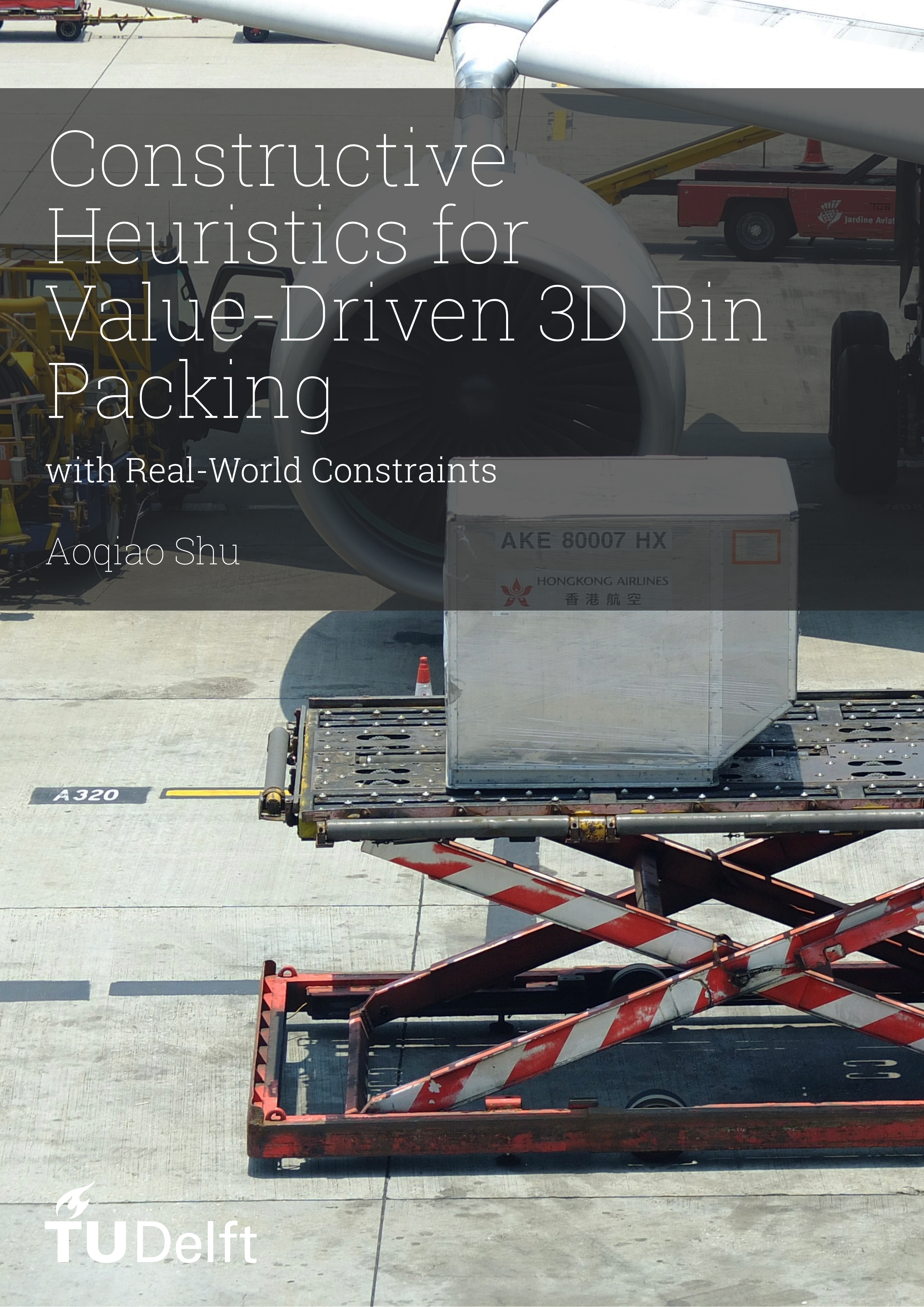
Constructive Heuristics for Value-Driven 3D Bin Packing

with Real-World Constraints

Aoqiao Shu



This page intentionally left blank.

Constructive Heuristics for Value-Driven 3D Bin Packing

with Real-World Constraints

by

Aoqiao Shu

Supervisor:	A. Bombelli
Project Duration:	December, 2024 - September, 2025
Faculty:	Faculty of Aerospace Engineering, Delft

TUDelft

Preface

This thesis marks the end of six years in Delft and the end of my student life. I'm thankful to the people who stayed close throughout.

First, to my supervisor, Alessandro Bombelli, thank you for your guidance and trust throughout this project. Long live all the mountains we moved together.

I'm also grateful to the friends who made daily life lighter. To Dequan for keeping me fed, Justas for saving me literally, Jiaxuan for Lego sessions, and Bryan for pushing me to the gym. We make these memories for ourselves.

Of course, none of this would have been possible without my family. Thank you for your support every step of the way, even far away in China. I had the best day with you today.

Finally, to Chenyu, thank you for being with me through it all. You stayed close without needing words. We're on each other's team.

Aoqiao Shu
Delft, October 2025

Contents

Nomenclature

Abbreviations

Abbreviation	Definition
3D-BPP	Three-dimensional Bin Packing Problem
BFD	Best Fit Decreasing
BS	Batch Size
EE	Equal-Equal
EEE	Equal-Equal-Equal
EP	Extreme Point
ES	Equal-Sparse
FFD	First Fit Decreasing
MB	Maximum Number of Bins
MILP	Mixed-Integer Linear Program
NC	Number of Supporting Corners
PD	Padding Distance
SA	Surface Area Ratio
SS	Sparse-Sparse
ULD	Unit Load Device

I

Scientific Article

This page intentionally left blank.

Constructive Heuristics for Value-Driven 3D Bin Packing with Real-World Constraints

Aoqiao Shu

This work presents a heuristic framework for value-driven three-dimensional bin packing for air cargo operation under practical constraints such as batch arrivals, box fragility, and stability requirements. The method builds on the extreme-point placement heuristic, enhanced with a two-stage box selection strategy that first filters boxes by value density and then sorts them with attention to geometric alignment. Stability checks including supporting surface area, corner support, and padding tolerance are introduced to ensure realistic placements, along with a dedicated procedure for safely placing fragile items. A batch-handling mechanism allows bins to be packed progressively as boxes arrive, using volume-based thresholds to lock or release partially filled bins. Performance is evaluated using several types of box sets that differ in shape regularity and distribution. Results show that relaxing stability conditions improves both value and volume metrics, and that sorting strategies which consider box dimensions outperform value-only sorting. The framework also highlights the limitations of packing overly uniform boxes and the quality of solutions decreased as the number of bins increases. The method aims to balance space usage and value considerations while accounting for constraints commonly found in logistics settings.

I. Introduction

Air cargo plays a crucial role in global trade. In 2023, the industry transported 61.4 million tonnes of freight, and the total value of goods moved by air was estimated at $8 trillion [1]. While that volume represents under 1 % of world trade by volume, air transport accounts for about 33 % of trade by value [1]. This disparity reflects the central role of air freight in moving high-value, time-sensitive goods.

Three-dimensional bin packing plays a critical role in air cargo operations. The problem involves selecting and placing a set of items (typically boxes) into one or more containers (bins or ULDs (Unit Load Devices)) in a way that maximises space usage or meets other performance goals such as load balance, stability, or value. From a computational standpoint, the three-dimensional bin packing problem is well known to be NP-hard, making exact optimisation impractical for larger instances [2]. This difficulty has motivated a wide range of heuristic and metaheuristic methods in the literature.

Beyond computational complexity, real-world conditions introduce further layers of difficulty. In practice, cargo cannot always be stacked freely as stability constraints must be respected, and heavy goods must not be placed on top of fragile objects [2, 3]. Batch arrivals of freight add further complexity, since packing decisions often must be made without full knowledge of the set of items to be loaded [4]. Moreover, shipments differ not only in size and weight but also in value and urgency, which creates a trade-off between efficient use of space and prioritisation of high-value goods. Traditional bin packing heuristics generally focus on maximising volume utilisation and thus do not handle these value aware priorities effectively [2].

This work develops a constructive heuristic for value-aware 3D bin packing under batch arrivals. The method builds upon the extreme-point (EP) heuristic, where placement options are restricted to a dynamic list of candidate coordinates inside each bin. This significantly reduces the search space while still producing competitive results. The algorithm extends the classic EP approach with several key additions:

- A two-stage box selection strategy that first filters boxes based on value density, then sorts them to encourage consistent stacking and efficient placement.

- A set of physical stability checks including minimum supported surface area, required supported corners, and an optional padding distance to allow for minor alignment gaps.
- A dedicated routine to handle fragile boxes safely by delaying their placement until a valid support structure is formed below.
- A batch-based arrival mechanism that simulates the incremental availability of boxes, with logic for unlocking and discarding bins based on current fill levels.

These features allow the algorithm to better reflect real-world constraints, particularly in cargo loading scenarios where value, fragility, and late arrivals must be considered simultaneously. The overall goal is to balance volume efficiency and value prioritisation, while keeping the algorithm adaptable to different packing environments.

The rest of this paper is structured as follows: section II described the state of the art, section III introduces the full methodology, including the extreme-point system, sorting logic, support validation, and batch management rules. This is followed by a description of the testing sets and parameter variations used in evaluation. In section IV, the results are then analysed with respect to six hypotheses, covering both algorithm design choices and input conditions. Finally, conclusions are drawn and recommendations for further research are provided.

II. Literature Review

A. Bin Packing Problem Overview

The three-dimensional Bin Packing Problem (3D-BPP) is a classic NP-hard problem where a set of items must be placed into a limited number of bins without overlap. The goal is usually to minimise the number of bins used or maximise volume utilisation. In air cargo, this corresponds to loading shipments into ULDs while respecting geometric and operational constraints.

Exact optimisation can be achieved by formulating the problem as a Mixed-Integer Linear Program (MILP), which can then be solved using exact methods such as Branch-and-Bound [3, 5]. These methods are theoretically capable of producing optimal solutions since they capture the full set of constraints and objectives. However, they are limited to very small instances, as runtimes increase exponentially with the number of items. For example, even advanced exact methods, such as Branch-and-Bound or modern MILP solvers, cannot handle the hundreds of items found in realistic cargo scenarios. As a result, exact methods mainly serve as benchmarks for evaluating heuristic solutions.

B. Heuristic and Metaheuristic Approaches

To address larger and more practical cases, heuristic algorithms are widely used. Constructive heuristics build a solution step by step using placement rules such as First Fit Dereasing (FFD) or Best Fit Decreasing (BFD)[6], or more structured schemes like deepest-bottom-left-fill. These methods are simple and efficient but often leave void spaces, reducing overall utilisation. To improve on this, more sophisticated approaches model free space explicitly through representations such as corner points [5], or extreme points [7]. Such representations focus placement decisions on meaningful candidate positions rather than arbitrary coordinates, leading to tighter packings.

Beyond greedy construction, metaheuristic frameworks have been applied to explore the solution space more broadly. Examples include tabu search [8], guided local search [9], GRASP [10], genetic algorithms [11], and simulated annealing [12]. These methods often start from a constructive heuristic and then improve it through reallocation, swaps, or repacking. While more computationally demanding, they can achieve higher utilisation or reduce bin count by considering the packing problem globally rather than one bin at a time. However, for large-scale operations where speed and reliability are critical, constructive heuristics remain the core building block.

C. Extreme-Point Based Packing Methods

Among constructive heuristics, extreme-point methods have become a dominant strategy. First formalised by Crainic et al. [7], the EP approach defines a set of candidate positions located at the walls of the container or adjacent to already placed boxes. These positions are chosen such that no box could be shifted further back or down from that point without collision. Restricting placement to extreme points significantly reduces the search space while still preserving the ability to find high-quality solutions. In practice, EP-based algorithms achieve denser packings than traditional layer-based or shelf heuristics, as each new item is placed flush against existing structures or container boundaries [7, 13]. Figure 1 illustrates how EPs are defined in both 2D and 3D packings.

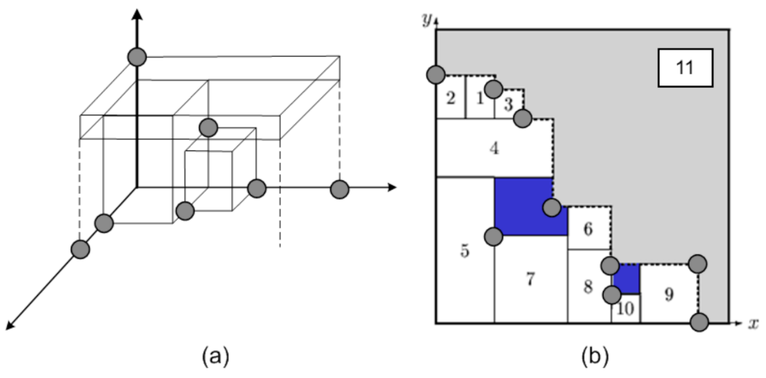


Figure 1. Example of extreme points defined in 3D (a) and 2D (b) packings [7]

EP heuristics have been extended and hybridised in various ways. For example, they can be embedded into multi-phase frameworks, or adapted to specific constraints like weight balance and orientation rules.

1. Merit Function Strategies

Because multiple extreme points are usually available, EP heuristics require a rule to decide which placement is most effective. This is handled by a merit function that assigns a score to each candidate placement. Different forms of merit functions have been studied, such as minimising leftover volume, minimising footprint expansion, or minimising the sum of residual gaps between the box and its surrounding space [7].

The residual space approach, for example, evaluates how well a box fits along each axis by measuring the distance between the box and the available limits. Placements with smaller gaps score better, encouraging tight fits. Variants such as the EP-BFD heuristic combine a residual space metric with ordering rules, achieving some of the best reported results for 3D packing benchmarks.

2. Box Selection and Queueing

Alongside the placement rule, the order in which boxes are selected is a crucial factor. Many algorithms use static sequences, for example sorting items by volume, base area, or height before packing [7]. This is simple and computationally cheap but may miss opportunities where a later box could fit better in the current space. Dynamic or adaptive sequencing, in contrast, evaluates the remaining unpacked items at each step and selects the one that best suits the current configuration [10, 14]. Although slower, such adaptive strategies generally lead to higher utilisation.

Despite these advances, most heuristic methods operate under the simplifying assumption that all items are of equal priority and must be packed. Selection is therefore driven by geometric fit alone. Very few constructive algorithms have considered item value or profit as part of the decision process. Similarly, classical models assume that all items are available upfront, ignoring practical situations where shipments arrive in batches. Traditional online heuristics handle items as they arrive but do not incorporate mechanisms to defer, repack, or prioritise items across batches.

