given by $c_{i,d} + \frac{1}{2}s'_{i,d}$, where $s'_{i,x} = l'_i$, $s'_{i,y} = w'_i$, and $s'_{i,z} = h'_i$ depending on the orientation. Masses m_i are used if available; otherwise, the volume $l_i w_i h_i$ serves as a proxy.

We accumulate the signed, mass-weighted offset of the CG from g_d using surplus and deficit terms $t_d^+, t_d^- \geq 0$. The absolute deviation is then linearized with $z_d \geq 0$.

Link min-coordinates to selection and container bounds:

$$0 \;\leq\; c_{i,d} \;\leq\; \left(R_d - \sum_{o \in O} s_{i,o,d}\, r_{i,o}\right) u_i \quad \forall i \in I,\ d \in D \tag{33}$$

Signed mass-weighted offset (definition):

$$\sum_{i \in I} m_i \left(c_{i,d} + \tfrac{1}{2} \sum_{o \in O} s_{i,o,d}\, r_{i,o}\right) - g_d \sum_{i \in I} m_i u_i = t_d^+ - t_d^- \quad \forall\, d \in D \tag{34}$$

Absolute-value linearization:

$$z_d \geq\; t_d^+ - t_d^- \quad \forall d \in D \tag{35}$$

$$z_d \geq\; -(t_d^+ - t_d^-) \quad \forall d \in D \tag{36}$$

with $t_d^+, t_d^-, z_d \geq 0$.

Objective (soft CG): We minimize the average absolute deviation across axes:

$$\min Z_3 \;=\; \frac{1}{|D|} \sum_{d \in D} z_d \tag{37}$$

Optional hard bound (per axis): To limit deviations, we optionally impose per-axis caps proportional to the axis length R_d and the total loaded mass:

$$t_d^+ \leq \alpha_d R_d \sum_{i \in I} m_i u_i \qquad \forall d \in D \tag{38}$$

$$t_d^- \leq \alpha_d R_d \sum_{i \in I} m_i u_i \qquad \forall d \in D \tag{39}$$

where $\alpha_d = 0.10$ reflects the $\pm 10\%$ tolerance around the ideal CG per axis.

Remarks:

(i) Equations (34), (35)–(36) define a *soft* CG objective, i.e., deviation is penalized but not bounded.

(ii) Adding (38)–(39) creates a *soft+hard* regime, where the deviation is minimized and additionally capped. Our experiments show this can reduce packability.

(iii) If $c_{i,d}$ duplicates existing placement variables (e.g., x_i, y_i, z_i), one can directly substitute and omit (33) to reduce redundancy.

6. **Variable domains.** Binary variables are u_i, $r_{i,o}$, $b_{i,k,p}$, and ULO_{ik}; all others are continuous and nonnegative. We define $b_{i,k,p}$ for all ordered pairs $i \neq k$, while ULO_{ik} exists only for precedence pairs with $\pi(i) < \pi(k)$. Effective sizes l_i', w_i', h_i' are defined by (5)–(7).

$$u_i \in \{0, 1\} \qquad \forall\, i \in I \tag{40}$$

$$r_{i,o} \in \{0,1\} \qquad \forall\, i \in I,\ o \in O \tag{41}$$

$$b_{i,k,p} \in \{0,1\} \qquad \forall\, i \neq k,\ p \in P \tag{42}$$

$$\mathrm{ULO}_{ik} \in \{0,1\} \qquad \forall\, i,k \in I :\ \pi(i) < \pi(k) \tag{43}$$

$$c_{i,d} \geq 0 \qquad \forall\, i \in I,\ d \in D \tag{44}$$

$$l'_i,\ w'_i,\ h'_i \geq 0 \qquad \forall\, i \in I \tag{45}$$

$$t_d^+,\ t_d^-,\ z_d \geq 0 \qquad \forall\, d \in D \tag{46}$$

4 Hybrid Genetic Algorithm Approach for CLP

The three-dimensional container loading problem (CLP) is an NP-hard combinatorial optimization problem, and its difficulty increases sharply when practical constraints such as box orientation, delivery precedence, cargo stability, and load balance are incorporated simultaneously. While exact methods like Mixed-Integer Programming (MIP) can model these constraints, they become computationally infeasible for large-scale or real-time instances.

To tackle this, we develop a *hybrid genetic algorithm* (HGA) tailored for CLP with balance constraints. The proposed HGA combines two complementary components: (i) a fast placement strategy that constructs feasible solutions using the Deepest–Bottom–Left Fill (DBLF) heuristic, and (ii) a customized multi-objective evolutionary framework based on NSGA-II that explores a diverse set of trade-off solutions.

The decoder built on DBLF acts as a domain-specific packing engine: it greedily places boxes using geometric feasibility checks and updates a candidate point list to ensure compact packing (Section 4.3). While this ensures high volume utilization and feasibility, it does not directly account for load balance or unloading convenience. These trade-offs are addressed in the second stage via a multi-objective GA, where population members represent box loading sequences, and fitness is evaluated

across three objectives: volume utilization, unloading obstacles, and CG balance. This hybridization allows the search to combine greedy exploitation with global exploration, producing feasible load plans that satisfy practical constraints while improving Pareto-optimality.

The remainder of this section presents the core components of the HGA: the genotype representation (Section 4.1) and initialization logic (Section 4.2), the DBLF-based decoder (Section 4.3), and the genetic operators and selection mechanism of the NSGA-II framework (Section 4.5).

4.1 Individual representation in genotype space

The representation of individuals in the genotype space is crucial for the effectiveness of the hybrid genetic algorithm. In classical combinatorial problems (e.g., 0–1 knapsack or TSP), simple binary or order-based encodings suffice. However, the proposed CLP involves both placement order and orientation decisions, along with practical constraints such as support and delivery priorities. To accommodate these, we use a diploid representation consisting of two chromosomes.

- **Chromosome 1** encodes a permutation of the n boxes, defining their loading sequence.
- **Chromosome 2** encodes the orientation of each box, selecting one of the six allowable rotation modes.

These chromosomes are decoded by the DBLF-based placement heuristic (Section 4.3), which attempts to construct a feasible 3D layout from the genotype. During decoding, each box is placed according to the order and orientation encoded in the chromosomes, while enforcing model constraints such as bounds, non-overlap, stacking support, and multi-drop precedence.

An illustration of this diploid representation for a standard instance is shown in Fig. 7. Chromosome 1 specifies the sequence (e.g., box 6 is loaded first), while Chromosome 2 defines orientations (e.g., box 6 is placed in rotation mode 6).

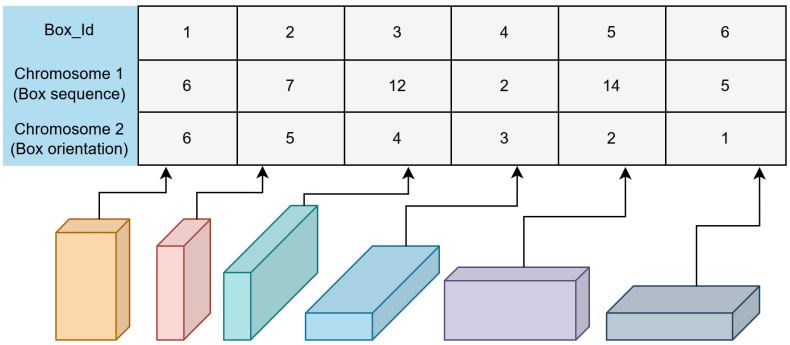


Fig. 7: Diploid representation of an individual in a population

For multi-drop delivery cases, the same structure is applied. As shown in Fig. 8, the permutation implicitly determines unloading priority. Boxes belonging to earlier delivery zones are expected to be unloaded first, so the decoder penalizes solutions that violate this through stacking or placement deeper in the container (see ULO-related constraints). The orientation chromosome operates identically.

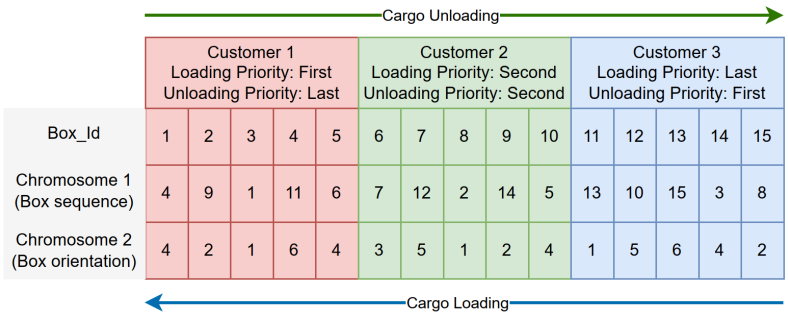


Fig. 8: Diploid representation of an individual for multi-drops deliveries

4.2 Population initialization

The initial population forms the starting point for the evolutionary process and strongly influences both convergence and diversity. In our HGA, the first generation is constructed to produce individuals that are structured yet varied, reflecting packing feasibility as well as practical realism.

Each individual is encoded with two chromosomes:

- A **box sequence chromosome** — a permutation of all boxes, constructed with grouping heuristics.
- A **rotation chromosome** — orientation values assigned to each box.

Box types (e.g., items with identical dimensions or the same destination) are first grouped together, after which the sequence chromosome is generated by randomly shuffling the order of groups and concatenating the items within each group. This ensures that boxes of the same type are packed adjacently. To further increase diversity, additional variants are included by sorting the full item list in descending order of volume, length, width, or height.

Orientation values are then randomly assigned from the set of allowed rotations, with the restriction that all items of the same type share the same rotation. This reflects practical handling consistency and reduces rotational noise. As a result, the initial population is diverse and well-structured with respect to type adjacency and consistent orientation.

Once both chromosomes are constructed, the resulting genotype is decoded into a load plan using the placement strategy (Section 4.3). The resulting 3D layout is first evaluated for geometric feasibility (non-overlap, boundary, and stability constraints). Center-of-gravity (CG) deviation is then quantified along each axis as described in Section 3.3 and treated as the third optimization objective. Layouts with larger

deviations are dominated by more balanced ones and thus receive lower rank in NSGA-II's selection process. This naturally penalizes unstable packings without discarding otherwise feasible solutions.

4.3 Box placement via DBLF strategy

Before placement begins, each individual's genotype (a permutation and orientation list) is translated into a sorted list of boxes using a multi-criteria ranking function which is described in Section 4.4. This ranking promotes boxes with higher customer and item priority, better stackability, and larger volume—guiding the placement order in a way that balances volume utilization and unloading ease.

