UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Last Mile Delivery Box Packing

Yuri Masakazu Mizusawa

Dissertação de Mestrado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: ______________________

Yuri Masakazu Mizusawa

Last Mile Delivery Box Packing

Master dissertation submitted to the Instituto de Ciências Matemáticas e de Computação – ICMC-USP, in partial fulfillment of the requirements for the degree of the Master Program in Computer Science and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Profa. Dra. Franklina Maria Bragion de Toledo

USP – São Carlos
February 2024

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

```
         Mizusawa, Yuri Masakazu
M685l       Last Mile Delivery Box Packin / Yuri Masakazu
         Mizusawa; orientadora Franklina Maria  Bragion de
         Toledo. -- São Carlos, 2024.
            125 p.


            Dissertação (Mestrado - Programa de Pós-Graduação
         em Ciências de Computação e Matemática
         Computacional) -- Instituto de Ciências Matemáticas
         e de Computação, Universidade de São Paulo, 2024.


            1. Otimização. 2. Empacotamento. 3. Heurísticas.
         I. Bragion de Toledo, Franklina Maria , orient. II.
         Título.
```

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Yuri Masakazu Mizusawa

Empacotamento de caixas de entregas de última milha

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientadora: Profa. Dra. Franklina Maria Bragion de Toledo

USP – São Carlos
Fevereiro de 2024

I dedicate this work to all the professors whose guidance and wisdom have shaped my career.
Their commitment to knowledge and mentorship has left an indelible mark on my growth.

ACKNOWLEDGEMENTS

Aos meus amigos,

Quero expressar minha gratidão a cada um de vocês. A presença de vocês em minha vida é fonte de apoio, amor e alegrias. Obrigado pela companhia.

À minha família,

Minha gratidão por tudo que fizeram por mim ao longo da vida. Seu amor, sacrifícios e orientação moldaram a pessoa que sou hoje

Aos mestres,

Obrigado por desafiarem, motivarem e acreditarem em mim. Vocês são verdadeiros guias, e sou grato por ter tido a sorte de ter educadores tão incríveis em minha vida.

Agradeço também a toda equipe do Instituto de Ciências Matemáticas e de Computação (ICMC).

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001

“It may help to understand human affairs to be clear that most of the great triumphs and tragedies of history are caused, not by people being fundamentally good or fundamentally bad, but by people being fundamentally people.”

— Neil Gaiman, Good Omens: The Nice and Accurate Prophecies of Agnes Nutter, Witch —

RESUMO

MIZUSAWA, Y. **Empacotamento de caixas de entregas de última milha**. 2024. 125 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2024.

O crescimento do comércio eletrônico resulta no aumento expressivo do envio de encomendas, o que torna essencial a otimização do processo de empacotamento. Este estudo investiga modelos de empacotamento visando minimizar o espaço vazio durante o transporte. Primeiramente, modelos de empacotemento são resolvidos com métodos exatos para analisar para sua eficiência e rapidez, destacando-se a eficácia da rotação dos itens e de formulações alternativas. Dois cenários são explorados, comparar conjuntos de caixas geradas neste trabalho com conjuntos da literatura, e permitir o empacotamento dos itens de um mesmo cliente em até duas caixas. Os dois cenários apresentaram uma redução significativa do espaço vazio. Na segunda parte do trabalho, é desenvolvida uma busca local que ajusta as dimensões das caixas. Esse método mostrou-se eficaz na redução do espaço vazio, apresentando implicações práticas para otimização logística no contexto do comércio eletrônico.

Palavras-chave: Otimização, Empacotamento 3D, Heurísticas.

ABSTRACT

MIZUSAWA, Y. **Last Mile Delivery Box Packing**. 2024. 125 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2024.

The growth of e-commerce is leading to a significant increase in parcel shipments, making it essential to optimize the packaging process. This study investigates packaging models that aim to minimize the amount of empty space during transportation. First, packing models are solved with exact methods to analyze their efficiency and speed, highlighting the effectiveness of item rotation and alternative formulations. Two scenarios are explored: comparing box sets generated in this work with sets from the literature, and allowing items from the same customer to be packed in two boxes. Both scenarios show a significant reduction in residual volume. In the second part of the work, a local search method is developed to adjust the dimensions of the boxes. This method is effective in reducing empty space and has practical implications for logistics optimization in the context of e-commerce.

Keywords: Optimization, Packing, 3D, Heuristics.

LIST OF FIGURES

Figure 1 – Bar-plot with quantity of solutions (a) and problems (b) over the years that were surveyed with the bibliographic review. This chart only includes papers that work with 3D. 34

Figure 2 – This figure depicts how Chen, Lee and Shen (1995) handles box-item placement, where all boxes ℓ have the same origin and item placement depends on $\alpha_{i\ell}$ variable. In Figure 2a, since both items have a different $\alpha_{i\ell}$ value, item i is placed on box 1 and item j is placed on box 2, they are allowed to overlap (light gray area). In Figure 2b, both items are placed on the same box 1 ($\alpha_{i1} = \alpha_{j1} = 1$). Therefore, the constraint (3.11) is active and consequently at least one non-overlapping constraints (3.5)-(3.22) is active (dark gray area is not allowed). 41

Figure 3 – Example of box placement for the Chen, Lee and Shen (1995) model depicting non-overlapping constraints. 45

Figure 4 – Box representation for the 2D problem used by Tsai, Malstrom and Kuo (1993). Item i is inside the box and item j is outside the box. This image is a representation of the variable α_i activation, for item i, $\alpha_i = 1$, forcing it into the box and for item j, $\alpha_j = 0$, allowing it to be outside the box and thus not be counted in the objective function. 47

Figure 5 – Item positioning examples for the Tsai, Malstrom and Kuo (1993) model. Figure 5a and Figure 5b depict active non-overlapping constraints (3.42) and (3.43), respectively. 49

Figure 6 – Example of box placement for the developed model. 50

Figure 7 – Solution for the toy instance with only one box. 53

Figure 8 – Solution for the toy instance with up to two boxes available. Dashed lines defines the boxes. 53

Figure 9 – Density plot displaying the volume of instances (orders) without factor adjustment for the items (a) and with adjustment (b). Vertical dotted lines represent the highest value in each dataset. 57

Figure 10 – An example with three distinct approaches to packing four orders, with boxes denoted by the dashed lines and items represented as grey squares. The red areas represent the residual volume. The residual volumes are ordered as follows: $V_{RES\ A} \geq V_{RES\ C} \geq V_{RES\ B}$. 75

Figure 11 – This figure depicts decreasing total residual volume by increasing a box dimension. Dashed lines represent the boxes, and the items are shown as grey squares. The red area ($V_{RES\ 1}$) shows the initial residual volume before the change, while the green area ($V_{RES\ 2}$) illustrates the residual volume after the adjustment. 76
Figure 12 – This figure depicts decreasing total residual volume by reducing a box dimension. Dashed lines represent the boxes, and the items are shown as grey squares. The red area ($V_{RES\ 3}$) shows the initial residual volume before the change, while the green area ($V_{RES\ 4}$) illustrates the residual volume after the adjustment. 76
Figure 13 – This figure depicts the neighborhood definition and limits in two dimensions. The red point shows the initial box that will be explored and the tiers range are defined by the black rectangles, with **A** the dimension increase and **R** reduction. 83
Figure 14 – Image with bar plot of objective value reduction (left y-axis) and line plot for times (right y-axis) of each local search (N1 and N2) for each tier (x-axis) from 1 to 6. Results obtained for Instance 1. 91
Figure 15 – This figure contains a bar plot with the objective value (V_{RES}) for both heuristics (H), H1 in blue and H2 in orange, and local search using initial solution H1 or H2 (LS^{H1}_{N2N1} or LS^{H1}_{N2N1}). Values are the mean for all instances, and the vertical bar as the standard deviation between instances. 95

LIST OF ALGORITHMS

Algorithm 1 – Constructive Heuristic - H1 79
Algorithm 2 – Box filter. 80
Algorithm 3 – Packing model function. 80
Algorithm 4 – Calculate V_{RES} for each partition 81
Algorithm 5 – Constructive Heuristic - H2 82
Algorithm 6 – Function to create a HB_t set from HB, with larger boxes. 84
Algorithm 7 – Skip boxes that knowingly do not provide a better solution or is infeasible, comparing with the current solution box. 86
Algorithm 8 – Skip boxes for that do not yield better results or are infeasible, comparing with the order. 86
Algorithm 9 – Pseudocode for Local Search with N1 87
Algorithm 10 – Function to create a HB_t set of boxes from HB, with smaller boxes. . . 123
Algorithm 11 – Skip boxes for that do not yield better results or are infeasible, comparing with the order. 124
Algorithm 12 – Pseudocode for Local Search with N2 125

LIST OF TABLES

Table 1 – List of journals considered on the initial search. 31
Table 2 – Inclusion and exclusion criteria. 31
Table 3 – Problem Type and Solution Approach used in this review. 32
Table 4 – Quantity of each type of problem. This table only includes papers that work with 3D. 32
Table 5 – Analyse of approaches used. This table only includes papers that work with 3D problems. 33
Table 6 – Analysing Problem Type and Solution Approach. This table only includes papers that deal with 3D problems. 33
Table 7 – Comparison table for some aspects of discussed articles. Model types are pallet loading problem (PLP), cutting stock problem (CSP) and bin packing problem (BPP). Where it is stated "NA" is for *not applicable*. Box generation concerns the origin of the boxes for the instance. All model are MIPs formulations. . . 37
Table 8 – Table with parameters and variables for Chen, Lee and Shen (1995) original model with rotation. 42
Table 9 – Table depicting active constraint if that positional variable is active (equal to 1), only one can be active. 44
Table 10 – Table showing how the orthogonal variables determine the six possible rotations of an item. The first four columns represent variable values and the last three columns represent the dimension of the item that is in each axis, where ’l’ is the length, ’w’ is the width and ’h’ is the height. 45
Table 11 – Table with parameters and variables for Tsai, Malstrom and Kuo (1993) original model without rotation and for only one box. 48
Table 12 – Overlapping variables and their values, adapted from Tsai, Malstrom and Kuo (1993). For each pair of items i and j (u_1^{ij}, u_2^{ij}, u_3^{ij}), we have at least one constraint active, and the value of (x_i, y_i, z_i) that should be respected. 50
Table 13 – Table with extra parameters for the adapted model. 51
Table 14 – Data of the toy problem instance. 52
Table 15 – Number of variables for each model. 54
Table 16 – Number of constraints for each model. 54
Table 17 – Probability and number of items for each instance Class [Hübner, Holzapfel and Kuhn Heinrich (2015)]. 55
Table 18 – Percentage difference between linear relaxations ($Q = 2$). 60

Table 19 – Run times for the two formulations of Z_{CLSwr} (without rotation) by the number of items considering AMB box set and $Q = 2$. 61
Table 20 – Run times for the two formulations of Z_{TMKa} (without rotation) by the number of items considering AMB box set and $Q = 2$. 61
Table 21 – Time in seconds for the instance that reached the time limit. 61
Table 22 – Total residual volume percentage (V_{RES}) for all instances with (w.) and without (w/o) rotation for only one box (Q=1) and up to two boxes (Q=2) for AMB box set. 62
Table 23 – Run times for the model Z_{CLSwr} (without rotation) by number of items considering AMB box set, $Q = 1$ and $Q = 2$. 63
Table 24 – Run times for model Z_{CLS} (with rotation) by number of items considering AMB box set, $Q = 1$ and $Q = 2$. 64
Table 25 – Summarized run times in seconds of all instances for Z_{CLS} (with rotation) for all different box sets (AMB, HB and HBnd), only one box (Q=1) and up to two boxes (Q=2). 65
Table 26 – Number of instances that reached the time limit (1,800 seconds) for Z_{CLS} (with rotation) for all different box sets (AMB, HB and HBnd), only one box ($Q = 1$) and up to two boxes ($Q = 2$) by item quantity. 65
Table 27 – Summary statistics of run times in seconds for all instances in all three box sets by item quantity for up to two boxes (Q=1). 66
Table 28 – Summary statistics of run times in seconds for all instances in all three box sets by item quantity for up to two boxes (Q=2). 67
Table 29 – Summary of residual volumes for all box sets. 68
Table 30 – Total residual volumes (V_{RES}) for all box sets by item quantity. 69
Table 31 – Summary of residual volumes (V_{res_k}) for all box sets by item quantity. 70
Table 32 – Quantity of instances that the model packed in one or two boxes for each box set when executed with $Q = 2$. 70
Table 33 – Table with runtimes in seconds and objective value (V_{RES}) for the Constructive Heuristic (H1) and the Fast Constructive Heuristic (H2), percentage difference calculated as $\frac{H1-H2}{H1}$ of the objective values. 90
Table 34 – Table with the variation of random starts for Fast Constructive Heuristic (H2), with mean and standard deviation (SD) of times and objective values (V_{RES}), executed for 20 boxes and 10 different seeds for each instance. 90
Table 35 – Table with time and objective value (V_{RES}) for H1 and H2 best run and percentile difference (Diff. = $\frac{(H2-H1)}{H1}$), executed for 20 boxes. 90
Table 36 – Table with results of tier exploration, where rows represent tiers and columns the runtime in second and percent objective value reduction from H2 solution for each local search (N1 and N2). 92

Table 37 – The experiment results obtained by the Local Search method using neighbourhood N1, N2, N1N2 and N2N1. Times do not include H2 runtimes. 93
Table 38 – Table comparing results from H1 with H2 run with N2N1 (H2&N2N1). Where rows have the instances and columns time in seconds and objective value (V_{RES}). Times include H2 runtimes for LS_{N2N1} 93
Table 39 – Table with runtimes, objective values (V_{RES}) and percentage reduction (Red.) of H1 and H1 solution run with LS_{N2N1}. 94
Table 40 – The table presents the runtimes in seconds and objective values for H1 and H2 executed as an initial solution for the local search LS_{N2N1}. Last column represents the percent difference of the values (Diff. = $\frac{LS^{H1}_{N2N1} - LS^{H2}_{N2N1}}{LS^{H1}_{N2N1}}$). 95
Table 41 – Table containing items dimensions. 103
Table 42 – Table with 20 orders and their items. 107
Table 43 – Table containing AMB dimensions. 109
Table 44 – Table containing HB dimensions. 113
Table 45 – Table containing HBnd dimensions. 117
Table 46 – Percentage difference between linear relaxations ($Q = 1$). 121
Table 47 – Run times for the two formulations of Z_{TMKa} (without rotation) by the number of items considering AMB box set and $Q = 1$. 122
Table 48 – Run times for the two formulations of Z_{CSLwr} (without rotation) by the number of items considering AMB box set and $Q = 1$. 122

CONTENTS

1 INTRODUCTION . . . 25
1.1 Research Objectives . . . 26
1.2 Document Outline . . . 26

2 LITERATURE REVIEW . . . 29
2.1 Bibliographic Review Method . . . 30
2.2 Collected Data . . . 31
2.3 Overview of Packing Problems . . . 34
2.4 Related Works . . . 35
2.5 Summary of Bibliography Review . . . 36

3 PACKING PROBLEM . . . 39
3.1 Problem Definition . . . 39
3.2 Mathematical Models . . . 40
3.2.1 Chen, Lee and Shen Model . . . 40
3.2.2 Chen, Lee and Shen Model Without Rotation . . . 46
3.2.3 Tsai, Malstrom and Kuo Model . . . 47
3.2.4 Adapted Tsai, Malstrom and Kuo Model . . . 50
3.2.5 Equivalent Constraints . . . 52
3.2.6 A Toy Problem . . . 52
3.2.7 Models Analysis . . . 53
3.3 Computational Experiments . . . 54
3.3.1 Instances Sets . . . 55
3.3.2 Box Sets . . . 55
3.3.3 Item Sets . . . 58
3.3.4 Exploration Goals . . . 58
3.3.5 Computational Results . . . 58
3.4 Conclusions . . . 71

4 BOX GENERATION PROBLEM . . . 73
4.1 Problem Definition - Box Generation . . . 74
4.2 Solution Approach . . . 74
4.2.1 Defining HB Set . . . 77
4.2.2 Constructive Heuristics . . . 77

4.3 Local Search 83
4.3.1 Neighbourhood definition 83
4.3.2 Local Search using Neighbourhood N1 85
4.3.3 Local Search using Neighbourhood N2 88
4.4 Computational Experiments 88
4.4.1 Instance Sets 88
4.4.2 Analysing the Constructive Heuristics 89
4.4.3 Analysing the Tier Dimension 91
4.4.4 Analysing the Local Search with Initial Solution H2 92
4.4.5 Comparing results from H1 with LS_{N2N1} *. 93*
4.4.6 Analysing the Local Search with Initial Solution H1 93
4.5 Conclusions 94

5 CONCLUSIONS AND FUTURE RESEARCHES 97

BIBLIOGRAPHY 99

APPENDIX A INSTANCE - ITEMS 103

APPENDIX B INSTANCE - ORDERS & ITEMS 107

APPENDIX C INSTANCE - BOX SET AMB 109

APPENDIX D INSTANCE - BOX SET HB 113

APPENDIX E INSTANCE - BOX SET HBND 117

APPENDIX F TABLES - RELAXATION AND TIMES RESULTS FOR Q=1 121

APPENDIX G ALGORITHMS - LOCAL SEARCH USING N2 123

CHAPTER

1 INTRODUCTION

In recent years, global trade has shown high growth, consequently transportation of goods around the world has also been increasing. In 2022, 131 billion packages were shipped worldwide, which is expected to double by 2026 [Statista (2021)]. Due to this increase, logistics chains are under pressure to reduce costs and environmental impacts.

Transportation costs and the use of packing materials, such as cardboard and fillers, are among the major challenges faced by logistics companies. These challenges can be addressed by implementing optimization strategies, such as mathematical modeling. These techniques have been well-documented in the literature for tackling various logistics issues, including routing, scheduling, and packing efficiency. In particular, packing optimization aims to improve the way goods are packaged into boxes, pallets, or containers.

In the logistics industry, packing is a daily challenge for many companies. The packing process can be divided into two main problems: (i) packing individual items of each order into boxes, and (ii) consolidating these boxes onto pallets for delivery. The first problem aims to minimize the number of boxes used and eliminate empty space in the boxes. The second problem aims to maximize the number of boxes or orders that can be consolidated onto a minimal number of pallets. It's important to note that both problems are interconnected and can be challenging to solve. They are classified as packing problems in the literature.

In order to solve these problems, it is essential to understand to which class of problems they belong. Packing problems like the ones mentioned can be classified into pallet loading, knapsack, bin packing, and container loading problems. The pallet loading problem aims to pack the maximum amount of similar boxes into one single pallet. The knapsack problem aims to fill a container with the most valuable like profit or less waste, with different kinds of boxes. The bin packing problem minimizes the number of containers needed to pack a set of boxes. Finally, the container-loading problem fills the minimum number of containers with a fixed number of boxes, which can be considered a version of three-dimensional (3D) bin packing, but container loading can only be 3D. Box packing is usually 3D, although some studies use the 2D version due to problem specifics, e.g., non-stackable material.

Another critical factor is the constraints this class of problems can require. First, it is mandatory for two and three-dimensional packing problems to have non-overlapping constraints of the items placement. Second, most real instances come with business or process-related constraints. There can be a wide variety of these constraints, such as volume limit, weight limit, balancing, cargo splitting, and stability, amongst others. Ali *et al.* (2022) covers most of them and mention several articles of each problem.

This study aims to optimize the packing process by identifying more efficient strategies for determining box sizes, with the goal of reducing material and delivery costs. The packing problems will be treated as a three-dimensional model and solved using exact methods. As described by Wäscher, Haußner and Schumann (2007) typology, the problem addressed in this work is known as the *Residual Bin Packing Problem* (RBPP).

1.1 Research Objectives

The main goal of this research is to minimize the wasted volume of boxes used to pack customer orders (items) in three dimensions. In order to do this, it is proposed to:

1. Study packing problems in the e-commerce scenario;
2. Define a set of boxes in the same context.

In the initial phase of this study, continuous packing models addressing the packing with multiple box selections are examined. Following that, a novel method for generating boxes and altering their dimensions is presented.

1.2 Document Outline

This work is divided into the following chapters.

First, a literature review is presented in Chapter 2. In this chapter, it is explained how the review was conducted, and its results are shown. Then, a brief discussion of packing problems is made, followed by a description of related works.

Chapter 3 describes the examined models. Subsequently, instances comprising orders, items, and boxes are generated for testing purposes. Finally, computational experiments are conducted to evaluate: i. the described models, ii. a novel box set, and iii. the packing of items in two or more boxes.

In Chapter 4, a new method for generating and improving boxes for a given set of orders is presented and evaluated. The problem of defining the optimal set of boxes for a given set of orders is defined. Then, two constructive heuristics are devised to provide an initial solution for the problem. Subsequently, a local search and its neighbourhoods are detailed and tested.

Finally, a summary of this study and future works are discussed in Chapter 5.

CHAPTER 2

LITERATURE REVIEW

Cutting and Packing Problems encompass various practical (day-to-day) and theoretical problems. These problems generally involve cutting, packing, and allocating items. Although many formulations exist, there are some common elements amongst them.

Over the years, several names were given and established to refer to specific problems and classes of problems. It can be considered that Cutting Stock, Knapsack, Pallet & Container Loading, and Bin Packing are the best known. Along with their variants (e.g., dimension, objective max-min, object format) or application-specific constraints (e.g., weight, overlapping), several problems can be defined.

Dowsland and Dowsland (1992) present a good overview of packing problems focused on modeling, exact solutions, and heuristic approaches. Although it is a bit outdated, it is still a good reference for the basic problems and their formulations. Also, Scheithauer (2018) is a very important introductory book, that describes and helps define packing models and their variants.

Even though classic problem classifications are widely used, it is sometimes challenging to classify a new problem. Therefore, a more extensive classification was demanded to better fit and differentiate those problems. With that in mind Dyckhoff (1990) proposed a more robust classification system for packing problems, later reviewed and improved by Wäscher, Haußner and Schumann (2007). The last one is largely used to define and classify packing problems and also used in this work as a reference for problem definitions.

Before classifying the studied problem, a better understanding of its nature is in order. The main idea for this work comes from e-commerce. Daily, thousands of customer orders must be packed and shipped. With the growth of e-commerce, companies are looking for ways to reduce packing costs. This strategy is becoming the aim of big companies, for instance, Wallmart$^{\text{TM}}$with the Sustainability Hub [Walmart (2022)] and UPS$^{\text{TM}}$with packaging optimization [UPS (2021)], both initiatives are focused on improving packing to reduce cost and wasted materials. Among these goals is deciding the number of boxes and box types available for packing customer orders, which reduces operational costs. However, it is not easy to define a set of boxes to maintain in stock aiming to minimize box and shipping costs. Some companies prefer to deal with a small number of types [Alonso *et al.* (2016)], while others use cost functions [Fontaine and Minner (2022)] to determine the boxes available for packing. The main problem is the balance between the number of boxes and the extra empty volume. In general, a small number of box types results in cheaper boxes (ordered in bulk) and large extra volume that requires more filling and shipping costs. On the other hand, many box types allow for better allocation of orders but might have extra costs to buy and maintain. It is proposed to deal with this problem using mathematical modeling, i.e., given a set of boxes, pack each customer order into the best-fitting box available. Consequently, it defines a set of boxes that better suit the orders thus minimizing costs.

According to Wäscher, Haußner and Schumann (2007), the problem studied can be designated as Residual Bin Packing Problem (RBPP). Following their classification system, this problem has an input minimization (waste minimization), composed of strongly heterogeneous regular small items and strongly heterogeneous regular large objects. As a result, the bibliographic review presented is focused on RBPP.

Besides formulations for those problems, there is concern about how to solve them. The more common solution methods are heuristic and exact solutions. Furthermore, there are math-heuristics, meta-heuristics, and even machine-learning methods. In the first phase of this study, exact methods were used with the Gurobi software. Subsequently, for the second phase, heuristics and local search techniques were used mixed with exact methods.

One of the most recent reviews about three-dimensional packing problems (3D-PPs) was presented by Ali *et al.* (2022). The authors used the model proposed by Webster and Watson (2002) in their systematic review. Therefore, we used a similar methodology to obtain an appropriate review.

This chapter is organized into five sections. First, in Section 2.1, some brief details of the bibliographic review are presented. The data collected is discussed in Section 2.2. In Section 2.3, it is discussed the topic of packing problems. Finally, related works are presented in Section 2.4 and a summary is made in Section 2.5.

2.1 Bibliographic Review Method

Articles were obtained using a set of keywords in Web of Science, Google Scholar, and private search engines of a selected set of journals with relevance to this field (see Table 1). It is important to highlight that the journal list is not an exclusion criteria, the idea was to focus the search on some journals to limit the results. The search strings used on those engines were: 3D, three-dimensional and packing. This choice restricted the search, complimented by the exclusion-inclusion criteria, which clarified the scope.