D. Research Gap: Profit and Batch Integration

EP heuristics provide a strong basis for efficient 3D packing, and merit functions and box selection rules play key roles in guiding them. However, the literature shows two major gaps. First, most constructive algorithms focus solely on space utilisation or bin count, neglecting profit considerations. In real-world cargo operations, it may be preferable to prioritise high-value shipments even at the cost of leaving some space unused. Second, batch arrivals are common in logistics, yet existing methods rarely address this. Items are usually assumed to be known in advance (offline) or packed immediately on arrival (online), leaving little flexibility to optimise across multiple waves of cargo.

Combining the EP framework with profit-aware selection and batch handling mechanisms therefore represents an open opportunity. Such integration would move constructive heuristics closer to operational needs by accounting not only for geometric efficiency but also for economic value and temporal arrival patterns.

III. Methodology

This section describes the methodology used in this work. The algorithm is based on the EP concept introduced by Crainic et al. [7]. Unlike methods such as linear programming, which aim for a global exact solution, the extreme point heuristic is a constructive approach that produces the best placement available at each step according to its rule.

The key idea behind the extreme point heuristic is to narrow the search space of possible positions inside the container while still accounting for all feasible future placements. To do this, the method defines a set of special coordinates, called extreme points, where the next box may be placed. These points lie either against the container walls or adjacent to already packed boxes, and they are chosen so that no box could be pushed further back or down from that position without colliding with something. Each extreme point therefore represents the lowest and furthest back feasible placement in a region of the bin.

When packing begins, the only available point is at the origin of the container, that is, the back–left–bottom corner with coordinate (0, 0, 0). Placing the first box at this position makes full use of the corner and maximises the available space in the container. Once this first box is in place, new extreme points appear on its exposed faces. In three dimensions, a single box can generate up to three *raw extreme points*, one aligned with each axis. Raw EPs are the immediate candidate positions that appear on the exposed faces of a placed box. These raw EPs are then projected outward along the other two axes until they meet either the container wall or the surface of another box. Once this projection is done, the valid EPs are obtained. Each raw EP can therefore give rise to two valid extreme points, meaning that a single placement may create as many as six new candidate positions.

The list of extreme points is updated each time a box is added. The point that was used for placement is removed, any new points generated by the placement are added, and any invalid points are discarded.

To illustrate the concept more clearly, a two-dimensional example is shown in Figure 2.

1) **Step 1:** When the container is empty, the origin (EP point A) is the only available extreme point. This is where the first box will be placed.
2) **Step 2:** Once box 1 is placed at EP point A, it is removed from the list because it is no longer valid. The placement of box 1 generates two raw extreme points, one extending along the x-axis and one along the y-axis. Each raw point is then projected along the other direction until it meets either the container wall or the surface of a box. The x-axis raw point, when projected along the negative y-axis, coincides with the container wall, creating EP point B. The y-axis raw point, when projected upward, meets the container wall, creating EP point C.
3) **Step 3:** Box 2 is placed at EP point B (it could also have been placed at EP point C, but EP point B was chosen arbitrarily). After this placement, new raw extreme points are created along the faces of box 2. Their projections produce EP point D on the container wall and EP point E against the side of box 1.

4) **Step 4:** Box 3 is placed at EP point D. This placement generates new raw extreme points. When projecting upward along the y-axis and left along the negative x-axis, the projection does not meet the wall directly but instead touches the surface of box 1. As a result, EP point G is created as the next valid point.

After the three boxes are placed, four extreme points remain in the list: EP point C, EP point E, EP point F, and EP point G. Note that EP point E is not removed, as it may still be useful for accommodating certain types of boxes. The same logic extends naturally to three dimensions, where each placement produces raw EPs that are projected to generate valid EPs in the 3D space. For example, when a raw EP is traced along the x-axis, it is then projected towards the negative y- and z-directions until it meets either the container wall or the surface of another box.

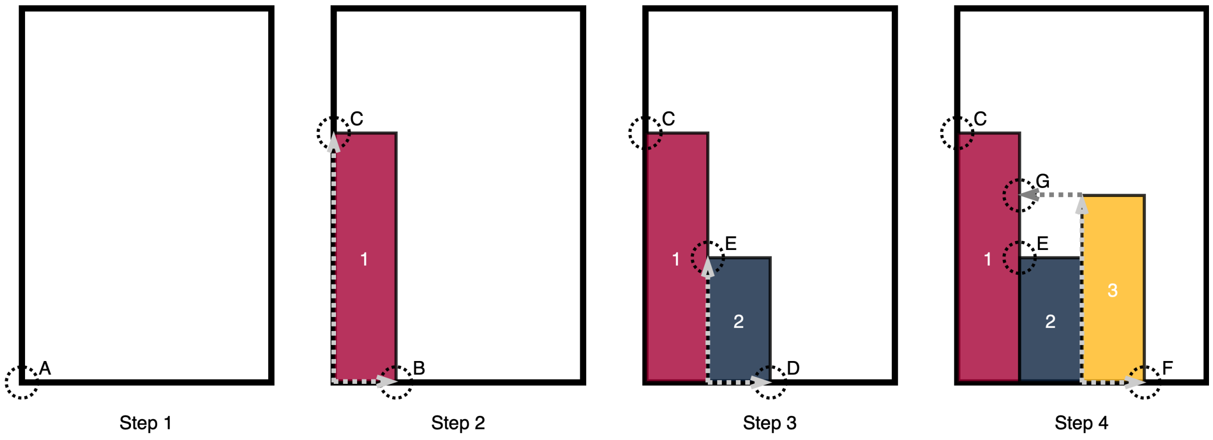


Figure 2. A two-dimensional illustration of the extreme point (EP) heuristic.

Further subsections of this section provide more detail on the overall procedure. They explain how the next box to be placed is selected and which extreme point it is assigned to, how fragile boxes are managed during packing, how batches of boxes arriving at different times are handled, and finally how the method is tested.

A. Extreme-Point Evaluation (Merit Function)

In Step 3 of Figure 2, box 2 is placed at EP Point B, although EP Point C would also have been a valid option. This situation highlights the need for a rule to decide which EP should be chosen when several are available. To address this, a merit function is used to evaluate each EP Point and assign a score, allowing the algorithm to select the most appropriate placement.

$$\begin{aligned} score_{3D} = {} & (RS_x - w)^p + (RS_y - d)^p + (RS_z - h)^p \\ & + (RS_x - w)^{p/2} \times (RS_y - d)^{p/2} \\ & + (RS_y - d)^{p/2} \times (RS_z - h)^{p/2} \\ & + (RS_z - h)^{p/2} \times (RS_x - w)^{p/2} \end{aligned} \tag{1}$$

$$\begin{aligned} score_{2D} = {} & (RS_x - w)^p + (RS_y - d)^p \\ & + (RS_x - w)^{p/2} \times (RS_y - d)^{p/2} \end{aligned} \tag{2}$$

The merit function in three dimensions is given in Equation 1, and a simplified form for the two-dimensional case is shown in Equation 2. In these equations, p is the exponent applied to each term, with a default value of 2. The terms RS_x, RS_y, and RS_z represent the residual space distances along the x-, y-, and z-axes, while w, d, and h denote the width, depth, and height of the candidate box. Residual space distance can be visualised by imagining a laser beam projected from an EP along one axis: the beam travels until it meets either the container wall or the surface of another box, and the distance covered is the residual space. This distance represents the maximum free length in that direction into which the candidate box can extend without overlap.

To help visualise this concept, a two-dimensional example of residual space is shown in Figure 3. In the figure, the grey dotted double-arrow lines represent the residual space distance measured from each EP Point. When the container is empty, the residual space extends across the full container, which corresponds to the maximum possible box size in both direction. At EP Point E, the residual space along the x-axis is very limited compared with EP Point G, while EP Point E offers a larger free distance along the y-axis. These differences mean that different EP Points are better suited to boxes of different shapes. For example, EP Point E would be more suitable for a tall, thin box (high aspect ratio), whereas EP Point G would better accommodate a more square-shaped box (low aspect ratio).

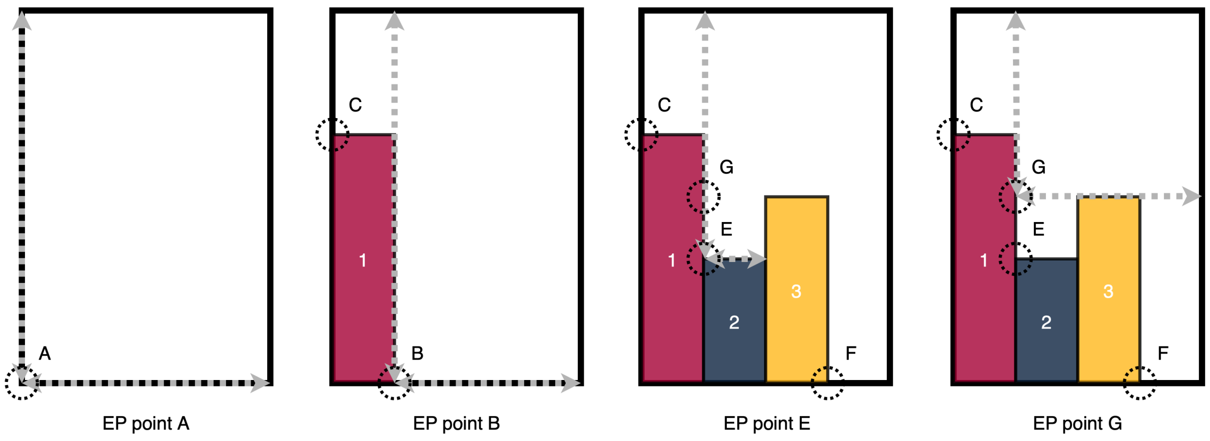


Figure 3. A two-dimensional illustration of residual space distances from different EP Points.

The three-dimensional merit function in Equation 1 balances two main objectives. First, it favours placements where the candidate box fits snugly against the residual space along one axis. For example, if $(RS_x - w) = 0$, the box exactly fills the available space in the x-direction, leaving no gap. This behaviour is encouraged by the squared terms $(RS_x - w)^p$, $(RS_y - d)^p$, and $(RS_z - h)^p$, which reward tight fits along individual axes. Second, the cross terms such as $(RS_x - w)^{p/2}(RS_y - d)^{p/2}$ reflect situations where the box aligns closely along two axes at once. This means the function also favours placements where a box not only fits snugly in one direction, but also touches boundaries in two dimensions, effectively "locking" the box in place. Together, these components guide the algorithm to prefer placements that make efficient use of space by reducing gaps along one or more axes.

The three-dimensional merit function in Equation 1 is designed so that lower scores indicate better placements. The first three terms, $(RS_x - w)^p$, $(RS_y - d)^p$, and $(RS_z - h)^p$, favour situations where the candidate box fits snugly against the residual space along one axis. For instance, if $(RS_x - w) = 0$, the box exactly fills the available space in the x-direction, leaving no gap. The cross terms, such as $(RS_x - w)^{p/2}(RS_y - d)^{p/2}$, represent cases where the box is close to fitting along two axes at the same time. This encourages placements where the box not only fits well in one direction but also touches boundaries in two dimensions, effectively "locking" the box into place and reducing unused space.

The parameter p controls how strongly the function penalises larger gaps. With $p = 2$ (the default), the penalty grows quadratically, giving a balanced preference to tighter fits. Larger values of p (e.g. $p = 4$) make the function much more sensitive to gap size, so the algorithm will strongly prefer placements that minimise free space. Smaller values of p reduce this effect, treating gap differences more evenly.

In some cases, two EP Points may appear along the same line, sharing the same coordinate on one axis but differing on the others. To avoid redundancy and reduce wasted space, the algorithm keeps only the point that is closer to the origin. For example, in Figure 3, if a box could be placed at either EP Point E or EP Point G, EP Point G would be discarded. Although both points share the same x coordinate, EP Point E is closer to the origin and therefore preferred.

After this filtering, the remaining EP Points are scored using the merit function. The algorithm then checks whether the best-scoring EP can properly support the box. Support can be satisfied in two ways: either by having a sufficient portion of the box's bottom surface resting on the container floor or on already packed boxes, or by ensuring that enough of the box's corners are supported by existing boxes or by the container structure (ULD cut).

If the support condition is met, the box is placed at the EP with the lowest merit score, and the EP list is updated for the next iteration. It is also worth noting that a padding distance parameter is introduced to allow for small gaps between the lower and upper boxes. If the gap is within this padding threshold, the upper box is still considered supported.

B. Box Selection from Queue

To incorporate value-driven decision making into the packing framework, the box selection procedure is divided into two steps. Step 1 applies a profit-based filtering, while Step 2 considers the dimensions of each box.

In Step 1, each box in the packing queue has a *value density* calculated using Equation 3. Value density is defined as the ratio of profit to the geometric mean of an item's volume and weight. In air cargo settings, value is normally measured either by weight or by volume. The geometric mean ensures that both constraints are treated equally, reflecting the fact that each represents an important capacity limitation—analogous to the two "sacks" in a multi-dimensional knapsack problem. The square root reduces the measure to the correct dimension, consistent with how value is assessed in practice.

$$value\_density = \frac{value}{\sqrt{volume \times weight}} \tag{3}$$

The boxes are then sorted in descending order of value density. The top $N\%$ are selected for the current batch, while the remaining boxes are placed in a waiting list for later consideration. Being placed in the waiting list does not mean that these boxes are excluded entirely; they may still be chosen in subsequent batches. Further details on how batches are handled are introduced in a later section.

Moving on to the second step, the focus is on placing the boxes selected from the high-value pool effectively into the bin. Three different ideas are considered.

The first idea is to arrange the boxes by height, starting from the tallest and moving downwards. For rotatable boxes, the shortest side is treated as the height so that they rest on their largest face, while non-rotatable boxes keep their fixed orientation. If two boxes share the same height, the one with the larger base area, measured as width multiplied by depth, is placed first. As shown in Figure 4, six boxes with different dimensions are considered. Assuming all boxes are rotatable, the box with dimension 3×7 is rotated to lie flat, so its acting dimension becomes 7×3. After this adjustment, the boxes are sorted by decreasing height values: height 5 first, then 4, and finally 3. Within each height group, tie-breaking is done by the base width.

A second idea is to give priority to the heights that appear most often in the batch. This encourages boxes with common heights to be packed together, making the layout more uniform. As in the first idea, rotatable boxes are placed on their largest face so that their shortest side is treated as height, while non-rotatable boxes remain fixed. Within each height group, boxes with larger base areas are placed first. In the example shown in Figure 5, after rotation correction, the most common height is 3, so the height-3 group is packed first, followed by the height-5 group, and finally the least frequent group with height 4.

The third idea takes a different approach by allowing rotatable boxes to freely rotate in order to minimise the total number of distinct heights across the batch. Instead of always assigning the shortest side as height, each rotatable box is oriented so that it matches the height of other boxes whenever possible. For example, consider a rotatable box with dimensions $(40, 30, 20)$ and a non-rotatable box with dimensions $(30, 50, 40)$, where the last number is the fixed height. In this case, the rotatable box can be oriented to take 40 as its height so that both boxes share the same height. As a result, they belong to one common height group rather than two separate ones, reducing variation and making stacking more consistent. As shown in Figure 6, after the initial box set is adjusted, the sorting algorithm clusters the boxes into three distinct height levels (3, 4, and 5), and the packing sequence proceeds from the highest group downwards.