Given an ordered list of rotated items (chromosome decoding), we sequentially place boxes into the container using a placement heuristic based on the Deepest–Bottom–Left Fill (DBLF) rule [98]. The strategy relies on maintaining a list of candidate placement points, which we extend, prune, and update dynamically during decoding. DBLF is widely adopted for its efficiency and compactness [16, 99, 100].

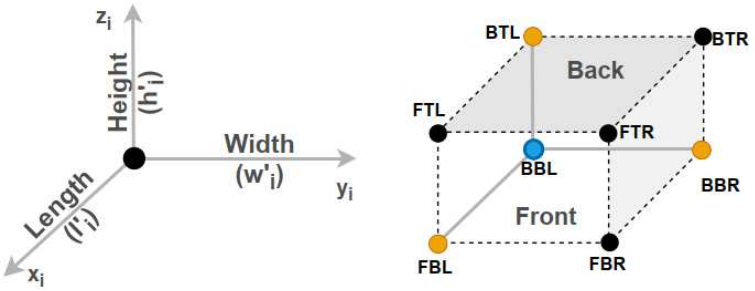


Fig. 9: Box-i with BBL origin and candidate placement points (blue: origin; light orange: labeled points $p \in \mathcal{P}$).

The container's origin is fixed at the front–bottom–left corner $(0, 0, 0)$, with $+x$ pointing toward the rear (door), $+y$ to the right, and $+z$ upward (see Fig. 3a). The placement process proceeds in several steps:

Step 1: candidate points:. We maintain an ordered list $\mathcal{P}$ of candidate placement points (for continuity with the figures, individual points shown as p correspond to elements of $\mathcal{P}$). Initially, $\mathcal{P} = \{(0, 0, 0)\}$ at the front–bottom–left corner (FBL). When a box-i is considered with an orientation selected by (4), its effective dimensions along the axes are l'_i, w'_i, h'_i from (5)–(7). We attempt to place box-i with its back–bottom–left (BBL) corner at a point $p \in \mathcal{P}$ (Fig. 9). The blue circle marks the box's origin. For reference, several corner names (FBL, BTL, BBR, etc.) appear in Fig. 9 details in Section 3.1.

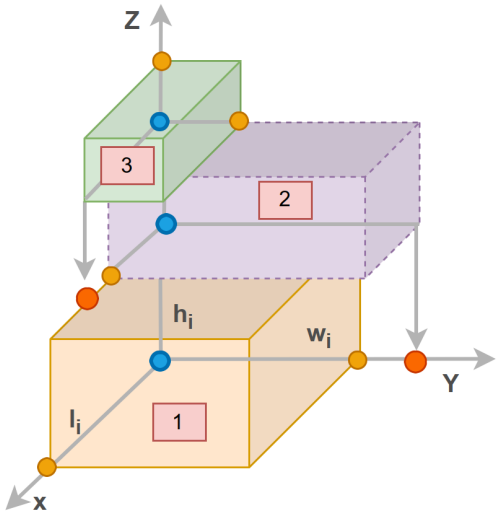


Fig. 10: Candidate points ($p \in \mathcal{P}$) after box placement.

Step 2: updating $\mathcal{P}$***.***

When the first item is placed in the container at the FBL, we generate up to three new points adjacent to its faces:

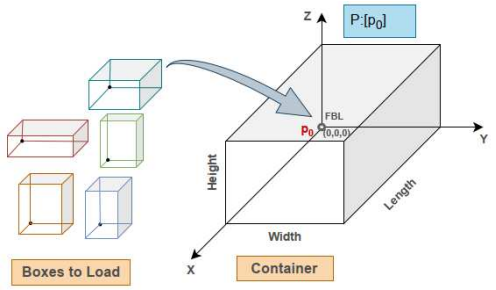
$$p^{(x)} = (x_i + l'_i,\, y_i,\, z_i), \quad p^{(y)} = (x_i,\, y_i + w'_i,\, z_i), \quad p^{(z)} = (x_i,\, y_i,\, z_i + h'_i)$$

If the item rests on the floor of the container ($z_i = 0$), these are exactly $(x_i + l'_i, y_i, 0)$, $(x_i, y_i + w'_i, 0)$ and (x_i, y_i, h'_i). If there are items underneath, $p^{(x)}$ and $p^{(y)}$ are *projected* onto the supporting surface directly below their (x, y) coordinates, bigger red circles in Fig.10 (standard in extreme-point methods). Points outside bounds are dropped immediately.

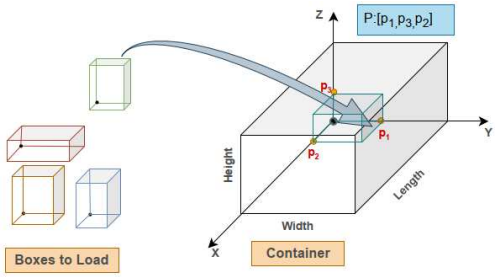
DBLF ordering and feasibility.

The list $\mathcal{P}$ is kept sorted lexicographically by *increasing* x (i.e., deeper in the container), then by *increasing* z (lower height), and finally by *increasing* y (closer to the left wall). Equivalently, the priority key is (x, z, y). For a candidate $p \in \mathcal{P}$ and item i, we test: (i) container bounds (23)–(25); (ii) pairwise non-overlap (8)–(16). If p is feasible, we place i, remove p, insert the new points $\{p^{(x)}, p^{(y)}, p^{(z)}\}$, filter duplicates and dominated points (greater than or equal in all three coordinates under the DBLF order), and resort $\mathcal{P}$. If the first point is infeasible, we test the next one, and so on, until either a feasible point is found or the list is exhausted.

Figures 11—13 walk through this process. In Fig. 11, the first item is placed at FBL, and three points are generated and sorted as (p_1, p_3, p_2) by the DBLF rule. The second item is then attempted at p_1; new points (p_4, p_5, p_6) are created, added, and the list becomes $(p_4, p_6, p_3, p_2, p_5)$. Fig. 12 shows infeasible trials: bounds violations (see (23)–(25)) and overlap (see (8)–(16)).

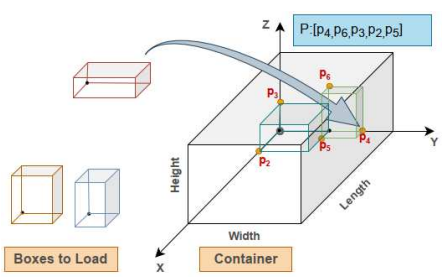


(a) Placement of first item at FBL

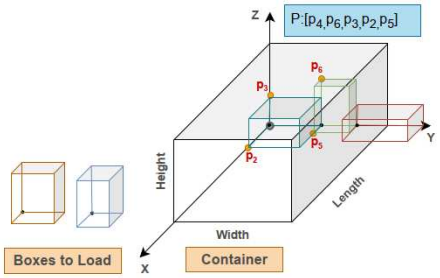


(b) Candidate set after first placement

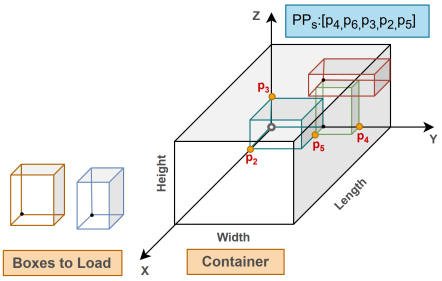
Fig. 11: DBLF progression: place, generate, sort.



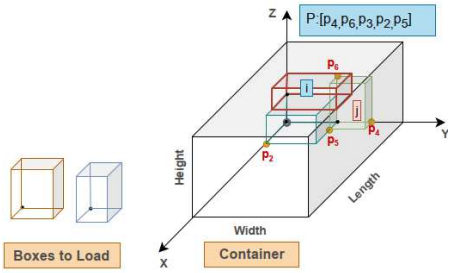
(a) Next candidate point



(b) Bounds violated; equations:(23)–(25)

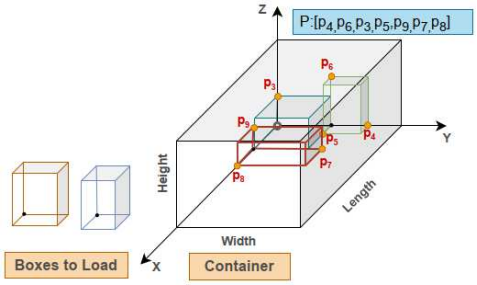


(c) Minimum support violated; equation:(31)

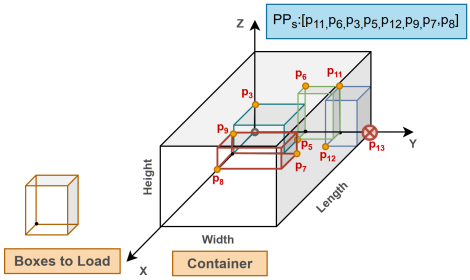


(d) Overlap; equations:(8)–(16)

Fig. 12: Feasibility checks at candidate points (illustrative violations).



(a) Feasible candidate for the current item



(b) Candidate point not generated due to violation of container bounds; equations:((23)–(25))

Fig. 13: Accepting/rejecting newly created candidates.

Fig.13 illustrates the outcome of the feasibility check on newly generated points. In subfigure (a), a valid point (e.g., p_4) results in a successful placement, while subfigure (b) highlights the rejection of candidate p_{13}, which is not created due to violation of

container boundary constraints (equations (23)–(25)). The updated placement list $\mathcal{P}$ is also shown for both cases.

Impact on balance:

As noted in [17], the DBLF strategy is effective for maximizing volume utilization and reducing computation time, but it introduces a systematic bias: boxes are preferentially placed deeper, lower, and closer to the left wall of the container. This concentrates mass in the front–bottom–left (FBL) region and can shift the center of gravity (CG) away from the container midpoint, particularly in the early stages of packing.

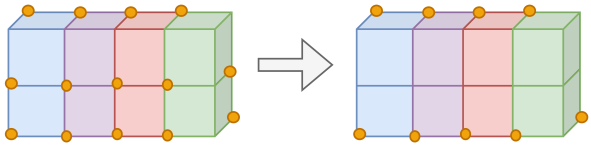
In our implementation, DBLF remains the primary ordering principle, but it is complemented by a balance-aware tie-breaking rule. For each candidate placement point (EP), we compute the relative moment differences between the left–right and front–rear halves of the container, using box volume as a proxy for weight. Among feasible EPs, those yielding smaller moment differences are preferred. This refinement mitigates the inherent FBL bias without undermining DBLF's efficiency in filling space.