Table 1 – List of journals considered on the initial search.

Journal
European Journal of Operational Research (EJOR)
Computers & Operations Research (COR)
OMEGA
JSTOR
International Transactions in Operational Research (ITOR)
Operations Research
Journal of Heuristics
Managment Science

The exclusion criteria specially represented a more robust filter. For instance, if a 3D packing article (following inclusion criteria) had any exclusion criteria, irregular objects, for example, this article would be excluded. Another important point is that, although some two-dimensional packing models can be adapted to three-dimensional packing problems, these papers were not included in this review. For this reason Lodi, Martello and Monaci (2002), and Lodi, Martello and Vigo (2002) are reviews about this topic that fill this gap. Inclusion and exclusion criteria are listed in Table 2.

Table 2 – Inclusion and exclusion criteria.

Inclusion Criteria
Packing
Three-dimensions
Heuristics
Exact Methods
Exclusion Criteria
Irregular objects (e.g.: non-rectangular, spherical, cylindrical)
Non-linear
On-line packing
One-dimension
Problem mixes (eg.: container loading & routing)

In order to have a better understanding of the collection of articles, during the criteria revision, the problem studied and the solution approach were noted and classified. This classification uses the generic name from the literature to Problem Type and Solution Approach, described in Table 3. Each class was stated as present or not in the article.

2.2 Collected Data

Initially, the research strategy resulted in 124 articles, of which 26 were excluded according to the criteria presented. The resulting 98 articles are in the range from 1989 up to 2022. Of those, nine are reviews.

Table 3 – Problem Type and Solution Approach used in this review.

Problem Type
Pallet Loading, Bin Packing, Knapsack, Cutting Stock Problem.
Solution Approach
Exact, Heuristic, Meta-Heuristic, Math-Heuristic, Deep learning.

For container loading, Zhao *et al.* (2016) and Bortfeldt and Wäscher (2013) were good reviews, with the second specifically for constraints. For pallet loading, Silva, Oliveira and Wäscher (2016) reviews advanced and showed future paths for the problem. Finally, the most recent review, includes online and offline packing with advances in the area and possible future work lines [Ali *et al.* (2022)].

Regarding dimensions, 74 articles included three-dimensional problems, but as it is one of the key search words, we cannot consider the result as the search terms are biased. Therefore, from here on, all results will only be considered if there are 3D models in the article. Also, it is important to consider that one article may deal with more than one type of problem and solution approach, therefore these numbers will not match the 74 mentioned.

Considering the Problem Types, there is a predominance of bin packing problems followed by container loading (both with almost half of the articles) and the other classes with roughly the same amount (see Table 4). However, it was noted that container loading is commonly associated with routing problems that were not included (exclusion criteria), probably due to the nature of the transportation problems. Knapsack, pallet loading, and cutting stock showed to have few publications for 3D.

Table 4 – Quantity of each type of problem. This table only includes papers that work with 3D.

Problem Type	**Quantity**
Bin Packing	38
Container Loading	30
Knapsack	9
Pallet Loading	5
Cutting Stock	4
Total	**86**

In analyzing the solution approaches, exact and heuristics methods were the most common solving methods, followed by meta-heuristics. Only two articles used math-heuristic and/or deep learning strategy (Table 5). These results were expected, as heuristics and exact methods are the more classic approaches for those problems. Math-heuristics are yet to be more widely explored, especially, as we see, for packing problems. The same logic can be applied to deep learning strategy, which is still starting to be used in optimization.

The same trend of more heuristics and exact methods, now evaluating the solution

Table 5 – Analyse of approaches used. This table only includes papers that work with 3D problems.

Solution Type	**Quantity**
Heuristic	40
Exact	26
Meta-Heuristic	10
Math Heuristic	1
Deep Learning	2
Total	79

approaches per problem, is noted [Table 6]. It is especially interesting to see that both deep learning articles are related to the bin packing problem [Hu *et al.* (2017), Jiang, Cao and Zhang (2021)], and the single math-heuristic was applied to the container loading problem [da Silva *et al.* (2020)].

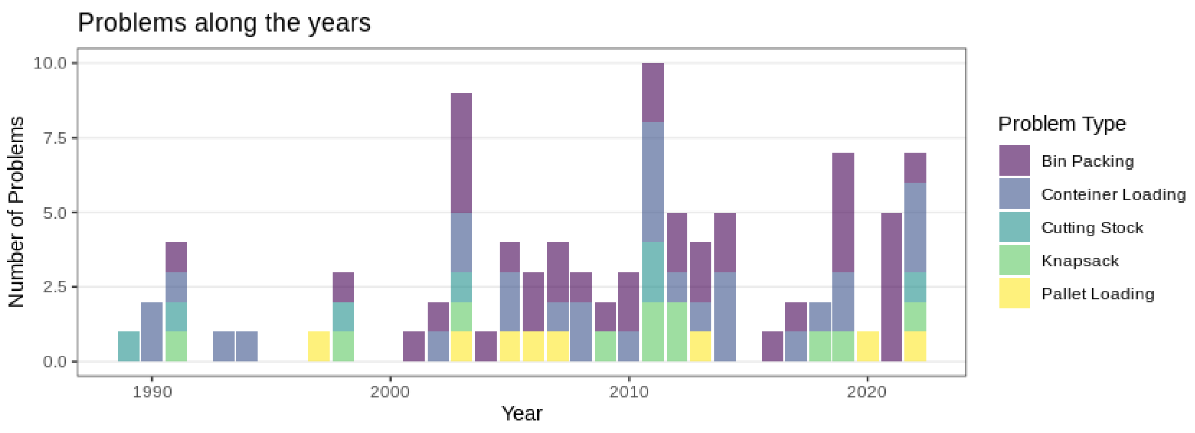
Table 6 – Analysing Problem Type and Solution Approach. This table only includes papers that deal with 3D problems.

Problem \ Solution	Deep Learning	Math Heuristic	Meta-Heuristic	Heuristic	Exact
Bin Packing	2	0	4	21	12
Container Loading	0	1	4	15	13
Knapsack	0	0	1	5	5
Pallet Loading	0	0	1	3	5
Cutting Stock	0	0	0	2	3

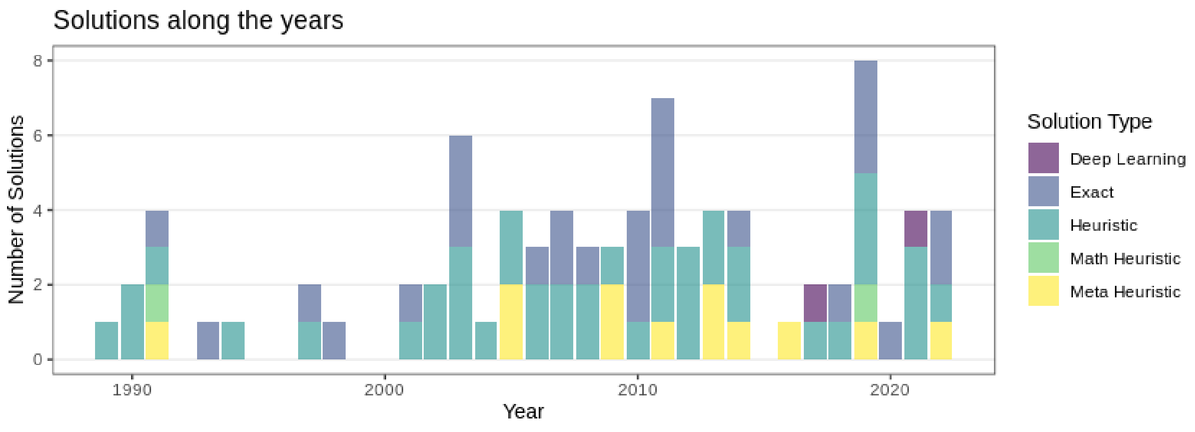
In Figure 1a, it is possible to observe the same dominance of bin packing and container loading problems to the pallet loading and cutting stock problems. It is interesting to note that the pallet loading problem appears consistently throughout the years in small quantities. In Figure 1b, heuristics and exact methods dominated the solving approaches. Despite its low quantity, math-heuristics appears two times with a 29 years gap, maybe showing a recent trend to use these methods. Also, the deep learning strategy only appeared after 2017 due to the growth and development of neural network methods over the last ten years.

Between 2000 and 2015, there was a consistent amount of publications with a slightly increasing trend. This trend was broken between 2015 to 2018, followed by a peak in 2019 and a significant decrease in 2020. The following year shows roughly the same trend in publications.

Figure 1 – Bar-plot with quantity of solutions (a) and problems (b) over the years that were surveyed with the bibliographic review. This chart only includes papers that work with 3D.



(a) Problem types along the years.



(b) Problem types along the years.

2.3 Overview of Packing Problems

The problem studied aims to minimize wasted volume by deciding a box (or boxes) that fit all items of each customer order. As packing problems usually share some characteristics, a short description of each problem is presented.

The container loading problem aims to fill one or more containers with a set of items. Variations of this problem include: all boxes must be packed in one (single container loading problem) or several containers (multi-container loading problem) and containers and items can be identical or heterogeneous [Dowsland and Dowsland (1992)]. The current problem is similar to heterogeneous multi-container loading as it fills multiple boxes (containers).

It is important to mention that container loading usually has additional real constraints depending on the problem context. They can include, for instance, weight distribution, loading order, and stack balance, amongst others. For a better insight, Bortfeldt and Wäscher (2013) provide a recent and extensive review on this subject. These types of constraints are not explored in this work.

Bin packing problems are a classic and well-known set of problems with a wide variety of applications and formulations. The problem, generally, states that it is necessary to pack a set of items into a finite number of bins in order to minimize one dimension (strip packing) or minimize the number of bins (bin packing) [Dowsland and Dowsland (1992)]. The similarities originated in minimizing the bins for packing (boxes in this case), for the 3D variant. Both mentioned problems, bin packing and container loading, appear as the more widely studied Table 4.

Lastly, the Knapsack Problem needs to fill one box (Knapsack Problem) or more (Multiple Knapsack Problem) from several items with different profits. The problem deals with one (Single Knapsack Problem) or several constraints (Multidimensional Knapsack Problem). Even though the objective function can be seen in a similar way, in the case studied here, the focus is to choose the items that best fit the knapsack (or knapsacks).

As can be seen from the classic problems descriptions, there is no standard definition, and one problem can have characteristics that fit two or more different classes. Following the above definition, the problem studied was classified in a more standard typology by Wäscher, Haußner and Schumann (2007) and ended as *Residual Bin Packing Problem* RBPP.

2.4 Related Works

Although a comprehensive survey of articles on packing problems was conducted, only a few are directly relevant to this work. In this short review, six articles were identified that addressed similar problems. Among them, two mathematical models for packing problems are presented, while the other two used packing problems within the context of sets of orders. The remaining two articles were excluded from this section as they related to non-linear problems.

Tsai, Malstrom and Kuo (1993) deal with the pallet loading problem. The authors described two mathematical models for the 2D and 3D pallet loading problem with mixed item sizes. The objective is to maximize the occupancy of one box by selecting items while avoiding overlapping. Although rotation is mentioned, it is not addressed by the authors. In their approach, items are placed in continuous positions in the box. This model is detailed in Subsection 3.2.3.

Chen, Lee and Shen (1995) deal with the container loading problem with multiple heterogeneous containers and a finite number of boxes. The objective is to minimize the sum of the volume of the containers used minus the total item volume, in other words, minimize wasted volume. The authors proposed a mathematical model for the problem that selects one or multiple boxes (containers) to pack all items. Rotation of the items is allowed. One noticeable characteristic is that the authors do not specify any constraint to limit box quantity, therefore, it is assumed that there is no restriction on boxes.

Similarly, Alonso *et al.* (2016) approach this problem with a different objective. Their main goal is to decide the boxes sizes chosen to reduce costs while packing a set of customer orders (each order contains a set of items). Even though it is mentioned that the number of boxes types influences the costs, due to the distribution company preferences, the authors used a fixed quantity of boxes ranging from 1 to 4. To solve the problem, they propose an integer linear programming model based on Beasley (1985) cutting stock problem for the box selection and packing was done in two dimensions with item rotation using heuristics.

Vieira *et al.* (2021) approach the same problem and objective, that is, to choose box sizes, but in their case, each box has all three dimensions open. In a following work, Vieira and Carvalho (2022), extend the problem to consider multi-container loading with open dimensions. Furthermore, the authors use a bi-objective mixed-integer non-linear function to optimize container volume and their number. Both models use open dimensions and end up falling into non-linear optimization. Therefore, they are not discussed in this work.

More recently, Fontaine and Minner (2022) developed a new method denominated by them as Branch & Repair, that is based on logic-based Benders decomposition [Hooker (2007)] and branch-and-check methods [Thorsteinsson (2001)]. The objective is to determine the best set of boxes to pack a set of orders. The problem, as introduced by them, is a three-dimensional bin selection problem (3D-BSP) that minimizes the cost of unused space (wasted volume) and the cost of maintaining a variety of boxes. Basically, the objective is to minimize extra space to pack the orders while accounting for the costs of increased box types. In this model, all orders are sub-problems of the 3D bin packing problem (3D-BPP) category and are solved individually.

2.5 Summary of Bibliography Review

Up to the closing of this review, these were the more relevant publications to the current work. In Table 7, a summary of the main differences among the cited articles is presented.

Tsai, Malstrom and Kuo (1993) and Chen, Lee and Shen (1995) addressed the packing problem to one order. In contrast, the works of Alonso *et al.* (2016) and Fontaine and Minner (2022) address a problem that shares similarities with the one examined in this work, incorporating the concept of multiple orders within the packing process.

The main difference between this research and Alonso *et al.* (2016) and Fontaine and Minner (2022) works is related to the rotation of the items. Furthermore, Alonso *et al.* (2016) deal with the definition of the boxes and Fontaine and Minner (2022) knew a priori the boxes available. In this research, the set of boxes needs to be defined to improve the packing of a given set of orders.

Table 7 – Comparison table for some aspects of discussed articles. Model types are pallet loading problem (PLP), cutting stock problem (CSP) and bin packing problem (BPP). Where it is stated "NA" is for *not applicable*. Box generation concerns the origin of the boxes for the instance. All model are MIPs formulations.

Article	Tsai et al.	Chen et al.	Alonso et al.	Fontaine et al.	Current Work
Problem Type	2D & 3D PLP	3D PLP	2D CSP	3D BPP & 3D BSP	3D BPP & 3D BSP
Box Selection	One	Multiple	Multiple	Multiple	Multiple
Items	Maximize	Pack all	Pack all	Pack all	Pack all
Box Generation	NA	No	Heuristics	Instance	Herz
Item Packing	NA	NA	Pattern	Sub-problem	Packing Problem
Item Rotation	No	Yes	2D	2D	3D
Solving Method	B&B	Lingo	CPLEX & Heuristics	B&R	Gurobi & Local Search
Study Case	-	Alternative Formulations	Order Split	Order Split	Order Split & Box Generation
Objective	Maximize Occupation	Minimize Volume	Minimize Volume	Minimize Volume	Minimize Volume

CHAPTER 3

PACKING PROBLEM

The problem studied belongs to the Last Mile Delivery category, which is a general description of the logistics of picking, packing, and delivering goods directly to the final customer (e-commerce, for instance) [Boysen, Fedtke and Schwerdfeger 2020]. More specifically, inside the picking and packing problems, companies also must decide what boxes to keep in stock to pack all items with minimal cost. In summary, the problem consists of packing the client's orders in a box (or boxes), aiming to minimize the packing and shipping costs, i.e., wasted box volume and box acquisition costs.

As discussed in Chapter 2, similar problems are approached in the literature. Tsai, Malstrom and Kuo (1993) presented a mathematical model to maximize the items packed in a box, but not dealing with the box selection. Chen, Lee and Shen (1995) pack items in boxes, with the objective of minimizing the total wasted volume.

This chapter introduces four models. Section 3.1 offers a detailed explanation of the problem under investigation. Section 3.2 describes the mathematical models from the literature that are considered in this work. Then, in Section 3.3, the results of computational experiments are presented. Last, in Section 3.4, a summary of this chapter and some ideas for future research are presented.

3.1 Problem Definition

Given an instance (order) consisting of items, the main goal of the studied problem is to pack all items into boxes while minimizing the total waste volume by selecting the box (or boxes) that best fits the items without overlapping.

Items and boxes are rectangular and can be rotated depending on the model. If rotation is not allowed, items are placed on the box with its length parallel to the x-axis. In addition, all items must be packed. When rotation is allowed, objects have the flexibility to rotate to any of the six possible positions for a rectangular object.

In the literature, these problems usually focus on packing all items in only one box to facilitate shipping logistics. However, according to Fontaine and Minner (2022) and Alonso *et al.* (2016), waste can be reduced if an order can be packed in more than one box. For this reason, the models were adapted to allow packing one instance in more than one box.

3.2 Mathematical Models

In this section, the four models studied are presented. The original Chen, Lee and Shen (1995) model and a version without rotation are described in Subsection 3.2.1 and Subsection 3.2.2, respectively. Subsequently, Subsection 3.2.3 and Subsection 3.2.4 present the original Tsai, Malstrom and Kuo (1993) model and its adaptation to accommodate multiple boxes. Last, Subsections 3.2.6 and 3.2.7 present a toy problem and a short discussion of the main differences among models, in this order.

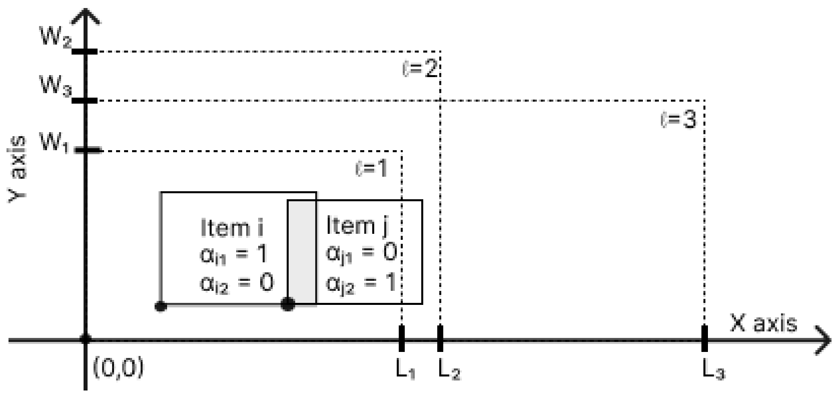
3.2.1 Chen, Lee and Shen Model

Chen, Lee and Shen (1995) proposed a model for the 3D packing problem considering that all items need to be packed. In this case, it is possible to choose one box or a set of boxes to pack the items. The objective is to minimize the total unused space in the boxes, ensuring to pack of all items.

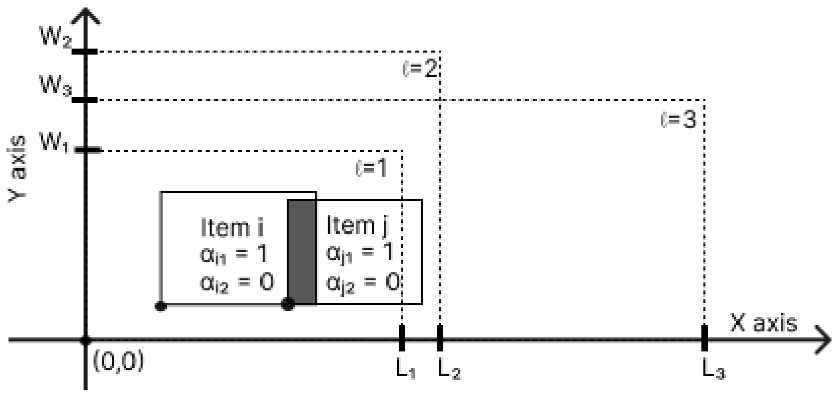
To represent the boxes, the authors use the origin point (0,0,0) and their length, height, and width, as illustrated in Figure 2. While boxes are overlapping, items can overlap (Figure 2a) only if they are in different boxes, i.e., $\alpha_{i\ell} \neq \alpha_{j\ell}$, variables for items i and j are not active at the same time. Otherwise, if both are active at the same time $\alpha_{i\ell} = \alpha_{j\ell} = 1$ (Figure 2b), at least one non-overlapping variable is activated.

In this model, item rotation is enabled through binary orthogonal variables. This is done by four variables that determine the axis along which each item dimension is oriented. By applying orthogonal constraints, achieving all six possible rotations for a rectangle is possible. For instance, when lx_i is set to one, it signifies that the length of the item is aligned with the x axis (for additional details on these constraints, please refer to the explanation in the following sections). The parameters and variables of this model are described in Table 8, the model is defined by (3.1) – (3.18), and labeled as Z_{CLS}.

Figure 2 – This figure depicts how Chen, Lee and Shen (1995) handles box-item placement, where all boxes ℓ have the same origin and item placement depends on $\alpha_{i\ell}$ variable. In Figure 2a, since both items have a different $\alpha_{i\ell}$ value, item i is placed on box 1 and item j is placed on box 2, they are allowed to overlap (light gray area). In Figure 2b, both items are placed on the same box 1 ($\alpha_{i1} = \alpha_{j1} = 1$). Therefore, the constraint (3.11) is active and consequently at least one non-overlapping constraints (3.5)-(3.22) is active (dark gray area is not allowed).



(a) Items are allowed to overlap.



(b) Items are not allowed to overlap.

Table 8 – Table with parameters and variables for Chen, Lee and Shen (1995) original model with rotation.

Parameters	
n	is the number of items;
v_i	is the volume of item i ($v_i = l_i \cdot h_i \cdot w_i$);
(l_i, w_i, h_i)	is, respectively, the length, the width and the height of item i;
m	is the number of boxes;
(L_ℓ, H_ℓ, W_ℓ)	is, respectively, the length, the height and the width of the box ℓ;
V_ℓ	is the volume of box ℓ ($V_\ell = L_\ell \cdot H_\ell \cdot W_\ell$);
(ML, MH, MW)	is, respectively, a large number for the length, the height and the width of the box ℓ.
Variables	
(x_i, y_i, z_i)	*front bottom left* positional coordinates of item i, as length, width and height, respectively (continuous variable);
a_{ij} (b_{ij})	is equal to 1 if item i is on the left (right) side (x-axis) of the item j and 0 otherwise (binary variable);
c_{ij} (d_{ij})	is equal to 1 if item i is behind (front) (y-axis) of the item j and 0 otherwise (binary variable);
e_{ij} (f_{ij})	is equal to 1 if item i is above (bellow) (z-axis) of the item j and 0 otherwise (binary variable);
lx_i, lz_i	binary variables that indicate if the length of item i is parallel to the x, y or z axis (length, width and height). For example, if lx_i assumes 1 if it's length is in x axis of the box, otherwise is equal to 0 (binary variable);
wy_i	binary variable that indicate if the width of item i is parallel to the x, y or z axis (length, width and height). For example, if wy_i assumes 1 if it's width is in y axis of the box, otherwise is equal to 0 (binary variable);
hz_i	binary variable that indicate if the height of item i is parallel to the x, y or z axis (length, width and height). For example, if hz_i assumes 1 if it's height is in z axis of the box, otherwise is equal to 0 (binary variable);
β_ℓ	assume the value 1 if box ℓ is used and 0 otherwise (binary variable);
$\alpha_{i\ell}$	is equal to 1 if item i is packed inside the box ℓ, and 0 otherwise (binary variable).

$$Z_{CLS} = \min \sum_{\ell=1}^{m} V_\ell \, \beta_\ell - \sum_{i=1}^{n} v_i \tag{3.1}$$

s.t.