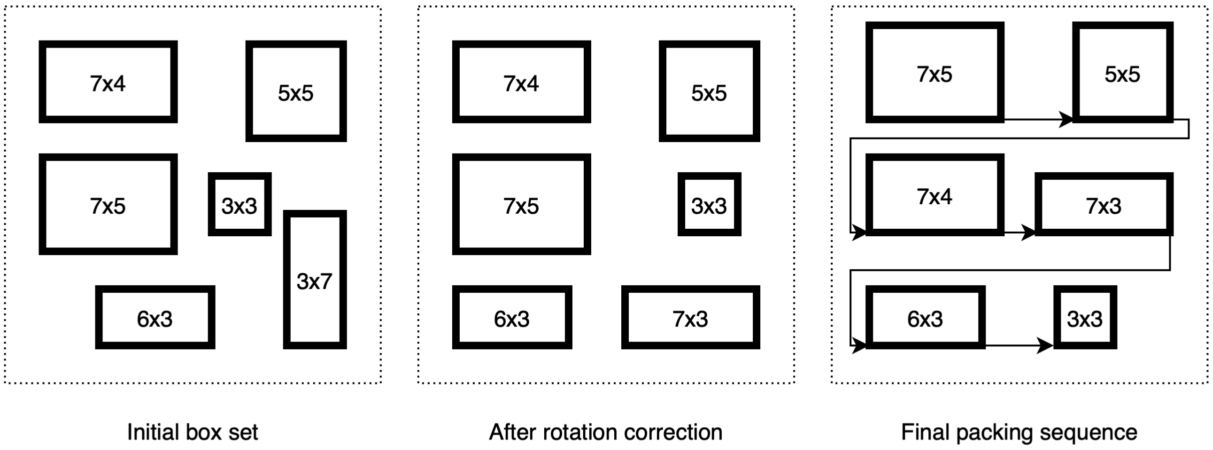


Figure 4. Example of sorting by descending height.

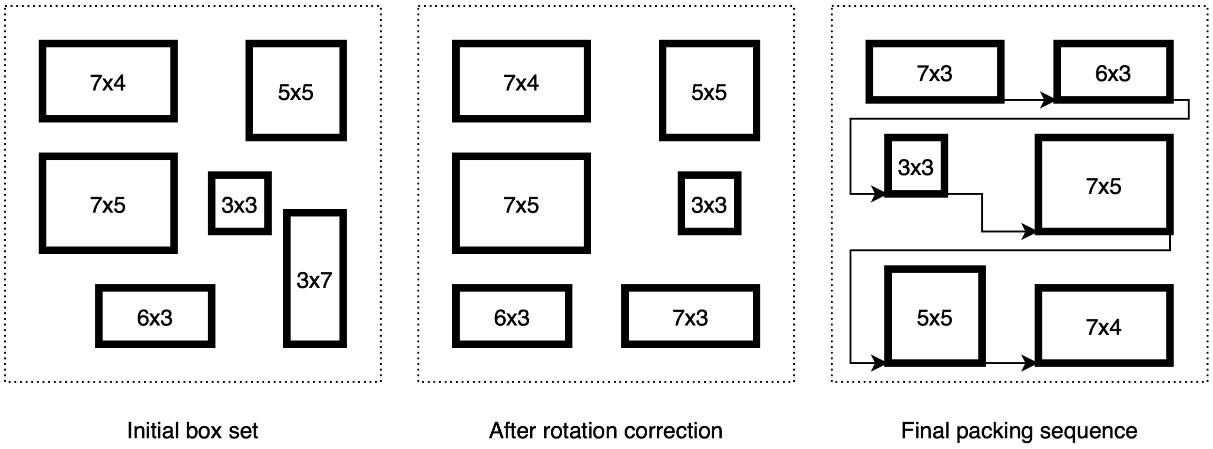


Figure 5. Example of sorting by the most frequent height.

Uniformity in height layers is beneficial because it produces flatter packing levels, improves the stability of stacked boxes, and reduces wasted vertical space that would otherwise occur from mismatched box heights. With uniform height, the boxes in a layer collectively form a consistent two-dimensional surface, which in effect partially reduces the three-dimensional packing problem to a two-dimensional one within that layer. This structure resembles the conditions under which guillotine cut strategies are often studied, although such methods are not exactly applied here. This simplification not only makes the arrangement easier to manage but can also lead to higher packing efficiency in terms of volume, as gaps caused by uneven layers are minimised.

C. Handling Fragile Boxes

Fragile boxes are treated with a dedicated procedure to make sure they are not placed in unsafe positions during packing. Because these boxes cannot take heavy loads on top and should not rest on other fragile items, their placement is handled differently from normal boxes.

The process starts with a 2D EP problem defined at the ceiling of the container. In this temporary stage, fragile boxes are positioned as if they "float" at the top, and candidate locations are identified using the 2D EP rules. This step does not commit the boxes to their final location but instead prepares possible placements for later validation. At the end of the main packing loop, or once the bin is locked, the fragile boxes are released from this temporary ceiling layer. They are allowed to drop vertically downwards until they reach the highest supported z-position. A support check is then carried out to ensure that the placement is stable. Support can come from the container floor or from the surfaces of other already

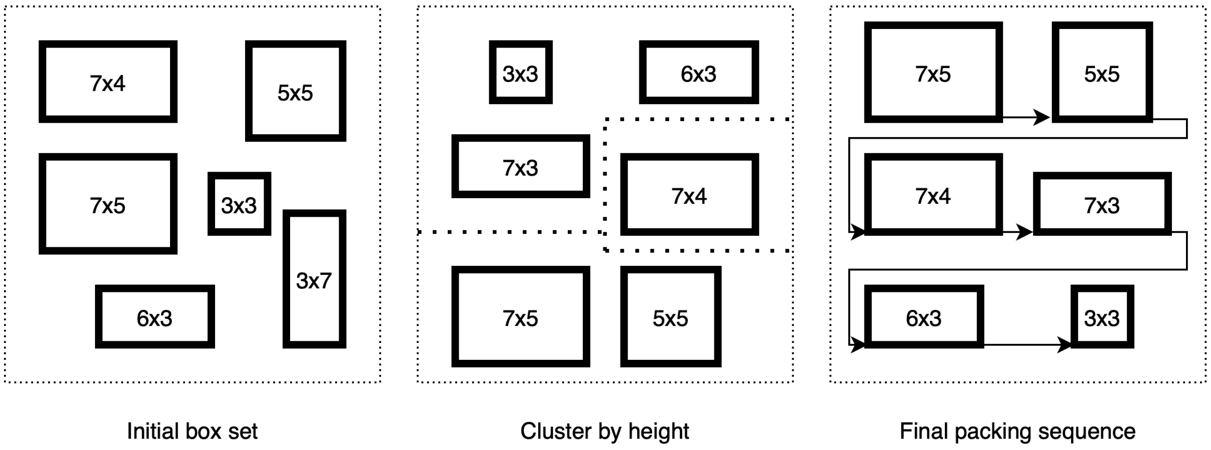


Figure 6. Example of minimising the number of distinct heights.

packed boxes (the same condition applied to normal boxes).

If a fragile box passes this support check, its position is confirmed and it becomes part of the final solution. If the support is insufficient, the fragile box is not fixed in place. Instead, it is returned to the main packing loop, where it can attempt to find a more suitable extreme point in later iterations. Any fragile box that cannot be safely supported after this process is ultimately discarded to protect its integrity.

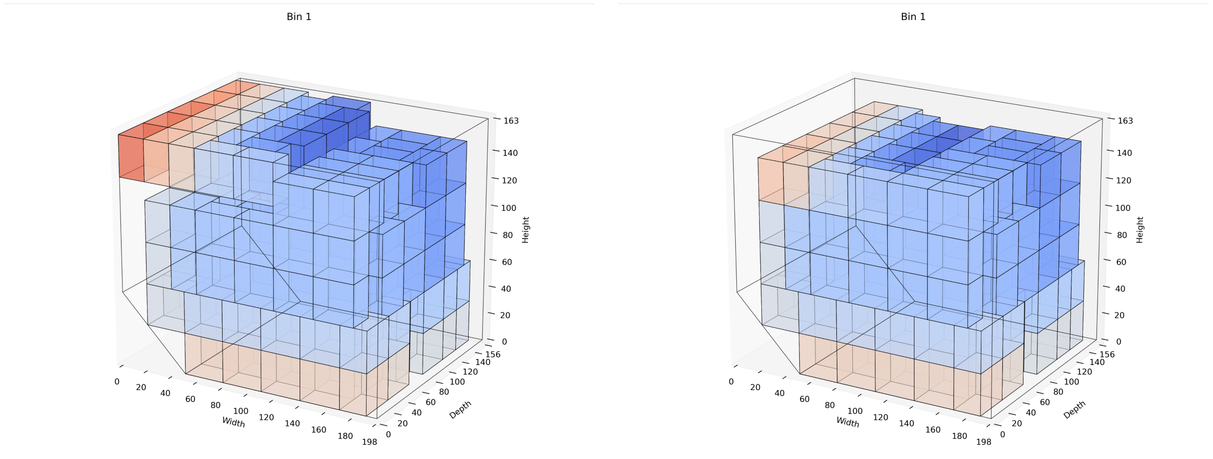


Figure 7. Fragile box handling: instance 1.

In the left part of Figure 7, corresponding to the initial phase, the fragile boxes are first placed at the upper left side of the bin, where they are temporarily positioned using the ceiling 2D EP problem. In the right figure, these boxes are then released and allowed to fall vertically until they reach the highest available support. Most of the fragile boxes succeed in finding stable positions and are confirmed, while some of the red boxes floating above the container cut fail to obtain sufficient support and are therefore discarded.

Similarly, in Figure 8, the fragile boxes are first shown floating above the packed structure in the left subfigure, where normal boxes are coloured in blue and the fragile boxes are displayed in varying colours. Once dropped, as shown in the right subfigure, the fragile boxes that find sufficient support are confirmed in place and highlighted in red, while unsupported ones are discarded.

This approach ensures that fragile boxes are only placed in positions where they are safe, while still giving them multiple chances to be included in the packing plan. By handling them separately through the temporary 2D EP process and the drop-and-support validation, the algorithm avoids unsafe placements

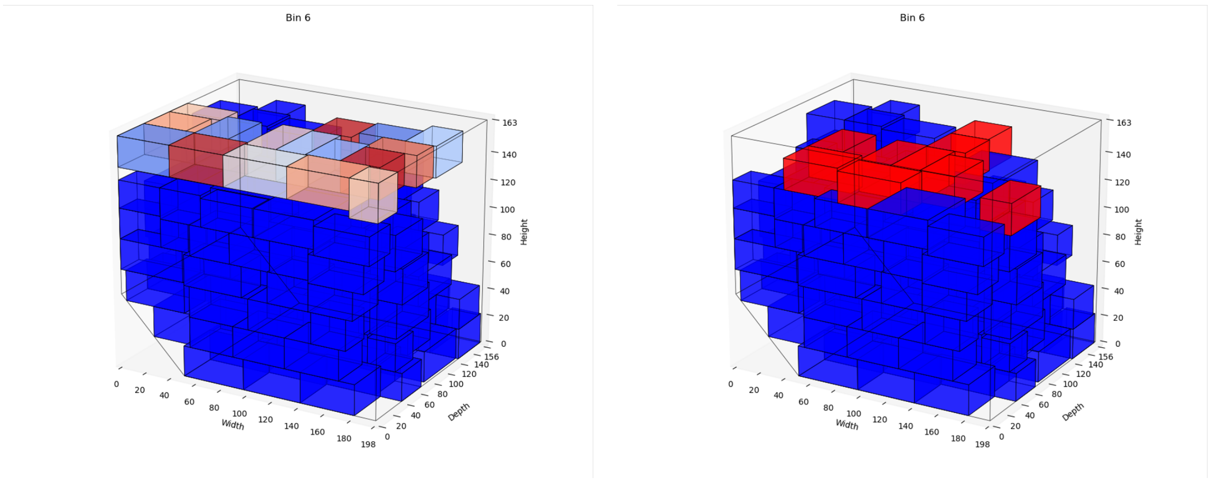


Figure 8. Fragile box handling: instance 2.

without disrupting the main packing flow.

Since fragile boxes make up only a very small fraction of the total, this additional procedure adds minimal computational cost while preserving packing efficiency.

D. Handling Batch Arrivals

This framework uses batches to simulate the arrival times of packages, treating them as discrete batches rather than continuous time steps. The batch arrival process begins when a batch of boxes arrives, with each batch consisting of a predefined set of boxes. After each batch is received, the algorithm attempts to place N% of thr top value boxes in the queue into the bins, filling up the available space. Once a batch is processed, the algorithm evaluates whether the bin is close to reaching its packing capacity by analysing both the volume and weight utilisation of the bin.

The system then analyses the volume utilisation of each bin to determine the next steps in the packing process. To manage this process, two important ratios are introduced: the unpacking ratio and the locking ratio.

As shown in Figure 9, the horizontal axis represents the volume utilisation of the bin, which is calculated as the ratio of the packed volume to the total available volume in the bin. The unpacking ratio is set to 50% and the locking ratio to 80%. These thresholds were selected because 50% is low enough to justify repacking while balancing computation resources, whereas 80% indicates that a bin is nearly full. The values are not fixed and can be adjusted according to the requirements of the application. The algorithm behaves with 50% and 80% as follows:

- If the bin's volume utilisation is below 50%, the bin is considered under-utilised. In this case, the packed boxes are unpacked and returned to the queue for re-placement. This allows the algorithm to try different packing configurations and potentially improve the packing density.
- If the bin's volume utilisation is between 50% and 80%, the bin is considered adequately filled, and no unpacking will occur. The algorithm will not attempt to change the packing, and the bin will wait for the next batch of boxes to be added in the following round.
- Once the volume utilisation exceeds 80%, the bin is considered locked. This means no further boxes can be added to this bin from the next batch, as it has reached its capacity limit. The bin is effectively "full" and no additional boxes are allowed to prevent overfilling.