Importantly, this local bias correction does not replace the formal CG modeling presented in Section 3.3. At the evolutionary level, overall CG deviation is still treated as a separate optimization objective, so that NSGA-II explicitly balances utilization, unloading obstacles, and stability. The two mechanisms therefore work in tandem: DBLF+moment-difference improves the quality of local placements, while the multi-objective GA applies global pressure toward balanced layouts.

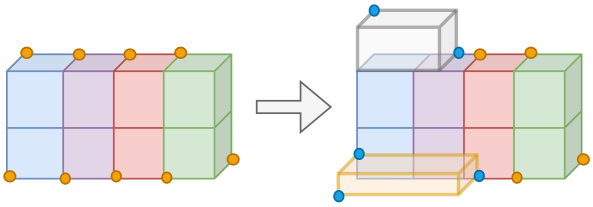
Pruning candidates:

To reduce decoding time, we remove candidates that are either dominated (by another point no worse in (x, z, y) under the DBLF order), outside bounds, or strictly covered by an already placed item or by a projected face.

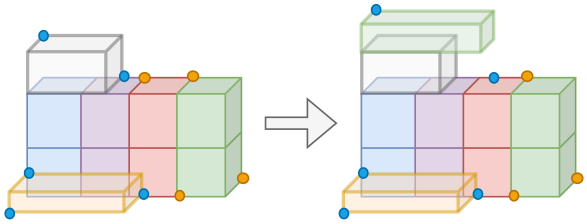
This placement strategy is called once per individual during genotype decoding in the NSGA-II cycle. See Algorithms 1 and 2 (Appendix A) for pseudocode.



(a) Initial candidate set including some infeasible points (e.g., insufficient clearance due to stacked boxes of similar orientation)



(b) Covered points removed (e.g., occluded by existing boxes)



(c) Covered points removed (e.g., occluded by box's projected faces)

Fig. 14: Sequential filtering of candidate placement points.

4.4 Box ranking

To balance maximizing volume utilization and minimizing unloading obstacles, we implemented a custom ranking function for the boxes in the population, which prioritizes the geometric and logistical attributes of the boxes before placement to efficiently meet both objectives. The ranking function prioritizes the boxes as follows:

To balance the conflicting goals of maximizing volume utilization and minimizing unloading obstacles and CG deviation, we implement a custom box ranking function that reorders each genotype before decoding via DBLF (Section 4.3).

1. **Customer Priority**: The primary ranking criterion is customer priority, which reflects the delivery order. This ensures boxes are loaded in sequence, following the $LIFO$ rule, to optimize unloading by reducing $ULOs$. It is crucial for satisfying constraint (C3).
2. **Stackability**: The next criterion, Stackability is used to resolve ties when boxes have the same customer and item priority. Stackable boxes can be safely placed on top of others without compromising constraint (C4). This influences the packing strategy by maximizing vertical space. Non-stackable boxes require special placement consideration and are ranked lower to avoid inefficiency and ensure transport stability.
3. **Volume**: Boxes are ranked by volume in descending order, with larger boxes given higher priority for placement to maximize space utilization.
4. **Base Area**: When two boxes have the same volume, their base area is used as the next tie-breaker. A larger base area indicates a bigger footprint, allowing for more efficient placement to ensure cargo stability and meet constraint (C4).
5. **Maximum Dimension**: Finally, if boxes have identical volume and base area, their maximum dimension (length or width) is used as a tie-breaker. Larger dimensions are preferred for better packing efficiency.

This multi-tiered ranking system prioritizes boxes based on the most important attributes, resulting in a more efficient packing solution. Applying these criteria ensured a balance between the two main objectives of this study, while considering logistical constraints. The next sorting criterion is applied only when boxes tie in the primary criterion.

This ranked list is then decoded to produce a 3D layout, which is evaluated across objectives by the NSGA-II engine (Appendix A, Algorithm 3).

4.5 Operators

The crossover and mutation operators are performed between customers within the same priority to prevent infeasibility because of priority constraint violation. For simplicity, we illustrate one customer's crossover and mutation operators.

1. **Crossover:** As order-based crossover has proven to suit permutation encoding, we have used a modified variant [101] suitable for our diploid chromosome representation. With this, two child individuals C_1 and C_2 are produced from two parent individuals P_1 and P_2. We start by selecting two random positions i and j, and the portion of the chromosome in $P_1(i)$ to $P_1(j)$ is copied onto $C_1(i)$ to $C_1(j)$, while $P_2(i)$ to $P_2(j)$ is copied onto $C_2(i)$ to $C_2(j)$. Afterwards, P_2 is swept circularly from $j+1$, filling the remaining values in C_1, and similarly, C_2 is filled with values from P_1. This process is illustrated in Fig. 15, where $i = 4$, $j = 7$ for customer 1, $i = 11$, $j = 7$ for customer 2, and $i = 18$, $j = 21$ for customer 3.
2. **Mutation:** The individuals generated through recombination undergo a two-step optimization (2-OPT) process using a mutation operator. First, we apply a 2-OPT mutation with a probability of p_{m1}. Two random positions, i and j, are selected within the individual, and the segment between these positions, $P(i) \dots P(j)$, is reversed. Following the 2-OPT mutation, we apply a random bit resetting mutation with a probability of p_{m2}. Each bit in the second chromosome has a probability p_{m2}

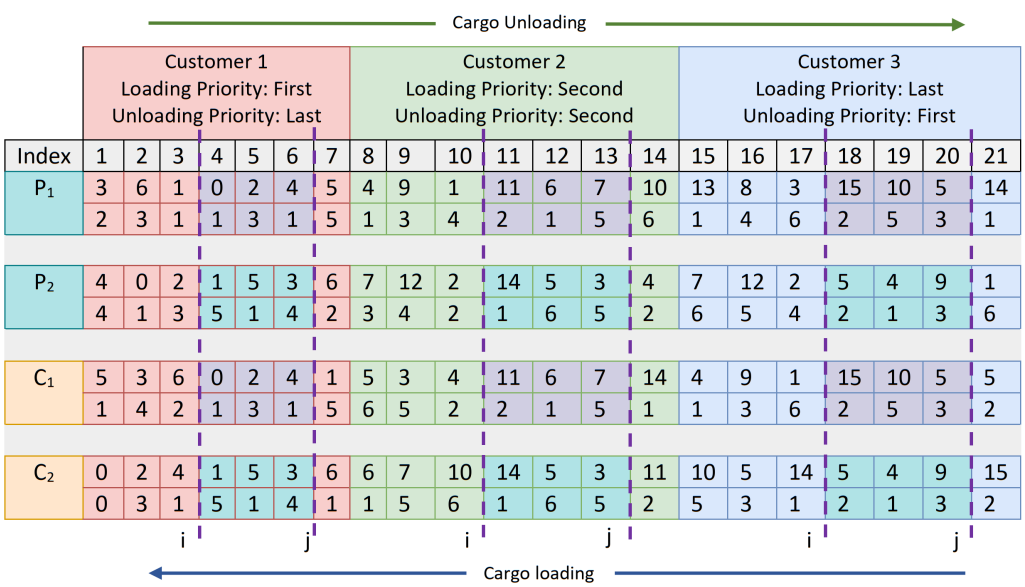


Fig. 15: Crossover operation

of being selected and is then randomly reset to a different rotation value according to its rotation category, as shown in Fig. 16.

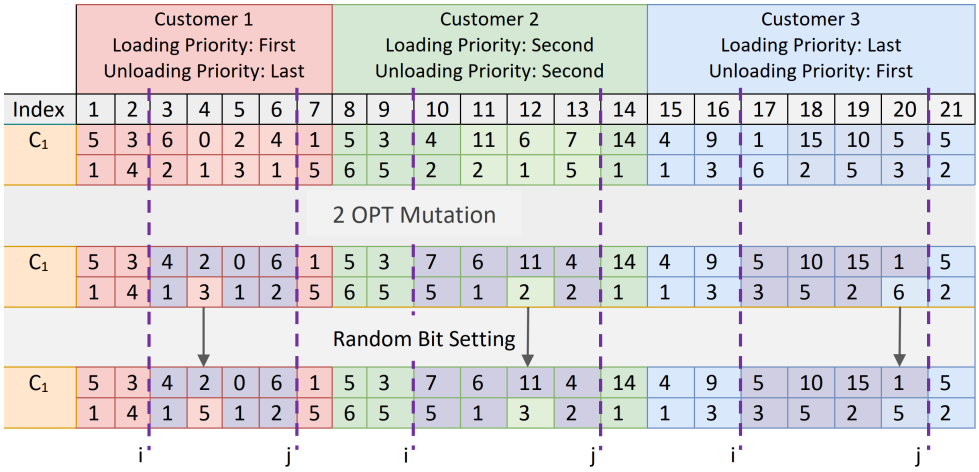


Fig. 16: Two-step mutation

3. **Selection:** The selection operator in GAs is essential to determine the individuals to contribute to the next generation. We employ binary tournament selection to choose individuals to produce offspring. This approach is simple and strikes a balance between exploration and exploitation. It increases the likelihood of selecting individuals with higher fitness levels, though it does not guarantee their selection. It helps to preserve genetic diversity within the population, preventing premature convergence and encouraging the exploration of diverse solutions to find optimal results.

Handling objective scale and conflict:

The proposed CLP formulation involves three conflicting objectives: maximizing volume utilization (Z_1), minimizing unloading obstacles (Z_2), and minimizing center-of-gravity deviation (Z_3). These objectives differ in nature and units: Z_1 is a percentage of container volume, Z_2 is a discrete count of blocking relations, and Z_3 is a normalized deviation across the three spatial axes.

To avoid domination by any single objective, we adopt the NSGA-II framework, which relies on Pareto dominance rather than explicit weighting or rescaling. Each objective is evaluated independently, and solutions are ranked by non-dominance with respect to the population. This allows natural trade-offs to emerge and preserves diversity along the approximate Pareto front [102, 103].

We also structured the objectives to be numerically well-behaved: Z_1 is normalized by container capacity, Z_3 averages deviations over axes relative to container dimensions, and Z_2 is bounded by the number of customer precedence pairs. In practice, all three objectives actively compete, and the algorithm consistently produces diverse trade-offs between utilization, accessibility, and balance.

5 Experimental Results

This section reports the computational experiments conducted to evaluate the proposed model and algorithms. All implementations were developed in Python 3.10, and tests were run on a workstation equipped with an Intel Core i7-12700H (12^{th} Gen, 2.30 GHz) processor and 32 GB of RAM. Results are presented in the following subsections.

5.1 Problem instances

The performance of the proposed model is evaluated using the benchmark instances of Bischoff and Ratcliff (BR). The primary evaluation metric is container volume utilization. To simulate multi-drop delivery conditions, we introduce modified BR instances that assign customer destinations and priorities, enabling the evaluation of unloading efficiency and overall load balance at the time of departure.