// Placement constraints

$$x_i + l_i \cdot lx_i + w_i \cdot (lz_i - wy_i + hz_i) + h_i \cdot (1 - lx_i - lz_i + wy_i - hz_i) \le L_\ell + ML(1 - \alpha_{i\ell}) \qquad i = 1,...,n; \ell = 1,...,m; \tag{3.2}$$

$$y_i + w_i \cdot wy_i + l_i \cdot (1 - lx_i - lz_i) + h_i \cdot (lx_i + lz_i - wy_i) \le W_\ell + MW(1 - \alpha_{i\ell}) \qquad i = 1,...,n; \ell = 1,...,m; \tag{3.3}$$

$$z_i + h_i \cdot hz_i + w_i \cdot (1 - lz_i - hz_i) + l_i \cdot lz_i \le H_\ell + MH\ (1 - \alpha_{i\ell}) \qquad i = 1,...,n; \ell = 1,...,m; \tag{3.4}$$

// Non-overlapping constraints

$$x_i + l_i \cdot lx_i + w_i \cdot (lz_i - wy_i + hz_i) + h_i \cdot (1 - lx_i - lz_i + wy_i - hz_i) \le x_j + ML(1 - a_{ij}) \qquad i = 1,...,n-1; j = i+1,...n; \tag{3.5}$$

$$x_j + l_j \cdot lx_j + w_j \cdot (lz_j - wy_j + hz_j) + h_j \cdot (1 - lx_j - lz_j + wy_j - hz_j) \le x_i + ML(1 - b_{ij}) \qquad i = 1,...,n-1; j = i+1,...n; \tag{3.6}$$

$$y_i + w_i \cdot wy_i + l_i \cdot (1 - lx_i - lz_i) + h_i \cdot (lx_i + lz_i - wy_i) \le y_j + MW(1 - c_{ij}) \qquad i = 1,...,n-1; j = i+1,...n; \tag{3.7}$$

$$y_j + w_j \cdot wy_j + l_j \cdot (1 - lx_j - lz_j) + h_j \cdot (lx_j + lz_j - wy_j) \le y_i + MW(1 - d_{ij}) \qquad i = 1,...,n-1; j = i+1,...n; \tag{3.8}$$

$$z_i + h_i \cdot hz_i + w_i \cdot (1 - lz_i - hz_i) + l_i \cdot lz_i \le z_j + MH(1 - e_{ij}) \qquad i = 1,...,n-1; j = i+1,...n; \tag{3.9}$$

$$z_j + h_j \cdot hz_j + w_j \cdot (1 - lz_j - hz_j) + l_j \cdot lz_j \le z_i + MH(1 - f_{ij}) \qquad i = 1,...,n-1; j = i+1,...n; \tag{3.10}$$

$$a_{ij} + b_{ij} + c_{ij} + d_{ij} + e_{ij} + f_{ij} \ge \alpha_{i\ell} + \alpha_{j\ell} - 1 \qquad i = i,...n-1; j = i+1,...n; \ell = 1,...,m; \tag{3.11}$$

// Orthogonality Constraints

$$lx_i + lz_i \le 1 \qquad i = 1...n \tag{3.12}$$

$$lz_i + hz_i \le 1 \qquad i = 1...n \tag{3.13}$$

$$wy_i \le lx_i + lz_i \qquad i = 1...n \tag{3.14}$$

$$wy_i \le lz_i + hz_i \qquad i = 1...n \tag{3.15}$$

$$lx_i + hz_i \le 1 + wy_i \qquad i = 1...n \tag{3.16}$$

// Items and boxes selection constraints

$$\sum_{\ell=1}^{m} \alpha_{i\ell} = 1 \qquad i = 1,...,n; \qquad (3.17)$$

$$\sum_{i=1}^{n} \alpha_{i\ell} \leq n\, \beta_\ell \qquad \ell = 1,...,m; \qquad (3.18)$$

// Variable domain

$$\alpha_{i\ell}, \beta_\ell \in \{0,1\} \qquad i = 1,...,n; \ell = 1,...,m; \qquad (3.19)$$

$$a_{ij}, b_{ij}, c_{ij}, d_{ij}, e_{ij}, f_{ij} \in \{0,1\} \qquad i = 1,...,n-1; j = i+1,...n; \qquad (3.20)$$

$$lx_i, lz_i, wy_i, hz_i \in \{0,1\} \qquad i = 1,...,n; \qquad (3.21)$$

$$x_i, y_i, z_i \geq 0 \qquad i = 1,...,n. \qquad (3.22)$$

The objective function (3.1) is designed to minimize the residual volume of the selected box or boxes. It computes this by subtracting the sum of the volumes of the items from the sum of the volumes of the selected boxes. To ensure that the items are positioned within the boxes, constraints (3.2) through (3.4) have been formulated. These constraints prevent the item's position plus its length (or width or height, respectively) from extending beyond the box boundaries, denoted as L_ℓ, W_ℓ, and H_ℓ, when the item is packed into box ℓ ($\alpha_{i\ell} = 1$). Additionally, ML, MW, and MH are defined as $\max(L_\ell) - \min(L_\ell)$, $max(W_\ell) - \min(W_\ell)$, and $\max(H_\ell) - \min(H_\ell)$, respectively.

Constraints (3.5) – (3.10) are non-overlapping constraints designed to determine the relative positioning of items to one another, as illustrated in Figure 3. Activation of these constraints is dependent on constraint (3.11), which occurs when two items are packed into the same box ($\alpha_{i\ell} = 1$ and $\alpha_{j\ell} = 1$), as illustrated in Figure 2. In such instances, at least one of the position variables (a_{ij}, b_{ij}, c_{ij}, d_{ij}, e_{ij}, f_{ij}) assumes a value of 1, thereby activating one non-overlapping constraint (refer to Table 9).

Table 9 – Table depicting active constraint if that positional variable is active (equal to 1), only one can be active.

Positional Variable	Active Constraint
a_{ij}	(3.5)
b_{ij}	(3.6)
c_{ij}	(3.7)
d_{ij}	(3.8)
e_{ij}	(3.9)
f_{ij}	(3.10)

However, these described constraints (*Placement* and *Non-overlapping constraints*) work in conjunction with the orthogonal variables (lx_i, lz_i , wy_i, and hz_i) to determine the rotation of items. Constraints (3.12) to (3.16) restrict the combinations of orthogonal variables, allowing for only six possible item rotations (refer to Table 10). The coordination between orthogonal variables and these constraints dictates that each dimension of an item aligns with a single axis at any given time, achieved by enabling or disabling specific dimensions within constraints (3.2) to (3.10). For instance, if $(lx_i, lz_i, wy_i, hz_i) = (1, 0, 1, 1)$, it signifies that the length, width, and height of the item are oriented along the x, y, and z axes, respectively. This ensures the correct alignment of each item dimension within non-overlapping and placement constraints.

Table 10 – Table showing how the orthogonal variables determine the six possible rotations of an item. The first four columns represent variable values and the last three columns represent the dimension of the item that is in each axis, where 'l' is the length, 'w' is the width and 'h' is the height.

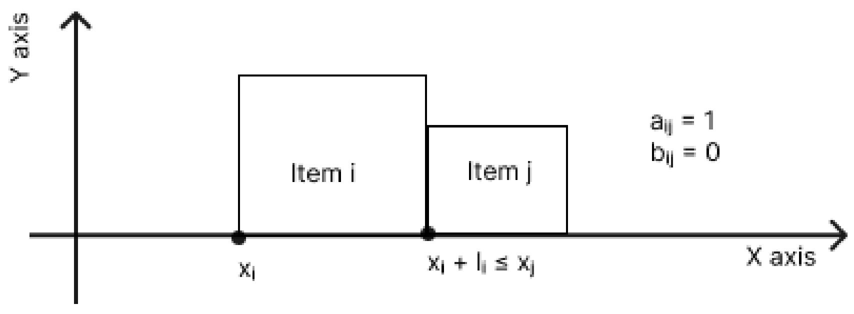
Orthogonal Variables				Axis Position		
lx	lz	wy	hz	x axis	y axis	z axis
1	0	1	1	l	w	z
0	0	0	1	w	l	z
1	0	0	0	l	z	w
0	1	1	0	z	w	l
0	1	0	0	w	z	l
0	0	0	0	z	l	w

Constraints (3.17) ensure that each item i is packed into only one box ℓ. Constraints (3.18) ensure that box ℓ is selected if at least one item i is packed in it.

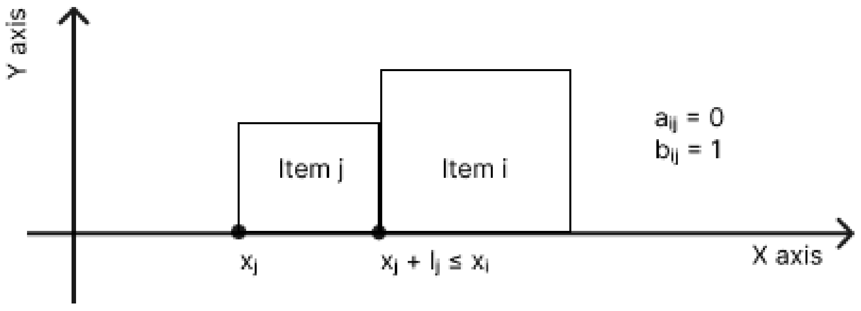
Remark. To limit the number of boxes the model can select (Q) to pack the items, the following constraint is added:

$$\sum_{\ell=1}^{m} \beta_\ell \leq Q \tag{3.23}$$

Figure 3 – Example of box placement for the Chen, Lee and Shen (1995) model depicting non-overlapping constraints.



(a) Item positioning forcing i to stay left of j.



(b) Item positioning forcing j to stay left of i.

3.2.2 Chen, Lee and Shen Model Without Rotation

As described in the previous section, Chen, Lee and Shen (1995) original model allows item rotation. Nevertheless, to facilitate the comparison with the Tsai, Malstrom and Kuo (1993) adapted model for multiple boxes (as outlined in Subsection 3.2.4), a version without rotation was defined, and labeled as Z_{CLSwr}. This modified version retains the same general model structure, with the difference being the orthogonal variables and constraints removal, dependent constraints were adjusted accordingly. The parameters and variables in this version remain consistent with the original model, and its formulation is represented by (3.24) – (3.34).

$$Z_{CLSwr} = \min \sum_{\ell=1}^{m} V_\ell \, \beta_\ell - \sum_{i=1}^{n} v_i \tag{3.24}$$

s.t.

// Placement constraints

$$x_i + l_i \le L_\ell + ML\,(1 - \alpha_{i\ell}) \qquad i = 1,\dots,n; \ell = 1,\dots,m; \tag{3.25}$$

$$y_i + w_i \le W_\ell + MW\,(1 - \alpha_{i\ell}) \qquad i = 1,\dots,n; \ell = 1,\dots,m; \tag{3.26}$$

$$z_i + h_i \le H_\ell + MH\,(1 - \alpha_{i\ell}) \qquad i = 1,\dots,n; \ell = 1,\dots,m; \tag{3.27}$$

// Non-overlapping constraints

$$x_i + l_i \le x_j + ML(1 - a_{ij}) \qquad i = 1,\dots,n-1; j = i+1,\dots n; \tag{3.28}$$

$$x_j + l_j \le x_i + ML(1 - b_{ij}) \qquad i = 1,\dots,n-1; j = i+1,\dots n; \tag{3.29}$$

$$y_i + w_i \le y_j + MW(1 - c_{ij}) \qquad i = 1,\dots,n-1; j = i+1,\dots n; \tag{3.30}$$

$$y_j + w_j \le y_i + MW(1 - d_{ij}) \qquad i = 1,\dots,n-1; j = i+1,\dots n; \tag{3.31}$$

$$z_i + h_i \le z_j + MH(1 - e_{ij}) \qquad i = 1,\dots,n-1; j = i+1,\dots n; \tag{3.32}$$

$$z_j + h_j \le z_i + MH(1 - f_{ij}) \qquad i = 1,\dots,n-1; j = i+1,\dots n; \tag{3.33}$$

$$\begin{aligned} & a_{ij} + b_{ij} + c_{ij} + d_{ij} + e_{ij} \\ & \qquad + f_{ij} \ge \alpha_{i\ell} + \alpha_{j\ell} - 1 \qquad i = 1,\dots,n-1; j = i+1,\dots n; \ell = 1,\dots,m; \end{aligned} \tag{3.34}$$

// Items and boxes selection constraints

Same as (3.17), (3.18) *and* (3.23)

// Variable domain

Same as (3.19) *to* (3.22)

The main difference between the models is that non-overlapping and placement constraints have been adjusted. With this change, all items length, width and height are aligned along the x, y and z axes, respectively.

3.2.3 Tsai, Malstrom and Kuo Model

Tsai, Malstrom and Kuo (1993) deals with 2D and 3D packing problems. First, the authors propose a model for the 2D packing problem considering that items can be placed in continuous positions. Then, a 3D packing model is presented, which is described here.

The objective is to maximize the total volume of items packed into a box. Therefore, the model does not require that all items be packed. To deal with this situation, the authors represent the box as shown in Figure 4, for the 2D problem. Items placed outside the rectangle defined by (X^o, Y^o) and $(X^o + L, Y^o + W)$ are considered unpacked. In addition, the items lengths are placed parallel to the x-axis and cannot be rotated. The variables and parameters of the model are defined in Table 11. The model presented by the authors is described by (3.35)–(3.51), and labeled as Z_{TMK}.

Figure 4 – Box representation for the 2D problem used by Tsai, Malstrom and Kuo (1993). Item i is inside the box and item j is outside the box. This image is a representation of the variable α_i activation, for item i, $\alpha_i = 1$, forcing it into the box and for item j, $\alpha_j = 0$, allowing it to be outside the box and thus not be counted in the objective function.

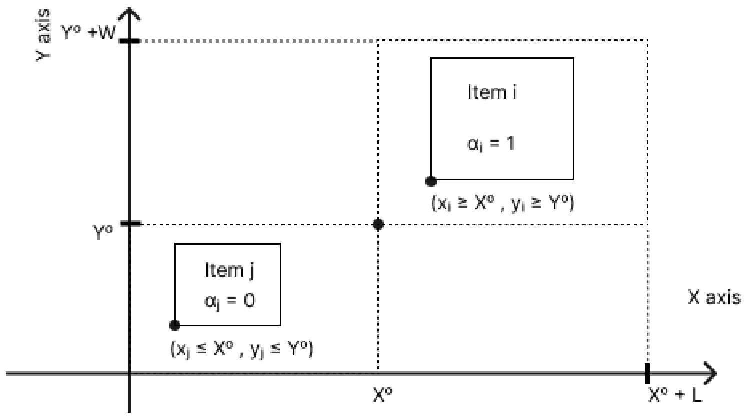


Table 11 – Table with parameters and variables for Tsai, Malstrom and Kuo (1993) original model without rotation and for only one box.

Additional Parameters	
(X^o, Y^o, Z^o)	is the initial positional coordinates of the box, as length, width and height, respectively.
Additional Variables	
α_i	assumes the value 1 if item i is packed inside the box and zero otherwise (binary variable);
$u_1^{ij}, u_2^{ij}, u_3^{ij}$	auxiliary variables used to activate the non-overlapping constraints between items i and j (binary variables).

$$Z_{TMK} = \max \sum_{i=1}^{n} v_i \; \alpha_i \tag{3.35}$$

s.t.

// Placement constraints

$$x_i \geq X^o \; \alpha_i \qquad i = 1, \ldots, n; \tag{3.36}$$

$$y_i \geq Y^o \; \alpha_i \qquad i = 1, \ldots, n; \tag{3.37}$$

$$z_i \geq Z^o \; \alpha_i \qquad i = 1, \ldots, n; \tag{3.38}$$

$$x_i + l_i \leq X^o + L_\ell \qquad i = 1, \ldots, n; \tag{3.39}$$

$$y_i + w_i \leq Y^o + W_\ell \qquad i = 1, \ldots, n; \tag{3.40}$$

$$z_i + h_i \leq Z^o + H_\ell \qquad i = 1, \ldots, n; \tag{3.41}$$

// Non-overlapping constraints

$$x_i + l_i \leq x_j + ML \; (u_2^{ij} + u_3^{ij}) \qquad i = 1, \ldots, n-1; j = i+1, \ldots n; \tag{3.42}$$

$$x_j + l_j \leq x_i + ML \; (u_1^{ij} + u_3^{ij}) \qquad i = 1, \ldots, n-1; j = i+1, \ldots n; \tag{3.43}$$

$$y_i + w_i \leq y_j + MW \; (u_1^{ij} + u_2^{ij}) \qquad i = 1, \ldots, n-1; j = i+1, \ldots n; \tag{3.44}$$

$$y_j + w_j \leq y_i + MW \; (2 - (u_1^{ij} + u_2^{ij})) \qquad i = 1, \ldots, n-1; j = i+1, \ldots n; \tag{3.45}$$

$$z_i + h_i \leq z_j + MH \; (2 - (u_2^{ij} + u_3^{ij})) \qquad i = 1, \ldots, n-1; j = i+1, \ldots n; \tag{3.46}$$

$$z_j + h_j \leq z_i + MH \; (2 - (u_1^{ij} + u_3^{ij})) \qquad i = 1, \ldots, n-1; j = i+1, \ldots n; \tag{3.47}$$

$$1 \leq u_1^{ij} + u_2^{ij} + u_3^{ij} \leq 2 \qquad i = 1, \ldots, n-1; j = i+1, \ldots n; \tag{3.48}$$

// Variable domain

$$u_1^{ij}, u_2^{ij}, u_3^{ij} \in \{0, 1\} \qquad i = 1, \ldots, n-1; j = i+1, \ldots n; \tag{3.49}$$

$$\alpha_i \in \{0, 1\} \qquad i = 1, \ldots, n; \tag{3.50}$$

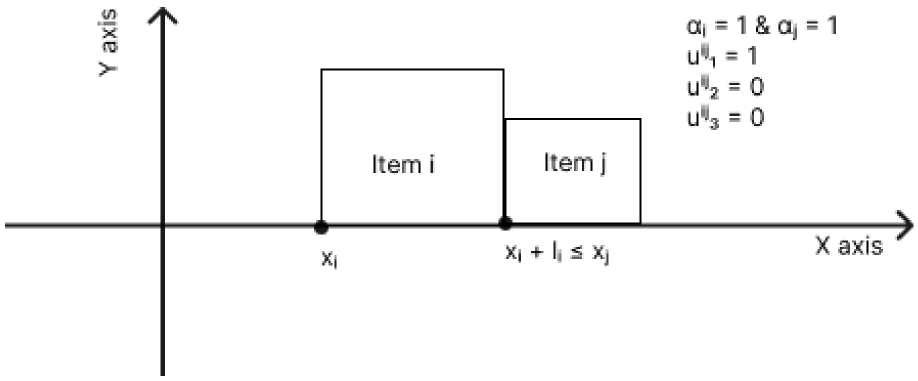
$$x_i, y_i, z_i \geq 0 \qquad i = 1, \ldots, n. \tag{3.51}$$

The objective function (3.35) maximizes the total volume of items packed in the box. The objective is connected to constraints (3.36) to (3.38). If α_i is active ($\alpha_i = 1$) then (x_i, y_i, z_i) is forced to be inside the box, i.e., being greater than (X^o, Y^o, Z^o). Otherwise, if α_j is not active

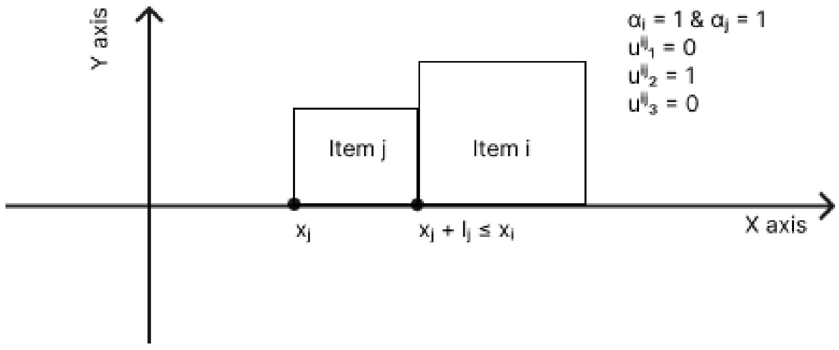
($\alpha_j = 0$), the coordinates of the item can be placed outside the box, i.e., items are placed before coordinates (X^o, Y^o, Z^o) as illustrated in Figure 4. Also, constraints (3.39) to (3.41) limit items to be inside the box, coordinates of item i are smaller than $(X^o + L - l_i, Y^o + W - w_i, Z^o + H - h_i)$.

Constraints (3.42) to (3.47) avoid items overlapping with the help of variables $(u_1^{ij}, u_2^{ij}, u_3^{ij})$. This makes that at least one non-overlapping constraint is always active (see constraints (3.48)) for each pair of items, as it can be seen in Table 12. This set of constraints works in pairs for each axis and each pair of items. For example, suppose the non-overlap is active for items i and j at the x axis (Constraints (3.42) and (3.43)). In this case, they impose that $x_i + l_i \leq x_j$ or $x_j + l_j \leq x_i$ when $u_3^{ij} = 0$ and if $u_1^{ij} = 1$ or $u_2^{ij} = 1$, respectively. See Figure 5 and Table 12 for more details.

Figure 5 – Item positioning examples for the Tsai, Malstrom and Kuo (1993) model. Figure 5a and Figure 5b depict active non-overlapping constraints (3.42) and (3.43), respectively.



(a) Item positioning forcing i to stay left of j.



(b) Item positioning forcing j to stay left of i.

Table 12 – Overlapping variables and their values, adapted from Tsai, Malstrom and Kuo (1993). For each pair of items i and j (u_1^{ij}, u_2^{ij}, u_3^{ij}), we have at least one constraint active, and the value of (x_i, y_i, z_i) that should be respected.

Variable values			RHS of each constraint						
u_1^{ij}	u_2^{ij}	u_3^{ij}	(3.42)	(3.43)	(3.44)	(3.45)	(3.46)	(3.47)	Active Constraint
1	0	0	x_j	M	M	M	2M	M	(3.8)
0	1	0	M	x_i	M	M	M	2M	(3.9)
0	0	1	M	M	y_j	2M	M	M	(3.10)
1	1	0	M	M	2M	y_i	M	M	(3.11)
0	1	1	2M	M	M	M	z_j	M	(3.12)
1	0	1	M	2M	M	M	M	z_i	(3.13)

3.2.4 Adapted Tsai, Malstrom and Kuo Model

As seen, the original Tsai, Malstrom and Kuo (1993) model can only handle one box at a time. In order to make multiple boxes possible (the studied problem), two essential points are added to this model. Initially, following the approach taken by Chen, Lee and Shen (1995), a variable $\alpha_{i\ell}$ is introduced to assign item i to box ℓ. The second point relates to the placement of the boxes. In the adjusted model, the boxes are arranged sequentially along the length-axis (x-axis) as shown in Figure 6. Thus, there is a need to add new parameters $(X_\ell^o, Y_\ell^o, Z_\ell^o)$ and $(X_\ell', Y_\ell', Z_\ell')$ to the model, along with constraints on the box placement limits in relation to $\alpha_{i\ell}$. Further details on the additional parameters can be found in Table 13, while the adapted model is defined by equations (3.52) – (3.56), and labeled as Z_{TMKa}.

Figure 6 – Example of box placement for the developed model.

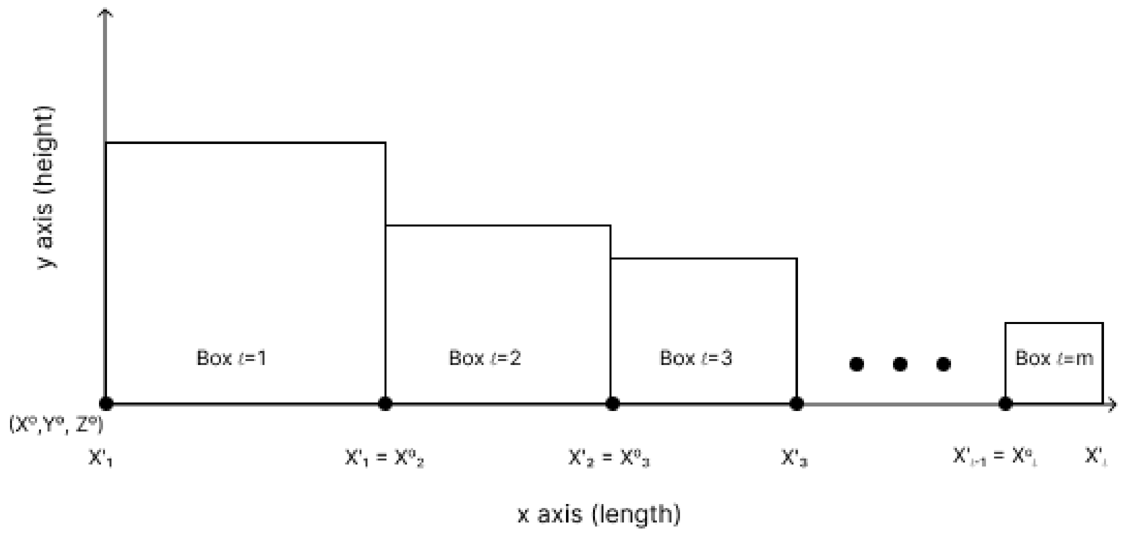


Table 13 – Table with extra parameters for the adapted model.