It is important to note that smaller batches of boxes may require several arrivals before the bins reach the unpack threshold. This occurs because smaller batches might not immediately fill the bin to a level that pass the unpacking condition. As a result, the algorithm will continue to receive new batches and attempt further placements until the unpacking ratio is met and the bin reaches the required volume to not

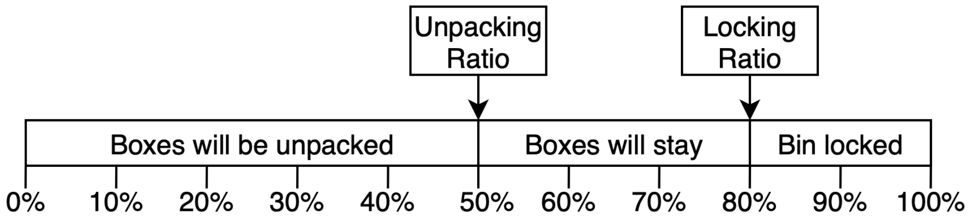


Figure 9. Unpacking and locking ratios based on bin volume utilisation.

trigger unpacking.

In the final batch, all remaining boxes are attempted for placement. This marks the last opportunity for the algorithm to fit any leftover items into the bins. Any boxes that cannot be placed due to space limitations or packing constraints are discarded. Discarding boxes ensures that only optimally packed bins are considered in the final solution, reducing the inefficiency of wasted space and guaranteeing that no further adjustments are needed.

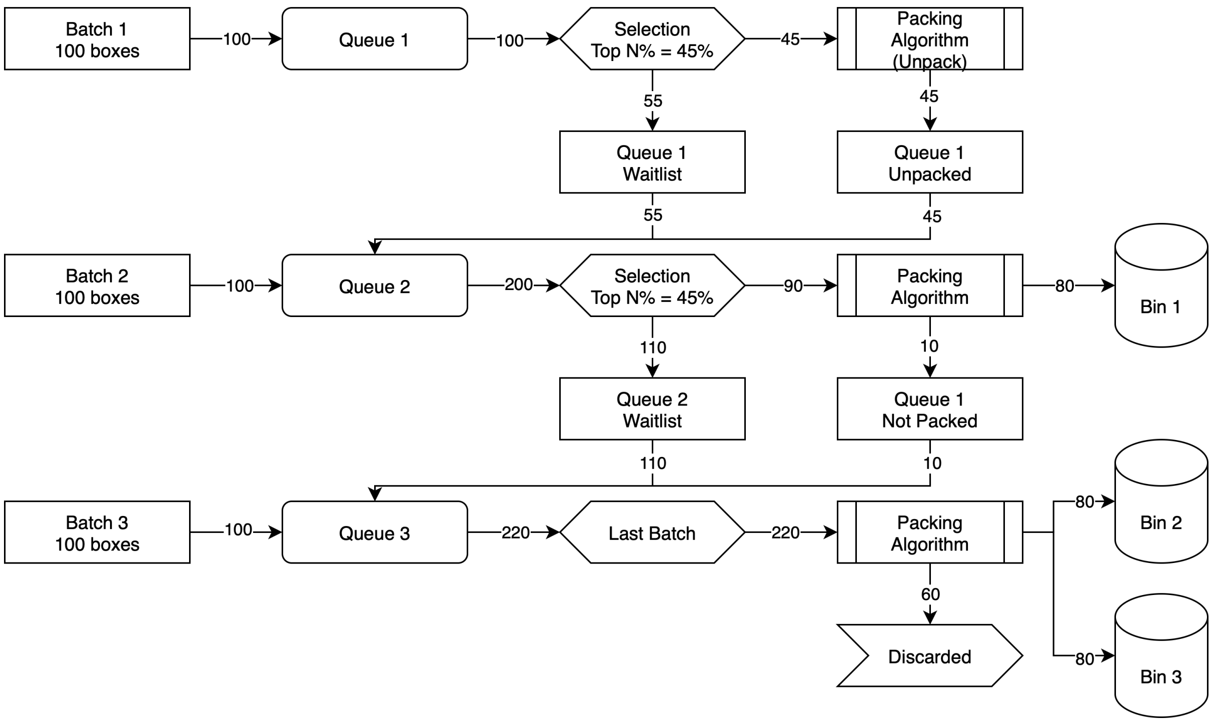


Figure 10. Batch arrival and processing example

Figure 10 shows an example of 300 boxes arriving in 3 batches of 100 boxes each, with the top N% set to 45%, a maximum of 3 bins, and assuming each bin having a fixed capacity of 80 boxes.

When the first batch arrives, the 100 boxes are analysed and sorted based on their value density. The top 45 boxes, representing 45% of the batch, are selected and fed into the packing queue. However, because these 45 boxes do not utilise enough of the bin's volume, the bin is unpacked, and these 45 boxes are released back into the queue. The previously waitlisted 55 boxes from the first batch are also added back into the queue, together with the 100 boxes from batch 2, forming a combined batch of 200 boxes.

Out of the combined 200 boxes, 90 boxes are selected for packing. Bin 1 fills quickly with 80 boxes. However, the remaining 10 boxes are not enough to fill another bin, Bin 2. Since Bin 2 does not yet reach the unpacking threshold, these 10 boxes are released back into the queue.

When batch 3 arrives, the 100 boxes from batch 3 are combined with the 110 boxes from the previous waitlist and 10 released boxes, creating a total of 220 boxes. As this is the last batch, all 220 boxes are attempted for placement into the bins. Bin 2 and Bin 3 are both filled with 80 boxes each. However, at this final stage, there is not enough space left for the remaining 60 boxes, and they are discarded.

This example illustrates how the algorithm handles the arrival of multiple batches, the selection of boxes based on value density, the management of bins based on unpacking and locking ratios, and the final attempt to place all remaining boxes.

E. Testing and Evaluation

Due to the flexibility of heuristic models, many parameters can be freely adjusted. To assess the quality of the proposed algorithm and identify the most effective configuration, a series of tests are carried out by varying key parameters. These include:

- Different types of box dimensions and properties
- Top value ratio (used for sorting and selecting high-value boxes)
- Number of batches
- Required support area ratio
- Padding distance between boxes
- Number of required supported corners
- Bin locking and unpacking ratios
- Total number of available bins

By systematically changing these parameters, the robustness and adaptability of the algorithm under different conditions can be evaluated.

1. The Testing Box Sets

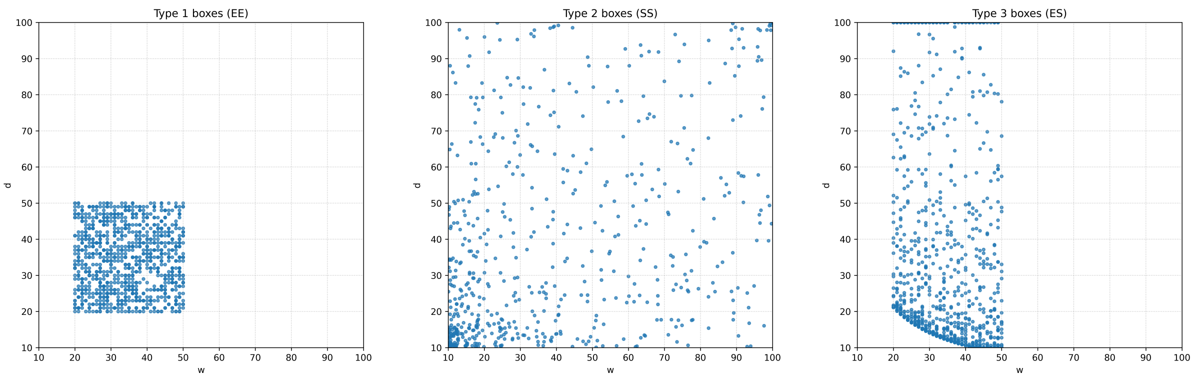
To study how box shape influences packing efficiency, four test sets are constructed with different width–depth combinations. The central goal is to evaluate whether the algorithm performs better with more regular (balanced) boxes or with more irregular (varied) ones.

To ensure fair comparison, Types 1 to 3 are designed to have the same total base area $\sum_i w_i d_i$ and similar height distributions, which leads to comparable total volume across these sets. This design isolates the effect of width–depth shape variation. Type 4 serves as a control case, also using the same total base area but allowing height to vary freely, resulting in a different total volume. This helps assess the impact of strict layering structure versus fully three-dimensional flexibility.

Type 1: Equal-Equal (EE): This is the most regular set. Both width (w) and depth (d) are drawn independently from the same discrete uniform range of 20 to 50 cm, leading to evenly balanced base dimensions. Height (h) is sampled from a narrow range of 25 to 30 cm to maintain consistent volume across boxes. As seen in Figure 11a, for 1000 boxes, the width–depth scatter plot shows a dense, symmetric grid covering a square area. The box bases are fairly square-like and uniform, which makes this set well-suited for layered packing strategies and minimises the challenges of irregular fitting.

Type 2: Sparse-Sparse (SS): This set is designed to have the same total base area as the Equal-Equal (Type 1) set, but with much higher variation between individual box sizes. The goal is to test how the packing algorithm responds when faced with a combination of very small and very large boxes, rather than a uniform spread. To achieve this, each box is first assigned a share of the total base area using a Dirichlet distribution with a very small concentration parameter. This creates an imbalanced allocation, where a few boxes receive large areas and many receive small ones. The sum of these areas still matches that of Type 1, maintaining consistency in total volume. Once each box is given a base area, it is split into width and depth using a random aspect ratio. The resulting dimensions are then clipped to ensure they fall within a practical range (typically 10 cm to 100 cm). The height is sampled uniformly from 25 cm to 30 cm, just as in Type 1. The outcome is shown in Figure 11b, where the base shapes are spread widely across the w-d space. Compared to Type 1, this set displays much more diversity in size and shape. Empirically, the standard deviation of base area is more than four times higher than in the Equal-Equal set. This provides a more challenging setting to evaluate the robustness of the packing algorithm under irregular and imbalanced input.

Type 3: Equal-Sparse (ES): This type serves as a hybrid between the balanced design of Type 1



(a) Type 1: Equal-Equal (EE) **(b) Type 2: Sparse-Sparse (SS)** **(c) Type 3: Equal-Sparse (ES)**

Figure 11. Width–depth scatter plots of box sets: (a) Type 1 (EE), (b) Type 2 (SS), and (c) Type 3 (ES).

and the irregular structure of Type 2. In this set, the width of each box is fixed using the same uniform distribution from 20 cm to 50 cm, while the depth is allowed to vary to create differences in base area. The process begins by assigning each box a target base area, using the same uneven distribution as in Type 2. However, because the width is fixed, the corresponding depth is computed by dividing the area by the width. If the resulting depth falls outside the allowed bounds (10 cm to 100 cm), the base area is adjusted through a clipping step to ensure the final depth stays within range. Once the depth is corrected, the base dimensions are set, and the height is again drawn uniformly from 25 cm to 30 cm. The result is a structure where widths are regular but depths vary significantly. As illustrated in Figure 11c, this generates a horizontal band of points in the w-d space, width remains in a narrow range while depth stretches broadly. This intermediate design allows testing whether the packing method benefits from partial regularity while still facing some variation.

Type 4: Equal-Equal-Equal (EEE): This control set extends the balance of Type 1 to all three dimensions. Width, depth, and height are each drawn independently from the same uniform range (20–50 cm), creating the most cube-like boxes in the test. This setup allows the boxes to be packed with fewer orientation constraints, testing the algorithm's performance under ideal, regular conditions. The main difference lies in the height, which now varies more broadly, removing the layer structure enforced in Types 1 to 3.

Types 1 to 3 share the same total base area and use narrow height ranges, resulting in similar total volumes. However, the spread of their base dimensions differs in variance as shown in Equation 4.

$$\mathrm{Var(EE)} < \mathrm{Var(ES)} < \mathrm{Var(SS)} \tag{4}$$

Type 4 shares the same total base area but allows height to vary over a wider range, which leads to different total volume. This setup enables a controlled comparison of how box regularity and dimensional variance affect packing efficiency, layer structure, and bin utilisation.

F. Hypotheses

Three performance measures are used in this study:

1) Packing volume utilisation
2) Value packed per bin
3) Value-to-volume ratio per bin

Each hypothesis is stated in terms of the expected effect on performance. While the main focus of some hypotheses is on volume utilisation or value packed, all three measures are analysed in section IV.

H1: Reducing the required supporting area ratio increases packing performance.

H2: Reducing the required number of supporting corners increases packing performance.

H3: Introducing padding distance increases packing performance.
H4: Packing similar box types (type 1 rather than type 2) increases packing performance.
H5: The choice of sorting method affects packing performance.
H6: Increasing the number of bins decreases the average packing efficiency per bin.

IV. Results & Discussion

This section presents the results of the study. The analysis is organised around the six hypotheses defined in section III. For each hypothesis, the effect of the tested change is examined using three primary performance measures: packing volume utilisation, value packed per bin, and value-to-volume ratio per bin. In addition, average density and average weight utilisation are included where relevant.

A. Support constraint sensitivity (H1–H3)

H1: Reducing the required supporting area ratio increases packing performance.
H2: Reducing the required number of supporting corners increases packing performance.
H3: Introducing padding distance increases packing performance.

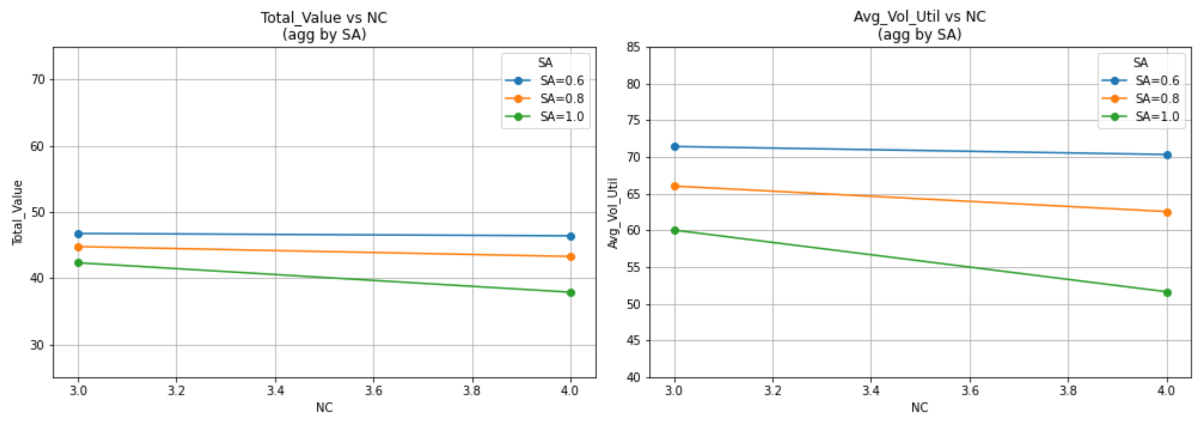


Figure 12. Effect of surface area (SA) requirement under varying NC.