The BR dataset [70] is widely used in container loading research. It consists of seven classes, $BR1$ through $BR7$, each containing 100 boxes to be packed into a single container. The classes differ in the degree of box heterogeneity, ranging from only a few box types ($BR1$) to highly diverse sets with up to 20 types ($BR7$). The container dimensions are fixed at $L = 587$, $W = 233$, and $H = 220$ cm.

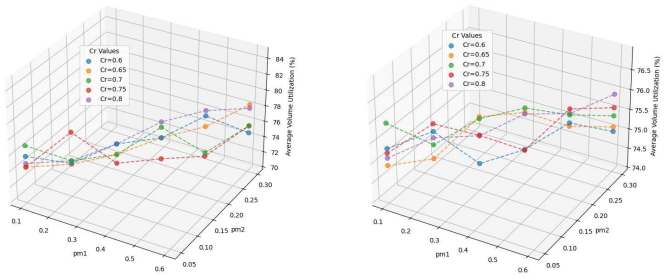
5.2 Parameter setting

The proposed multi-objective CLP is solved with NSGA-II, which requires tuning of several parameters. A preliminary design-of-experiments study tested the following ranges:

1. Generations (G): $[1, 200]$
2. Population size (pop_size): $[1, 50]$
3. Crossover rate (C_p): $[0, 0.8]$
4. Mutation rates: $pm_1 \in [0, 0.6]$, $pm_2 \in [0, pm_1/2]$

To capture the effect of box diversity, ten instances were randomly drawn from $BR1$ and $BR7$, representing the extremes of weak and strong heterogeneity. Performance across different combinations of C_p, pm_1, and pm_2 was compared. Fig.17 shows typical results for one instance of each class. A detailed summary is given in TableB1 (Appendix B).

During experimentation, the population size was fixed at 50, while crossover and mutation rates were systematically varied. The optimal settings derived from this analysis are reported in Table 1 and adopted in the NSGA-II implementation.



(a) $BR1$, Problem Instance 1

(b) $BR7$, Problem Instance 1

Fig. 17: Sensitivity of NSGA-II performance to parameter settings for one instance each of $BR1$ and $BR7$.

Table 1: Parameter configurations for $NSGA-II$

Parameters	Value
Generations	200
Population Size	50
Crossover Rate	0.8
Mutation rate	pm_1: 0.6
	pm_2: 0.3

5.3 Comparison of Benchmark Datasets

To validate the proposed $NSGA-II-$ based CLP model, this study conducted experiments on a basic dataset before applying the algorithm to the full problem scope. The analysis involves seven core BR datasets, with results compared to those from prior studies [56, 70, 104, 105].

Table 2: Utilization results with other methods.

Test case	**BR1**	**BR2**	**BR3**	**BR4**	**BR5**	**BR6**	**BR7**
Box Type	3	5	8	10	12	15	20
Bischoff et al.	81.76	81.70	82.98	82.60	82.76	81.50	80.51
Bischoff and Ratcliff	83.79	84.44	83.94	83.71	83.80	82.44	82.01
Gehring	85.80	87.26	88.10	88.04	87.86	87.85	87.68
Kuo (*NSGA-II*)	91.66	86.53	**89.56**	**91.99**	89.83	**93.30**	91.07
this study (*NSGA-II*)	**92.23**	**87.36**	88.57	89.53	**89.86**	90.85	**91.68**

The best results are listed in bold.

Bischoff et al. [70] and Bischoff and Ratcliff [104] employed heuristic methods that did not rely on parameter tuning, focusing instead on rule-based strategies. In contrast, Gehring [105] implemented a GA with specific configurations, including a population size of 50 chromosomes, 300 iterations, and crossover and mutation rates of 0.5 each. Kuo [56] utilized both $NSGA-II$ and Multi-objective Particle Swarm Optimization; however, this discussion focuses exclusively on NSGA-II. In Kuo's study, NSGA-II was configured with a population size of 200, a maximum of 200 generations, and crossover and mutation rates of 0.8 and 0.6, respectively.

Table 2 provides a comparative summary of the average container utilization achieved by each algorithm across the seven datasets. We applied the basic CLP constraints, Geometry, Orientation and Weight limit (sub-section 3.2). The results indicate that the $NSGA\text{-}II$ algorithm introduced in this study outperforms the

benchmark algorithms in several BR test instances. This performance improvement emphasizes the effectiveness and robustness of the proposed algorithm in solving container loading problems, even under varying operational conditions.

5.4 Experimental validation with modified BR data

The standard BR dataset was modified to align with the proposed multi-objective CLP formulation and to incorporate the practical constraints defined in Section 3.2. Each box was assigned to a delivery customer to reflect multi-drop scenarios, and stackability rules were enforced to capture vertical stability requirements. Because the original BR instances do not specify weights, synthetic weights were assigned in proportion to box volume, so that larger boxes were treated as heavier. This approximation is consistent with prior CLP studies and provides a realistic proxy for mass distribution, enabling evaluation of center-of-gravity (CG) deviation (Section 3.3). CG balance was thus modeled as a third optimization objective alongside container utilization and unloading obstacles (ULO). Table 3 reports the results for container utilization and ULO across the modified BR instances. NSGA-II consistently achieved high average

Table 3: Results for modified BR test instances: volume utilization (percent) and unloading obstacles.

Metric	**BR1**	**BR2**	**BR3**	**BR4**	**BR5**	**BR6**	**BR7**
Customers	3	5	8	10	12	15	20
NSGA-II (Avg.) [%]	83.74	74.70	84.90	86.57	83.45	88.95	84.88
NSGA-II (st.dev) [pp]	6.12	5.89	2.91	3.56	3.49	3.14	4.10
NSGA-II (Max) [%]	92.66	86.63	89.65	90.99	89.38	90.30	91.07
NSGA-II (Min) [%]	69.65	64.48	75.68	76.96	73.60	83.14	74.77
Avg. # of ULOs	5	8	10	11	13	12	14

Notes: Standard deviation (st.dev) is reported in absolute *percentage points (pp)*, i.e. the direct difference between percentages. ULOs = unloading obstacles.

utilization, typically in the range of 83–89%, with low standard deviations (3–6 percentage points), indicating stable performance across runs. The maximum utilizations approach 91% even in strongly heterogeneous instances (BR7). The number of customers per BR class (from 3 in BR1 to 20 in BR7) directly affects unloading complexity. As expected, the average number of ULOs increases with the number of customers, yet the values remain moderate (5–14), showing that the algorithm effectively manages delivery accessibility without severely sacrificing space efficiency.

Table 4: Average CG coordinates and deviations for modified BR instances. Container midpoints are (293.5, 116.5, 110.0) cm.

BR Class	**Avg. CG (cm) [x, y, z]**	**Avg. deviation (%) [x, y, z]**
BR1	(292.7, 115.8, 109.3)	(0.3, 0.6, 0.6)
BR2	(295.3, 117.7, 110.6)	(0.6, 1.0, 0.5)
BR3	(292.2, 116.9, 110.8)	(0.4, 0.3, 0.7)
BR4	(298.0, 118.8, 110.1)	(1.5, 1.9, 0.1)
BR5	(284.6, 116.2, 109.6)	(3.0, 0.2, 0.3)
BR6	(290.4, 114.3, 110.8)	(1.1, 1.9, 0.7)
BR7	(298.4, 114.5, 110.7)	(1.7, 1.7, 0.7)

Table 4 presents the average center-of-gravity (CG) positions and corresponding axis-wise deviations across modified BR instances, measured relative to the geometric midpoint of the container (293.5, 116.5, 110.0) cm. The results confirm that NSGA-II maintains a well-balanced load distribution in all tested cases.

For BR1–BR3, which represent instances with lower heterogeneity and fewer customers, the CG deviations remain extremely small—below 1% across all axes. In particular, horizontal deviations in the x and y directions average below 0.6%, while vertical deviations stay within 0.7%. For BR4–BR7, despite higher complexity and increased customer diversity, deviations remain operationally negligible: no axis

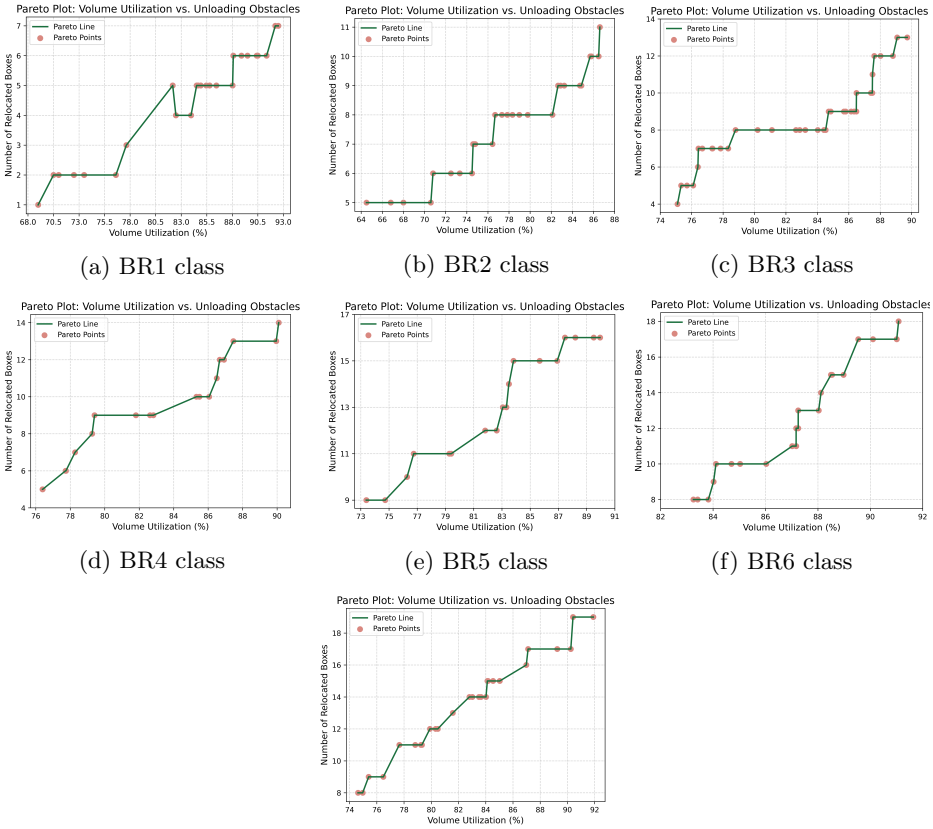
exceeds a 3% shift. The highest observed deviation is 3.0% along the x-axis in BR5, but the average deviations in all other cases remain below 2%.