Additional Parameters	
$(X^o_\ell, Y^o_\ell, Z^o_\ell)$	as initial positional coordinates of box ℓ, as length, height and width, respectively;
$(X'_\ell, Y'_\ell, Z'_\ell)$	as final positional coordinates of box ℓ, as length, height and width, respectively;

$$Z_{TMKa} = \min \sum_{\ell=1}^{m} V_\ell \cdot \beta_\ell - \sum_{i=1}^{n} v_i \tag{3.52}$$

s.t.

// Placement constraints

$$x_i \geq X^o_\ell - X^o_\ell(1-\alpha_{i\ell}) \qquad i=1,\ldots,n;\ \ell=1,\ldots,m; \tag{3.53}$$

$$x_i + l_i \leq X^{'}_\ell + ML\ (1-\alpha_{i\ell}) \qquad i=1,\ldots,n;\ \ell=1,\ldots,m; \tag{3.54}$$

$$y_i + w_i \leq Y^{'}_\ell + MW\ (1-\alpha_{i\ell}) \qquad i=1,\ldots,n;\ j=1,\ldots,m; \tag{3.55}$$

$$z_i + h_i \leq Z^{'}_\ell + MZ\ (1-\alpha_{i\ell}) \qquad i=1,\ldots,n;\ \ell=1,\ldots,m; \tag{3.56}$$

// Non-overlapping constraints

Same as (3.42) *to* (3.48)

// Items and boxes selection constraints

Same as (3.17), (3.18) *and* (3.23)

// Variable domain

Same as (3.49), (3.19) *and* (3.51)

The objective function (3.52) is the same as in Chen, Lee and Shen (1995), minimizes the residual volume ($V_\ell - \sum_{i=1}^{n} v_i$) for selected boxes. Constraints (3.53) and (3.56) set the upper and lower limits for the placement of the items depending on which box they are packed in. In this model, all boxes are aligned along the x axis. If variable $\alpha_{i\ell}$ is equal to one for the pair (i,ℓ), then the value of x_i is restricted by constraints (3.53) and (3.54) to the limits of box ℓ, i.e., $X^0_\ell \leq x_i \leq X^{'}_\ell - l_i$. Similarly, constraints (3.55) and (3.56) ensure that the items are within the limits of Y'_ℓ and Z'_ℓ. The main difference is that each box ℓ does not need the equivalent (3.53) for the y axis and z axis, because for them the lower bound is zero (3.51).

Non-overlapping of items is ensured as defined by Tsai, Malstrom and Kuo (1993) in constraints (3.42) to (3.48). Analogous to Chen, Lee and Shen (1995), constraints (3.17) ensure that each item i is packed into only one box ℓ, and constraints (3.18) ensure that box ℓ is selected if at least one item i is packed into it. Constraints (3.23) limit the number of boxes the model can select, with the limit being Q.

3.2.5 Equivalent Constraints

Some optimization problems can be represented by alternative formulations. In this work, three models for the same problem are studied: the original Chen, Lee and Shen (1995) model, it's version without rotation and the Tsai, Malstrom and Kuo (1993) adapted model for multiple boxes. In these models, constraint (3.18), which ensures the activation of a box if that box is used, can be replaced without losing generality by:

$$\alpha_{i\ell} \leq \beta_\ell \qquad\qquad i = 1, \ldots n;\ \ell = 1, \ldots m. \tag{3.57}$$

This new constraint, originally used in facility location problems, can reduce the gap between the linear relaxation and the convex hull [Wolsey 2020]. As a drawback, the number of constraints increases from n to $n \cdot m$, therefore computational experiments were conducted in Subsection 3.3.5 to evaluate whether this constraint can improve the performance of the models.

3.2.6 A Toy Problem

This toy problem considers an order consisting of five items and a total volume of 11,680. There are four boxes available to pack these items. Boxes 1 and 2 are designed to be small (8,000 and 12,000 volumes, respectively). The entire order cannot fit in a single box. However, it is possible to fit the order into these two boxes by splitting the items. Box 3 with a volume of 27,000 was created to accommodate the complete order with a small amount of extra space. Alternatively, box 4 is capable of fitting the order as well, with more leftover space than Box 3. An example of the data is available in Table 14.

Table 14 – Data of the toy problem instance.

	Length	Width	Height	Volume
Item 1	20	5	30	3000
Item 2	10	20	20	4000
Item 3	10	18	20	3600
Item 4	5	8	18	720
Item 5	8	15	3	360
Box 1	20	20	20	8000
Box 2	20	20	30	12000
Box 3	30	30	430	27000
Box 4	40	40	40	64000

The toy instance was solved with only one box, i.e. adding the model constraint (3.23) with $Q = 1$. Then it was solved again, allowing the model to choose up to two boxes ($Q = 2$). In the first version, all items were assigned to box 3 ($30 \times 30 \times 30$) with a waste volume of 15,320 (objective function). The solution is displayed in Figure 7. In the second version, items 2 and 3 are placed in box 1 ($20 \times 20 \times 20$), and items 1, 4, and 5 are placed in box 2 ($20 \times 20 \times 30$).

The total volume wasted is 8,320, which is less than the first one, as anticipated. The solution obtained is displayed in Figure 8.

Figure 7 – Solution for the toy instance with only one box.

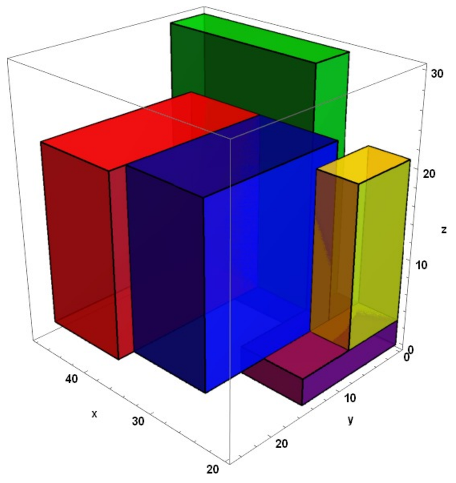
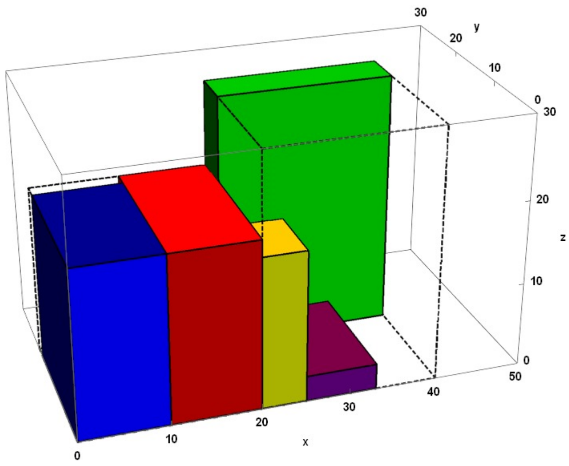


Figure 8 – Solution for the toy instance with up to two boxes available. Dashed lines defines the boxes.



3.2.7 Models Analysis

The models, namely Z_{CLS}, Z_{CLSwr}, and Z_{TMKa}, share a common objective function, which is the minimization of wasted volume while selecting the most suitable box(es). The exception is Z_{TMK}, which aims to maximize the number of items packed in a single box. In this study, the model from Tsai, Malstrom and Kuo (1993) (Z_{TMK}) was adapted to enable both box selection and item allocation (Z_{TMKa}). Additionally, the model by Chen, Lee and Shen (1995) was simplified by removing item rotation, resulting in Z_{CLSwr}.

Table 15 – Number of variables for each model.

Model	Continuous Variables	Binary Variables
Z_{CLS}	$3n$	$nm+m+6n(n-1)/2+4n$
Z_{CLSwr}	$3n$	$nm+m+6n(n-1)/2$
Z_{TMK}	$3n$	$n+3n(n-1)/2$
Z_{TMKa}	$3n$	$nm+m+3n(n-1)/2$

This new approach to assigning boxes in (x, y, z) coordinates is proposed to achieve a smaller number of variables. The proposal is to consider all boxes aligned along the x-axis and to limit the x-axis position variable whenever an item i is assigned to a box ℓ (constraints (3.53) and (3.54), defining box limits), as described in Subsection 3.2.4. The first reason to consider this regards the number of binary variables, Z_{TKMa} has a lower number of binary variables when compared to the similar formulation of Z_{CLSwr} (see Table 15). Then, the main advantage of this method is that it uses the same non-overlapping constraints for all boxes with the three binary variables for each pair of items (see Table 16). To the best of our knowledge, a similar approach has not been employed in previous literature on this topic.

Table 16 – Number of constraints for each model.

Model	Number of constraints
Z_{CLS}	$3mn+6n(n-1)/2+mn(n-1)/2+6n+m$
Z_{CLSwr}	$3mn+6n(n-1)/2+mn(n-1)/2+n+m$
Z_{TMK}	$6\,n+7n(n-1)/2$
Z_{TMKa}	$4nm+7n(n-1)/2+n+m$

In comparison, Chen, Lee and Shen (1995) (Z_{CLS} and Z_{CLSwr}) use constraints (3.11) that force the activation of non-overlapping constraints (3.5 to 3.10) depending on where the item is placed. This requires the use of six binary variables (a_{ij}, b_{ij}, c_{ij}, d_{ij}, e_{ij}, f_{ij}) for each pair of items (Subsection 3.2.1), in contrast to three (u_1^{ij}, u_2^{ij}, u_3^{ij}) from Tsai, Malstrom and Kuo (1993). This is one advantage of this last model.

Moreover, the original model (Z_{CLS}) contains additional variables and constraints to account for the item rotation. Specifically, lx_i, lz_i, wy_i, and hz_i are the four additional variables that result in an extra $4n$ variables. Furthermore, the new orthogonal constraints result in an additional $5n$ constraints. Even though such an approach increases the model complexity, adopting item rotation can improve item allocation solutions.

3.3 Computational Experiments

To conduct the computational experiments, 1,000 instances (orders) were generated based on the literature. A specific instance is defined by the number of items and the dimensions of each item. Two methods were utilized to determine the boxes available for packing the items. First, the box set from the same source presented in Fontaine and Minner (2022) was utilized. Then, two different box sets were created by applying Herz's grid strategy [Herz (1972)]. The instances were solved using the three sets of boxes. Subsection 3.3.1 provides a detailed description of the instances. The results of the computational experiments are reported and discussed in Subsection 3.3.5.

Computational experiments were conducted on a system consisting of an Intel® Core^{TM} i7-7700 CPU 3.60GHz x 8 with 15.5 GiB RAM and OS Ubuntu 20.04.4 LTS 64-bit. The models were developed using Julia version 1.7.2, and the instances were solved using Gurobi Optimizer version 9.5.1 build v9.5.1rc2 (linux64), with a time limit of 1,800 seconds.

3.3.1 Instances Sets

Hübner, Holzapfel and Kuhn Heinrich (2015) presented an overview of multichannel shopping by investigating several aspects of this business, including picking and packing. According to the authors, the number of items in each order typically follows a probability distribution that tends to have fewer items (50% with 1 to 2 items). Table 17 shows the four order classes defined by the authors, where the first column defines the class, and the second and third columns show the probability of each class and the number of items in each class. For instance, an order has a 25 % probability of having 2 to 3 items. The authors do not address the specifics of the items and boxes in their study.

Table 17 – Probability and number of items for each instance Class [Hübner, Holzapfel and Kuhn Heinrich (2015)].

Class	Probability	Number of Items
1	50%	1 to 2
2	25%	2 to 3
3	8%	3 to 4
4	16%	4 or more

The number of items for each instance is defined based on the probability presented in Table 17, considering the limit of ten items for class 4. Subsections 3.3.2 and 3.3.3 explain the three types of boxes used and how items are generated, respectively.

3.3.2 Box Sets

Two strategies were used for defining the available boxes. The first strategy utilized the set of boxes from Fontaine and Minner (2022). The second strategy involved generating boxes using the Herz grid strategy [Herz (1972)]. The goal is to analyze whether using Herz boxes can improve the packing of an item set.

Fontaine and Minner (2022) consider a collection of boxes used by third-party sellers of a large e-commerce company in their computational experiments [Amazon Boxes (2021)]. The box set includes 123 boxes with dimensions ranging from 13 to 133 inches in length, 5 to 147 cm in height, and 13 to 94 cm in width. The volume distribution of these boxes, referred to as AMB, is shown in Figure 9b. A complete table of all AMB boxes can be found in Appendix C.

The second set of boxes, referred to as Herz Boxes (HB), was generated using the Herz Grid [Herz (1972)]. Herz originally proposed this strategy to generate a combination of elements that could be cut from an object. Later, the method was adapted to define the grid of points for discrete packing models. Without loss of optimality, the Herz grid generates a reduced grid of feasible points for placing items in a box. In this work, the generated boxes are expected to pack the items better than other types of boxes by reducing the number points that lead to poor packing.

The set of points generated by the Herz grid for each dimension is defined in an analogous way. For instance, when considering the length dimension, the set of points P_L is given by

$$P_L = \{x \mid x = \sum_{i=1}^{m} l_i b_i,\ 0 \leq x \leq L_j - \min\{l_i | i = 1,...,m\},\ b_i \in \mathbb{Z}^+,\ i = 1,...,m\}, \tag{3.58}$$

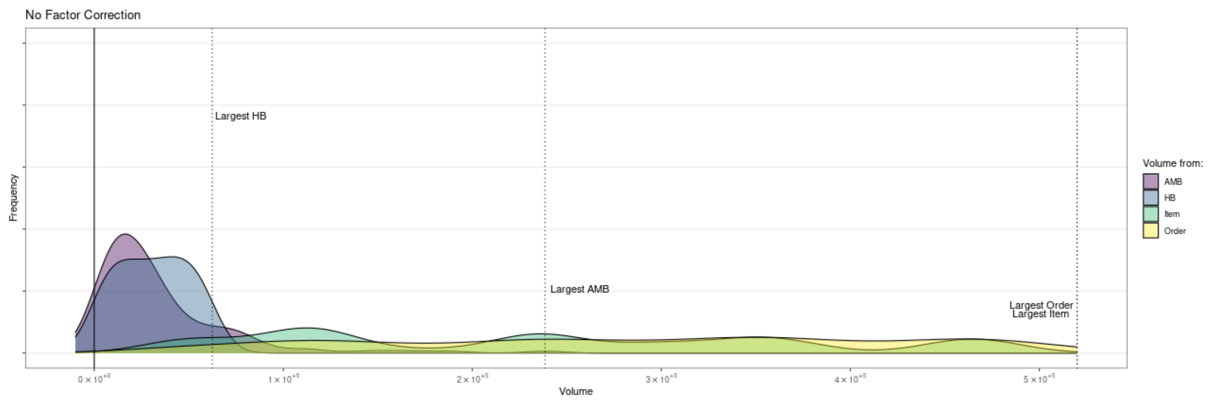
where l_i represents the length of item i and b_i represents the quantity of times item i can fit into the box. L_j refers to the length of the box j, while m represents the total number of item types in the instance. This set of points encompasses all possible placement of the items in the box on the axis x. For example, given an instance with a box with a length equal to 16 ($L = 16$) and two items with length $l_1 = 4$ and $l_2 = 6$, then $b_i = \{0,1,2\}$ to $i = 1,2$. The P_L is equal to:

$$P_L = \{0,4,6,8,10,12\}.$$

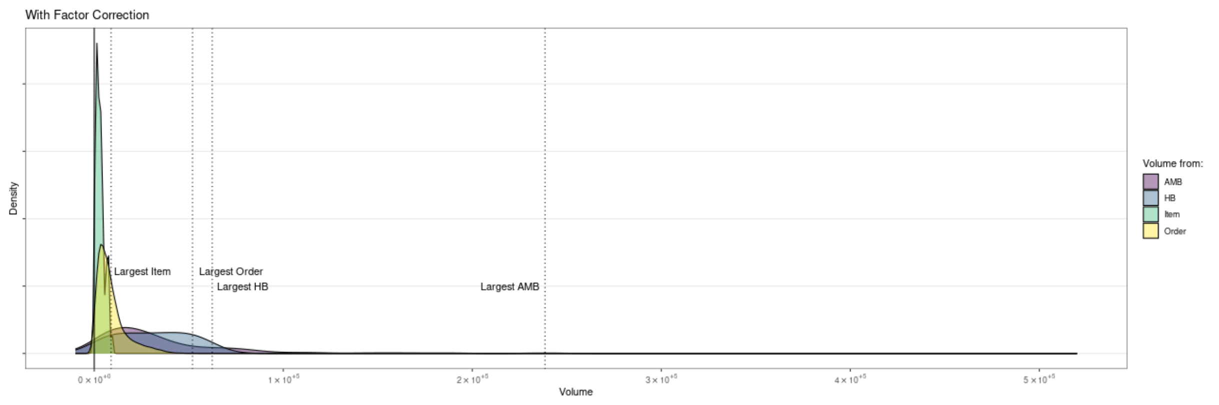
This strategy was applied to each instance. The limit (L_j) used was the worst scenario, which is the sum of the dimensions of all items aligned in a row for each instance. This resulted in a grid of 162, 166, and 181 points for each (x, y, z) axis, based on the characteristics of each instance. These points resulted in 4,867,452 combinations of possible points to define a box, an improvement over the more than 6,000,000 when using a grid with a discretization step of one. From this, a set of 123 boxes was randomly selected, sampling was done without replacement, and a random number generator was used to determine the selected boxes. Also, boxes with a volume greater than 120% of the largest instance volume and not respecting $L \geq W \geq H$ were excluded. The entire set can be seen in Appendix E.

It can be seen in Figure 9 that by comparing the volume distribution of the boxes generated, the HB box set has boxes that are larger than the item and instances volumes (Figure 9a). This could result in additional residual volume during the packing process. To overcome this issue, a pre-processing step was taken to improve these boxes. Grid points higher than two standard deviations of the mean for each axis were excluded and the box volumes were sampled from the probabilities of the instance volume distribution. This strategy aimed to exclude larger boxes and forced the box volumes to be similar to the instance volumes (see Figure 9c). This third set of boxes is referred to as HBnd (HB new distribution).

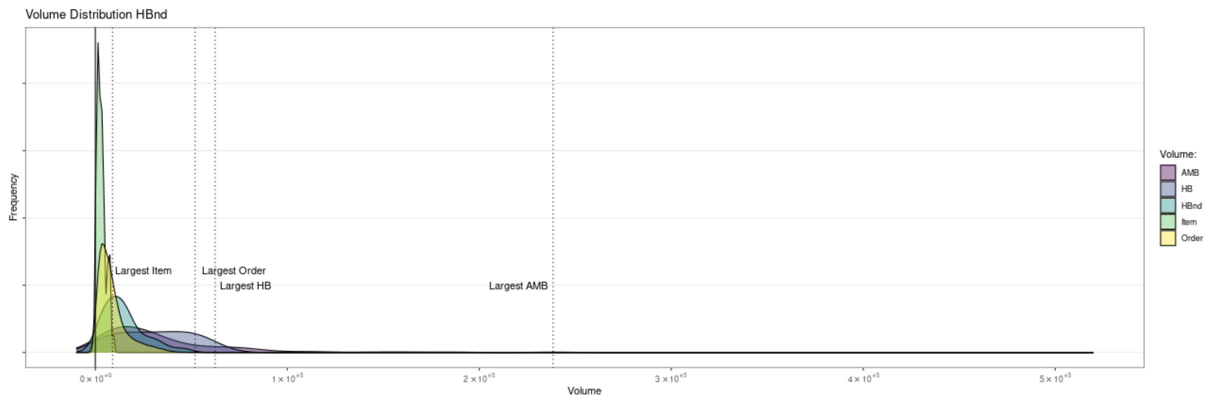
Figure 9 – Density plot displaying the volume of instances (orders) without factor adjustment for the items (a) and with adjustment (b). Vertical dotted lines represent the highest value in each dataset.



(a) Distribution of volumes without factor correction.



(b) Distribution of volumes with factor correction.



(c) Distribution of volumes including HBnd box set.

A similar method was introduced by Brinker and Gündüz 2016. The authors have devised a set of techniques to be employed in the packaging processes of e-commerce companies. One of the methods employed is the usage of the p-median model to determine the ideal package sizes for the given demand while minimizing the used space, i.e., the wasted volume. To define the available boxes, the authors set the dimensions of the boxes as follows: the minimum is equal to the maximum dimension of the smallest item, and the maximum is equal to a number defined by the company. A fixed step is defined to determine the coordinates. Also, by eliminating rotation and applying $L \geq W \geq H$, they got 2,600 different box sizes. However, since the step and maximum size are manually selected, it is possible to generate boxes with dimensions that are not suitable for packing the instances. On the other hand, this strategy has the advantage of reducing the number of box sizes.

3.3.3 Item Sets

A set of 90 items with dimensions ranging from 25 to 115 is presented by Fanslau and Bortfeldt (2010). The volume distribution among the items, instances, and boxes is illustrated in Figure 9. Figure 9a shows that the volumes of the instances and items exceed those of the box sets. The dimensions of the items were divided by a factor of 4.64 to adjust their volume and ensure that they fit within the box volume. The adjustment is illustrated in Figure 9b, which demonstrates that all item and instance volumes fit within the box volumes with the adjustment, leaving a small amount of extra space. The items for each instance were randomly selected from a customized set, sampling was done with replacement and a random number generator was used to determine the items. The items can be found in Appendix A.

3.3.4 Exploration Goals

With the packing models and box sets defined, it is now possible to begin the exploration of the first objective: the study of packing in the context of e-commerce.

First, the described formulations are evaluated to identify linear relaxation and runtime performance differences in terms, linear relaxation and runtime. This is followed by an evaluation of item rotation, considering both residual volume and runtime. The primary objective of these tests is to compare the models that define the best boxes for each order.

Then, the novel box generation strategy is tested to evaluate its impact on residual volume reduction. For this purpose, the three box sets are executed using the selected model for all instances and the results are compared for evaluation.

3.3.5 Computational Results

Computational experiments were performed on 1,000 instances (orders). The instances were generated as described in Subsection 3.3.1. Firstly, the number of items is generated according to Table 17 probabilities. Then, the items were selected as explained in Subsection 3.3.3. As an example, the first 20 instances are available in Appendix B.

This subsection employs these instances to: i) explore alternative formulations and compare the models (Subsection 3.3.5); ii) examine the impacts of rotation on the chosen model (Subsection 3.3.5); and iii) compare different box sets (Subsection 3.3.5). Two scenarios are considered to achieve these goals: i) all the items can be packed into a single box ($Q = 1$), or ii) items can be packed into up to two boxes ($Q = 2$), as defined by constraint (3.23) in Subsection 3.2.1.

Models are defined as Z_{TMKa}, Z_{CLSwr} and Z_{CLS}, for the adapted Tsai, Malstrom and Kuo (1993), Chen, Lee and Shen (1995) without and with rotation respectively (Section 3.2). Additionally, alternative constraints, as described in Subsection 3.2.5, are represented by Z_1 and Z_2. For example, if Z_{TMKa} is executed with the original constraint it is reefed as Z_{TMKa1} and for the alternative constraint as Z_{TMKa2}, same for the other two models.

A pre-processing phase is used to reduce the number of boxes available for each instance. For $Q = 1$ or $Q = 2$, boxes that do not pack the smallest item in dimensions or volumes are eliminated. For $Q = 2$, we also discard boxes larger than the box used in the solution of the instance considering $Q = 1$.

Phase 1 - Performance of Alternative Formulations

Models without rotation, as defined in Subsection 3.2.4 [Tsai, Malstrom and Kuo (1993)] and Subsection 3.2.2 [Chen, Lee and Shen (1995)] and the alternative formulations described in Subsection 3.2.5 were first solved in their linear relaxation version (Z^{Rel}) to analyze the dual bounds. In this analysis, all 1,000 instances were used, considering packaging in only one box ($Q = 1$) and two boxes ($Q = 2$) of the AMB box set. In a second study, the original version of the formulations was solved to compare the time required to solve this set of instances.

Formulations without rotation are defined as Z_{TMKa1} for the multiple box Tsai, Malstrom and Kuo (1993) adapted model and Z_{CLSwr1} for the Chen, Lee and Shen (1995) model without rotation, where in these formulations the original constraint $\sum_{i=1}^{n} \alpha_{i\ell} \leq n\ \beta_\ell$ is used. The alternative formulation is defined as Z_{TMKa2} and Z_{CLSwr2} with the constraint $\alpha_{i\ell} \leq \beta_\ell$, as described in Subsection 3.2.5.