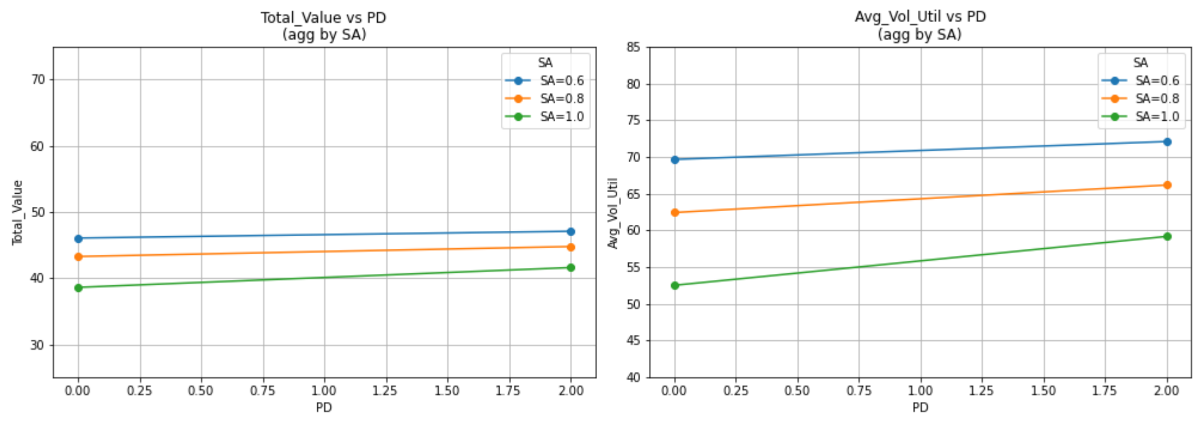


Figure 13. Effect of SA under varying padding distance (PD).

Results: These three factors all control how strict the stability condition is when placing a box on top of others. To isolate their effect, only Type 1 (Equal-Equal) boxes are used in this experiment.

The required supporting (surface area ratio (SA)) is varied from 0.6 to 1.0. Lower values allow partial support, giving more flexibility in stacking. The required number of supporting corners (NC) is set to either 3 or 4, where fewer required corners make it easier to place boxes. For the padding distance (PD),

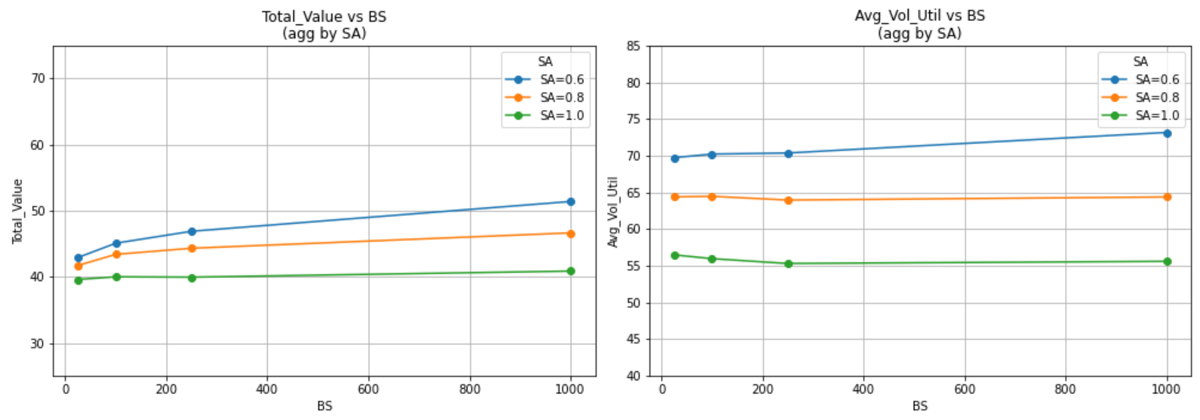


Figure 14. Effect of SA under varying batch size (BS).

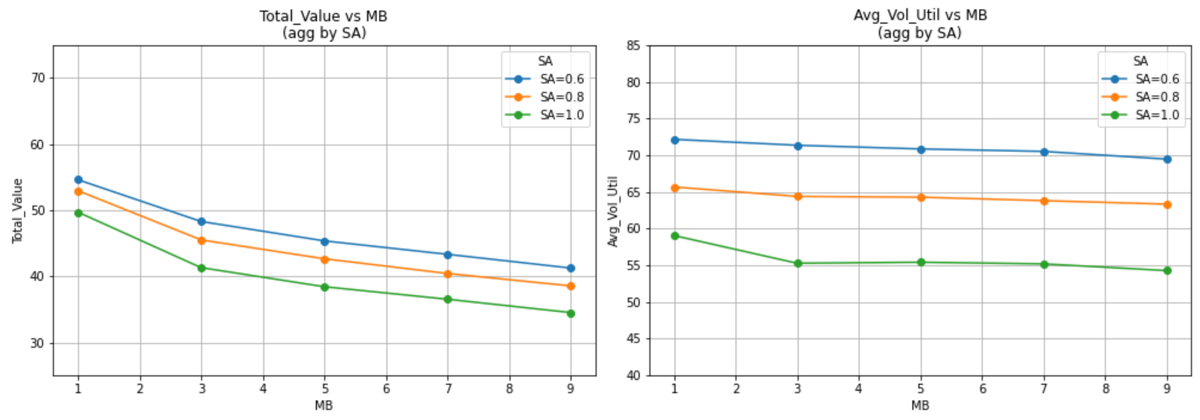


Figure 15. Effect of SA under varying max number of bins (MB).

two values are tested: 0 cm and 2 cm, with the latter allowing small gaps between touching boxes, which can relieve tight packing constraints.

In addition to these three main factors, other parameters such as batch size (BS), maximum number of bins (MB), sorting method, and bin unpack/lock ratio are varied across runs to provide a diverse set of scenarios. This ensures the observations are not tied to one specific setting but reflect consistent trends across conditions.

The results with respect to differences in the required supporting surface area ratio are shown in Figure 12, Figure 13, Figure 14, and Figure 15. In all cases, both the total value packed per bin and the average volume utilisation improve when the required supporting surface area ratio is set to 0.6. This setting consistently yields the highest performance, indicating that relaxing the surface support requirement allows the algorithm more flexibility in placing boxes effectively.

Similarly, when considering the number of required supporting corners, a comparable trend is observed. When the required number of corners is set to 4, the overall solution quality declines significantly. As shown in Figure 16, in the most restrictive case—where both the supporting surface area ratio is set to 100% and the number of corners is set to 4, the aggregated average volume utilisation drops to around 52%. In contrast, when both constraints are relaxed (area ratio at 60% and corners at 3), the utilisation reaches approximately 72%.

Lastly, when considering the padding distance, a consistent trend is also observed. As shown in Figure 17, when the padding distance is set to 2 cm, both the average volume utilisation and total packed value are noticeably higher than when no padding is allowed (0 cm). This shows that introducing a small tolerance improves flexibility in support detection, which in turn leads to better packing performance.

Discussion: All three hypotheses are supported by the results. Relaxing the required supporting

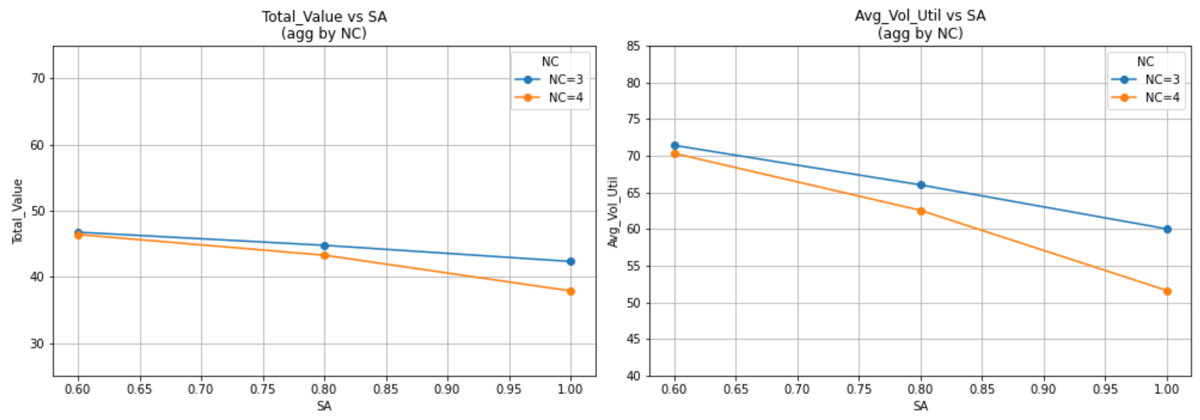


Figure 16. Effect of required surface area ratio at different corner support conditions.

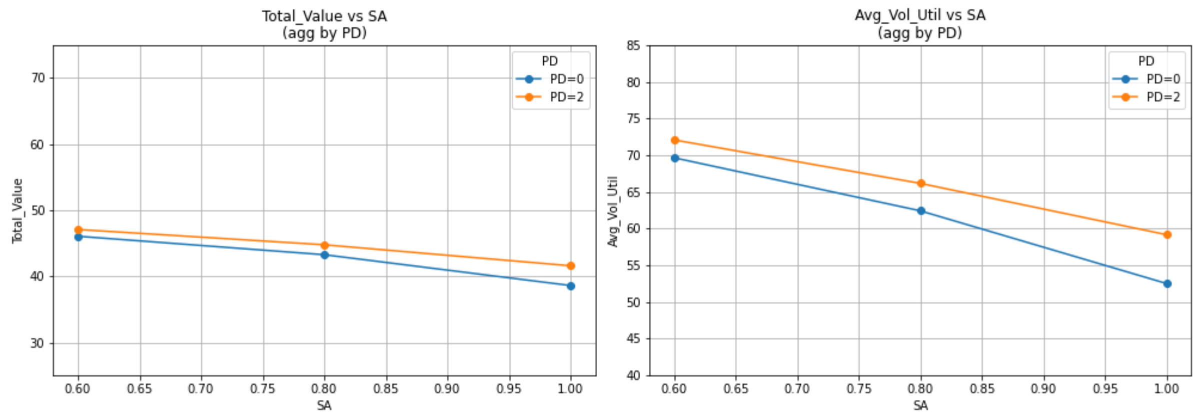


Figure 17. Effect of padding distance on packing performance, aggregated by supporting surface area ratio.

surface area ratio, reducing the number of required supporting corners, and introducing a small padding distance all lead to improved packing performance. In each case, more flexible stability conditions allow the algorithm to place more boxes successfully, resulting in higher total packed value and better volume utilisation.

Among the three factors, the supporting surface area ratio shows the strongest influence. Even when the number of corners is kept relaxed, a strict area ratio of 1.0 leads to noticeably worse results. On the other hand, setting the area ratio to a more permissive value like 0.6, even with a strict corner requirement of 4, still achieves relatively strong performance. This indicates that the area support condition is more dominant than the corner count. The effect of padding distance is more modest but consistently positive. Allowing a small vertical buffer (e.g., 2 cm) improves results, particularly in scenarios where small alignment gaps would otherwise prevent larger boxes from being placed on top. This flexibility helps recover viable placements that would be lost under stricter no-gap conditions.

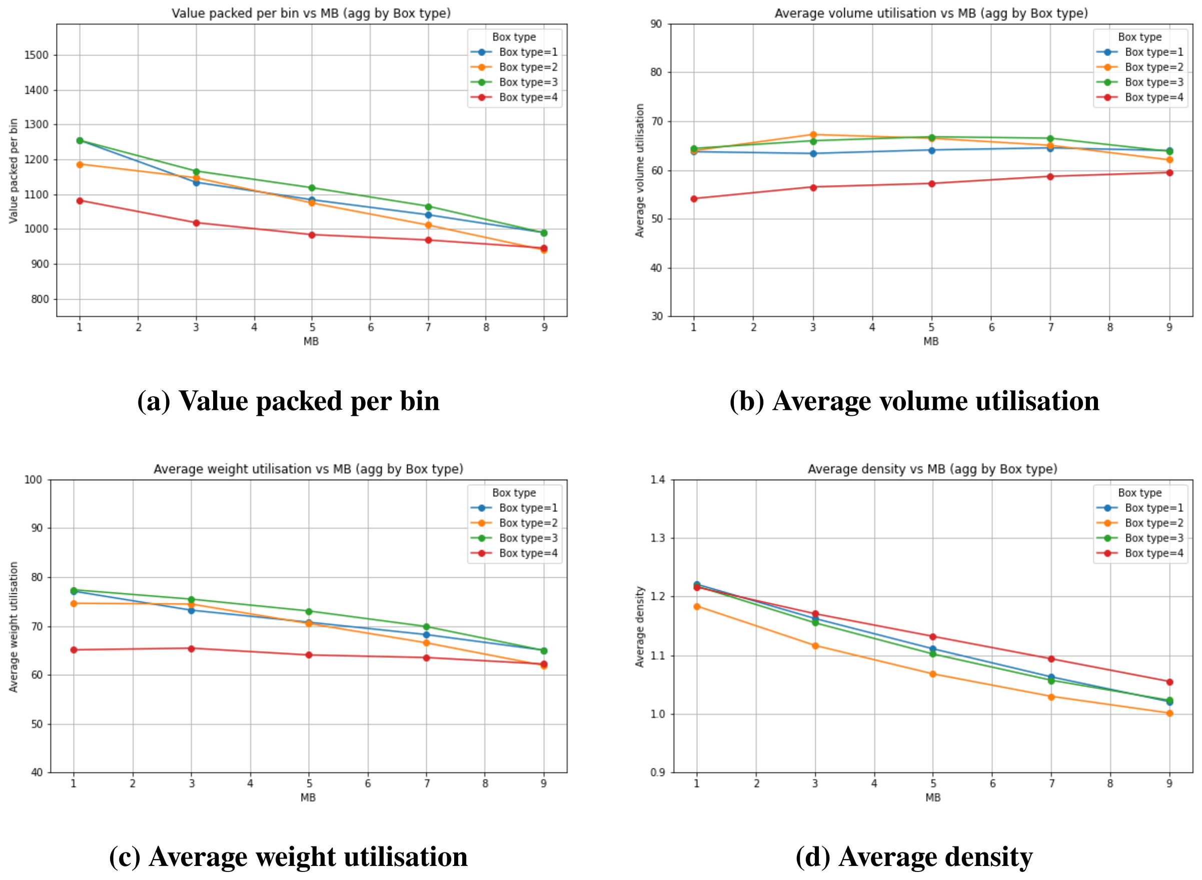
B. Hypothesis 4

H4: Packing similar box types (type 1 rather than type 2) increases packing performance.

Results: To evaluate the effect of different box dimension types on packing performance, tests were conducted using all four box types defined earlier. Recall that Type 1 (Equal-Equal) contains boxes with relatively consistent width and depth, while Type 2 (Sparse-Sparse) introduces a wide spread of box base areas. Type 3 (Equal-Sparse) lies between the two, with one fixed side and one varied side. Type 4 (Equal-Equal-Equal) consists of cube-like boxes with all dimensions drawn from the same balanced distribution.