This high degree of mass symmetry suggests that NSGA-II does not introduce systematic imbalance in any direction. There is no consistent drift toward a specific corner (e.g., front–left–bottom), and the CG remains close to the container's center in both low and high heterogeneity settings. Fig. 18 presents approximate Pareto fronts for the modified BR instances, showing the trade-off between volume utilization and unloading obstacles (ULOs). Because NSGA-II is a heuristic method, these fronts represent high-quality approximations of the true Pareto set rather than guaranteed optima.

The progression from $BR1$ to $BR7$ reflects increasing heterogeneity in box types and customer deliveries, which in turn broadens the trade-off surface. For simpler cases such as $BR1$ and $BR2$, the fronts are compact: high utilization can be achieved with only a few ULOs. In contrast, for more complex cases such as $BR6$ and $BR7$, the fronts are more dispersed, revealing a stronger conflict—solutions with very high utilization typically involve more unloading obstacles.

Across all classes, NSGA-II consistently identifies a diverse set of non-dominated layouts. This provides decision-makers with practical flexibility: in simpler instances they can favor utilization, while in more complex cases a compromise between packing efficiency and unloading convenience is more appropriate.

The smooth progression of these approximate Pareto fronts across problem classes shows that the framework scales naturally with problem difficulty. These results highlight the method's practical value: even without guaranteeing the true Pareto set, the algorithm produces high-quality trade-offs suitable for operational multi-drop loading scenarios.



(a) BR1 class (b) BR2 class (c) BR3 class

(d) BR4 class (e) BR5 class (f) BR6 class

(g) BR7 class

Fig. 18: Pareto analysis of unloading obstacles and volume utilization.

5.5 Validation with MIP

To benchmark the quality of heuristic solutions, we implemented a mixed-integer programming (MIP) model of the CLP (Section 3.3) using OR-Tools. As is common in 3D packing, a direct formulation with orientation, non-overlap, and boundary constraints quickly became intractable: even small BR instances yielded poor utilization (35–50%) with runtimes exceeding 3600 s.

Warm-starting with DBLF:

To improve tractability, we integrated a placement-point based warm start. Specifically, the deepest-bottom-left-fill (DBLF) heuristic was used to generate an initial set of extreme-point placements, which were injected into the solver using OR-Tools' `SetHint` functionality.[1] This mechanism allows the solver to start from a feasible or nearly feasible layout. Although the solver is not obliged to follow the hint, in practice, hints can accelerate convergence. Without warm-starting, the raw MIP often ran for over an hour and still produced poor layouts with only 35–50% utilization. With DBLF hints, utilization rose upto 60–80% on BR instances, and the solver typically converged well before the 300s cutoff.

The effectiveness of this warm-start stems from two complementary factors: (i) DBLF prioritizes the placement of larger boxes early, which aligns well with the MIP's objective of maximizing packed volume; and (ii) in our modified BR instances, customer delivery priorities were assigned inversely to box volume—smaller boxes correspond to earlier delivery. Consequently, DBLF tends to place large, late-delivery boxes deeper into the container, while leaving space near the door for smaller, early-delivery boxes. This structure not only improves volume utilization but also helps preserve the last-in–first-out (LIFO) unloading order, thereby implicitly supporting customer priorities even without hard constraints.

Unloading obstacles (ULO):

Delivery sequencing creates a clear tension between packing density and accessibility at later stops. We experimented with two formulations:

- *Hard constraints*: enforcing a strict last-in–first-out (LIFO) rule along the container length. This left the solver with almost no flexibility, and it typically placed only a handful of boxes before running out of feasible options.

[1] See OR-Tools documentation: SetHint.

- *Soft penalties*: relaxing the rule by allowing violations, but introducing slack variables that were penalized in the objective. This produced feasible layouts, but volume utilization fell by roughly 10–15% compared to runs without ULO.

These results showed that ULO cannot be imposed as a hard feasibility condition without destroying packing efficiency. Instead, it is better modeled as a separate optimization objective—as done in our NSGA-II approach—so the algorithm can balance accessibility against utilization.

Center-of-gravity (CG):

We also explored how to incorporate CG balance within the MIP model. Imposing a hard bound of ±10% deviation from the container midpoint along all three axes proved too restrictive: in most runs, the solver either returned infeasible solutions or packed only a small number of boxes.

To address this, we treated CG deviation as a soft objective. Specifically, we introduced a linearized formulation that minimizes the average mass-weighted deviation of the loaded center of gravity from the container's geometric center. No hard bound was imposed, although the model supports optional axis-wise caps if needed. This soft treatment aligns with our NSGA-II formulation, which also minimizes CG deviation as one of three competing objectives.

Comparison of Results:

To evaluate the quality of heuristic solutions produced by our hybrid GA, we compare them against solutions obtained from the MIP model for a representative instance in each BR class. Table 5 compares the performance of the MIP and NSGA-II approaches on a representative instance from each BR class. For all BR classes, NSGA-II consistently outperforms MIP in terms of volume utilization. The improvement ranges from a 10.0% gain for BR7 to nearly 20% for BR1 and BR2. These results confirm

the effectiveness of the heuristic in achieving dense packings, particularly under tight computational budgets.

Table 5: Comparison of MIP and NSGA-II on representative BR instances in terms of volume utilization and unloading obstacles (ULOs).

Metric	**BR1**	**BR2**	**BR3**	**BR4**	**BR5**	**BR6**	**BR7**
Customers	3	5	8	10	12	15	20
MIP Utilization [%]	75.60	73.12	77.50	75.34	79.41	78.87	80.97
NSGA-II Utilization [%]	90.66	87.63	88.65	90.99	89.38	91.30	89.07
Utilization Gap [%]	19.9	19.8	14.4	20.8	12.6	15.8	10.0
MIP ULOs [#]	10	14	20	13	40	50	59
NSGA-II ULOs [#]	5	8	10	11	13	12	14

The difference in unloading obstacles (ULOs) is even more striking. MIP produces significantly higher ULO counts, especially in the more heterogeneous BR classes. For instance, MIP generates 40–59 ULOs in BR5–BR7, compared to just 13–14 by NSGA-II. Even in simpler instances like BR1–BR3, the ULOs from MIP are roughly double those of NSGA-II. This contrast highlights a key limitation of the MIP model: without explicitly optimizing for unloading priority, the solver tends to prioritize volume packing at the expense of accessibility. Although warm-starting with DBLF (via `SetHint`) helped improve convergence and overall utilization, it had little to no effect on ULO reduction. NSGA-II, by contrast, treats ULOs as an explicit objective and successfully balances density with delivery sequencing.

These trends also reflect the increasing challenge of managing customer heterogeneity. As the number of customers rises from 3 in BR1 to 20 in BR7, ULOs grow sharply under MIP (from 10 to 59), but remain relatively stable under NSGA-II (5 to 14). This indicates that NSGA-II scales better with respect to operational constraints and customer complexity, making it a more practical solution for real-world multi-drop scenarios.

Table 6: Comparison of MIP and NSGA-II on representative BR instances: CG coordinates and deviations. Container midpoint is (293.5, 116.5, 110.0) cm.

Metric	**BR1**	**BR2**	**BR3**	**BR4**
MIP CG (cm) [x,y,z]	(293.0, 115.9, 108.4)	(295.7, 117.4, 111.1)	(292.5, 116.8, 111.5)	(296.4, 117.8, 110.2)
NSGA-II CG (cm) [x,y,z]	(291.1, 115.1, 109.0)	(298.4, 118.8, 110.5)	(289.6, 117.3, 110.7)	(295.3, 117.2, 110.1)
MIP CG deviation [%]	(0.2, 0.5, 1.5)	(0.8, 0.8, 1.0)	(0.3, 0.2, 1.3)	(1.0, 1.1, 0.2)
NSGA-II CG deviation [%]	(0.8, 1.2, 0.9)	(1.7, 1.9, 0.5)	(1.3, 0.6, 0.7)	(0.6, 0.6, 0.1)
Metric	**BR5**	**BR6**	**BR7**	
MIP CG (cm) [x,y,z]	(288.2, 116.4, 109.5)	(291.6, 115.5, 110.6)	(296.7, 115.4, 111.8)	
NSGA-II CG (cm) [x,y,z]	(295.4, 115.9, 111.1)	(292.2, 115.8, 110.5)	(289.7, 116.4, 109.7)	
MIP CG deviation [%]	(1.8, 0.1, 0.5)	(0.7, 0.9, 0.6)	(1.1, 0.9, 1.7)	
NSGA-II CG deviation [%]	(0.7, 0.5, 1.0)	(0.4, 0.6, 0.5)	(1.3, 0.1, 0.3)	

CG balance comparison:

Table 6 compares the CG coordinates and deviation values produced by MIP and NSGA-II across all BR classes. The deviations are calculated per axis as absolute differences from the container midpoint (293.5, 116.5, 110.0) cm, expressed as a percentage of axis length.

Across all instances, both methods maintain CG deviations within the operational threshold of ±10% per axis. However, NSGA-II generally achieves tighter CG control, especially in the vertical (z) axis where precise balance is crucial for safe handling. For example, in BR5 and BR7, NSGA-II reports vertical deviations of 1.0% and 0.3%, respectively, while MIP shows 0.5% and 1.7%. Similarly, in the horizontal (x and y) directions, NSGA-II often performs slightly better or at least comparably to MIP.

It is worth noting that the MIP results benefit from DBLF-based warm-starting via `SetHint`, which naturally places heavier (larger) boxes toward the container base and center. This heuristic indirectly improves CG distribution, even though the solver does not explicitly optimize for CG in its objective. Nevertheless, because NSGA-II models CG deviation as a standalone minimization objective, it can fine-tune placements more effectively to achieve balanced mass distributions.

These differences, though numerically small, may have practical implications in real-world loading scenarios governed by safety thresholds and stability regulations. The consistent CG performance of NSGA-II reinforces the value of multiobjective modeling when addressing stability-sensitive constraints.

6 Conclusion

This study tackled a complex and practically relevant variant of the Container Loading Problem (CLP), where space efficiency must be balanced with delivery accessibility and cargo stability. We proposed a tri-objective formulation that jointly optimizes

volume utilization, minimizes unloading obstacles (ULOs), and controls the center of gravity (CG) to ensure safe and stable packing.