Analysing the dual bounds for the formulations

The linear relaxation versions were solved to measure the impact of the described alternative formulations on the dual bounds. Since the Z_{TMKa1} and Z_{CLSwr1} formulations give the same results, only the data for the former are shown, the same for the results for Z_{TMKa2} and Z_{CLSwr2}. Table 18 shows the summary of the results, including the mean, median and standard deviation (SD) of the percentage improvement of the objective functions for all instances grouped by number of items. The percentage improvement for each instance is calculated as $\frac{Z^{Rel}_{TMKa2} - Z^{Rel}_{TMKa1}}{Z^{Rel}_{TMKa2}} \cdot 100$. From the table, it can be seen that the alternative formulation (Z^{Rel}_{TMKa2}) shows a small improvement for the dual bound over the standard formulation (Z^{Rel}_{TMKa1}) with an increase of 1.5 to 1.9% on average. This result is consistent with the literature [Wolsey 2020], which suggests that this type of formulation provides better dual bounds. Similar results were observed for $Q = 1$, but the gain was minimal, less than 0.2% in this case. Detailed results for $Q = 1$ can be found in Appendix F in Table 46.

Table 18 – Percentage difference between linear relaxations ($Q = 2$).

	Improvement		
Items	Mean	Median	SD
1	0.00%	0.00%	0.00%
2	1.82%	1.38%	1.82%
3	1.90%	1.57%	1.50%
4	1.84%	1.47%	1.54%
5	1.60%	1.47%	0.67%
6	1.49%	1.52%	0.60%
7	1.60%	1.40%	0.73%
8	1.40%	1.35%	0.62%
9	1.61%	1.46%	0.84%
10	1.42%	1.41%	0.46%

Analysing the run times for the formulations

Even though the alternative formulation results in better dual bounds, analyzing the computational times is important. Execution times for the mixed-integer models are shown in Table 19 for Z_{CLSwr} and in Table 20 for Z_{TMKa}. The tables contain a summary with mean, median, standard deviation (SD) and total time in seconds. Each row reports the data of instances with the number of items described in the first column. The quantity of instances is presented in the last column.

From these results, it is clear that there is no significant difference between the original and alternative formulations in terms of time (see the mean and total times). Even if the alternative formulation has slightly better times, this is not consistent, as seen in the instances with 8 and 9 items in Table 20. Also, Table 21 shows the only instance that reached the time limit in the Z_{TMKa2} formulation. This instance has been removed from the Tables 19 and 20 to avoid distortion of the data. This instance was complex to solve for all models as the times are well above the average. It is also possible to see the inconsistency of the time because, although Z_{CLSwr2} has shorter times, this is not seen for this instance.

Based on the results obtained, it is possible to compare the performance of the models. It is easy to see that Chen, Lee and Shen (1995) (Z_{CLSwr}) outperforms Tsai, Malstrom and Kuo (1993) (Z_{TMKa}). This difference is probably due to the different box allocation strategy, which in the case of Z_{TMKa} ends up using a large bigM on constraints (3.54, define box limits) and (3.42 - 3.43, define non-overlapping). These results are particularly noticeable in instances with 8 to 10 items. Although the mean and median of both formulations are close, the total time is much higher for Z_{TMKa}, which can be explained by the higher standard deviation, i.e. there is a high variation in the solving times. In summary, since Chen, Lee and Shen (1995) showed superior performance and the alternative formulation provided slightly better times, these will be used from now on.

Table 19 – Run times for the two formulations of Z_{CLSwr} (without rotation) by the number of items considering AMB box set and $Q = 2$.

	Z_{CLSwr1}				Z_{CLSwr2}				
Items	Mean	Median	SD	Total	Mean	Median	SD	Total	Instances
1	<1	<1	<1	<1	<1	<1	<1	<1	251
2	<1	<1	<1	<1	<1	<1	<1	<1	380
3	<1	<1	<1	1	<1	<1	<1	1	155
4	<1	<1	<1	2	<1	<1	<1	1	71
5	<1	<1	<1	1	<1	<1	<1	<1	18
6	<1	<1	<1	5	<1	<1	<1	4	25
7	<1	<1	<1	21	<1	<1	<1	15	31
8	2	1	2	61	2	1	2	59	31
9	8	4	12	121	4	3	4	72	16
10*	10	7	11	221	10	4	14	216	22
Total	2	1	1	434	2	1	2	370	1000

* One instance reaches the time limit and is not present in the data.

Table 20 – Run times for the two formulations of Z_{TMKa} (without rotation) by the number of items considering AMB box set and $Q = 2$.

	Z_{TMKa1}				Z_{TMKa2}				
Items	Mean	Median	SD	Total	Mean	Median	SD	Total	Instances
1	<1	<1	<1	<1	<1	<1	<1	<1	251
2	<1	<1	<1	<1	<1	<1	<1	<1	380
3	<1	<1	<1	1	<1	<1	<1	1	155
4	<1	<1	<1	2	<1	<1	<1	2	71
5	<1	<1	<1	2	<1	<1	<1	1	18
6	<1	<1	<1	6	<1	<1	<1	5	25
7	<1	<1	<1	25	<1	<1	<1	18	31
8	5	2	9	163	6	1	13	182	31
9	19	6	30	302	28	3	67	451	16
10*	113	10	285	2374	59	7	126	1237	22
Total	14	2	3	2877	9	1	21	1898	1000

* One instance reaches the time limit and is not present in the data.

Table 21 – Time in seconds for the instance that reached the time limit.

	Formulation	
Model	1	2
Z_{CLSwr}	409 s	1,570 s
Z_{TMKa}	1,658 s	1,800 s

Phase 2 - Analysis of the effects of rotation

In this second phase of computational experiments, the original Chen, Lee and Shen (1995) model with items rotation allowed (Z_{CLS}) is compared with the adapted model without rotation (*CLSwr*). The aim is to observe how much space could be saved compared to the increased model complexity of the rotation and, thus, higher computational times. All instances were solved by both models for only one box ($Q = 1$) and up to two boxes ($Q = 2$) for the AMB box set.

Analysing waste reduction for rotation

The residual waste space was used to analyze the performance approaches with and without rotation. For this comparison all instances are considered together, $V_{RES} = 100\,\frac{\sum V_\ell - \sum v_i}{\sum V_\ell}$, where V_ℓ is the volume of the boxes selected to pack all instances and v_i is the volume of each item i of all instances. This measure reflects the percentage of wasted space considering the packing of all instances together. For example, if this number is reduced, the total volume transported has also been reduced.

Rotation is expected to better allocate the items in the boxes, thus reducing the residual volume. This is shown to be true in these experiments. Table 22 shows the total residual volume percentage (V_{RES}), where the first column represents the number of items of instances, followed by the residual volume percentage without (w/o Rotation) and with (w. Rotation) rotation and the difference between them (Diff.), for $Q = 1$ and $Q = 2$, the last column represents the number of instances and the last line the total for all instances. It is possible to see that there is an average reduction of about 16% for both Qs when there is rotation. This shows that rotation can have a significant impact on residual volume reduction.

Table 22 – Total residual volume percentage (V_{RES}) for all instances with (w.) and without (w/o) rotation for only one box (Q=1) and up to two boxes (Q=2) for AMB box set.

	Q=1			Q=2			
Items	w/o Rotation	w. Rotation	Diff.	w/o Rotation	w. Rotation	Diff.	Instances
1	69.0%	69.0%	0.0%	69.0%	69.0%	0.0%	251
2	59.0%	37.6%	-21.5%	58.1%	36.8%	-21.2%	380
3	52.7%	29.5%	-23.3%	50.0%	26.7%	-23.3%	155
4	47.8%	27.1%	-20.7%	42.8%	22.7%	-20.1%	71
5	44.7%	25.0%	-19.6%	41.1%	18.5%	-22.6%	18
6	42.0%	23.4%	-18.6%	35.6%	19.0%	-16.6%	25
7	41.7%	19.3%	-22.4%	34.0%	15.1%	-18.9%	31
8	38.4%	18.5%	-20.0%	33.4%	14.6%	-18.8%	31
9	37.6%	16.3%	-21.2%	29.8%	13.7%	-16.1%	16
10	34.9%	15.7%	-19.2%	28.0%	13.0%	-15.0%	22
Total	52.4%	35.6%	-16.8%	49.7%	33.8%	-15.9%	1000

Analysing increased run times for rotation

However, this reduction is accompanied by an increase in computational times, as can be seen by comparing the times without rotation in Table 23 and with rotation Table 24. Both tables show time statistics with mean, median, standard deviation (SD) and total by items with the total sum on the last line.

For a single box ($Q = 1$), the results indicate a significant increase in runtimes, with times going from 73 seconds to 11,655 seconds. The impact is even more pronounced when considering two boxes ($Q = 2$), where runtimes increase from 370 seconds to 67,398 seconds. This outcome highlights the time impact resulting from the added complexity of rotation.

Analyzing the data by the number of items, it can be seen that the time increase starts to reach higher values (mean, median and total) in instances with 6 items. This increase shows a high variation (standard deviation) and is accompanied by 21 ($Q = 2$ only) instances with 8 or more items reaching the time limit. In contrast, instances without rotation did not reach the time limit (see Table 26).

The results indicate the potential of using rotation to reduce the residual volume. Nevertheless, it is important to consider the number of items as instances with six or more items lead to considerably longer computational times due to the increased model complexity caused by rotation in combination with the number of items.

Table 23 – Run times for the model Z_{CLSwr} (without rotation) by number of items considering AMB box set, $Q = 1$ and $Q = 2$.

	Q=1				Q=2				
Items	Mean	Median	SD	Total	Mean	Median	SD	Total	Instances
1	<1	1	<1	2	0	0	0	0	251
2	<1	<1	<1	5	0	0	0	1	380
3	<1	<1	<1	3	0	0	0	1	155
4	<1	<1	<1	2	0	0	0	1	71
5	<1	<1	<1	1	0	0	0	1	18
6	<1	<1	<1	3	0	0	0	4	25
7	<1	<1	<1	5	0	0	0	15	31
8	<1	<1	<1	9	2	1	2	59	31
9	1	<1	2	18	5	3	4	72	16
10	1	1	1	26	10	4	14	216	22
Total	1	1	1	73	2	0	5	370	1,000

Table 24 – Run times for model Z_{CLS} (with rotation) by number of items considering AMB box set, $Q = 1$ and $Q = 2$.

	Q=1				Q=2				
Items	Mean	Median	SD	Total	Mean	Median	SD	Total	Instances
1	<1	<1	<1	<1	<1	<1	<1	<1	251
2	<1	<1	<1	16	<1	<1	<1	1	380
3	<1	<1	<1	15	<1	<1	<1	5	155
4	<1	<1	<1	14	<1	<1	<1	18	71
5	<1	<1	<1	9	2	1	1	28	18
6	1	1	1	33	10	4	12	259	25
7	4	2	5	112	68	25	130	2,104	31
8	22	5	66	669	522	227	608	16,175	31
9	111	22	176	1,776	849	630	702	13,586	16
10	410	95	574	9,011	1,601	1,800	412	35,221	22
Total	12	<1	106	11,655	305	3	279	67,398	1,000

Phase 3 - Box Set Comparison

The final part of this section compares three different box sets to evaluate the box generation strategies explained in Subsection 3.3.2. The box sets (AMB, HB, and HBnd) are used to solve all instances with Z_{CLS} model, for only one box ($Q = 1$) and up to two boxes ($Q = 2$). This experiment assesses whether the box generation strategy reduces the residual volume and determines the implications of allowing for packing each order in one or two boxes.

Analysing the run times for box sets

The first test evaluates the effect of various box sets on computational time. The results are presented in Table 25, and in detail in Table 27 ($Q = 1$) and Table 28 ($Q = 2$). Each table summarizes computational times for each box set, including mean, median, third quartile (Q3), standard deviation (SD), minimum, maximum and total times. The first table is grouped by box set, while the second and third are detailed by item quantity, with the last column representing the number of instances.

From Table 25, it can be seen that the runtime was affected by the box set used. This is evident from the results of $Q = 1$, which shows that the HBnd box set has lower computational times than the other box sets, and from $Q = 2$ where the HB box set has the lowest times. Additionally, Table 26 displays the number of instances that have reached the time limit per quantity of items. The table shows that the HB box set with $Q = 1$ had 7 instances that reached the time limit. Meanwhile, no instances from the other box sets reached the time limit for $Q = 1$, while around several instances did so for $Q = 2$. Furthermore, it can be seen that the longer runtimes are for instances with more than 6 elements Table 27 and Table 28 due to the complexity of the problem.

It is important to highlight that the mean and median for all box sets are generally low, especially the median, which indicates that 50% of the instances have low runtimes. This is due to two reasons. First, there are more instances with fewer items, as explained in Subsection 3.3.1. Second, for instances with more items, the standard deviation tends to be higher, indicating a large variation in runtimes between instances. For example, for $Q = 1$, the fastest 10 item instances run in 1 to 5 seconds (Table 27) while the slowest 497 to 1,800 (time limit), and similar results for $Q = 2$.

Table 25 – Summarized run times in seconds of all instances for Z_{CLS} (with rotation) for all different box sets (AMB, HB and HBnd), only one box (Q=1) and up to two boxes (Q=2).

Box Set	Q	Mean	Median	Q3	SD	Min	Max	Total
AMB	1	12	<1	<1	106	<1	1800	11,655
AMB	2	67	<1	<1	308	<1	1800	67,398
HB	1	21	<1	<1	164	<1	1800	21,398
HB	2	55	<1	<1	273	<1	1800	55,153
HBnd	1	6	<1	<1	57	<1	1420	6,190
HBnd	2	70	<1	<1	314	<1	1800	69,567

Table 26 – Number of instances that reached the time limit (1,800 seconds) for Z_{CLS} (with rotation) for all different box sets (AMB, HB and HBnd), only one box ($Q = 1$) and up to two boxes ($Q = 2$) by item quantity.

	Q=1			Q=2		
Items	AMB	HB	HBnd	AMB	HB	HBnd
8	0	0	0	2	1	4
9	0	0	0	3	5	7
10	2	7	0	16	13	17
Total	0	7	0	21	19	18

Table 27 – Summary statistics of run times in seconds for all instances in all three box sets by item quantity for up to two boxes (Q=1).

Box Set	Items	Mean	Median	Q3	SD	Min	Max	Total	Instances
AMB	1	<1	<1	<1	<1	<1	<1	<1	251
	2	<1	<1	<1	<1	<1	<1	16	380
	3	<1	<1	<1	<1	<1	<1	15	155
	4	<1	<1	<1	<1	<1	<1	14	71
	5	<1	<1	1	<1	<1	1	9	18
	6	1	1	2	1	<1	3	33	25
	7	4	2	3	5	<1	22	112	31
	8	22	5	15	66	1	370	669	31
	9	111	22	102	176	3	530	1,776	16
	10	410	95	576	574	5	1,800	9,011	22
HB	1	<1	<1	<1	<1	<1	<1	<1	251
	2	<1	<1	<1	<1	<1	<1	14	380
	3	<1	<1	<1	<1	<1	<1	14	155
	4	<1	<1	<1	<1	<1	1	15	71
	5	1	<1	1	1	<1	2	11	18
	6	1	1	2	1	<1	4	30	25
	7	17	3	5	62	1	343	524	31
	8	32	9	28	53	1	249	981	31
	9	153	21	145	276	4	856	2443	16
	10	789	480	1,800	748	4	1,800	17,365	22
HBnd	1	<1	<1	<1	<1	<1	<1	<1	251
	2	<1	<1	<1	<1	<1	<1	10	380
	3	<1	<1	<1	<1	<1	<1	11	155
	4	<1	<1	<1	<1	<1	1	12	71
	5	1	<1	1	<1	<1	1	10	18
	6	2	1	2	3	<1	17	51	25
	7	7	2	8	13	<1	71	207	31
	8	42	5	31	101	<1	478	1,299	31
	9	144	16	73	357	1	1,420	2,297	16
	10	104	30	135	135	1	497	2,293	22

Table 28 – Summary statistics of run times in seconds for all instances in all three box sets by item quantity for up to two boxes (Q=2).

Box Set	Items	Mean	Median	Q3	SD	Min	Max	Total	Instances
AMB	1	<1	<1	<1	<1	<1	<1	<1	251
	2	<1	<1	<1	<1	<1	<1	1	380
	3	<1	<1	<1	<1	<1	<1	5	155
	4	<1	<1	<1	<1	<1	1	18	71
	5	1	1	2	1	<1	4	28	18
	6	10	4	12	12	<1	42	259	25
	7	68	25	58	129	5	598	2,104	31
	8	522	227	938	608	5	1,800	16,175	31
	9	849	630	1,470	702	24	1,800	13,586	16
	10	1,601	1,800	1,800	412	250	1,800	35,221	22
HB	1	<1	<1	<1	<1	<1	<1	<1	251
	2	<1	<1	<1	<1	<1	<1	1	380
	3	<1	<1	<1	<1	<1	<1	4	155
	4	<1	<1	<1	<1	<1	1	13	71
	5	2	<1	2	5	<1	21	38	18
	6	8	3	12	9	<1	30	194	25
	7	71	22	52	169	2	922	2,197	31
	8	357	191	411	463	5	1,800	11,076	31
	9	727	337	1,800	759	21	1,800	11,634	16
	10	1,363	1,800	1,800	617	163	1,800	29,994	22
HBnd	1	<1	<1	<1	<1	<1	<1	<1	251
	2	<1	<1	<1	<1	<1	<1	2	380
	3	<1	<1	<1	<1	<1	<1	10	155
	4	<1	<1	<1	<1	<1	1	25	71
	5	4	2	3	7	<1	22	73	18
	6	14	7	16	17	<1	77	347	25
	7	106	58	91	208	2	1,150	3,298	31
	8	498	207	649	585	25	1,800	15,429	31
	9	994	779	1,800	763	31	1,800	15,907	16
	10	1,567	1,800	1,800	481	331	1,800	34,475	22

Analysing waste reduction for the box sets

To explore residual volume reduction for each instance, the residual wasted space is calculated as $V_{res_k} = \frac{\sum V_\ell - \sum v_i}{\sum V_\ell} 100$, where in this case V_ℓ is the volume of the selected box(es) to pack the items of instance k and v_i is the volume of its items. This second measure allows to observe the variation among the instances. For example, the mean and the standard deviation allow identifying where the box set has a high or low residual volume variation.

Table 29 contains the residual volume for all box sets and for $Q = 1$ and $Q = 2$, where columns 3 to 7 show the summary statistics for V_{res_k} with mean, median, standard deviation (SD), minimum, maximum and the last column (Total Per.) the total residual volume (V_{RES}). Table 30 shows the total residual volume (V_{RES}) by item for $Q = 1$ and $Q = 2$ for all box sets by item quantity, with last column representing number of instances. Last, Table 31 shows the V_{res_k} with the mean, median and standard deviation (SD) for all box sets and $Q = 1$ and $Q = 2$ by item quantity and box set.

When comparing box sets based on residual volume, HBnd exhibits the best performance, which is evident in both scenarios ($Q = 1$ and $Q = 2$), as shown in Table 29. This is because the means and total percentages for HBnd are lower, proving that this box generation strategy can improve packing by reducing the total box volume. This can also be observed with more detail in Table 30 and Table 31, which provide additional information by item. The first table shows that HBnd generally has a lower or equal total residual volume (V_{RES}) performance compared to the other box sets for each item quantity. The second table (V_{res_k}) presents the same pattern. However, it is now possible to see that, in most cases, the standard deviation is lower for HBnd, indicating that it achieved better packing in more cases.

Table 29 – Summary of residual volumes for all box sets.

				V_{res_k}			V_{RES}
Box Set	Q	Mean	Median	SD	Min	Max	Total Per.
AMB	1	40.2%	33.7%	21.5%	9.4%	92.5%	35.6%
AMB	2	38.9%	32.1%	22.4%	7.9%	92.5%	33.8%
HB	1	41.0%	36.8%	20.6%	7.3%	92.8%	34.2%
HB	2	39.8%	34.6%	21.0%	6.8%	92.8%	34.2%
HBnd	1	38.3%	33.2%	19.1%	4.7%	92.2%	34.0%
HBnd	2	36.7%	31.1%	20.0%	4.7%	92.2%	32.0%

An exception is found for the HB box set with 6 or more items when $Q = 1$. It can be observed that this box set has progressive lower total residual volumes compared to the other two, see Table 30. The reason for this can be explained by the volume distribution of boxes in each set illustrated in Figure 9c. HB has more large boxes than the other two sets, resulting in better packing efficiency due to having more options. However, this phenomenon is no longer observed in the $Q = 2$ scenario, because larger instances can be split into two boxes. This result suggests that HB and HBnd box sets could be further explored.

Table 30 – Total residual volumes (V_{RES}) for all box sets by item quantity.

	AMB		HB		HBnd		
Items	Q=1	Q=2	Q=1	Q=2	Q=1	Q=2	Instances
1	69.0%	69.0%	67.8%	67.8%	65.6%	65.6%	251
2	37.6%	36.8%	39.5%	38.8%	36.0%	35.5%	380
3	29.5%	26.7%	31.8%	29.4%	30.7%	27.4%	155
4	27.1%	22.7%	26.1%	23.4%	25.6%	21.3%	71
5	25.0%	18.5%	22.1%	20.6%	21.8%	18.9%	18
6	23.4%	19.0%	19.7%	18.2%	19.7%	16.1%	25
7	19.0%	15.1%	17.0%	15.1%	20.1%	15.3%	31
8	18.5%	14.6%	15.9%	14.2%	18.8%	13.8%	31
9	16.7%	13.8%	13.9%	12.3%	20.1%	13.2%	16
10	15.7%	13.0%	11.5%	11.3%	14.8%	12.5%	22
Total	35.6%	33.8%	35.3%	34.2%	34.0%	32.0%	1000

Comparing $Q = 1$ with $Q = 2$ in waste reduction

Finally, the efficiency of allowing the model to choose up to two boxes ($Q = 2$) versus allowing only one box for packing ($Q = 1$) is compared in terms of residual volume reduction (Tables 29 to 31). It can be observed that the reduction remains constant, around 2%, see Table 29, for all box sets. However, the reduction increases for larger item quantities. In particular, this reduction ranges from 0.2% (for 2 items) up to 3% (for 10 items), see Table 30. These results indicate that some cases benefit from this strategy. These results are further supported by Table 32. The table displays the number of instances packed into two boxes, revealing that about 20% of the instances benefit from this kind of packing. It is important to note, however, that the runtimes for this strategy ($Q = 2$) are significantly higher than for a single box ($Q = 1$), see Table 25, especially for instances with 6 or more items, see Tables 27 and 28.

In summary, the box generation strategy for the HBnd box set has proven to be effective in reducing residual volume. Additionally, computational times can vary among different box sets. Moreover, using the strategy of packing instances in up to two boxes further reduces residual volume at the cost of increased runtime, at least for larger instances.

Table 31 – Summary of residual volumes (V_{res_k}) for all box sets by item quantity.

		Q=1			Q=2		
Box Set	Items	Mean	Median	SD	Mean	Median	SD
AMB	1	66.0%	74.6%	21.4%	66.0%	74.6%	21.4%
	2	37.8%	35.2%	13.9%	37.3%	34.0%	14.1%
	3	29.2%	29.3%	7.3%	27.1%	26.9%	7.5%
	4	27.1%	25.9%	6.3%	23.5%	22.7%	5.6%
	5	25.1%	24.1%	4.8%	19.1%	19.3%	4.0%
	6	23.1%	23.1%	4.2%	18.9%	19.0%	2.8%
	7	18.6%	18.6%	4.1%	15.1%	15.5%	3.1%
	8	18.3%	18.2%	3.8%	14.8%	15.7%	2.5%
	9	16.7%	16.3%	3.0%	14.0%	14.9%	2.3%
	10	15.4%	15.1%	3.3%	12.9%	13.1%	1.9%
HB	1	64.9%	69.5%	16.2%	64.9%	69.5%	16.2%
	2	40.1%	37.9%	15.3%	39.2%	36.5%	15.0%
	3	32.7%	32.5%	8.8%	30.1%	29.7%	7.9%
	4	27.3%	26.6%	8.0%	24.6%	25.4%	6.6%
	5	22.6%	22.5%	4.4%	21.0%	20.6%	4.1%
	6	20.0%	19.5%	4.9%	18.4%	18.8%	3.9%
	7	17.3%	17.0%	3.9%	15.3%	15.2%	2.8%
	8	15.9%	15.7%	2.1%	14.3%	14.3%	2.2%
	9	14.2%	13.8%	3.1%	12.4%	13.6%	2.5%
	10	11.4%	11.6%	2.6%	11.0%	10.9%	2.5%
HBnd	1	60.8%	62.5%	18.2%	60.8%	62.5%	18.2%
	2	35.8%	35.2%	12.7%	35.2%	34.5%	12.8%
	3	30.9%	30.8%	7.8%	27.5%	27.6%	7.3%
	4	26.3%	25.4%	7.3%	22.1%	22.0%	6.5%
	5	21.8%	21.7%	3.9%	19.2%	19.8%	4.0%
	6	19.8%	19.4%	5.5%	16.5%	16.6%	3.8%
	7	19.5%	18.9%	6.1%	15.2%	14.4%	2.9%
	8	18.5%	18.4%	4.2%	13.8%	13.9%	2.0%
	9	19.5%	20.8%	4.9%	13.3%	13.6%	1.6%
	10	14.5%	14.6%	4.1%	12.2%	12.4%	2.2%

Table 32 – Quantity of instances that the model packed in one or two boxes for each box set when executed with $Q = 2$.