Box value is defined as value = weight × k, where k is a boosting factor that equals 2 with 10% probability and 1 otherwise, introducing occasional uneven increases in box value. Weight is calculated as the product of box volume and a randomly sampled density, with density drawn from a uniform distribution between 150 and 450 kg/m^3. As a result, value density becomes proportional to $k \times \sqrt{\text{density}}$. Since weight is usually not a limiting factor, when two boxes share the same dimensions, the heavier one will normally be selected. However, because $\sqrt{3} > 2$, boxes with a value boost (i.e., $k = 2$) will be prioritised in sorting, even if they are lighter.

The results are shown in Figure 18. The x-axis indicates the maximum number of bins allowed, and each line represents performance for a different box type. Four performance metrics are included: value packed per bin, average volume utilisation, avergae weight utilisation, and average density.



(a) Value packed per bin **(b) Average volume utilisation**

(c) Average weight utilisation **(d) Average density**

Figure 18. Packing performance for different box types (Equal-Equal, Sparse-Sparse, Equal-Sparse, Equal-Equal-Equal).

Focusing on Types 1 to 3, which are constructed to have similar volume profiles and height ranges, the difference in performance is not as clear-cut as initially expected. Across most metrics, the lines for the three types often overlap or cross each other. No single type consistently outperforms the others across all conditions. This suggests that box regularity, as defined here, may not strongly determine overall packing outcome.

One small distinction is observed in the average density per bin, estimated by dividing average weight utilisation by volume utilisation. In this case, Type 2 appears to produce bins with slightly lower average density compared to the other types. The reason for this is unclear, and may be due to incidental box selection or sorting effects.

Comparing Type 1 and Type 4, it is important to note that Type 4 uses a broader height range and has a higher average height (27.5 cm for Type 1 vs. 35 cm for Type 4). This leads to fewer boxes being packed into each ULD, even when volume utilisation is similar. Figure 18b shows that Type 4 consistently yields the lowest volume utilisation among all types. This is likely due to reduced flexibility in forming compact height layers, making it harder to achieve tight vertical stacking.

Discussion: The results do not support Hypothesis 4. Among box Types 1 to 3, which have similar height and volume, no clear performance difference is observed. Their metrics overlap and vary without a consistent trend. Although Type 2 shows slightly lower average density, the reason is unclear. Type 4, which differs in height profile, performs worse due to reduced flexibility in forming height layers. Overall, base dimension regularity does not reliably improve packing efficiency under the current method.

C. Hypothesis 5

H5: The choice of sorting method affects packing performance.

Results: The tests compare two sorting methods: value only sorting (Method 3) and value sorting with free rotation to reduce the number of distinct heights (Method 4). Pure height based rules, such as tallest first or most frequent height, were not included. In batches with low variance in box size (box types 1 to 3), these rules offer little beyond the fallback behaviour already present in Method 4. In batches with type 4 boxes, reducing the number of distinct heights can be beneficial, and this effect is already captured by Method 4. In practice, when heights cluster, the most frequent height rule operates as a fallback within Method 4. The numbering of methods remains at 3 and 4 because Methods 1 and 2 were considered in the design phase but not tested.

The results are presented in Figure 19. For volume utilisation, using sorting Method 4 (value plus dimension) yield better result. As for when checking for values, although method 4 performs better, the difference is not that siginificant.

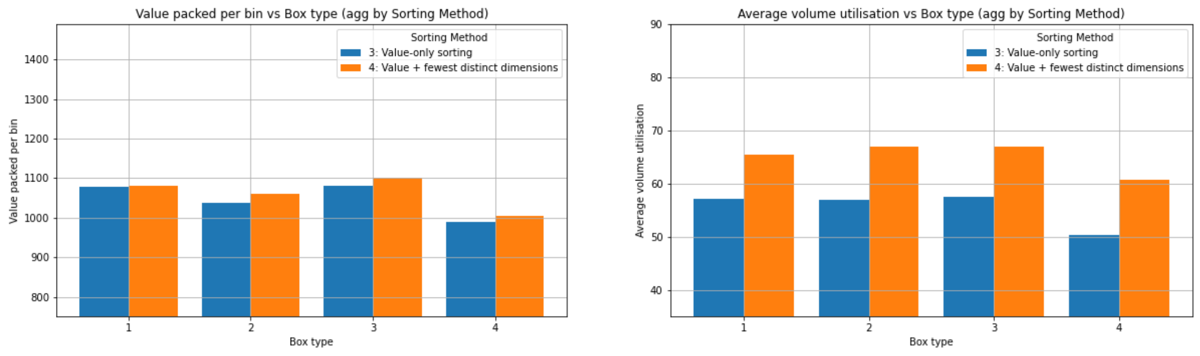


Figure 19. Results for differnet Box type

Discussion: The results confirm that incorporating dimension information into the sorting method is key to improving packing efficiency. Method 4, which accounts for both value and box dimensions, consistently achieves better volume utilisation than value-only sorting (Method 3). This shows that reducing height variation and encouraging more structured layers helps minimise wasted space. Even in value-focused metrics, Method 4 performs at least as well, reinforcing the benefit of dimension-aware sorting.

D. Hypothesis 6

H6: Increasing the number of bins decreases the average packing efficiency per bin.

Results: The results related to changes in the maximum number of bins (from 1 to 9) are shown in Figure 18 and Figure 20. In terms of volume utilisation, increasing the number of bins leads to a generally non-increasing trend. The algorithm fills the first few bins more effectively, but later bins tend to have lower utilisation.

When considering value packed per bin, the decline is clearer. As the number of bins increases, average value per bin drops. This occurs because high-value boxes are prioritised early in the sorting process, filling the first few bins, while remaining lower-value boxes are spread across the later bins.

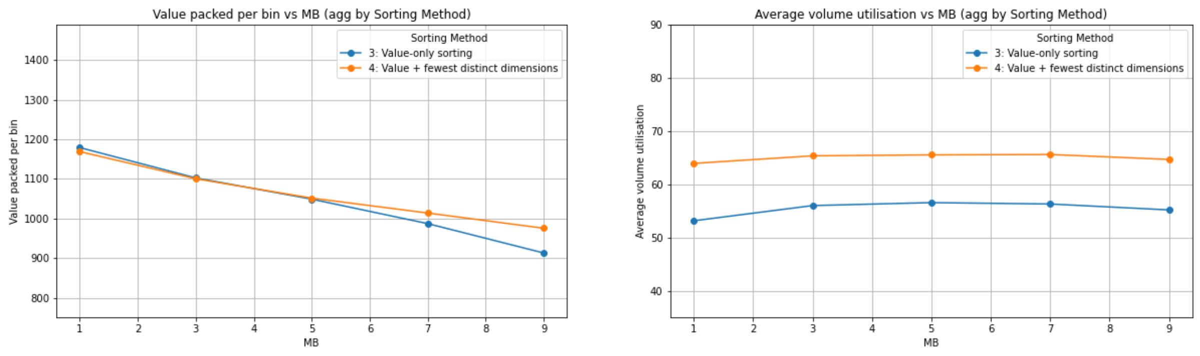


Figure 20. Results for differnet number of bins

Discussion: These results support the hypothesis. Increasing the number of bins reduces the average value and space usage per bin. The algorithm front loads high-value boxes, leading to lower efficiency in later bins.

V. Conclusion and Recommendations

This work presents a heuristic framework for value driven three-dimensional bin packing under realistic constraints such as batch arrivals, stability requirements, and handling of fragile boxes. The algorithm is built upon the extreme point placement principle, and extended with a two stage box selection mechanism: first filtering by value density and then applying a dimension-aware sorting method.

The framework introduces and evaluates multiple conditions relevant to practical packing: minimum supporting surface area, required number of supported corners, and a tolerance based padding distance to bridge minor misalignments. A dedicated process is also included to handle fragile boxes safely, placing them only when enough structural support is available. Batch arrivals are modelled using a flexible unpack/lock thresholding strategy, simulating staged box availability in a way that mimics real world logistics flows.

The testing confirms several key observations. First, relaxing the stability constraints either by lowering the required support area, reducing the number of supporting corners, or allowing minimal gaps through padding—directly improves packing performance. These conditions provide the algorithm with more placement opportunities, especially when box alignment or bin shape creates awkward voids.

Second, the use of dimension-aware sorting (Method 4), particularly through reducing the number of distinct height layers, leads to clear gains in volume utilisation. This effect becomes more prominent when boxes are varied in shape, such as in Type 4 sets. By encouraging consistent stacking layers, the algorithm reduces vertical waste and achieves tighter fits. On the other hand, pure value based sorting (Method 3) performs similarly in value terms but falls short in spatial efficiency.

Contrary to expectations, the hypothesis that more regular box shapes would result in better performance was not supported. In fact, irregular shapes as seen in the Sparse-Sparse (Type 2) set often gave better results. These boxes, with a wider spread of sizes, seem to offer more flexibility in filling small voids that occur naturally during packing. Uniform boxes, while easier to stack, can leave persistent unused pockets, especially near bin edges or under partial layers.

Finally, increasing the number of available bins reduces the average performance per bin. When more bins are allowed, the algorithm tends to pack more boxes overall, but the quality of each bin decreases, particularly in terms of value per bin. This is expected, as high-value boxes are used early, leaving lower-value boxes to fill the remaining space. The result is lower per-bin efficiency despite higher total throughput.

Recommendations

Although the framework includes a top value filtering mechanism in the first stage of box selection, the results from this feature were inconsistent and are not included in the analysis. In theory, filtering the highest value boxes from each batch should increase overall value efficiency, especially when space is tight. However, under batch handling, this filtering introduced irregularities, particularly when small batch sizes and strict lock ratios were used. Further controlled testing is recommended to better understand how value filtering interacts with batch arrivals. This could involve varying the proportion of selected boxes more dynamically, or adapting the filtering based on how full the bins are.

Additionally, further improvements could be made by incorporating better height grouping methods. While Method 4 reduced height variation by allowing rotation, it still relied on implicit grouping. More explicit layer formation or clustering strategies could be used to structure layers before placement. This would be particularly helpful in scenarios with non-rotatable or fragile items.

Furthermore, more testing could be done to simulate real life scenarios more closely, as the current model fixes the total to 1000 packages with equal sized batches. This does not capture the actual flow of operations, where order volumes rise and fall across the day. Introducing uneven batch sizes and variable arrival patterns would provide a more accurate picture of system performance and show how well the methods adapt under realistic demand changes.

Lastly, while padding distance proved useful in reducing over-constraint in support checks, future versions could explore adaptive padding rules (e.g., tolerating larger gaps for lighter boxes or requiring tighter fits near bin walls). This could further balance support realism and packing flexibility.

Overall, the framework provides a practical and adaptable method for solving complex packing problems, with promising results under a range of realistic conditions. However, several areas, particularly around batch interaction remain open for further tuning and refinement.

References

[1] Air Transport Action Group, "Aviation: Benefits Beyond Borders 2024," , 2024.

[2] Bortfeldt, A., and Wäscher, G., "Constraints in container loading – A state-of-the-art review," *European Journal of Operational Research*, Vol. 229, No. 1, 2013, pp. 1–20. doi: 10.1016/j.ejor.2012.12.006.

[3] Paquay, C., Schyns, M., and 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*, Vol. 23, No. 1-2, 2016, pp. 187–213. doi: 10.1111/itor.12111.

[4] Gutin, G., Jensen, T., and Yeo, A., "Batched bin packing," *Discrete Optimization*, Vol. 2, No. 1, 2005, pp. 71–82. doi: 10.1016/j.disopt.2004.11.001.

[5] Martello, S., Pisinger, D., and Vigo, D., "The Three-Dimensional Bin Packing Problem," *Operations Research*, Vol. 48, No. 2, 2000, pp. 256–267. doi: 10.1287/opre.48.2.256.12386.

[6] Baker, B. S., "A new proof for the first-fit decreasing bin-packing algorithm," *Journal of Algorithms*, Vol. 6, No. 1, 1985, pp. 49–70. doi: 10.1016/0196-6774(85)90018-5.

[7] Crainic, T. G., Perboli, G., and Tadei, R., "Extreme point-based heuristics for three-dimensional bin packing," *Informs Journal on computing*, Vol. 20, No. 3, 2008, pp. 368–384. doi: 10.1287/ijoc.1070.0250.

[8] Lodi, A., Martello, S., and Vigo, D., "TSpack: a unified tabu search code for multi-dimensional bin packing problems," *Annals of Operations Research*, Vol. 131, 2004, pp. 203–213. doi: 10.1023/B:ANOR.0000039519.03572.08.

[9] Faroe, O., Pisinger, D., and Zachariasen, M., "Guided local search for the three-dimensional bin-packing problem," *Informs journal on computing*, Vol. 15, No. 3, 2003, pp. 267–283. doi: 10.1287/ijoc.15.3.267.16080.

[10] Parreño, F., Alvarez-Valdés, R., Tamarit, J. M., and Oliveira, J. F., "A maximal-space algorithm for the container loading problem," *INFORMS Journal on Computing*, Vol. 20, No. 3, 2008, pp. 412–422. doi: 10.1287/ijoc.1070.0254.

[11] Li, X., Zhao, Z., and Zhang, K., "A genetic algorithm for the three-dimensional bin packing problem with heterogeneous bins," *IIE Annual Conference. Proceedings*, Institute of Industrial and Systems Engineers (IISE), 2014, p. 2039.

[12] Egeblad, J., and Pisinger, D., "Heuristic approaches for the two-and three-dimensional knapsack packing problem," *Computers & Operations Research*, Vol. 36, No. 4, 2009, pp. 1026–1049. doi: 10.1016/j.cor.2007.12.004.

[13] Paquay, C., Limbourg, S., and Schyns, M., "A tailored two-phase constructive heuristic for the three-

dimensional multiple bin size bin packing problem with transportation constraints," *European Journal of Operational Research*, Vol. 267, No. 1, 2018, pp. 52–64. doi: 10.1016/j.ejor.2017.11.010.

[14] Oliveira, Ó., and Gamboa, D., "Adaptive sequence-based heuristic for the two-dimensional non-guillotine bin packing problem," *Hybrid Intelligent Systems: 18th International Conference on Hybrid Intelligent Systems (HIS 2018) Held in Porto, Portugal, December 13-15, 2018 18*, Springer, 2020, pp. 370–375. doi: 10.1007/978-3-030-14347-3_36.

This page intentionally left blank.

II

Research Proposal

This page intentionally left blank.

1 Introduction

The global air cargo industry has shown strong growth following the COVID-19 pandemic, driven primarily by an increase in e-commerce, supply chain shifts, and technological advancements. According to the International Air Transport Association (IATA), global air cargo demand in 2021 was 6.9% higher than pre-pandemic levels from 2019 and 18.7% above 2020 figures (IATA, 2021). Boeing further forecasts a continued annual growth rate of approximately 4%, expecting air cargo traffic to double over the next two decades (Boeing, 2024).