To address this, we developed a hybrid multi-objective optimization framework based on NSGA-II, with a placement decoder that combines an extreme-point (EP) strategy and the deepest-bottom-left-fill (DBLF) heuristic. EPs define the candidate positions for the next box, while DBLF ranking favors compact packing and high utilization. Since DBLF tends to bias the center of gravity toward the front-bottom-left corner, EPs are adaptively re-ranked in reverse DBLF order whenever balance requirements would otherwise be violated. Beyond volume utilization, the framework incorporates a broad set of real-world constraints, including vertical support, stackability, delivery priorities, weight limits, and geometric feasibility, ensuring that generated load plans remain both physically stable and operationally practical under tight spatial restrictions.

Our experimental results on modified BR benchmark instances demonstrate that the proposed method consistently produces high-quality, balanced, and accessible layouts across varying degrees of box heterogeneity. CG deviations remain well within operational limits, and ULO counts are substantially lower than those produced by exact MIP models. Comparative analysis with MIP-based solutions showed that while exact methods are useful for benchmarking small instances, our heuristic approach scales better, achieves higher utilization, and handles multi-drop constraints more effectively. We also experimented with DBLF-based warm-starting in the MIP models, which in some cases narrowed the gap by improving utilization and convergence speed. However, this effect was not consistent across objectives; for example, warm-starting did not lead to better solutions for minimizing unloading obstacles (ULOs), highlighting both the potential and the limitations of using heuristic guidance in exact solvers.

The approach supports decision-making by generating diverse Pareto-optimal load plans with 3D visualization, allowing warehouse operators to select solutions that best align with their operational priorities—whether it be delivery orders, cargo stability, or packing density.

Future research could extend the CG balance objective beyond the departure state to explicitly consider stability throughout the delivery route. In multi-drop scenarios, every unloading action alters the mass distribution, and a plan that is balanced at departure may become unstable after several stops. Addressing this challenge would require methods that anticipate the evolving CG trajectory over the entire delivery sequence, ensuring that the container remains balanced not just initially but at each intermediate state. This raises important questions about the operational consequences of CG violations during transit, such as safety risks, regulatory compliance, or unloading difficulties at customer sites. It also opens avenues for corrective strategies, including selective re-stacking, shifting of boxes, or the use of flexible loading zones that allow adjustments during the route. Developing algorithms that can model and optimize these dynamic aspects of balance would significantly enhance the realism of CLP studies and bring the framework closer to practical deployment in modern logistics.

Acknowledgements. The research leading to these results has received funding from the European Union's Horizon Europe research and innovation program under grant agreement No 101057294, corresponding to the project titled AIDEAS (AI-Driven industrial Equipment product lifecycle boosting Agility, Sustainability and resilience).

Declarations

- **Funding**

The research leading to these results has received funding from the European

Unions Horizon Europe research and innovation program under grant agreement n 101057294, corresponding to the project titled AIDEAS (AI Driven industrial Equipment product life cycle boosting Agility, Sustainability, and resilience).

- **Author contribution**

Conceptualization, JLML, JAPG and ME; methodology, JLML, JAPG and ME; formal analysis, ME, JLML, JAPG and WMM; investigation, ME; writing original draft preparation, ME; writing review and editing, ME and SOA; writing computer codes, ME; running computer programs, ME; results analysis, ME; supervision, JLML and JAPG; funding acquisition, JLML. All authors have read and agreed to the published version of the manuscript.

- **Data availability**

The data used in this research work is publicly available at https://people.brunel.ac.uk/~mastjjb/jeb/orlib/thpackinfo.html.

- **Informed consent**

All authors consent to participating in this research work. In addition, all the authors also consent to the publishing of this research work

Appendix A Algorithms

Algorithm 1 Placement point generation and update of $\mathcal{P}$ list

Input: List L of already placed boxes
Input: Candidate list $\mathcal{P}$ of placement points
Input: Box i placed at (x_i, y_i, z_i) with size (l'_i, w'_i, h'_i)
Initialize $H_x^{\max} \leftarrow 0$, $H_y^{\max} \leftarrow 0$
if $z_i = 0$ **then** ▷ Placed on container floor
    Insert $(x_i + l'_i, y_i, 0)$ and $(x_i, y_i + w'_i, 0)$ into $\mathcal{P}$
else
    for each box $j \in L$ **do** ▷ Project onto support below $(x_i,\ y_i)$
        if $x_i + l'_i \geq x_j$ **then**
            if $z_j + h'_j \leq z_i$ **and** $z_j + h'_j > H_x^{\max}$ **then**
                $H_x^{\max} \leftarrow z_j + h'_j$
        else if $x_j + l'_j \geq x_i$ **then**
            $H_x^{\max} \leftarrow z_j$
        if $y_i + w'_i \geq y_j$ **then**
            if $z_j + h'_j \leq z_i$ **and** $z_j + h'_j > H_y^{\max}$ **then**
                $H_y^{\max} \leftarrow z_j + h'_j$
        else if $y_j + w'_j \geq y_i$ **then**
            $H_y^{\max} \leftarrow z_j$
        if $H_x^{\max} = z_i$ **and** $H_y^{\max} = z_i$ **then**
            break
    Insert $(x_i + l'_i, y_i, H_x^{\max})$ and $(x_i, y_i + w'_i, H_y^{\max})$ into $\mathcal{P}$
Insert $(x_i, y_i, z_i + h'_i)$ into $\mathcal{P}$
if $x_i + l'_i \geq L$ **then**
    Remove $(x_i + l'_i, y_i, z_i)$ from $\mathcal{P}$
Remove (x_i, y_i, z_i) from $\mathcal{P}$

Algorithm 2 Deepest–Bottom–Left Fill (DBLF) decoder

Input: List B of boxes to be packed (ordered and rotated)
Input: Candidate list $\mathcal{P}$ of placement points
for each box $i \in B$ **do**
    $packed \leftarrow$ **false**
    Compute effective dimensions (l'_i, w'_i, h'_i) based on rotation (see (5)–(7))
    Sort $\mathcal{P}$ lexicographically by (x, z, y) ▷ DBLF priority
    for each point $p \in \mathcal{P}$ **do**
        if box i is feasible at p (bounds, overlap, stability) **then**
            $packed \leftarrow$ **true**
            Place box i at p
            Remove p from $\mathcal{P}$ and update via Algorithm 1
            break

Algorithm 3 Multi-objective NSGA-II with DBLF decoding

Input: Population size N, number of generations G, crossover probability C_p, mutation probabilities p_{m1}, p_{m2}
Output: Non-dominated set S of high-quality loading plans

Initialization:
Generate initial population $P = \{p_1, \ldots, p_N\}$ using heuristic initialization (Section 4.2)
Apply box ranking to each genotype (Section 4.4)
Decode each individual using DBLF strategy (Algorithm 2)
Evaluate objectives Z_1, Z_2, Z_3 (Section 3.3)
Perform non-dominated sorting to assign Pareto fronts and crowding distance
for generation $g = 1$ to G **do**
Selection and Crossover:
    for pairwise mating until N offspring **do**
        Select two parents via binary tournament (based on rank and crowding distance)
        Apply order-based crossover on Chromosome 1 (box order) with probability C_p
        Apply partial-mapped crossover on Chromosome 2 (rotations) with probability C_p
Mutation:
    for each offspring p **do**
        if rand() $\leq p_{m1}$ **then**
            Apply 2-OPT mutation on box order
        if rand() $\leq p_{m2}$ **then**
            Randomly reset rotation genes (see Fig. 16)
Evaluation:
    Apply box ranking (Section 4.4)
    Decode all offspring via DBLF (Algorithm 2)
    Evaluate objectives and constraints
    Merge parent and offspring populations
    Perform non-dominated sorting, assign crowding distances
    Select the best N individuals for the next generation
Return Pareto front $S = \{s \in P : \text{rank}(s) = 1\}$

Appendix B Tables

Table B1: Variation in NSGA-II performance with parameter changes

Crossover rate C_p	Second mutation rate pm_1	Second mutation rate pm_2	Performance indicator	Volume utilization	Avg.# of ULOs
BR1 Class, Each problem instance has 3 kind of boxes					
0.6	[0.1,0.2,0.3,0.4,0.5,0.6]	[0.5,0.1,0.15,0.2,0.25,0.3]	Average	0.773	
			st.dev	0.043	5
0.65	[0.1,0.2,0.3,0.4,0.5,0.6]	[0.5,0.1,0.15,0.2,0.25,0.3]	Average	0.764	
			st.dev	0.045	5
0.7	[0.1,0.2,0.3,0.4,0.5,0.6]	[0.5,0.1,0.15,0.2,0.25,0.3]	Average	0.745	
			st.dev	0.042	7
0.75	[0.1,0.2,0.3,0.4,0.5,0.6]	[0.5,0.1,0.15,0.2,0.25,0.3]	Average	0.756	
			st.dev	0.047	9
0.8	[0.1,0.2,0.3,0.4,0.5,0.6]	[0.5,0.1,0.15,0.2,0.25,0.3]	Average	0.779	
			st.dev	0.040	3
BR7 Class, Each problem instance has 20 kinds of boxes					
0.6	[0.1,0.2,0.3,0.4,0.5,0.6]	[0.5,0.1,0.15,0.2,0.25,0.3]	Average	0.743	
			st.dev	0.063	17

Table B1 continued from previous page

0.65	[0.1,0.2,0.3,0.4,0.5,0.6]	[0.5,0.1,0.15,0.2,0.25,0.3]	Average	0.737	
			st.dev	0.064	10
0.7	[0.1,0.2,0.3,0.4,0.5,0.6]	[0.5,0.1,0.15,0.2,0.25,0.3]	Average	0.751	
			st.dev	0.062	15
0.75	[0.1,0.2,0.3,0.4,0.5,0.6]	[0.5,0.1,0.15,0.2,0.25,0.3]	Average	0.716	
			st.dev	0.064	14
0.8	[0.1,0.2,0.3,0.4,0.5,0.6]	[0.5,0.1,0.15,0.2,0.25,0.3]	Average	0.769	
			st.dev	0.067	8

Note: *st. dev* stands for *standard deviation*, *Avg* stands for *Average* ,*ULOs* stands for *unloading obstacles*

References

[1] Sridhar, R., Chandrasekaran, M., Sriramya, C., Page, T.: Optimization of heterogeneous Bin packing using adaptive genetic algorithm. IOP Conference Series: Materials Science and Engineering **183**(1), 012026 (2017) https://doi.org/10.1088/1757-899X/183/1/012026 . Publisher: IOP Publishing. Accessed 2023-10-17

[2] Salkin, H.M., De Kluyver, C.A.: The knapsack problem: a survey. Naval Research Logistics Quarterly **22**(1), 127–144 (1975)