	Used Boxes	
Box Set	One	Two
AMB	768	232
HB	805	195
HBnd	750	250

3.4 Conclusions

This chapter first presents and discusses alternative formulations for 3D *Residual Bin Packing Problems*. Then the generation of box sets and item instances is described. Next, two models and an alternative formulation for each are compared to obtain a better dual bound. Then, it is investigated how rotation can improve volume reduction at the expense of computational time. Lastly, a comparison is conducted between each box set and the use of two boxes to determine whether these strategies reduce residual volume.

The initial part of the computational experiments showed that the alternative formulation (Z_{CLS2wr} and Z_{TMKa2}) improved the dual bound. However, the gain obtained did not result in better computational times. This outcome is likely due to the limited number of items in the instances tested. Therefore, even though a better dual bound was achieved, the difference was minor. Considering this situation, since this research does not cover instances with more than 10 items, exploring a greater number of items would be interesting for further research to determine if the alternative formulations deliver better outcomes in these cases.

The comparison of models Z_{CLSwr} and Z_{TMKa} showed that the former had lower run times, indicating that it is a better formulation. As mentioned in Subsection 3.3.5, the main difference lies in the method used to allocate the boxes. This feature made the formulation Z_{TMKa} dependent on a larger BigM, resulting in poorer performance. Therefore, the Chen, Lee and Shen (1995) model with the alternative constraint was used for the remaining work.

With that established, a study on rotation was performed, showing a significant reduction in residual volume (20%). However, instances with larger item sets (6 or more) have a significant increase in computational time. Therefore, this strategy is valid, but the benefits of longer runtimes must be considered.

Finally, the box sets and box allocation strategies ($Q = 1$ and $Q = 2$) were tested. The results show that allowing the model to pack up to two boxes ($Q = 2$) can improve packing efficiency by about 2%. However, the gains are small, and the computational time increases significantly. Therefore, this strategy is useful as long as time is not a concern. Regarding the box generation strategy, the HBnd box set showed superior overall performance. This suggests that the box generation strategy discussed in Subsection 3.3.2 is valid. Additionally, the HB box set demonstrated better performance in instances with 6 or more items. This result implies that the strategy utilized to generate both box sets is efficient and warrants further investigation.

In summary, this study has demonstrated the effectiveness of the methods used here to help companies determine an appropriate set of boxes to reduce the residual volume of their packaging. The selection of box dimensions using the Herz grid that while ensuring a similar volume distribution of orders proved to be effective. Additionally, packing one order in up to two boxes further reduced the residual volume.

A critical aspect is whether implementing these strategies has business or practical limitations. For instance, several companies do not allow rotation on one or more axes, which could result in poorer performance. In contrast, this study allows rotation on all axes, thus exploring all possibilities. Another factor to consider is that packing items in two boxes may result in logistical and practical complications, leading to cost escalation or impracticability. Further studies could explore these practical limitations in detail.

Another important aspect in determining the set of boxes is the cost of manufacturing the boxes, which has not been considered in this work. Typically, these expenses are linked to the number of boxes of the same type and the amount of material required for production. Investigating the same strategies while considering the cost would be an interesting aspect to approach.

As previously stated, the HBnd box set exhibited the best overall performance, with HB being better in instances with more items. As discussed in Subsection 3.3.5, this may be attributed to the smaller volume distribution of the first set as opposed to the second set, which allows the first set to pack smaller instances better and the second set to pack larger instances better. Since both sets were generated by random sampling, the question arises as to whether there are ways to improve the performance of the selected boxes further. To achieve this objective, it is essential to investigate the strategy used to generate the boxes. Thus, Chapter 4 aims to improve this strategy by utilizing a heuristic approach with this objective in focus.

CHAPTER 4
BOX GENERATION PROBLEM

In Chapter 3, it was discussed how to improve item packing using a given set of boxes. The proposed approach is efficient and indicates two new research challenges: i) define the number of box types available (set of boxes); and ii) alter the dimensions of each box, focusing on reducing total residual volume.

The first issue can be addressed by considering the solution of all orders together. However, this results in a much more complex mathematical model, consequently harder to solve. Similarly, the second issue may include new box types to the existing set. Nevertheless, this approach becomes unreasonable due to the extensive array of potential box variations.

As outlined in Section 2.3, other authors have addressed these issues. For instance, Fontaine and Minner (2022) aim to define a smaller set of boxes from a larger set capable of packing all orders with minimal residual volume. While this approach addresses the challenge of limiting box types, it does not generate new boxes. Alonso *et al.* (2016) solve a 2D cutting stock packing model using a set of packed orders to define and restrict the number of used boxes. In their approach, boxes are generated in prepacked patterns that are improved.

In this chapter, a new method is introduced to address the difficulties associated with limiting the number of box types and the strategy for generating new boxes. This approach aims to consolidate these challenges into a unified problem.

In Section 4.1, the problem is defined. Following this, Section 4.2 describes the solution approach, followed by two constructive heuristics in Subsection 4.2.2. Then, Section 4.3 describes a local search with two different neighbourhoods. Computational experiments for the heuristics and local search are described in Section 4.4. Last, a summary of the results is made in Section 4.5.

4.1 Problem Definition - Box Generation

The definition of the box set has two main objectives: i) reduce the residual space to pack the orders, and ii) fix the number of boxes in the company stock. Two extreme solutions exist: i) use one ideal box to pack each order, resulting in the smallest residual space, and ii) use just one box where all orders can be packed, i.e., a box big enough to pack every order. In Figure 10, four orders are packed according to these two extreme solutions with Figure 10a as the first example and Figure 10b the second. In Figure 10c, it is presented a intermediary solution.

The first solution is optimal regarding residual volume, but it is unfeasible for companies that have to deal with thousands of orders every day. On the other hand, the second solution results in a high residual volume, which leads to high transport costs. The problem studied consists of defining a set of boxes that allows a balance between these two main objectives.

The objective is to minimize the sum of all packed orders ($i = 1, ..., o$) residual volumes ($V_{res\ i}$) (4.1).

$$V_{RES} = \min \sum_{i=1}^{o} V_{res\ i} \tag{4.1}$$

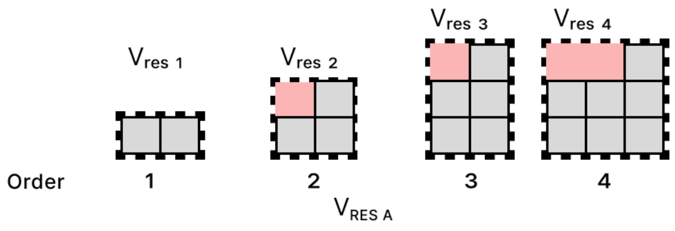
4.2 Solution Approach

The solution method developed started with a suggested number of boxes. Based on the number of boxes, a box set is generated considering a set of representative orders. These boxes undergo slight dimension adjustments to modify the overall residual volume. The box dimension can be increased or reduced, as depicted in Figures 11 and 12, respectively.

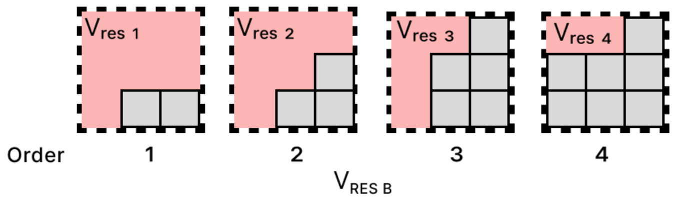
In these figures, each scenario illustrates four orders, with some items (represented by grey rectangles) packed within a box (indicated by the dashed line). In both scenarios, the residual volume is outlined in red prior to the alteration of the box and in green post-adjustment. In Figure 11a, the box containing order 1 is increased to reduce the total residual volume ($V_{RES\ 2} \leq V_{RES\ 1}$), then it packs orders 1, 2 and 3, as shown in Figure 11b. Correspondingly, the second strategy involves reducing the box dimensions, as illustrated in Figure 12. Here, orders 1, 2, and 3 are packed in one box, while order 4 is placed in a larger second box (Figure 12a). The dimension of the first box is reduced. Orders 1 and 2 are packed in the new box, and orders 3 and 4 are packed in the second box, Figure 12a. This new solution results in a smaller total residual volume ($V_{RES\ 4} \leq V_{RES\ 3}$).

In summary, the solution method is based on two steps: i) defining an initial box set and ii) improving this set. In the next section, the definition of box sets is detailed. In sequence, two constructive heuristics are proposed to define the initial box set. Finally, a local search heuristic is developed to improve the box set and, consequently, the solution of the problem.

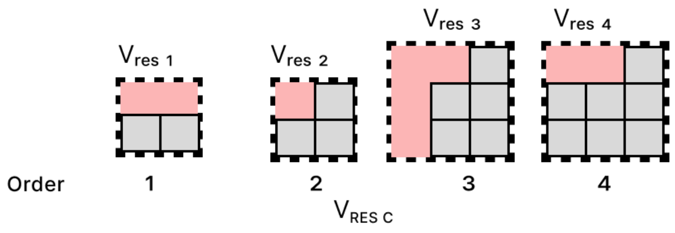
Figure 10 – An example with three distinct approaches to packing four orders, with boxes denoted by the dashed lines and items represented as grey squares. The red areas represent the residual volume. The residual volumes are ordered as follows: $V_{RES\ A} \geq V_{RES\ C} \geq V_{RES\ B}$.



(a) Four orders packed in four different boxes, minimizing residual volume.

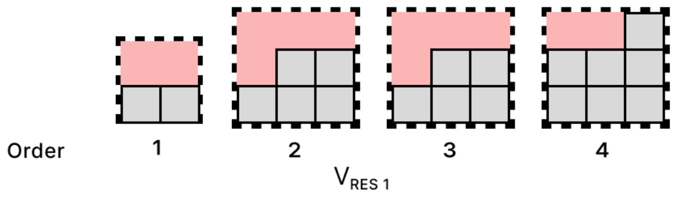


(b) Four orders packed in the same box limited by largest order, maximal residual volume.

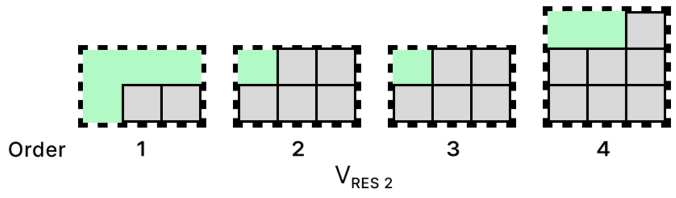


(c) Two smaller orders packed in one box and two larger orders on another, showing a mid term scenario.

Figure 11 – This figure depicts decreasing total residual volume by increasing a box dimension. Dashed lines represent the boxes, and the items are shown as grey squares. The red area ($V_{RES\ 1}$) shows the initial residual volume before the change, while the green area ($V_{RES\ 2}$) illustrates the residual volume after the adjustment.

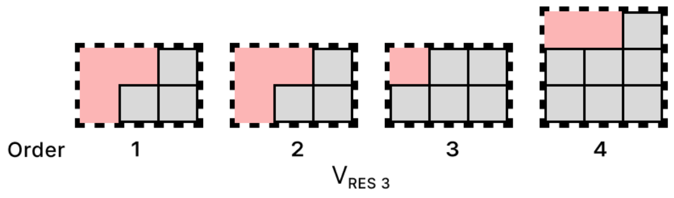


(a) Order 1 packed in one box and orders 2, 3 and 4 in a larger one.

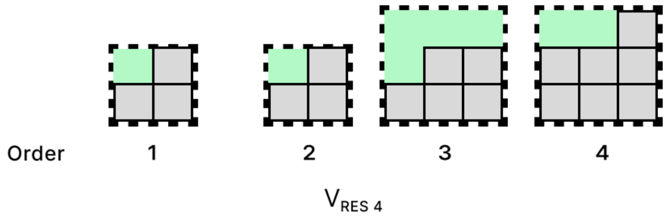


(b) Box packing order 1 dimension is increased, now also packing orders 2 and 3.

Figure 12 – This figure depicts decreasing total residual volume by reducing a box dimension. Dashed lines represent the boxes, and the items are shown as grey squares. The red area ($V_{RES\ 3}$) shows the initial residual volume before the change, while the green area ($V_{RES\ 4}$) illustrates the residual volume after the adjustment.



(a) Order 1, 2 and 3 packed in one box, and order 4 in a larger box.



(b) Orders 1 and 2 are packed in a smaller box, and order 3 is now packed in the same box as order 4.

4.2.1 Defining HB Set

Before discussing the constructive and local search heuristics, it is important to define the initial box set, which encompasses the boxes subject to exploration.

In this set, each box is defined by coordinates (x,y,z) representing its length, width, and height, defined as HB. Consequently, $HB = \{(x_0,y_0,z_0),\dots,(x_n,y_n,z_n)\}$, where n is the number of boxes in the set and without loss of generality $(x_i \geq y_i \geq z_i)$.

The dimensions of boxes are defined based on the combination of three vectors corresponding to each dimension of a box. These vectors are defined using the Herz grid (as described in Subsection 3.3.2). For example, based the following sets X, Y and Z:

$$X = \{1,6,9,14\}$$
$$Y = \{5,7,8,12\}$$
$$Z = \{1,3,7,15\}$$

it is obtained:

$$HB = \{(6,5,1),(6,5,3),\dots(14,12,7)\}$$

4.2.2 Constructive Heuristics

Two constructive heuristics were developed, each one with different objectives. The first one focuses on the quality of the solution, and the second heuristic was designed to be faster. Both heuristics finish with a feasible solution.

Constructive Heuristic - H1

This heuristic starts by grouping orders into partitions based on their volumes then, for each partition, determines the first box that accommodates the largest volume order. Once a box from an initial set of boxes is chosen, the smaller orders are packed into it. If any order remains unpacked, the box size is increased, and the process is repeated until all orders of the partition are accommodated. A detailed description of the heuristics is present by Algorithm 1. The algorithm inputs are: the set of orders to pack, the set of boxes available and the desired number of boxes. The solution is the set of boxes selected to pack the orders.

First, the set of orders ($\mathcal{O} = \{1,...,n\}$) is sorted in non-decreasing order based on their volume (line 2), which is the sum of the volumes of items composing each order. Subsequently, all orders are divided into partitions ($P = 1,...,m$). Each partition corresponds to a set of orders that will be packed into the same box. In this classification, each partition is associated with an integer, where the smaller integers are assigned to the smallest volume boxes. Consequently, if a box's volume increases beyond the volume of the next partition, the partitions are reclassified to maintain this order.

For each partition, the algorithm searches for the first box that can accommodate the last order (the larger volume), see Lines 6 to 32. The biggest order of the partition (cO) is selected to help in the definition of a box able to pack all orders of this partition. In Lines 14 to 21, the box able to pack cO order is defined. In Lines 22 to 29, it is verified if this box can be used to pack all the others in this partition. If not, a bigger box that is able to pack is chosen. This box is saved as part of the solution (Line 30), and the next partition is analyzed. Finally, all orders are packed using the generated boxes (line 33). This step is crucial for reallocating orders that might benefit from smaller boxes, initially assigned to partitions with larger boxes. The result is a list ($solution$) containing the box dimensions for each partition.

In Algorithm 1, the function $boxFilter$ (Algorithm 2) identifies the index of the first box meeting two criteria: having at least one dimension larger than the largest item dimension of order $\mathscr{O}[cO]$, and the box volume being equal to or greater than the order volume. This filtering step helps reduce the search time by excluding obviously infeasible boxes.

After filtering, the eligible boxes undergo testing (lines 16 - 21). Each order is packed using the packing model discussed in Subsection 3.2.1, utilizing the function *packingModel* (Algorithm 3). This function takes an order and a box or a set of boxes as input, and it packs the order in the box that minimizes the residual volume. If a feasible solution is found, the function returns the dimensions of the solution box. Otherwise, it returns an empty set, indicating that no feasible packing was achieved.

Algorithm 1 – Constructive Heuristic - H1

```
 1: Inputs:
 2: $\mathscr{O}$ as the set of orders
 3: $HB$ as the set of $(x,y,z)$ Herz box coordinates
 4: $m$ as number of partitions
 5: function ConstructiveHeuristicH1($\mathscr{O}$, $HB$, $m$)
 6:     Sort the orders of set $\mathscr{O}$ in non-decreasing order of volume
 7:     $opp \leftarrow \lfloor \frac{length(\mathscr{O})}{m} \rfloor$ // define the number of order per partition
 8:     $solution \leftarrow$ empty list // storage of solution box for each partition
 9:     $rO \leftarrow 1$
10:     // Finds the first largest box that fits all boxes
11:     for $p \leftarrow 1$ to $m$ do
12:         // Define the index of explored orders for each partition
13:         if $p = m$ then
14:             $cO \leftarrow n$ // last order of partition p
15:             $pO \leftarrow n - 1$ // second last order of partition p
16:         else
17:             $cO \leftarrow p \cdot opp$ // last order of partition p
18:             $pO \leftarrow p \cdot opp - 1$ // second last order of partition p
19:         end if
20:         $b \leftarrow$ boxFilter($\mathscr{O}[cO], HB$)
21:         $box \leftarrow \emptyset$
22:         // Find the first box that fits the last partition's order
23:         while $box = \emptyset$ do
24:             $box \leftarrow$ packingModel($\mathscr{O}[cO], HB[b]$)
25:             if $box = \emptyset$ then
26:                 $b \leftarrow b + 1$
27:             end if
28:         end while
29:         // Test the defined box for all remaining orders in the partition
30:         while $rO \leq pO$ do
31:             $box \leftarrow$ packingModel($\mathscr{O}[rO], HB[b]$)
32:             // If order is not packed, move to the next box ,
33:             // otherwise move to next order
34:             if $box = \emptyset$ then
35:                 $b \leftarrow b + 1$
36:             else
37:                 $rO \leftarrow rO + 1$
38:             end if
39:         end while
40:         $solution[p] \leftarrow HB[b]$ // save chosen box b to partition p
41:         $rO \leftarrow rO + 2$
42:     end for
43:     Run all orders $\mathscr{O}$ with $solution$ boxes to reassign orders's partitions
44:     return $solution$
45: end function
```

Algorithm 2 – Box filter.

```
1: Inputs:
2: order as the items dimensions of one order
3: HB as the set of $(x, y, z)$ Herz box coordinates
4: function BOXFILTER(order,HB)
5:     $itL \leftarrow$ largest item dimension from order
6:     $boxIndex \leftarrow \emptyset$
7:     $i \leftarrow 1$
8:     // Finds the first box that:
9:     // Order volume is smaller than box volume
10:    // Largest item dimension is less than largest box dimension
11:    for $box \in HB$ do
12:        $boxL \leftarrow$ largest dimension form box
13:        $i \leftarrow i + 1$
14:        if $boxL \geq itL$ and $volume(box) \geq volume(order)$ then
15:            $boxIndex \leftarrow i$
16:            return boxIndex
17:        end if
18:    end for
19:    return boxIndex
20: end function
```

Algorithm 3 – Packing model function.

```
1: Inputs:
2: order as the items dimensions of one order
3: B as the set of $(x, y, z)$ box coordinates
4: function PACKINGMODEL(order,B)
5:     // With the order and the set B of box(xes), packing is solved with $Z_{CLS2}$
6:     if $Z_{CLS2}$ is feasible then
7:         $bestBox \leftarrow$ the solution box for B
8:     else
9:         $bestBox \leftarrow \emptyset$
10:    end if
11:    return bestBox
12: end function
```

Constructive Heuristic - H2

The Constructive Heuristic H2 aims to produce a feasible solution rapidly. The method involves randomly generating a specified number of boxes (m) from the Herz grid ($HBnd$), ensuring that each generated box can accommodate at least one order. The complete pseudocode is outlined in Algorithm 5.

This algorithm is structured into three phases, followed by a redistribution step. In the first phase, it generates m boxes from the HB set, where each box corresponds to one partition. Subsequently, it tries to pack all orders in one of these boxes (lines 2 - 5). If there exist any infeasible orders (not packed), in the second phase, the dimensions of the largest box (partition m) are increased until all orders become feasible (lines 6 - 19). Finally, boxes unused by any orders are removed. If any box is removed, the partition with the largest residual volume (V_{RES}, Algorithm 4) is identified (partition p), and the first feasible box smaller than box p is added to the set of boxes. For this, a subset of boxes that at least one dimension is smaller than the box of the partition p but greater than $p-1$ is obtained from HB as HB_{mid}. This process continues until the desired number of partitions is achieved (lines 20 - 40). All orders of the instance are repacked using the available generated boxes (line 32).

Lastly, all orders are packed using the solution boxes (line 41), as in the previous heuristic. The result is a vector ($solution$) containing the box dimensions for each partition.

Algorithm 4 – Calculate V_{RES} for each partition

```
1:  **Inputs:**
2:  $\mathcal{O}$ as the set of orders
3:  $m$ as number of partitions
4:  **function** RESIDUALVOL($\mathcal{O}$, $m$)
5:      // For each partition calculate the total residual volume ($V_{RES}[p]$)
6:      **for** $p \leftarrow 1$ to $m$ **do**
7:          $V_{RES}[p] \leftarrow 0$
8:          **for** $o \in \mathcal{O}_p$ **do**
9:              $V_{RES}[p] \leftarrow V_{RES}[p] + V_{res}[o]$
10:         **end for**
11:     **end for**
12:     **return** $V_{RES}$
13: **end function**
```

Algorithm 5 – Constructive Heuristic - H2

```
1: **Inputs:**
2: $\mathcal{O}$ as the set of orders
3: $HB$ as the set of $(x,y,z)$ Herz box coordinates
4: $m$ as number of partitions
5: function ConstructiveHeuristicH2($\mathcal{O}$, $HB$, $m$)
6:     Sort $\mathcal{O}$ in non-decreasing order of volume
7:     $solution \leftarrow$ Sample $m$ boxes from $HB$ based on $\mathcal{O}$ volume distribution
8:     Sort $solution$ in a non-decreasing order by box volumes
9:     Run $packingModel$ for all orders $\mathcal{O}$ with $solution$ boxes, assign each feasible order to
   the respective partition
10:    $\mathcal{O}_{infeasible} \leftarrow$ All infeasible orders from $\mathcal{O}$
11:    $HB_{upper}$ is the subset of boxes from $HB$,with all dimensions larger than $solution[m]$ box
12:    $b \leftarrow 1$
13:    $o \leftarrow 1$
14:    // For all infeasible orders find the first feasible box starting from the largest box
15:    while $o \leq |\mathcal{O}_{infeasible}|$ do
16:        $box \leftarrow$ packingModel($\mathcal{O}_{infeasible}[o], HB_{upper}[b]$)
17:        if $box = \emptyset$ then
18:            $b \leftarrow b+1$
19:            $solution[m] \leftarrow HB[b]$
20:        else
21:            $o \leftarrow o+1$
22:        end if
23:    end while
24:    Remove boxes from $solution$ that do not pack any order
25:    $s \leftarrow |solution|$
26:    $V_{RES} \leftarrow$residualVol($\mathcal{O}$,$s$)
27:    // Generate boxes on the highest residual volume partition, until there are m boxes
28:    while $s < m$ do
29:        $p \leftarrow$ partition with the highest $V_{RES}$
30:        $HB_{mid} \leftarrow \{box \in HB \mid solution[p-1] \leq box \leq solution[p]\}$
31:        $b \leftarrow 1$
32:        $Ot \leftarrow \mathcal{O}_{p-1} \cup \mathcal{O}_p$
33:        // Accept the first box that pack at least one order as a solution
34:        for $o \leftarrow 1$ to $|Ot|$ do
35:            $box \leftarrow$ packingModel($Ot[o], HB_{mid}[b]$)
36:            if $box \neq \emptyset$ then
37:                $solution[s+1] \leftarrow HB[b]$
38:                Sort $solution$ in a non-decreasing order by box volume
39:                $V_{RES} \leftarrow$residualVol($\mathcal{O}$,$s$)
40:                $s \leftarrow s+1$
41:                Break
42:            else
43:                $b \leftarrow b+1$
44:            end if
45:        end for
46:    end while
47:    Run all orders $\mathcal{O}$ with $solution$ boxes to reassign orders's partitions
48:    return $solution$
49: end function
```

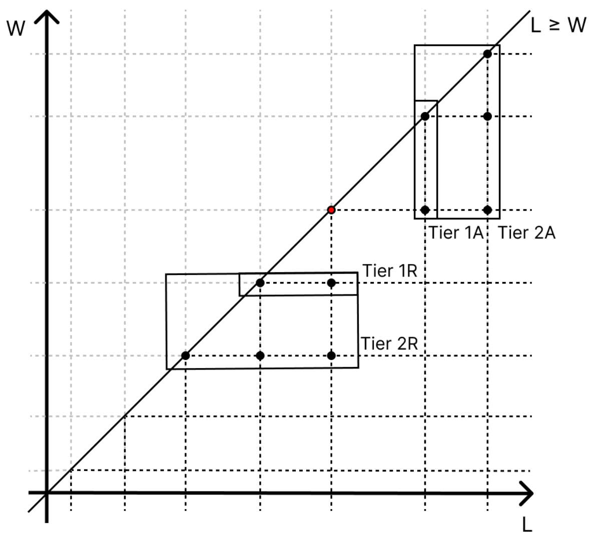
4.3 Local Search

A local search heuristic is proposed to refine the initial packing solution obtained by H1 or H2. The method uses two neighborhoods based on the Herz grid (Subsection 3.3.2) to generate new boxes. Note that the model described in Subsection 3.2.1 was used to determine the item packing.