Major cargo operators have also benefited significantly from this increased demand. FedEx and UPS recorded unprecedented parcel volumes driven by a surge in e-commerce activity during and after the pandemic. FedEx alone reported a 25% rise in daily package volume compared to pre-pandemic levels, prompting substantial investment in new aircraft and infrastructure (Reuters, 2021).

Despite this growth, optimising cargo operations remains a challenging task due to the complexity of packing goods efficiently into aircraft Unit Load Devices (ULDs). Efficient loading is crucial because poorly organised cargo can lead to reduced utilisation of available space, increased fuel consumption due to imbalance, and higher operational costs. Addressing these issues requires effective strategies that incorporate both spatial optimisation and load balancing.

This research addresses the optimisation challenge in air cargo loading by developing a novel approach combining three-dimensional bin packing techniques with load balancing and revenue optimisation considerations. The approach builds on recent advancements in heuristic methods, specifically the use of extreme-point-based heuristics, to effectively position cargo items within aircraft containers, minimising wasted space and ensuring balanced weight distribution. Additionally, by recognising similarities to the three-dimensional knapsack problem, this research explicitly considers revenue and profitability criteria, allowing cargo operators to selectively prioritise higher-value shipments when facing spatial constraints. Current bin-packing heuristics, although effective for maximising space usage within individual containers, typically neglect global load distribution and revenue-driven decisions across multiple containers, leading to suboptimal economic and operational outcomes when applied at scale.

Therefore, this research proposes an iterative methodology integrating three key components: a refined merit function to evaluate cargo placement considering load balance and revenue optimisation; dynamic sequencing of cargo items that adaptively responds to evolving spatial constraints; and multi-container optimisation that simultaneously assesses potential placements across several containers. The objective is to overcome the limitations of traditional sequential packing methods, reducing wasted space and enhancing the overall efficiency of cargo loading processes.

The primary research question is: *How can a 3D knapsack-based extreme-point heuristic approach be developed to maximise the overall profit in air cargo loading?* This question will be addressed by developing a model that integrates profitability metrics and operational costs, dynamically sequences cargo items, and simultaneously evaluates multiple container placements. The model aims to improve operational efficiency, support profitability, and enhance environmental sustainability through reduced fuel consumption associated with improved weight distribution.

The rest of this document is structured as follows: Chapter 2 reviews established bin packing techniques, followed by Chapter 3, which highlights gaps in current approaches. Then, Chapter 4 specifies the central research question and associated subquestions, and finally, Chapter 5 presents the research plan.

2 Literature Review

The bin packing problem (BPP) is a classic combinatorial optimisation problem in which a set of items must be packed into a minimum number of fixed-size bins without overflow. This problem is NP-hard, making it computationally challenging to solve optimally for large instances. BPP and its variants have significant practical importance in logistics and industry, with applications ranging from loading cargo into containers and pallets to cutting stock materials and resource allocation in computing. These real-world contexts demand efficient packing of items, highlighting the need for effective solution methods.

Exact solution methods such as mixed-integer linear programming (MILP) can solve small BPP instances but struggle with larger ones due to exponential complexity. For example, Paquay et al. (2016) proposed a MILP formulation for a three-dimensional bin packing problem in an air cargo context, but even with advanced solvers these approaches are limited by high computation times. Similarly, Martello et al. (2000) introduced the concept of corner points and developed an exact branch-and-bound algorithm for 3D bin packing; while effective for a single bin or very small problems, such exact algorithms become impractical as instance size grows. This computational intractability has motivated the development of heuristic and metaheuristic methods, which provide near-optimal solutions in reasonable time for large-scale BPP instances.

Numerous heuristic approaches have been proposed to tackle multi-dimensional bin packing. Building on the concept of corner points, Crainic et al. (2008) introduced the concept of extreme points – a reduced set of candidate positions obtained by projecting item corners – and derived a constructive heuristic that significantly improves packing efficiency. To deal with multiple bins, metaheuristics have been applied: Lodi et al. (2004) developed TSpack, a unified tabu search framework for multi-dimensional bin packing, which iteratively attempts to reallocate items among bins to reduce their number. Faroe et al. (2003) proposed a guided local search (GLS) strategy that iteratively removes one bin from the solution by redistributing its items, effectively guiding the solution toward using fewer bins. In a similar way, Crainic et al. (2009) designed a two-level tabu search called TS2PACK that first seeks to minimise the number of bins and then optimises the packing of items within each bin. Other local search based methods have also emerged: Ceschia and Schaerf (2013) presented a local search approach for a complex multi-drop, multi-container loading variant of BPP, and Trivella and Pisinger (2016) proposed a multi-level local search heuristic using interval graphs to improve packing arrangements and even address load-balancing concerns.

Another class of effective heuristics is based on greedy randomised strategies. Parreño et al. (2008) introduced a Greedy Randomised Adaptive Search Procedure (GRASP) that uses a maximal-space heuristic, dynamically choosing placements that maximise the remaining free space in a container. This GRASP approach was later enhanced: Parreño et al. (2010) combined GRASP with a Variable Neighborhood Descent (VND) local search to further refine packing solutions, and Alvarez-Valdes et al. (2013) integrated GRASP with Path Relinking to intensify the search for improved packings. In addition to these, other heuristic approaches have contributed to the literature: Chan et al. (2006); Egeblad and Pisinger (2009) developed a simulated annealing based heuristic for multi-dimensional packing problems; and Baldi et al. (2012) investigated a generalised bin packing variant (with additional profit considerations) and proposed effective heuristic solutions.

2.1. Extreme Points in Bin Packing

2.1.1. Foundation Work

Martello et al. (2000) – Corner Points: Instead of allowing items to be placed anywhere in a bin, they restricted attention to certain "promising" positions. Corner points are defined as the non-dominated locations where a new box can potentially be placed given the current layout. In essence, these are the empty corners formed by the placed items and container walls – no other feasible point lies simultaneously below and to the left/back of a corner point in all dimensions. By considering only such corner positions (e.g. the first box starts at the origin corner, and subsequent boxes at the exposed corners of existing ones), their algorithm dramatically pruned the search space of placements. This concept ensured that any optimal packing can be achieved by placing each new item at a corner of previously placed items (or the bin) without losing generality. Martello et al. leveraged corner points in an exact branch-and-bound algorithm for the 3D Bin Packing Problem (3D-BPP) and also as a heuristic rule, showing substantial improvement over arbitrary placement strategies in terms of packing density.

Crainic et al. (2008) – Extreme Points: Crainic, Perboli, and Tadei built upon the corner point idea with a refined concept called extreme points. Extreme points (EPs) are essentially a generalisation of corner points with a more systematic generation rule. These are the set of viable empty coordinates (against bin walls or against the surfaces of already-packed items) where a new item could be placed such that it touches an existing surface without overlapping anything. The extreme point rule they proposed is independent of any specific packing instance – it provides a generic way to enumerate candidate positions inside the container.

Key Improvements of Extreme-Point Methods: Extreme point heuristics offer clear benefits over earlier placement rules:

- Reduced Search Space: By focusing only on corner/extreme point positions, the algorithm avoids scanning every possible coordinate in the bin. This yields a finite set of meaningful placement options for each item, greatly improving efficiency while still ensuring no optimal solution is missed. Earlier "anywhere fit" heuristics or layer-based methods left too many possibilities or underutilised space.
- Better Space utilisation: EPs allow filling of complex gaps. New items are always placed flush against existing ones or walls, which helps to consume leftover space more tightly. This often leads to higher packing density compared to approaches that, say, stack items in fixed layers or rows. Crainic et al. note that their EP-based packings use bin volume more fully, reducing voids and ultimately requiring fewer bins.
- Generality and Constraints Handling: The extreme point rule is generic and does not depend on problem-specific tuning. It can naturally incorporate additional practical constraints. For example, Crainic et al. emphasise that EPs can handle cases like pre-placed/fixed items or exclusion zones within a container. Later researchers extended EP heuristics to accommodate weight limits, fragility, and stability constraints without altering the fundamental placement generation logic. This is harder to achieve in methods that pack by predefined layers or splits.
- Improved Solution Quality: Overall, heuristics based on extreme points produced some of the best-known results for 3D-BPP in the late 2000s. In fact, Crainic et al. reported that even when applied to 2D bin packing, their EP-based algorithm outperformed dedicated 2D algorithms – a testament to the effectiveness of the approach.

2.2. Sequencing

After defining where items can be placed (the extreme points), the next step is to determine in what order to place the items. The sequencing of items significantly impacts the packing outcome – a good order can avoid dead space, while a poor order may leave unusable gaps or force early bin closures. Researchers have explored both static and dynamic sequencing (deciding the next item on the fly during the packing process).

Static sequencing means the packing order is predetermined before the packing process begins. All items are sorted according to some key (for example, by volume, weight, base area, etc.), and then that fixed order is followed during packing. Classic algorithms like First-Fit Decreasing in bin packing use static ordering – e.g. sort boxes by descending size, then pack in that order. The appeal of static sequencing is its simplicity: it requires only an initial sort and then a straightforward placement loop, which is computationally efficient. However, a static sequence does not adjust to the actual packing configuration, so it might not yield the optimal fill for every instance.In contrast, dynamic sequencing (also called online or adaptive ordering) means the next item to pack is chosen on the fly based on the current state of the packing. Instead of a fixed list, the algorithm decides at each step which remaining item is best to place next, often by evaluating some heuristic criterion (like which choice leaves the least leftover space or best fills a particular gap). Eley (2003) introduced an early example of dynamic ordering: his heuristic considers the unused volume remaining after a potential placement and picks the next box that minimises wasted space. In other words, it dynamically selects the item that "fits" best in the moment. This adaptability usually improves space utilisation (since the algorithm can react to the configuration of packed items) at the cost of more computation per placement. Dynamic sequencing tends to explore the packing state more thoroughly – Zhao et al. (2016) note that static methods use a one-time sort (e.g. by volume or height), whereas dynamic methods continuously adjust the choice based on the evolving free space. The trade-off is that dynamic sequencing requires evaluating multiple candidates at each step, which can be slower. Nevertheless, many studies have found that dynamic ordering often yields better-packed solutions than a single fixed ordering, especially for highly heterogeneous item sets.In summary, sequencing plays a pivotal role: a static sequence offers speed and simplicity by committing to a predefined order, while a dynamic sequence offers adaptability by choosing items based on the current packing situation. Both approaches appear in the literature, and many heuristics incorporate one or the other (or even hybrids) to balance run-time and packing efficiency.

Both Crainic et al. (2008) and Paquay et al. (2018) use static sequencing to optimise packing efficiency. Crainic et al. employ the Clustered Area-Height rule, sorting items by base area and height before placement via an extreme point heuristic. Paquay et al. adopt the same ordering in their two-phase heuristic, ensuring a fixed sequence throughout packing. In both cases, sequencing is predetermined and unchanged, directly influencing packing performance by reducing bin usage and improving space utilisation. A variety of other heuristic bin packing studies in the last two decades have also addressed sequencing. Many follow the pattern of using static sorted orders. For example, Mack and Bortfeldt (2012) developed a wall-building algorithm for large 3D bin packing problems, in which items are sorted (by size) and packed into "walls" or layers one by one. This is a static sequencing approach: the algorithm predetermines an order (and grouping into layers) then places each item in turn. Similarly, Costa and Captivo (2016) proposed a layer-based heuristic (for mixed container loading) that relies on sorting boxes by base area to create layers, again a static sequence within each layer. On the other hand, some researchers explicitly employ dynamic sequencing. Eley (2003), as mentioned, is a notable early example – his heuristic chooses the next box based on the current free space, effectively re-evaluating the choice at each step rather than sticking to a global sorted list. Another dynamic approach is seen in Parreño et al. (2008), who designed a GRASP-based 3D packing algorithm: in the constructive phase, they do not use a fixed order, but rather repeatedly select the next item from a candidate list based on a pseudo-random greedy function that accounts for current space utilisation (a form of dynamic, adaptive sequencing). This allows different orderings in different iterations of the GRASP, seeking an improved packing by adjusting sequences. Metaheuristic and AI-based methods also treat sequencing as a key decision: Li et al. (2014) developed a genetic algorithm that encodes the packing sequence in the chromosome and uses a decoder to pack items in that order, effectively searching for an optimal static sequence through evolution. Meanwhile, Oliveira and Gamboa (2020) proposed an adaptive sequence-based heuristic which iteratively refines the item order: they generate an initial sequence, pack items, then adjust the order based on the result and repeat, blending static and dynamic ideas by searching the space of sequences. In specialise applications like multi-stop loading, the sequence might be externally constrained Junqueira et al. (2012) and Martínez et al. (2015) consider vehicle loading where the packing order must align with delivery drop-off order (to avoid rehandling), essentially imposing a predetermined static sequence by route design. These studies highlight that sequencing methods vary widely: some stick to one fixed order (static), others choose items adaptively (dynamic), and some hybrid approaches generate and evaluate multiple sequences to find a good one.

2.3. Merit Function

In heuristic bin packing algorithms, a merit function (or scoring function) is a criterion used to evaluate how "good" a potential placement is. It assigns a score to an item–position (or item–bin) combination, guiding the heuristic to choose which bin or position to pack next. For example, a simple merit function might be the negative leftover volume in a bin after placing an item – equivalent to choosing the bin where the item leaves the least unused space (the classic Best Fit heuristic). In effect, the merit function is the decision rule that drives the packing process, prioritising certain placements over others based on desired outcomes (e.g. maximising space usage or maintaining balance). Merit functions are crucial because they directly influence packing efficiency and feasibility. A well-designed merit function helps a heuristic emulate the objective of the bin packing problem (like high space utilisingation or minimal bin count) while respecting constraints. Different greedy heuristics (First Fit, Best Fit, etc.) can be seen as using different merit rules – First Fit uses a boolean merit (fit in the first open bin), whereas Best Fit uses a quantitative merit (minimise leftover space). In more complex 2D/3D packing, especially with additional constraints, the merit function often combines multiple factors into a single score. It effectively acts as a proxy objective guiding each placement step toward a good overall solution.

Various strategies have been proposed in the literature to define merit functions for 3D bin packing, each reflecting different priorities: Crainic et al. (2008) defined the Residual Space (RS) at an extreme point as the free volume around that point in each dimension (length, width, height). Among the several merit functions they tested, the best was one that maximises the utilisingation of this residual space. In practice, this merit function selects the placement that minimises the leftover gaps along each axis after accommodating the new box. Formally, for a given box i and a candidate position (extreme point) with residual free lengths (RS_e^x, RS_e^y, RS_e^z), their merit function was: $MF_1 = (RS_e^x w_k) + (RS_e^y d_k) + (RS_e^z h_k)$. This sum of differences is small when the box nearly fills the available space in all directions – effectively a Best Fit criterion generalised to 3D space. By choosing the EP that minimises MF_1 (i.e. leaves the least total slack in x,y,z), the algorithm achieves a very tight packing. Crainic et al. reported that this residual-space metric outperformed other rules in terms of packing more items per bin. The resulting heuristic, often dubbed EP-BFD (Extreme Point Best Fit Decreasing), was shown to outperform earlier constructive approaches on packing efficiency, though it did not consider weight balance (weight was assumed homogeneous or not constrained in their 2008 study).