[3] Zhao, X., Bennell, J.A., Bektaş, T., Dowsland, K.: A comparative review of 3d container loading algorithms. International Transactions in Operational Research **23**(1-2), 287–320 (2016)

[4] González, Y., Miranda, G., León, C.: A multi-level filling heuristic for the multi-objective container loading problem. In: International Joint Conference SOCO'13-CISIS'13-ICEUTE'13: Salamanca, Spain, September 11th-13th, 2013 Proceedings, pp. 11–20 (2014). Springer

[5] Toffolo, T.A., Esprit, E., Wauters, T., Berghe, G.V.: A two-dimensional heuristic decomposition approach to a three-dimensional multiple container loading problem. European Journal of Operational Research **257**(2), 526–538 (2017)

[6] Dereli, T., Daş, G.S.: Development of a decision support system for solving container loading problems. Transport **25**(2), 138–147 (2010)

[7] Araya, I., Moyano, M., Sanchez, C.: A beam search algorithm for the biobjective container loading problem. European Journal of Operational Research **286**(2), 417–431 (2020)

[8] Bortfeldt, A., Gehring, H.: A hybrid genetic algorithm for the container loading problem. European Journal of Operational Research **131**(1), 143–161 (2001)

[9] Huang, W., He, K.: A caving degree approach for the single container loading problem. European Journal of Operational Research **196**(1), 93–101 (2009)

[10] Lai, X., Hao, J.-K., Yue, D.: Two-stage solution-based tabu search for the multi-demand multidimensional knapsack problem. European Journal of Operational Research **274**(1), 35–48 (2019)

[11] Miranda, G., Lančinskas, A., González, Y.: Single and multi-objective genetic algorithms for the container loading problem. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 291–292 (2017)

[12] Zheng, J.-N., Chien, C.-F., Gen, M.: Multi-objective multi-population biased random-key genetic algorithm for the 3-d container loading problem. Computers & Industrial Engineering **89**, 80–87 (2015)

[13] Eley, M.: A bottleneck assignment approach to the multiple container loading problem. OR Spectrum **25**, 45–60 (2003)

[14] Zhu, W., Lim, A.: A new iterative-doubling greedy–lookahead algorithm for the single container loading problem. European Journal of Operational Research **222**(3), 408–417 (2012)

[15] Pollaris, H., Braekers, K., Caris, A., Janssens, G.K., Limbourg, S.: Vehicle routing problems with loading constraints: state-of-the-art and future directions. Or Spectrum **37**, 297–330 (2015)

[16] Gajda, M., Trivella, A., Mansini, R., Pisinger, D.: An optimization approach for a complex real-life container loading problem. Omega **107**, 102559 (2022)

https://doi.org/10.1016/j.omega.2021.102559 . Accessed 2023-10-17

[17] Karabulut, K., İnceoğlu, M.M.: A Hybrid Genetic Algorithm for Packing in 3D with Deepest Bottom Left with Fill Method. In: Yakhno, T. (ed.) Advances in Information Systems. Lecture Notes in Computer Science, pp. 441–450. Springer, Berlin, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30198-1_45

[18] George, J.A., Robinson, D.F.: A heuristic for packing boxes into a container. Computers & Operations Research **7**(3), 147–156 (1980) https://doi.org/10.1016/0305-0548(80)90001-5 . Accessed 2023-10-26

[19] Lim, A., Ying, W.: A new method for the three dimensional container packing problem. In: IJCAI, pp. 342–350 (2001). Citeseer

[20] Kurpel, D.V., Scarpin, C.T., Junior, J.E.P., Schenekemberg, C.M., Coelho, L.C.: The exact solutions of several types of container loading problems. European Journal of Operational Research **284**(1), 87–107 (2020)

[21] Bortfeldt, A., Wäscher, G.: Constraints in container loading – A state-of-the-art review. European Journal of Operational Research **229**(1), 1–20 (2013) https://doi.org/10.1016/j.ejor.2012.12.006 . Accessed 2024-04-18

[22] Ramos, A.G., Silva, E., Oliveira, J.F.: A new load balance methodology for container loading problem in road transportation. European Journal of Operational Research **266**(3), 1140–1152 (2018)

[23] Wang, L., Guo, S., Chen, S., Zhu, W., Lim, A.: Two natural heuristics for 3d packing with practical loading constraints. In: PRICAI 2010: Trends in Artificial Intelligence: 11th Pacific Rim International Conference on Artificial Intelligence, Daegu, Korea, August 30–September 2, 2010. Proceedings 11, pp. 256–267 (2010). Springer

[24] Baldi, M.M., Perboli, G., Tadei, R.: The three-dimensional knapsack problem with balancing constraints. Applied Mathematics and Computation **218**(19), 9802–9818 (2012)

[25] Trivella, A., Pisinger, D.: The load-balanced multi-dimensional bin-packing problem. Computers & Operations Research **74**, 152–164 (2016)

[26] Lim, A., Ma, H., Qiu, C., Zhu, W.: The single container loading problem with axle weight constraints. International Journal of Production Economics **144**(1), 358–369 (2013)

[27] Bortfeldt, A., Gehring, H.: A hybrid genetic algorithm for the container loading problem. European Journal of Operational Research **131**(1), 143–161 (2001) https://doi.org/10.1016/S0377-2217(00)00055-2 . Accessed 2024-01-24

[28] Moon, I., Nguyen, T.V.L.: Container packing problem with balance constraints. OR spectrum **36**(4), 837–878 (2014)

[29] Egeblad, J., Pisinger, D.: Heuristic approaches for the two-and three-dimensional knapsack packing problem. Computers & Operations Research **36**(4), 1026–1049 (2009)

[30] Bortfeldt, A., Wäscher, G.: Constraints in container loading–a state-of-the-art review. European Journal of Operational Research **229**(1), 1–20 (2013)

[31] Colombi, M., Corberán, Á., Mansini, R., Plana, I., Sanchis, J.M.: The directed profitable rural postman problem with incompatibility constraints. European Journal of Operational Research **261**(2), 549–562 (2017)

[32] Jamrus, T., Chien, C.-F.: Extended priority-based hybrid genetic algorithm for the less-than-container loading problem. Computers & Industrial Engineering

96, 227–236 (2016) https://doi.org/10.1016/j.cie.2016.03.030

[33] Junqueira, L., Morabito, R., Sato Yamashita, D.: Three-dimensional container loading models with cargo stability and load bearing constraints. Computers & Operations Research **39**(1), 74–85 (2012) https://doi.org/10.1016/j.cor.2010.07.017 . Accessed 2024-05-27

[34] Gendreau, M., Iori, M., Laporte, G., Martello, S.: A Tabu Search Algorithm for a Routing and Container Loading Problem. Transportation Science **40**(3), 342–350 (2006) https://doi.org/10.1287/trsc.1050.0145 . Publisher: INFORMS. Accessed 2024-05-27

[35] Tarantilis, C.D., Zachariadis, E.E., Kiranoudis, C.T.: A Hybrid Metaheuristic Algorithm for the Integrated Vehicle Routing and Three-Dimensional Container-Loading Problem. IEEE Transactions on Intelligent Transportation Systems **10**(2), 255–271 (2009) https://doi.org/10.1109/TITS.2009.2020187 . Conference Name: IEEE Transactions on Intelligent Transportation Systems. Accessed 2024-05-27

[36] Amossen, R.R., Pisinger, D.: Multi-dimensional bin packing problems with guillotine constraints. Computers & Operations Research **37**(11), 1999–2006 (2010) https://doi.org/10.1016/j.cor.2010.01.017 . Accessed 2024-05-27

[37] Fanslau, T., Bortfeldt, A.: A Tree Search Algorithm for Solving the Container Loading Problem. INFORMS Journal on Computing **22**(2), 222–235 (2010) https://doi.org/10.1287/ijoc.1090.0338 . Publisher: INFORMS. Accessed 2024-05-27

[38] Martinez-Sykora, A., Alvarez-Valdes, R., Bennell, J., Tamarit, J.M.: Constructive procedures to solve 2-dimensional bin packing problems with irregular pieces

and guillotine cuts. Omega **52**, 15–32 (2015) https://doi.org/10.1016/j.omega.2014.10.007 . Accessed 2024-05-27

[39] Egeblad, J., Garavelli, C., Lisi, S., Pisinger, D.: Heuristics for container loading of furniture. European Journal of Operational Research **200**(3), 881–892 (2010) https://doi.org/10.1016/j.ejor.2009.01.048 . Accessed 2024-05-27

[40] Hodgson, T.J.: A combined approach to the pallet loading problem. IIE Transactions **14**(3), 175–182 (1982)

[41] Haessler, R.W., Talbot, F.B.: Load planning for shipments of low density products. European Journal of Operational Research **44**(2), 289–299 (1990)

[42] Bortfeldt, A., Gehring, H.: On the treatment of restrictions in storage space optimization using the example of a genetic algorithm for the container loading problem. Logistics Management: Intelligent I+K Technologies, 83–100 (1999)

[43] Egeblad, J., Nielsen, B.K., Odgaard, A.: Fast neighborhood search for two-and three-dimensional nesting problems. European Journal of Operational Research **183**(3), 1249–1266 (2007)

[44] Silva, E., Ramos, A.G., Oliveira, J.F.: Load balance recovery for multi-drop distribution problems: A mixed integer linear programming approach. Transportation Research Part B: Methodological **116**, 62–75 (2018)

[45] Bischoff, E.E.: Stability aspects of pallet loading. Operations-Research-Spektrum **13**, 189–197 (1991)

[46] Moura, A., Oliveira, J.F.: A GRASP approach to the container-loading problem. IEEE Intelligent Systems **20**(4), 50–57 (2005) https://doi.org/10.1109/MIS.2005.57 . Conference Name: IEEE Intelligent Systems. Accessed 2024-01-21

[47] Oliveira, L.d.A., Lima, V.L., Queiroz, T.A., Miyazawa, F.K.: The container loading problem with cargo stability: a study on support factors, mechanical equilibrium and grids. Engineering optimization **53**(7), 1192–1211 (2021)

[48] Parreño, F., Alvarez-Valdés, R., Tamarit, J.M., Oliveira, J.F.: A maximal-space algorithm for the container loading problem. INFORMS Journal on Computing **20**(3), 412–422 (2008)

[49] Pisinger, D.: Heuristics for the container loading problem. European journal of operational research **141**(2), 382–392 (2002)

[50] Amossen, R.R., Pisinger, D.: Multi-dimensional bin packing problems with guillotine constraints. Computers & operations research **37**(11), 1999–2006 (2010)

[51] Malaguti, E., Durán, R.M., Toth, P.: Approaches to real world two-dimensional cutting problems. Omega **47**, 99–115 (2014)