4.3.1 Neighbourhood definition

The first neighbourhood, defined as *Dimension Increase (N1)*, concentrates on increasing the dimensions of the tested box. The second, known as *Dimension Reduction (N2)*, focuses on reducing the dimensions of the tested box. Both neighborhoods are applied to each partition one at a time.

Figure 13 – This figure depicts the neighborhood definition and limits in two dimensions. The red point shows the initial box that will be explored and the tiers range are defined by the black rectangles, with **A** the dimension increase and **R** reduction.



Each neighborhood employs the Herz grid to select a range of potential boxes (neighbors) of the tested partition box. These boxes are extracted from the set $HB = (x_o, y_o, z_o), \ldots, (x', y', z')$, with (x, y, z) beginning with the dimensions of the box in the tested partition (x_p, y_p, z_p) and extending up to t tiers for each dimension, see Figure 13. In the figure, it can be seen tiers that increase the box dimension (1A and 2A) and tiers that reduce the dimensions (1R and 2R). Each tier represents the number of elements a dimension may increase or decrease. This process results in a set of boxes, denoted as $HB_t = (x_p, y_p, z_p), \ldots, (x_{p+t}, y_{p+t}, z_{p+t})$, which becomes the neighborhood. Notice that $HB_1 \subseteq HB_2 \subseteq \ldots HB_t$.

To better understand how a tier is made, one must consider how the HB set is constructed, see Subsection 4.2.1. There, it is possible to see that this set is constructed from three vectors. A tier is characterized by the t elements obtained from these vectors associated with a specified box. With those elements, the HB set is narrowed down to a tier, which is denoted as HB_t.

For example, considering the three vectors HBx, HBy, HBz obtained using the Herz grid:

$$HBx = \{1,6,9,14\}, \qquad HBy = \{5,7,8,12\}, \qquad HBz = \{1,3,7,15\},$$

The HB set for these vector would be:

$$HB = \{(6,5,1),(6,5,3),\dots(14,12,7)\},$$

Exploring box $(6,5,3)$ with a tier $t = 1$ the vector will be:

$$HBx = \{6,9\}, \qquad HBy = \{5,7\}, \qquad HBz = \{3,7\},$$

$$HB_1 = \{(6,5,3),(9,5,3),(9,7,3),(9,7,7)\},$$

Exploring box $(6,5,3)$ with a tier $t = 2$ the vector will be:

$$HBx = \{6,9,14\}, \qquad HBy = \{5,7,8\}, \qquad HBz = \{3,7,15\},$$

$$HB_2 = \{(6,5,3),(9,5,3)\dots,(14,8,7)\}.$$

The strategy to obtain HB_t set described above is used by Algorithm 6 for dimension increase and Algorithm 10 for dimension reduction.

Algorithm 6 – Function to create a HB_t set from HB, with larger boxes.

```
1: **Inputs:**
2: $box$ as $(x,y,z)$ coordinates for current box
3: $HBx, HBy, HBz$ as vectors of Herz for $x,y$ and $z$ axis, respectively
4: $t$ as tier size
5: **function** CREATETIERUP($box$, $HBx$, $HBy$, $HBz$,$t$)
6:     $HBx_t \leftarrow t$ coordinates from $HBx$ that are larger than the $x$ coordinate from $box$
7:     $HBy_t \leftarrow t$ coordinates from $HBy$ that are larger than the $y$ coordinate from $box$
8:     $HBz_t \leftarrow t$ coordinates from $HBz$ that are larger than the $z$ coordinate from $box$
9:     $HB_t \leftarrow$ boxes combining all dimensions from $(HBx_t, HBy_t, HBz_t)$ respecting $(L \geq W \geq H)$
10:    **return** $HB_t$
11: **end function**
```

4.3.2 Local Search using Neighbourhood N1

This local search aims to improve the initial solution using the Dimension Increase Neighbourhood (N1). The objective is to identify partitions where the residual volume can be reduced via a trade-off, by increasing the residual volume of a given partition (p) it is expected to reduce the subsequent larger partition ($p+1$) residual volume via the packing of its smaller orders, as illustrated in Section 4.1 (refer to Figure 11). The pseudocode outlining this local search strategy is provided in Algorithm 9 and will be further elaborated upon in this subsection.

The algorithm uses the initial solution obtainable from either H1 or H2 ($solution$). It also uses three vectors: HBx, HBy, and HBz. Additional arguments include the instance with orders categorized into partitions ($\mathscr{O}$), the size of tiers to be explored (t), and the number of boxes in the initial solution (m).

The algorithm initiates by storing the best solution (line 2). Subsequently, for each partition, beginning with the smallest and excluding the last one, boxes are examined with dimensions increased from the current partition box (lines 4 - 32). The box in the last partition is not assessed, as there is no advantage in increasing its dimensions, given the absence of orders above it that could benefit from such adjustments.

The neighbor boxes are generated using the function $createTierUp$ (line 5), as previously detailed in this section. Subsequently, each box from the set HB_t is evaluated, and the first box that results in a better solution is accepted as the new solution (lines 6 - 30). This replacement is conditional upon two factors: first, the dimensions of the new box ($HB_t[b]$) must not match those of the current box ($newBox \neq box$), and second, the volume of the new box must be greater than or equal to that of the current box, as determined by the function $skipBox$ (refer to Algorithm 7).

The process of testing a neighbor box occurs in two distinct steps. First, the increased residual volume of the current partition (p) is calculated (line 9) and subsequently stored as $increase$. Then, all orders from the subsequent partition ($p+1$) are packed into this new box. If this yields a feasible outcome, the reduction in residual volume is computed as $reduction$ (lines 13 - 16), else nothing changes since this order is still in the same box.

It is essential to notice that this latter part is subject to specific conditions to avoid obvious infeasible box dimensions. These conditions are determined by the function $prepackingN1$ (refer to Algorithm 8), which excludes boxes whose largest dimensions are not greater than the largest item's dimension and boxes whose volumes are smaller than the order total volume.

Upon completion of the repacking, the reduction in residual volume is evaluated. If the reduction ($reduction$) surpasses the initial increase ($increase$), a new solution is accepted (lines 19 - 28). The new solution is stored (line 20), orders are reassigned to updated partitions (line 21), and the partition is reset to the previous one, if it is not the initial one (lines 22 - 27).

This process is repeated for all partitions until the second last ($m-1$), where a new solution will be formed. In Section 4.4 the results of this neighborhood will be discussed.

Algorithm 7 – Skip boxes that knowingly do not provide a better solution or is infeasible, comparing with the current solution box.

```
1:  Inputs:
2:  newBox as $(x,y,z)$ coordinates for neighbour box
3:  box as $(x,y,z)$ coordinates for current box
4:  function SKIPBOX(newBox, box)
5:      // Test if newBox dimensions are equal to box dimensions
6:      if $newBox = box$ then
7:          return $False$
8:      end if
9:      // Test if newBox volume is larger than box volume
10:     if $volume(box) \leq volume(newBox)$ then
11:         return $False$
12:     end if
13:     return $True$
14: end function
```

Algorithm 8 – Skip boxes for that do not yield better results or are infeasible, comparing with the order.

```
1:  Inputs:
2:  order as as an order with its items dimensions and volume
3:  newBox as $(x,y,z)$ coordinates of neighbour box
4:  function PREPACKINGN1(order, newBox)
5:      // Test newBox feasibility
6:      // Ensure that smallest item dimensions fits the box
7:      $itL \leftarrow$ largest item dimension from order
8:      $nboxL \leftarrow$ largest dimension form newBox
9:      if $nboxL \leq itL$ then
10:         return $False$
11:     end if
12:     // Ensure that the item volume is smaller than box volume
13:     if $volume(order) \leq volume(newBox)$ then
14:         return $False$
15:     end if
16: end function
```

Algorithm 9 – Pseudocode for Local Search with N1

```
Inputs:
$\mathscr{O}$ // as the set of orders
$HBx, HBy, HBz$ as vector of Herz point for $x, y$ and $z$ axis, respectively
$t$ as tier size
$m$ as number of partitions
function LocalSearchN1($\mathscr{O}$, $solution$, $HBx$, $HBy$, $HBz$, $t$, $m$)
    $bestSolution \leftarrow solution$
    $p \leftarrow 1$
    // For each partition explore the neighbour boxes
    while $p \leq (m-1)$ do
        $HB_t \leftarrow$ createTierUp($solution[p]$, $HBx$, $HBy$, $HBz$,$t$)
        // Iteration through neighbour boxes
        for $b \leftarrow 1$ to $|HB_t|$ do
            $reduction \leftarrow 0$
            // Testing only boxes with obvious better results
            if skipBox($HB_t[b]$, $solution[p]$) then
                // Calculate the increase in residual volume for
                // all orders from the current partition (p)
                $increase \leftarrow volume(HB_t) - volume(solution[p]) \cdot |O_p|$
                // Try to pack all orders from the next partition in the neighbour box
                for $order \in \mathscr{O}_{p+1}$ do
                    // Testing only obviously feasible boxes
                    if prepackingN1($order$, $newBox$) then
                        $box \leftarrow$ packingModel($order$, $HB_t[b]$)
                        // If packed, calculate the residual volume reduction
                        if $box \neq \emptyset$ then
                            $reduction \leftarrow reduction//$
                            $+volume(solution[p+1]) - volume(HB_t)$
                        end if
                    end if
                end for
                // If the reduction is larger than the increase,
                // accept new box as a solution and go to next partition,
                // else, go to next box
                if $reduction > increase$ then
                    $bestSolution[partition] \leftarrow HB_t[b]$
                    Reorganize $\mathscr{O}$ for partitions $p$ and $p+1$
                    if $p = 1$ then
                        $p \leftarrow 0$
                    else
                        $p \leftarrow p - 2$
                    end if
                    Break
                end if
            end if
        end for
        $p \leftarrow p + 1$
    end while
    return $bestSolution$
end function
```

4.3.3 Local Search using Neighbourhood N2

This second local search explores better solutions using the Dimension Reduction Neighbourhood (N2). The objective is to decrease the box dimensions of a given partition p, aiming to enhance the packing efficiency of p and $p-1$. However, as the dimensions are reduced, some orders of p may become infeasible, requiring repacking into partitions with larger boxes, leading to increased residual volume (refer to Figure 12). A better solution is obtained if the extra residual volume from the repacked orders is less than the reduction gained by the improved packing of the explored box. Further details regarding this are described in Section 4.1.

Although there are variations in the implementation of N1 and N2, the overall concept is similar. Therefore, the algorithms of this local search are described in Appendix G.

4.4 Computational Experiments

In this section, five instances are generated in order to analyze the developed constructive and local search heuristics. First, we evaluated the quality of initial solutions obtained by the constructive heuristics H1 and H2. In sequence, an analysis of solution quality and computational times is presented. Following this, the local searches are tested with the initial solution provided by heuristic H1 and H2. However, before that, the tier is tested to determine an appropriately sized neighborhood with a reasonable runtime. Then, the local searches are executed independently using the defined tier, employing either N1 or N2, and then in sequence, N1 followed by N2 and vice versa. The resulting improvements in solution quality and computational runtimes are compared and discussed.

Computational experiments were conducted on a system consisting of an Intel® Core^{TM} i7-7700 CPU 3.60GHz x 8 with 15.5 GiB RAM and OS Ubuntu 20.04.4 LTS 64-bit. The heuristics and local searches were developed using Julia version 1.7.2, and the packing models were solved using Gurobi Optimizer version 9.5.1 build v9.5.1rc2 (linux64), with a time limit of 1,800 seconds.

4.4.1 Instance Sets

The instances utilized are similar to those described in Subsection 3.3.1. However, the main distinction lies in treating instances rather than considering each order as an individual instance, a single instance is composed of a set of orders and their respective items. However, the generation process is the same as outlined in that section.

A total of five instances were generated, each comprising 250 orders. Each order contains one to five items. A reduction was made in the maximum items per order, opting for a limit of five items as opposed to the previous 10. This reduction was implemented to mitigate the increase in computational time resulting from orders with 6 up to 10 items, as discussed in Subsection

3.3.5. These adjustments were made due to the reliance on the heuristics and local searches on packing utilizing the model described in Subsection 3.2.1.

4.4.2 Analysing the Constructive Heuristics

The constructive heuristics H1 and H2 were applied to the above instances considering $m = 20$, i.e., a set of 20 boxes. The computational results are presented in Table 33. The first column represents the instances. The following two columns present the runtime and the objective value (V_{RES}) obtained by heuristic H1 for each instance. Columns 4-5 present the same information for heuristic H2. The final column presents the Percentage Difference of the objective value ($\frac{H1-H2}{H1}$).

This table shows that H1 yields superior results, achieving an overall 35.5% lower objective value compared to H2. However, regarding runtimes, H2 demonstrates better performance, with a mean of 118 seconds compared to 8,662 seconds (approximately 2.4 hours) for H1. Both results were expected, given that H1 was designed to provide a tighter solution by exploring a broader range of boxes, resulting in a better objective value. On the other hand, H2 aims for a quick initial solution, prioritizing speed over quality, with the intention of subsequent enhancements by other methods.

The heuristic H2 starts with a random set of boxes, leading to variations in the solution obtained for a given instance. To assess the variability in the solutions generated, the 5 instances were executed 10 times each with seeds ranging from 1 to 10, resulting in 50 solutions. Table 34 contains the mean and standard deviation (SD) of runtime and objective value (V_{RES}) for the mentioned tests. The runtimes remained relatively constant, averaging around 103 seconds with minimal variation (approximately 6.5 seconds). In contrast, the objective value exhibited a higher variability of approximately 11.7% (168,277 out of 1,432,654). While this variability is not substantial, it can impact subsequent methods by providing a wide range of viable solutions, as will be further explored in this work.

Last, Table 35 shows a comparison of H1 results with the best results for H2, where rows represent the instances and columns time in seconds, objective value (V_{RES}) for H1 and H2 in this order, and last percentile increase of the objective value of H1 to H2. In this table, it is possible to see that in H2 best solutions, objective values are, on average 23.0% higher, and times are significantly lower when compared with H1.

Table 33 – Table with runtimes in seconds and objective value (V_{RES}) for the Constructive Heuristic (H1) and the Fast Constructive Heuristic (H2), percentage difference calculated as $\frac{H1-H2}{H1}$ of the objective values.

	H1		H2		Percentage
Instance	Time (s)	V_{RES}	Time (s)	V_{RES}	Difference
1	9,251	926,646	137	1,509,856	-62.9%
2	8,518	1,001,971	113	1,564,715	-56.2%
3	8,172	1,134,401	117	1,199,237	-5.7%
4	8,713	1,004,537	112	1,306,586	-30.1%
5	8,655	933,352	109	1,196,526	-28.2%
Mean	8,662	1,000,181	118	1,355,384	-35.5%

Table 34 – Table with the variation of random starts for Fast Constructive Heuristic (H2), with mean and standard deviation (SD) of times and objective values (V_{RES}), executed for 20 boxes and 10 different seeds for each instance.

	Time (s)		V_{RES}	
Instance	Mean	SD	Mean	SD (%)
1	103.5	10.4	1,425,987	12.4%
2	102.5	2.6	1,532,823	10.1%
3	103.3	3.8	1,401,037	10.0%
4	105.4	9.0	1,454,074	8.5%
5	104.2	4.2	1,349,352	15.2%
Mean	103.8	6.5	1,432,654	11.7%

Table 35 – Table with time and objective value (V_{RES}) for H1 and H2 best run and percentile difference (Diff. = $\frac{(H2-H1)}{H1}$), executed for 20 boxes.

	H1		H2 Best		
Instance	Time (s)	V_{RES}	Time (s)	V_{RES}	Diff. (%)
1	9,251	926,646	106	1,204,120	29.9%
2	8,518	1,001,971	101	1,332,366	33.0%
3	8,172	1,134,401	110	1,205,427	6.3%
4	8,713	1,004,537	104	1,270,432	26.5%
5	8,655	933,352	106	1,138,233	22.0%
Mean	8,662	1,000,181	105	1,230,116	23.0%

4.4.3 Analysing the Tier Dimension

Before discussing the local search experiments, it is necessary to define the size of each neighborhood, i.e., the tiers dimension. The computational experiments were conducted using Instance 1 with the initial solution H2 with tiers ranging from 1 to 6 for N1 and N2. The results are presented in Figure 14 and Table 36. They provide the same data in different formats to facilitate the visualization.

In Figure 14, a bar plot represents the percentage objective value reduction of the initial solution (Reduction, left axis), where N1 is depicted in light blue and N2 in blue. Additionally, a line plot shows the time in seconds (right axis), with N1 in light purple and N2 in purple. The x-axis represents the tier. Table 36 presents the same data, with the tiers on the rows and columns displaying the time and objective value reduction (Red.) for both N1 and N2. Percentage values indicate how much the initial solution is reduced.

Based on these results, it is possible to observe that increasing the tiers can lead to better results and higher reduction rates. However, this improvement comes at the expense of increased solving time. Tier 2 demonstrates the most optimal balance, showcasing the best results within a reasonable time frame for both neighborhoods. Beyond Tier 2, the solving times start to escalate significantly without corresponding significant improvements in results.

Figure 14 – Image with bar plot of objective value reduction (left y-axis) and line plot for times (right y-axis) of each local search (N1 and N2) for each tier (x-axis) from 1 to 6. Results obtained for Instance 1.

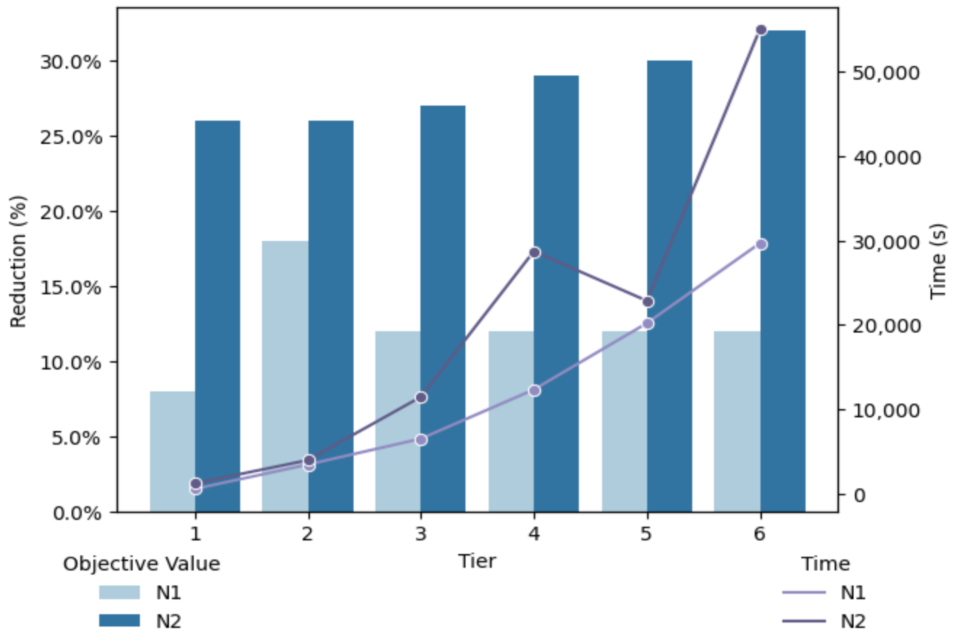


Table 36 – Table with results of tier exploration, where rows represent tiers and columns the runtime in second and percent objective value reduction from H2 solution for each local search (N1 and N2).

	LS_{N1}		LS_{N2}	
Tier	Time (s)	Red.	Time (s)	Red.
1	597	8.2%	1,248	26.3%
2	3,481	17.6%	4,002	26.5%
3	6,538	12.1%	11,482	27.5%
4	12,351	12.2%	28,707	29.2%
5	20,207	12.2%	22,862	30.5%
6	29,663	12.2%	54,955	32.1%

4.4.4 Analysing the Local Search with Initial Solution H2

With the initial solution provided by H2, the local search is run to assess its ability to improve the solution. First, the local search was applied using neighbourhoods N1 and N2 individually, labeled as LS_{N1} and LS_{N2}, respectively. Subsequently, both neighbourhoods are run in sequence: N1 followed by N2 (LS_{N1N2}), and vice versa (LS_{N2N1}). All variants of the local search used tier 2. The obtained results are summarized in Table 37. In this table, rows represent each instance, and columns display the outcomes of each neighbourhood in terms of runtime in seconds (not including H2 runtimes) and the percentage reduction from the objective value obtained by H2 (Red.= $\frac{H2-LS_N}{H2}$, where N is either N1, N2, N1N2 and N2N1).

Results indicate that LS_{N2} had a better performance than LS_{N1}, achieving an average reduction of 25.5% compared to the 11.2% achieved by LS_{N1}. However, considering runtimes, the scenario changes. In this case, LS_{N1} exhibits a faster runtime, averaging 2,183 seconds, while LS_{N2} takes approximately three times that. The disparity in runtimes can be attributed to the larger number of improved solutions by LS_{N2}.

LS_{N1N2} and LS_{N2N1} achieved a similar reduction, approximately 28.9% and 29.6%, respectively, demonstrating the effectiveness of the exploration in sequence. Regarding runtimes, LS_{N2N1} had a slight advantage, taking about 300 seconds less on average. This outcome suggests that employing both neighbourhoods together is beneficial. Moreover, the total runtimes are slightly lower than the sum of the times when each local search is run separately (8,327 seconds).

These findings suggest that employing LS_{N2N1} is, albeit marginally, the best approach evaluated for implementing these local searches. Consequently, this will be the chosen method moving forward.

Table 37 – The experiment results obtained by the Local Search method using neighbourhood N1, N2, N1N2 and N2N1. Times do not include H2 runtimes.

	LS_{N1}		LS_{N2}		LS_{N1N2}		LS_{N2N1}	
Instance	Time (s)	Red.	Time (s)	Red.	Time (s)	Red.	Time (s)	Red.
1	3,356	17.6%	4,062	26.5%	8,885	**40.1%**	5,788	31.6%
2	2,076	9.5%	5,824	31.0%	5,648	31.9%	7,509	**34.4%**
3	1,637	2.7%	6,494	24.8%	9,813	20.8%	8,171	**30.4%**
4	1,704	15.7%	6,244	24.1%	6,600	28.8%	7,990	**29.8%**
5	2,140	10.7%	8,098	21.1%	9,622	**23.1%**	9,154	21.8%
Mean	2,183	11.2%	6,144	25.5%	8,114	28.9%	7,722	**29.6%**

4.4.5 Comparing results from H1 with LS_{N2N1}

In this section, the results obtained by the best constructive heuristic (H1) are compared with the best Local Search (LS_{N2N1}). In Table 38, the results are summarized, where rows represent instances, and columns show runtimes (including H2 runtimes for LS_{N2N1}) and objective function values (V_{RES}) for H1 and LS_{N2N1}, respectively. The last column (Diff. = $\frac{(H1-LS)}{H1}$) is the percent difference of the objective values.

The objective values achieved by LS_{N2N1} are better when compared to H1. On average, the method was able to outperform H1 by 5.1%. Detailed analysis of the results reveals that all instances presented either a superior or comparable performance, except for Instance 1, which exhibited poorer performance. However, it is important to highlight that Instance 3 significantly impacts this result, lowering the mean. Finally, in terms of runtimes, LS_{N2N1} showcased shorter times on average (7,840 seconds) compared to H1 (8,662 seconds).