Building on earlier work, Paquay et al. (2018) developed a tailored two-phase constructive heuristic for the air-cargo 3D bin packing problem. In Phase 1, it packs boxes into a single ULD type using an improved extreme-point strategy, and in Phase 2 it assigns the packing pattern into multiple available ULD types. A distinctive aspect of this heuristic is the use of two alternative merit functions to guide box placement, reflecting a trade-off between pure space utilisingation and load balance. The first merit function is an adoption of Crainic's residual-space metric: it picks the extreme point that best fits the box dimensions, essentially minimising leftover space just like the EP-BFD rule. The second merit function directly addresses weight distribution: it computes the distance between the current centre of gravity (CG) of the load in a ULD and the geometric centre of that ULD's floor area. In formula form, MF_1 is proportional to the distance $MF_2 = \sqrt{(x_{CG} - x_M) + (y_{CG} - y_M)}$, i.e. how far the load's centre shifts toward the edges. A smaller MF_2 means the weight is more centrally balanced. By selecting placements that minimise this distance, the heuristic tries to keep the load balanced as it packs. Paquay et al. allow the algorithm to use either MF_1 or MF_2 and compare their performance. They found that MF_1 (residual space focused) achieved higher packing efficiency (better volume utilisingation and fewer bins) than MF_2 in general. However, using MF_1 alone can result in ULD loads that are unbalanced (CG off-centre). To handle this, their heuristic includes a post-processing step: after packing with the chosen merit function, if a ULD's load is out of the allowed balance range, the solution is adjusted (e.g., by swapping or shifting boxes between sides of the container) to improve weight distribution. This two-phase approach (pack for efficiency, then re-balance if needed) shows how merit functions can be applied in sequence to satisfy multiple objectives. In summary, Paquay et al. (2018) demonstrated two extremes of merit function design – one maximising space usage and one enforcing weight balance – and showed that a hybrid strategy can achieve both high fill rates and compliance with air cargo balance requirements.

In the last decade, research on bin packing for logistics has trended toward more integrated approaches. Rather than optimising a single criterion, newer algorithms consider multiple facets. For example, multi-objective optimisation and metaheuristics are used to handle conflicting goals. Recent studies like Gajda et al. (2022) describe industry cases where a heuristic sorts items based on multiple criteria and uses randomisation to explore different packings, managing to satisfy a host of constraints (load balancing, stability, priorities, etc.) while still optimising value or volume. The success of such methods is evidenced by their real-world adoption which report significant cost and CO_2 savings when their algorithm (balancing all these factors) was put into daily use at a logistics company. This underscores a trend: practical heuristics now aim to be both efficient and comprehensive in constraint handling. Merit functions are evolving from simple greedy rules to components of sophisticated frameworks (like GRASP, tabu search, or MILP-guided heuristics) that can juggle multiple objectives. Additionally, there is a trend of customising merit functions to specific applications: for instance, load sequencing considerations (when loading an aircraft, the order and orientation might matter for damage prevention) can be added as criteria. We also see research on stochastic bin packing where item sizes or weights may vary, requiring merit functions that hedge against uncertainty. All these advances build on the fundamental idea of a merit function – encapsulating what "good" looks like in a packing – but extend it with modern computing techniques. The air cargo application, with its strict requirements, has been a driving force in this evolution, pushing algorithms to achieve high utilisingation and high safety. As computing power and techniques improve, we can expect merit functions to become even more adaptive, possibly learning from data (e.g. using machine learning to predict which placement will lead to better final outcomes). The literature of the past decade clearly shows an increasing integration of classic heuristic merit-function approaches with advanced optimisation and simulation to tackle the rich, real-world bin packing problems in air transportation and beyond.

3 Research Gap

Existing research on air cargo loading includes heuristic algorithms, Mixed-Integer Linear Programmingmodels, and extreme point-based methods. These approaches primarily treat the problem as a three-dimensional bin packing task: the goal is to pack all given shipments into as few Unit Load Devices as possible, maximising volume utilisation. Advanced heuristics like the extreme point method have improved packing efficiency by identifying promising placement positions inside a container. While such methods enhance space usage and enforce weight balance constraints, they exhibit several limitations in addressing practical air cargo objectives. Key gaps in the literature include:

3D Knapsack Model – Profit-Driven Selection: Nearly all loading algorithms assume every shipment must be loaded, focusing on minimizing the number of ULDs used. Little work allows intentionally leaving out low-priority or low-profit items. In other words, the 3D loading problem is rarely formulated as a knapsack (choosing the most valuable subset to load) rather than a bin-packing problem. Some revenue-management studies have modeled cargo acceptance as a multi-dimensional knapsack to maximize yield (Pak & Dekker, 2004), but this concept has not been widely translated into 3D loading strategies for air cargo. In reality, air cargo operators may choose not to load certain boxes if they are low-value or too costly in terms of space or weight. The lack of a 3D knapsack approach means current methods may pack flights to capacity without considering if discarding or delaying certain shipments could yield higher overall profit.

Profit-Based Optimisation – Lack of Financial Metrics in Objective: Most studies optimise physical criteria – maximising filled volume or evenly distributing weight – without directly incorporating revenue or cost drivers into the objective function. Space utilisation and center-of-gravity balance are treated as ends in themselves, rather than means to improve profitability. As a result, a loading plan that looks efficient geometrically might not be optimal economically. For example, an arrangement that shifts the aircraft's center of gravity slightly off ideal can increase drag and fuel burn, incurring higher operating cost – yet standard algorithms do not factor in this cost. Revenue contributions of shipments and fuel consumption penalties are generally ignored in placement decisions. Integrating such financial factors into the merit function (i.e. making the algorithm explicitly profit-driven) remains an open research area. A profit-aware loading algorithm would choose and position cargo not just to fit, but to maximise net revenue, addressing a crucial gap in air cargo operations.

Extreme Point Heuristics Across Multiple ULDs – Sequential Loading vs. Concurrent Optimisation: Traditional EP-based heuristics handle one container at a time. The algorithm generates extreme points and packs items in the current ULD until no more items fit; only then is a new ULD opened for loading. This sequential bin-by-bin strategy can be limiting. It may fill one pallet to its brim (or weight limit) while another pallet remains empty, potentially leading to suboptimal distribution of weight and volume across the aircraft. There is no mechanism to dynamically allocate items among multiple ULDs in parallel for a better global outcome. A novel approach would be to generate and consider extreme points simultaneously across multiple ULDs, allowing the loader to decide, for instance, that an item should go into a second ULD even if the first isn't entirely full, because it improves overall balance or profit. Such a multi-ULD EP heuristic could yield a more flexible and efficient loading strategy – balancing the load between containers and maximising total loaded value – yet this idea remains largely unexplored in current research.

In summary, the literature to date has laid a solid foundation in packing algorithms for air cargo, but it has left significant gaps in terms of practical objectives. Shifting from a pure loading-maximisation mindset to a more profit-optimised, selective loading paradigm (the 3D knapsack view), and enhancing heuristic strategies to consider multiple ULDs concurrently, are promising directions. Addressing these gaps is important for improving air cargo operations – enabling airlines to make smarter decisions about what to load and where to place it in order to maximise revenue and minimise cost while still meeting all physical constraints. Each of these gaps represents an opportunity to better align loading algorithms with real-world operational goals, leading to more profitable and efficient air cargo flights.

4 Research Question

4.1. Main Research Question

How can a 3D knapsack-based extreme-point heuristic approach be developed to maximise the overall profit in air cargo loading?

4.2. Subquestions

4.2.1. Merit Function and Item Characteristics

RQ1: How can a merit function be designed and refined to account for both revenue (e.g., item value) and operational costs (e.g., fuel consumption resulting from centre-of-gravity shifts), while also considering item-specific properties such as weight, orientation constraints, and fragility?

4.2.2. Dynamic Sequencing Parameters

RQ2: Which parameters should drive the sequencing of items (e.g., dimensions, weight, value) and how can they be tuned dynamically so that the placement algorithm adapts to evolving constraints during the loading process?

4.2.3. Multi-ULD Extreme Points

RQ3: How does simultaneously generating extreme points across multiple ULDs compare to a sequential container-by-container approach, in terms of finding a more profitable or efficient overall loading plan?

4.2.4. Mitigating Local Greediness

RQ4: How can the algorithm prevent locally optimal decisions at each placement step from leading to suboptimal overall solutions, and what strategies (e.g., partial lookahead, re-ordering) can be used to improve global performance?

4.2.5. Benchmarking Against Existing Models

RQ5: How can the proposed approach be compared and validated against existing bin packing or knapsack-based models, and what metrics (e.g., total profit, computational time, space utilisation) are most appropriate for such comparisons?

5 Research Plan

This chapter outlines a concise, three-stage plan to develop and evaluate a 3D bin-packing model for air cargo operations. The plan begins by replicating a reference model, then moves on to refining the merit function (including an aircraft-specific cost element), and concludes with incorporating a dynamic sequencing approach. Data set generation and a brief schedule are also discussed.

5.1. Stage 1: Model Replication

In the first stage, a baseline extreme-point packing framework will be replicated, following the methodology presented by Paquay et al. (2018). This replication ensures that the basic packing logic is correctly implemented and provides a benchmark for later comparisons. Key tasks include reproducing static box sequencing and verifying the framework's results against published benchmarks.

5.2. Stage 2: Merit Function Tuning (with Aircraft Model)

Next, the merit function will be extended to account for item-specific revenue and the aircraft's operational costs, such as fuel usage due to centre-of-gravity shifts. By combining profitability measures (e.g., box value) with approximate penalties for off-centre loading, the packing algorithm can balance revenue gains against weight distribution concerns. The parameters within the merit function will be systematically tuned to ensure that space utilisation, profit maximisation, and basic safety constraints (e.g., fragility) are maintained.

5.3. Stage 3: Dynamic Sequencing

The final stage replaces the existing static ordering of items with a dynamic sequencing procedure. Rather than relying on a single sorted list, the algorithm will continually update which box to load next based on evolving capacity, weight distribution, and estimated profit contribution. This adaptive approach aims to avoid the pitfalls of locally optimal placements that may harm overall performance. Comparisons between static and dynamic strategies will be made to assess potential gains in solution quality and robustness.

5.4. Data Set Generation

Since operational air cargo data sets are not publicly available, synthetic instances will be generated to model realistic box characteristics (dimensions, weight, fragility, value). In generating these, the distributions and statistics provided by Brandt and Nickel (2019) will be taken into consideration. Their analysis of booking data from over 400 flights includes detailed distributions of item dimensions, weight, density, and volume utilisation, which help inform realistic test scenarios. However, the realism of such data remains limited, as cargo characteristics can vary across regions and have likely shifted due to events such as the COVID-19 pandemic.

References

Alvarez-Valdes, R., Martinez, A., & Tamarit, J. (2013). A branch & bound algorithm for cutting and packing irregularly shaped pieces. *International Journal of Production Economics*, *145*(2), 463–477.

Baldi, M. M., Crainic, T. G., Perboli, G., & Tadei, R. (2012). The generalized bin packing problem. *Transportation Research Part E: Logistics and Transportation Review*, *48*(6), 1205–1220.

Boeing. (2024). World air cargo forecast 2024-2043 [Accessed: 2025-03-02].

Brandt, F., & Nickel, S. (2019). The air cargo load planning problem-a consolidated problem definition and literature review on related problems. *European Journal of Operational Research*, *275*(2), 399–410.

Ceschia, S., & Schaerf, A. (2013). Local search for a multi-drop multi-container loading problem. *Journal of Heuristics*, *19*, 275–294.

Chan, F. T., Bhagwat, R., Kumar, N., Tiwari, M., & Lam, P. (2006). Development of a decision support system for air-cargo pallets loading problem: A case study. *Expert Systems with Applications*, *31*(3), 472–485.

Costa, M. d. G., & Captivo, M. E. (2016). Weight distribution in container loading: A case study. *International Transactions in Operational Research*, *23*(1-2), 239–263.

Crainic, T. G., Perboli, G., & Tadei, R. (2008). Extreme point-based heuristics for three-dimensional bin packing. *Informs Journal on computing*, *20*(3), 368–384.

Crainic, T. G., Perboli, G., & Tadei, R. (2009). Ts2pack: A two-level tabu search for the three-dimensional bin packing problem. *European Journal of Operational Research*, *195*(3), 744–760.

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

Eley, M. (2003). A bottleneck assignment approach to the multiple container loading problem. *OR spectrum*, *25*, 45–60.

Faroe, O., Pisinger, D., & Zachariasen, M. (2003). Guided local search for the three-dimensional bin-packing problem. *Informs journal on computing*, *15*(3), 267–283.

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

IATA. (2021). Air cargo market analysis december 2021 [Accessed: 2025-03-02].

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

Li, X., Zhao, Z., & Zhang, K. (2014). A genetic algorithm for the three-dimensional bin packing problem with heterogeneous bins. *IIE Annual Conference. Proceedings*, 2039.

Lodi, A., Martello, S., & Vigo, D. (2004). Tspack: A unified tabu search code for multi-dimensional bin packing problems. *Annals of Operations Research*, *131*, 203–213.

Mack, D., & Bortfeldt, A. (2012). A heuristic for solving large bin packing problems in two and three dimensions. *Central European Journal of Operations Research*, *20*, 337–354.

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

Martínez, D. A., Alvarez-Valdes, R., & Parreño, F. (2015). A grasp algorithm for the container loading problem with multi-drop constraints. *Pesquisa Operacional*, *35*, 1–24.

Oliveira, Ó., & Gamboa, D. (2020). Adaptive sequence-based heuristic for the two-dimensional non-guillotine bin packing problem. *Hybrid Intelligent Systems: 18th International Conference on Hybrid Intelligent Systems (HIS 2018) Held in Porto, Portugal, December 13-15, 2018 18*, 370–375.

Pak, K., & Dekker, R. (2004). Cargo revenue management: Bid-prices for a 0-1 multi knapsack problem. *Available at SSRN 594991*.

Paquay, C., Limbourg, S., & Schyns, M. (2018). A tailored two-phase constructive heuristic for the three-dimensional multiple bin size bin packing problem with transportation constraints. *European Journal of Operational Research*, *267*(1), 52–64.

Paquay, C., Schyns, M., & Limbourg, S. (2016). 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.

Parreño, F., Alvarez-Valdés, R., Oliveira, J. F., & Tamarit, J. M. (2010). A hybrid grasp/vnd algorithm for two-and three-dimensional bin packing. *Annals of Operations Research*, *179*, 203–220.

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

Reuters. (2021). Fedex profit soars with pandemic-fueled delivery demand [Accessed: 2025-03-02].

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

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