[52] Polyakovskiy, S., M'Hallah, R.: Just-in-time two-dimensional bin packing. Omega **102**, 102311 (2021)

[53] Alonso, M.T., Alvarez-Valdés, R., Iori, M., Parreño, F.: Mathematical models for multi container loading problems with practical constraints. Computers & Industrial Engineering **127**, 722–733 (2019)

[54] Nascimento, O.X., Queiroz, T.A., Junqueira, L.: Practical constraints in the container loading problem: Comprehensive formulations and exact algorithm. Computers & Operations Research **128**, 105186 (2021)

[55] Silva, E.F., Leão, A., Toledo, F.M.B., Wauters, T.: A matheuristic framework for the three-dimensional single large object placement problem with practical

constraints. Computers & Operations Research **124**, 105058 (2020)

[56] Kuo, R.J., Ho, P.-C., Zulvia, F.E.: Application of metaheuristics algorithm on a multi-objective container loading problem considering container's utilization and vehicle's balance. Applied Soft Computing **143**, 110417 (2023) https://doi.org/10.1016/j.asoc.2023.110417 . Accessed 2024-04-08

[57] Nascimento, O.X.d., Queiroz, T., Junqueira, L.: Practical constraints in the container loading problem: Comprehensive formulations and exact algorithm. Computers & Operations Research **128**, 105186 (2021) https://doi.org/10.1016/j.cor.2020.105186 . Accessed 2024-01-02

[58] Fekete, S.P., Schepers, J.: A new exact algorithm for general orthogonal d-dimensional knapsack problems. In: European Symposium on Algorithms, pp. 144–156 (1997). Springer

[59] Martello, S., Pisinger, D., Vigo, D.: The three-dimensional bin packing problem. Operations research **48**(2), 256–267 (2000)

[60] Silva, E.F., Toffolo, T.A.M., Wauters, T.: Exact methods for three-dimensional cutting and packing: A comparative study concerning single container problems. Computers & Operations Research **109**, 12–27 (2019)

[61] Fasano, G.: A mip approach for some practical packing problems: Balancing constraints and tetris-like items. Quarterly Journal of the Belgian, French and Italian Operations Research Societies **2**(2), 161–174 (2004)

[62] Hifi, M., Kacem, I., Nègre, S., Wu, L.: A linear programming approach for the three-dimensional bin-packing problem. Electronic Notes in Discrete Mathematics **36**, 993–1000 (2010)

[63] Junqueira, L., Morabito, R., Yamashita, D.S.: Three-dimensional container loading models with cargo stability and load bearing constraints. Computers & Operations Research **39**(1), 74–85 (2012)

[64] Paquay, C., Schyns, M., Limbourg, S.: A mixed integer programming formulation for the three-dimensional bin packing problem deriving from an air cargo application. International Transactions in Operational Research **23**(1-2), 187–213 (2016)

[65] Tsai, J.-F., Wang, P.-C., Lin, M.-H.: A global optimization approach for solving three-dimensional open dimension rectangular packing problems. Optimization **64**(12), 2601–2618 (2015)

[66] Alonso, M., Alvarez-Valdes, R., Iori, M., Parreño, F., Tamarit, J.: Mathematical models for multicontainer loading problems. Omega **66**, 106–117 (2017)

[67] Eley, M.: Solving container loading problems by block arrangement. European Journal of Operational Research **141**(2), 393–409 (2002)

[68] Bortfeldt, A., Gehring, H., Mack, D.: A parallel tabu search algorithm for solving the container loading problem. Parallel computing **29**(5), 641–662 (2003)

[69] Mack, D., Bortfeldt, A., Gehring, H.: A parallel hybrid local search algorithm for the container loading problem. International Transactions in Operational Research **11**(5), 511–533 (2004)

[70] Bischoff, E.E., Janetz, F., Ratcliff, M.S.W.: Loading pallets with non-identical items. Cutting and Packing **84**(3), 681–692 (1995) https://doi.org/10.1016/0377-2217(95)00031-K

[71] Morabito, R., Arenales, M.: An and/or-graph approach to the container loading

problem. International Transactions in Operational Research **1**(1), 59–73 (1994)

[72] Terno, J., Scheithauer, G., Sommerweiß, U., Riehme, J.: An efficient approach for the multi-pallet loading problem. European Journal of Operational Research **123**(2), 372–381 (2000)

[73] Loh, T.: A packing algorithm for hexahedral boxes. In: Proceedings of the Industrial Automation 92 Conference, Singapore (1992)

[74] Srinivas, M., Patnaik, L.M.: Genetic algorithms: A survey. computer **27**(6), 17–26 (1994)

[75] Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN'95-international Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995). ieee

[76] Glover, F., Laguna, M.: Tabu Search. Springer, ??? (1998)

[77] Karaboga, D., Akay, B.: A comparative study of artificial bee colony algorithm. Applied mathematics and computation **214**(1), 108–132 (2009)

[78] Yang, X.-S., Hossein Gandomi, A.: Bat algorithm: a novel approach for global engineering optimization. Engineering computations **29**(5), 464–483 (2012)

[79] Cheng, M.-Y., Prayogo, D.: Symbiotic organisms search: a new metaheuristic optimization algorithm. Computers & Structures **139**, 98–112 (2014)

[80] Kuo, R.-J., Zulvia, F.E.: The gradient evolution algorithm: A new metaheuristic. Information Sciences **316**, 246–265 (2015)

[81] Rajabioun, R.: Cuckoo optimization algorithm. Applied soft computing **11**(8), 5508–5518 (2011)

[82] Dokeroglu, T., Sevinc, E., Kucukyilmaz, T., Cosar, A.: A survey on new generation metaheuristic algorithms. Computers & Industrial Engineering **137**, 106040 (2019)

[83] Wong, W., Ming, C.I.: A review on metaheuristic algorithms: recent trends, benchmarking and applications. In: 2019 7th International Conference on Smart Computing & Communications (ICSCC), pp. 1–5 (2019). IEEE

[84] Jamrus, T., Chien, C.-F.: Extended priority-based hybrid genetic algorithm for the less-than-container loading problem. Computers & Industrial Engineering **96**, 227–236 (2016)

[85] Domingo, B.M., Ponnambalam, S., Kanagaraj, G.: Particle swarm optimization for the single container loading problem. In: 2012 IEEE International Conference on Computational Intelligence and Computing Research, pp. 1–6 (2012). IEEE

[86] Cano, I., Torra, V.: Particle swarm optimization for container loading of nonorthogonal objects. In: International Conference on Artificial Intelligence and Soft Computing, pp. 403–410 (2010). Springer

[87] Tlili, T., Faiz, S., Krichen, S.: A particle swarm optimization for solving the one dimensional container loading problem. In: 2013 5th International Conference on Modeling, Simulation and Applied Optimization (ICMSAO), pp. 1–4 (2013). IEEE

[88] Koonsintananan, S., Kimpan, W.: Applied particle swarm optimization in solving container loading problem for logistics. In: 2019 11th International Conference on Knowledge and Smart Technology (KST), pp. 88–93 (2019). IEEE

[89] Wang, Y., Li, H., Lei, Z., Ma, D., Fang, Y.: Progressively-refined tree search

for container loading problem. In: 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), pp. 2520–2528 (2019). IEEE

[90] Bayraktar, T., Ersöz, F., Kubat, C.: Effects of memory and genetic operators on artificial bee colony algorithm for single container loading problem. Applied Soft Computing **108**, 107462 (2021)

[91] Rakotonirainy, R.G., Vuuren, J.H.: Improved metaheuristics for the two-dimensional strip packing problem. Applied Soft Computing **92**, 106268 (2020)

[92] Mostaghimi Ghomi, H., St Amour, B.G., Abdul-Kader, W.: Three-dimensional container loading: A simulated annealing approach. International Journal of Applied Engineering Research **12**(7), 1290 (2017)

[93] Dereli, T., Sena Das, G.: A hybrid simulated annealing algorithm for solving multi-objective container-loading problems. Applied Artificial Intelligence **24**(5), 463–486 (2010)

[94] Zhou, Q., Hao, J.-K., Wu, Q.: A hybrid evolutionary search for the generalized quadratic multiple knapsack problem. European Journal of Operational Research **296**(3), 788–803 (2022)

[95] García, J., Maureira, C.: A knn quantum cuckoo search algorithm applied to the multidimensional knapsack problem. Applied Soft Computing **102**, 107077 (2021)

[96] Júnior, R.R., Yanasse, H.H., Morabito, R., Junqueira, L.: A hybrid approach for a multi-compartment container loading problem. Expert Systems with Applications **137**, 471–492 (2019)

[97] V. Romero, S., Osaba, E., Villar-Rodriguez, E., Oregi, I., Ban, Y.: Hybrid approach for solving real-world bin packing problem instances using quantum annealers. Scientific Reports **13**(1), 11777 (2023)

[98] Karabulut, K., İnceoğlu, M.M.: A hybrid genetic algorithm for packing in 3d with deepest bottom left with fill method. In: International Conference on Advances in Information Systems, pp. 441–450 (2004). Springer

[99] Crainic, T.G., Perboli, G., Tadei, R.: Extreme Point-Based Heuristics for Three-Dimensional Bin Packing. INFORMS Journal on Computing **20**(3), 368–384 (2008) https://doi.org/10.1287/ijoc.1070.0250 . Publisher: INFORMS. Accessed 2024-05-22

[100] Feng, X., Moon, I., Shin, J.: Hybrid genetic algorithms for the three-dimensional multiple container packing problem. Flexible Services and Manufacturing Journal **27**(2), 451–477 (2015) https://doi.org/10.1007/s10696-013-9181-8 . Accessed 2024-05-22

[101] Wang, H., Chen, Y.: A hybrid genetic algorithm for 3d bin packing problems. In: 2010 IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA), pp. 703–707 (2010). https://doi.org/10.1109/BICTA.2010.5645211

[102] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Transactions on Evolutionary Computation **6**(2), 182–197 (2002) https://doi.org/10.1109/4235.996017

[103] Elahi, M., Afolaranmi, S.O., Martinez Lastra, J.L., Perez Garcia, J.A.: A comprehensive literature review of the applications of ai techniques through the lifecycle of industrial equipment. Discover Artificial Intelligence **3**(1), 43 (2023)

[104] Bischoff, E.E., Ratcliff, M.: Issues in the development of approaches to container loading. Omega **23**(4), 377–390 (1995)

[105] Gehring, H., *et al.*: A genetic algorithm for solving the container loading problem. International transactions in operational research **4**(5-6), 401–418 (1997)