Table 38 – Table comparing results from H1 with H2 run with N2N1 (H2&N2N1). Where rows have the instances and columns time in seconds and objective value (V_{RES}). Times include H2 runtimes for LS_{N2N1}

	H1		LS_{N2N1}		Diff. (%)
Instance	Time (s)	V_{RES}	Time (s)	V_{RES}	V_{RES}
1	9,251	926,646	5,925	1,032,576	11.4%
2	8,518	1,001,971	7,622	1,026,406	2.4%
3	8,172	1,134,401	8,288	834,659	-26.4%
4	8,713	1,004,537	8,101	917,581	-8.7%
5	8,655	933,352	9,264	936,153	0.3%
Mean	8,662	1,000,181	7,840	949,475	-5.1%

4.4.6 Analysing the Local Search with Initial Solution H1

With the conclusion that the local search is more effective and faster than H1, the question arises: Could H1 be further improved using these methods? To explore this, H1 is employed as an initial solution of LS_{N2N1}. The results are presented in Table 39, with instances listed as rows, runtimes in seconds and objective value reduction ($VRES$) for H1 and LS_{N2N1}^{H1}, and the corresponding percentage reduction (Red. = $\frac{(H1-LS_{N2N1})}{H1}$).

From this table, it is evident that there is still potential to improve the objective value by applying the local search. The results indicate on average there is a 15.5% reduction from the initial objective value.

Upon examining the runtimes, a significant reduction is observed compared to the local search when used with H2. This reduction can be attributed to the fewer number of new solutions found, given that H1 already provides a superior solution. Overall, it is valuable to continue refining the H1 solution. However, it is crucial to consider the time aspect. Despite the reduced execution times for the local searches, H1 still requires a longer runtime. Therefore, the decision to implement these enhancements would depend on the available time for making such determinations.

Table 39 – Table with runtimes, objective values (V_{RES}) and percentage reduction (Red.) of H1 and H1 solution run with LS_{N2N1}.

	Time		V_{RES}		
Instance	H1	LS_{N2N1}^{H1}	H1	LS_{N2N1}^{H1}	Red. (%)
1	9,251	5,109	926,646	813,661	12.2%
2	8,518	5,445	1,001,971	863,860	13.8%
3	8,172	8,096	1,134,401	848,187	25.2%
4	8,713	4,442	1,004,537	927,546	7.7%
5	8,655	5,919	933,352	779,348	16.5%
Mean	8,662	5,802	1,000,181	846,520	15.1%

4.5 Conclusions

In summary, analysing constructive heuristics, it was observed that H1 had better results but worse runtime than H2, see Figure 15 and Table 40. As expected, the local search heuristics presented the best results and the worst runtimes.

The local search method based on altering the dimensions of already existing boxes works and is viable. The method significantly improved initial solutions when used with both heuristics. Furthermore, compared to a constructive heuristic, it showed better results. This answers the proposition made for this chapter: while generating a limited number of boxes, its dimensions were altered to provide the best packing for a set of orders.

Figure 15 – This figure contains a bar plot with the objective value (V_{RES}) for both heuristics (H), H1 in blue and H2 in orange, and local search using initial solution H1 or H2 (LS^{H1}_{N2N1} or LS^{H1}_{N2N1}). Values are the mean for all instances, and the vertical bar as the standard deviation between instances.

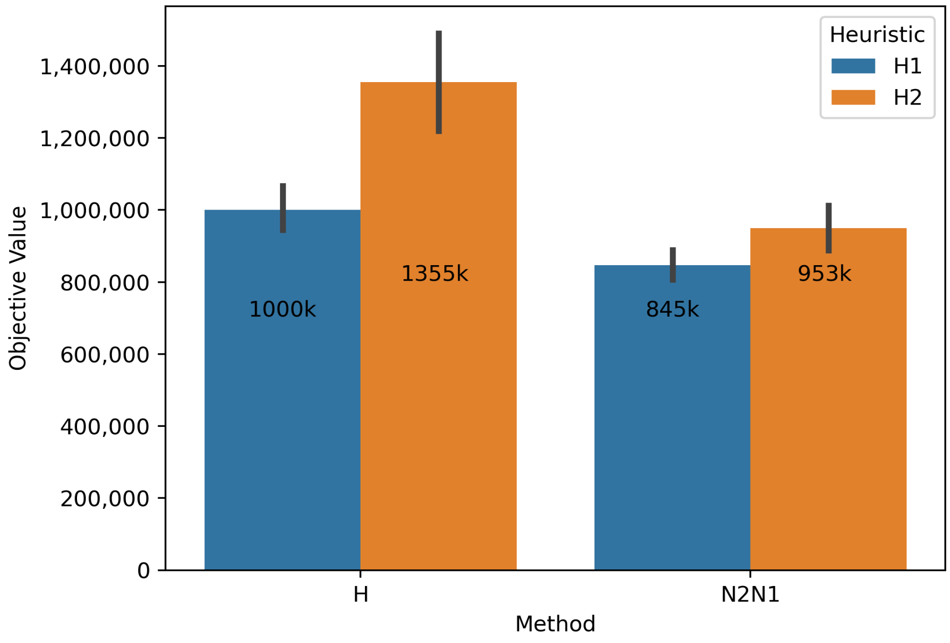


Table 40 – The table presents the runtimes in seconds and objective values for H1 and H2 executed as an initial solution for the local search LS_{N2N1}. Last column represents the percent difference of the values (Diff. = $\frac{LS^{H1}_{N2N1} - LS^{H2}_{N2N1}}{LS^{H1}_{N2N1}}$).

	LS^{H1}_{N2N1}		LS^{H2}_{N2N1}		Diff. (%)
Instance	Time (s)	V_{RES}	Time (s)	V_{RES}	V_{RES}
1	14,360	813,661	5,925	1,032,576	-26.9%
2	13,627	863,860	7,622	1,026,406	-18.8%
3	13,281	848,187	8,288	834,659	1.6%
4	13,822	927,546	8,101	917,581	1.1%
5	13,764	779,348	9,264	936,153	-20.1%
Mean	13,771	846,520	7,840	949,475	-12.2%

CHAPTER

5

CONCLUSIONS AND FUTURE RESEARCHES

In this work, two goals were established: i) to study packing models in the e-commerce scenario, and ii) to define an optimal set of boxes to pack a set of orders. To accomplish these objectives, first, packing models were studied and analysed in Chapter 3. Subsequently, a new methodology to define boxes was proposed. Based on a local search, this methodology generates and improves boxes and is described in Chapter 4.

The first objective, following the definition of the models, had three phases. Phases 1 and 2 aimed to choose the best model regarding residual volume reduction and time. The computational experiments revealed that Z_{CLS} exhibited superior performance. Despite its longer runtimes, particularly evident with 6 to 10 items, it demonstrated the most substantial reduction in residual volume. Consequently, the decision is to use the Z_{CLS} model in Phase 3.

In Phase 3, two aspects of the boxes were analyzed. First, a novel methodology for generating boxes was tested. Then, the impact of employing up to two boxes for packing was measured. The results show that the proposed HBnd box generation method exhibited superior performance. Concerning the packing of orders into two boxes, the outcomes, in general, suggested a further reduction in residual volume alongside an increase in computational times. This implies that both methods with the selected model were efficient in the packing processes.

The second objective encompassed two challenges: fixing the number of boxes and generating new boxes. To address these, two constructive heuristics were formulated to provide an initial solution to the problem. Also, a local search employing two neighborhoods that altered the dimensions of the boxes was outlined. The local search LS_{N2N1} demonstrated a notable reduction in the initial solutions. Additionally, in terms of the cost-benefit between time and objective value, executing the local search with H2 as an initial solution proved more favorable.

Even though this work achieved its objectives, some aspects could still be explored. One such aspect is to consider the box acquisition costs. Including these costs could significantly impact the solutions obtained in this study, as box costs are not exclusively determined by the box area and can be quite complex, as highlighted by Fontaine and Minner (2022).

A limitation of the proposed methods is their reliance on exact methods for packing. While these methods ensure optimal packing solutions, the model complexity escalates quickly as the number of items increases. Alternative packing methods, such as those presented by [Alonso *et al.* (2016)], could offer an interesting alternative, particularly in realistic scenarios involving thousands of daily orders. In such cases, clustering orders and defining a set of boxes for each cluster could also help reduce the packing volume.

Last, the described neighborhoods and the HBnd box set could still be used by other meta-heuristics, such as Variable Neighborhood Search (VNS) or Genetic Algorithms (GA). Since both are versatile for this problem, alternative strategies could be tested. For example, experimenting with the generation of populations of sampled HBnd boxes by a GA algorithm would be interesting.

In summary, this work showed that packing models can be a viable option for e-commerce, coupled with the option to generate boxes (HBnd). Additionally, a method to enhance a box set by adjusting its dimensions was developed and tested.

BIBLIOGRAPHY

ALI, S.; RAMOS, A. G.; CARRAVILLA, M. A.; OLIVEIRA, J. F. On-line three-dimensional packing problems: A review of off-line and on-line solution approaches. **Computers & Industrial Engineering**, v. 168, p. 108122, (2022). ISSN 0360-8352. Available: <https://www.sciencedirect.com/science/article/pii/S0360835222001929>. Citations on pages 26, 30, and 32.

ALONSO, M.; ALVAREZ-VALDES, R.; PARREñO, F.; TAMARIT, J. Determining the best shipper sizes for sending products to customers. **International Transactions in Operational Research**, v. 23, n. 1-2, p. 265–285, (2016). Available: <https://onlinelibrary.wiley.com/doi/abs/10.1111/itor.12128>. Citations on pages 30, 36, 40, 73, and 98.

AMAZON BOXES, S. **Amazon Boxes Sizes**. Accessed November 22, 2022, (2021). Available: <https://www.boxdimensions.com/>. Citations on pages 55 and 109.

BEASLEY, J. An algorithm for the two-dimensional assortment problem. **European Journal of Operational Research**, v. 19, n. 2, p. 253–261, (1985). ISSN 0377-2217. Available: <https://www.sciencedirect.com/science/article/pii/0377221785901791>. Citation on page 36.

BORTFELDT, A.; WäSCHER, G. Constraints in container loading – a state-of-the-art review. **European Journal of Operational Research**, v. 229, n. 1, p. 1–20, (2013). ISSN 0377-2217. Available: <https://www.sciencedirect.com/science/article/pii/S037722171200937X>. Citations on pages 32 and 34.

BOYSEN, N.; FEDTKE, S.; SCHWERDFEGER, S. Last-mile delivery concepts: a survey from an operational research perspective. **OR Spectrum**, Springer Science and Business Media LLC, v. 43, n. 1, p. 1–58, Sep. 2020. Available: <https://doi.org/10.1007/s00291-020-00607-8>. Citation on page 39.

BRINKER, J.; GüNDüZ, H. I. Optimization of demand-related packaging sizes using a p-median approach. **The International Journal of Advanced Manufacturing Technology**, Springer Science and Business Media LLC, v. 87, n. 5-8, p. 2259–2268, Mar. 2016. Available: <https://doi.org/10.1007/s00170-016-8630-4>. Citation on page 57.

CHEN, C.; LEE, S.; SHEN, Q. An analytical model for the container loading problem. **European Journal of Operational Research**, v. 80, n. 1, p. 68–76, (1995). ISSN 0377-2217. Available: <https://www.sciencedirect.com/science/article/pii/037722179400002T>. Citations on pages 35, 36, 39, 40, 41, 42, 45, 46, 50, 51, 52, 53, 54, 59, 60, 61, 62, 71, and 121.

da Silva, E.; LEãO, A.; TOLEDO, F.; WAUTERS, T. A matheuristic framework for the three-dimensional single large object placement problem with practical constraints. **Computers & Operations Research**, v. 124, p. 105058, (2020). ISSN 0305-0548. Available: <https://www.sciencedirect.com/science/article/pii/S0305054820301751>. Citation on page 33.

DOWSLAND, K. A.; DOWSLAND, W. B. Packing problems. **European Journal of Operational Research**, v. 56, n. 1, p. 2–14, (1992). ISSN 0377-2217. Available: <https://www.sciencedirect.com/science/article/pii/037722179290288K>. Citations on pages 29, 34, and 35.

DYCKHOFF, H. A typology of cutting and packing problems. **European Journal of Operational Research**, v. 44, n. 2, p. 145–159, (1990). ISSN 0377-2217. Cutting and Packing. Available: <https://www.sciencedirect.com/science/article/pii/037722179090350K>. Citation on page 29.

FANSLAU, T.; BORTFELDT, A. A tree search algorithm for solving the container loading problem. **INFORMS Journal on Computing**, v. 22, n. 2, p. 222–235, (2010). Available: <https://doi.org/10.1287/ijoc.1090.0338>. Citations on pages 58 and 103.

FONTAINE, P.; MINNER, S. A branch-and-repair method for three-dimensional bin selection and packing in e-commerce. **Operations Research**, v. 0, n. 0, p. 1–16, (2022). Available: <https://doi.org/10.1287/opre.2022.2369>. Citations on pages 30, 36, 40, 55, 73, and 98.

HERZ, J. C. Recursive computational procedure for two-dimensional stock cutting. **IBM J. Res. Dev.**, IBM Corp., USA, v. 16, n. 5, p. 462–469, sep (1972). ISSN 0018-8646. Available: <https://doi.org/10.1147/rd.165.0462>. Citations on pages 55 and 56.

HOOKER, J. N. Planning and scheduling by logic-based benders decomposition. **Operations Research**, v. 55, n. 3, p. 588–602, (2007). Available: <https://doi.org/10.1287/opre.1060.0371>. Citation on page 36.

HU, H.; ZHANG, X.; YAN, X.; WANG, L.; XU, Y. Solving a new 3d bin packing problem with deep reinforcement learning method. **arXiv preprint arXiv:1708.05930**, (2017). No citation.

HÜBNER, A.; HOLZAPFEL, A.; KUHN HEINRICH, j. . O. M. R. Operations management in multi-channel retailing: an exploratory study. v. 8, n. 3, p. 84–100, dec (2015). Available: <https://link.springer.com/article/10.1007/s12063-015-0101-9#citeas>. Citations on pages 19 and 55.

JIANG, Y.; CAO, Z.; ZHANG, J. Solving 3d bin packing problem via multimodal deep reinforcement learning. In: **Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems**. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, (2021). (AAMAS '21), p. 1548–1550. ISBN 9781450383073. Citation on page 33.

LODI, A.; MARTELLO, S.; MONACI, M. Two-dimensional packing problems: A survey. **European Journal of Operational Research**, v. 141, n. 2, p. 241–252, (2002). ISSN 0377-2217. Available: <https://www.sciencedirect.com/science/article/pii/S0377221702001236>. Citation on page 31.

LODI, A.; MARTELLO, S.; VIGO, D. Recent advances on two-dimensional bin packing problems. **Discrete Applied Mathematics**, v. 123, n. 1, p. 379–396, (2002). ISSN 0166-218X. Available: <https://www.sciencedirect.com/science/article/pii/S0166218X0100347X>. Citation on page 31.

SCHEITHAUER, G. **Introduction to Cutting and Packing Optimization**. Springer International Publishing, (2018). Available: <https://doi.org/10.1007/978-3-319-64403-5>. Citation on page 29.

SILVA, E.; OLIVEIRA, J. F.; WäSCHER, G. The pallet loading problem: a review of solution methods and computational experiments. **International Transactions in Operational Research**, v. 23, n. 1-2, p. 147–172, (2016). Available: <https://onlinelibrary.wiley.com/doi/abs/10.1111/itor.12099>. Citation on page 32.

STATISTA, P. B. **Statista**. Statista Accessed September 30, 2022, (2021). Available: <https://www.statista.com/statistics/1139910/parcel-shipping-volume-worldwide/>. Citation on page 25.

THORSTEINSSON, E. S. Branch-and-check: A hybrid framework integrating mixed integer programming and constraint logic programming. In: WALSH, T. (Ed.). **Principles and Practice of Constraint Programming — CP 2001**. Berlin, Heidelberg: Springer Berlin Heidelberg, (2001). p. 16–30. ISBN 978-3-540-45578-3. Citation on page 36.

TSAI, R. D.; MALSTROM, E. M.; KUO, W. Three dimensional palletization of mixed box sizes. **IIE Transactions**, Taylor & Francis, v. 25, n. 4, p. 64–75, (1993). Available: <https://doi.org/10.1080/07408179308964305>. Citations on pages 35, 36, 39, 40, 46, 47, 48, 49, 50, 51, 52, 53, 54, 59, 60, and 121.

UPS. **Cube optimization**. Accessed November 08, 2022, (2021). Available: <https://www.ups.com/media/en/CubeOptimizationSalesSheet.pdf>. Citation on page 29.

VIEIRA, M. V. C.; CARVALHO, M. Lexicographic optimization for the multi-container loading problem with open dimensions for a shoe manufacturer. **4OR-A QUARTERLY JOURNAL OF OPERATIONS RESEARCH**, SPRINGER HEIDELBERG, TIERGARTENSTRASSE 17, D-69121 HEIDELBERG, GERMANY, (2022). ISSN 1619-4500. Citation on page 36.

VIEIRA, M. V. C.; FERREIRA, F.; DUQUE, J. C. M.; ALMEIDA, R. M. P. On the packing process in a shoe manufacturer. **Journal of the Operational Research Society**, Taylor & Francis, v. 72, n. 4, p. 853–864, (2021). Available: <https://doi.org/10.1080/01605682.2019.1700765>. Citation on page 36.

WALMART. **Walmart Sustainability Hub**. Accessed November 08, 2022, (2022). Available: <https://www.walmartsustainabilityhub.com/>. Citation on page 29.

WEBSTER, J.; WATSON, R. T. Analyzing the past to prepare for the future: Writing a literature review. **MIS Quarterly**, Management Information Systems Research Center, University of Minnesota, v. 26, n. 2, p. xiii–xxiii, (2002). ISSN 02767783. Available: <http://www.jstor.org/stable/4132319>. Citation on page 30.

WOLSEY, L. A. **Integer Programming**. USA: John Wiley & Sons, Ltd, 2020. Citations on pages 52 and 60.

WäSCHER, G.; HAUßNER, H.; SCHUMANN, H. An improved typology of cutting and packing problems. **European Journal of Operational Research**, v. 183, n. 3, p. 1109–1130, (2007). ISSN 0377-2217. Available: <https://www.sciencedirect.com/science/article/pii/S037722170600292X>. Citations on pages 26, 29, 30, and 35.

ZHAO, X.; BENNELL, J. A.; BEKTAş, T.; DOWSLAND, K. A comparative review of 3d container loading algorithms. **International Transactions in Operational Research**, v. 23, n. 1-2, p. 287–320, (2016). Available: <https://onlinelibrary.wiley.com/doi/abs/10.1111/itor.12094>. Citation on page 32.

APPENDIX

A

INSTANCE - ITEMS

Table 41 containing all 90 items from Fanslau and Bortfeldt (2010) not adjusted by factor (4.64), as described in Subserction 3.3.3 .

Table 41 – Table containing items dimensions.

Index	Length	Width	Height	Volume
1	25	74	64	118,400
2	67	49	90	295,470
3	26	96	41	102,336
4	79	81	37	236,763
5	47	57	87	233,073
6	26	69	29	52,026
7	58	114	71	469,452
8	71	114	58	469,452
9	88	53	77	359,128
10	25	74	64	118,400
11	58	114	71	469,452
12	98	87	78	665,028
13	56	59	37	122,248
14	47	87	57	233,073
15	94	53	92	458,344
16	38	44	64	107,008
17	94	97	37	337,366
18	37	59	56	122,248
19	42	27	44	49,896
20	49	65	115	366,275
21	98	87	78	665,028

22	100	101	69	696,900
23	38	44	64	107,008
24	84	114	72	689,472
25	100	101	69	696,900
26	84	114	72	689,472
27	25	64	74	118,400
28	44	64	38	107,008
29	57	47	87	233,073
30	37	56	59	122,248
31	79	81	37	236,763
32	106	48	35	178,080
33	106	48	35	178,080
34	37	81	79	236,763
35	71	114	58	469,452
36	26	29	69	52,026
37	67	49	90	295,470
38	42	74	77	239,316
39	37	97	94	337,366
40	26	96	41	102,336
41	100	101	69	696,900
42	25	74	64	118,400
43	94	53	92	458,344
44	94	53	92	458,344
45	49	65	115	366,275
46	44	38	64	107,008
47	37	97	94	337,366
48	98	87	78	665,028
49	77	88	53	359,128
50	42	74	77	239,316
51	67	49	90	295,470
52	37	97	94	337,366
53	25	74	64	118,400
54	92	94	53	458,344
55	42	44	27	49,896
56	79	81	37	236,763
57	88	53	77	359,128
58	42	44	27	49,896

59	100	101	69	696,900
60	49	65	115	366,275
61	105	74	115	893,550
62	79	81	37	236,763
63	26	29	69	52,026
64	74	115	105	893,550
65	26	96	41	102,336
66	78	87	98	665,028
67	26	69	29	52,026
68	67	49	90	295,470
69	98	87	78	665,028
70	94	97	37	337,366
71	37	56	59	122,248
72	78	87	98	665,028
73	37	56	59	122,248
74	47	57	87	233,073
75	44	42	27	49,896
76	67	49	90	295,470
77	94	53	92	458,344
78	42	44	27	49,896
79	58	114	71	469,452
80	47	57	87	233,073
81	58	114	71	469,452
82	57	47	87	233,073
83	74	25	64	118,400
84	49	65	115	366,275
85	69	26	29	52,026
86	26	96	41	102,336
87	67	49	90	295,470
88	47	57	87	233,073
89	98	87	78	665,028
90	88	53	77	359,128

APPENDIX

B

INSTANCE - ORDERS & ITEMS

Table 42 containing the first 20 orders of instance described in Subsection 3.3.1, with dimensions of the items adjusted by factor (4.64).

Table 42 – Table with 20 orders and their items.

Order Number	Length	Width	Height	Volume
1	14	11	19	2,926
1	21	19	17	6,783
2	12	10	19	2,280
2	9	16	17	2,448
2	9	9	6	486
3	9	14	8	1,008
3	19	11	17	3,553
4	11	14	25	3,850
4	21	19	17	6,783
5	20	20	11	4,400
5	22	22	15	7,260
5	21	19	17	6,783
6	9	9	6	486
6	23	16	25	9,200
6	14	11	19	2,926
6	21	19	17	6,783
6	19	11	17	3,553
7	5	16	14	1,120
7	17	17	8	2,312
8	12	25	15	4,500
9	20	11	20	4,400

9	5	16	14	1,120
10	6	21	9	1,134
10	15	25	12	4,500
10	21	19	17	6,783
11	5	16	14	1,120
11	8	17	17	2,312
12	11	14	25	3,850
12	8	12	13	1,248
13	21	19	17	6,783
13	16	25	23	9,200
14	16	5	14	1,120
15	18	25	16	7,200
15	14	11	19	2,926
15	20	11	20	4,400
16	9	16	17	2,448
16	6	21	9	1,134
17	17	17	8	2,312
17	19	11	17	3,553
17	10	19	12	2,280
17	11	14	25	3,850
17	18	25	16	7,200
17	9	14	8	1,008
17	17	17	8	2,312
17	22	22	15	7,260
17	20	21	8	3,360
17	17	17	8	2,312
18	10	12	19	2,280
18	9	16	17	2,448
19	14	11	19	2,926
19	10	12	19	2,280
20	12	25	15	4,500

APPENDIX

C

INSTANCE - BOX SET AMB

Table 43 containing the 123 Amazon boxes extracted from Amazon Boxes (2021). Extraction date was on 21 of September 2022.

Table 43 – Table containing AMB dimensions.

Box Id	Box Name	Length	Width	Height	Volume
1	Z14	46	33	14	21,252
2	130 (Z13)	34	21	27	19,278
3	20	23	15	13	4,485
4	120 (Z12)	39	24	17	15,912
5	2BK	48	42	30	60,480
6	100 (Z10)	34	28	13	12,376
7	28 (ZM1)	25	20	11	5,500
8	50 (Z05)	33	29	9	8,613
9	1B2 (BO1)	40	32	9	11,520
10	3A3 (B75)	52	79	9	36,972
11	60	30	23	13	8,970
12	B70 (PD)	76	18	15	20,520
13	56	20	28	15	8,400
14	80	30	41	8	9,840
15	D37	64	51	35	114,240
16	10	23	15	9	3,105
17	81	38	25	11	10,450
18	40	25	16	18	7,200
19	70	46	13	15	8,970
20	170 (Z17)	44	34	20	29,920
21	1BK (B73)	53	41	12	26,076

22	118	36	28	15	15,120
23	110 (Z11)	46	36	9	14,904
24	176	46	36	25	41,400
25	143	41	31	18	22,878
26	93	30	24	18	12,960
27	0A0	23	17	6	2,346
28	BP0 (V4)	25	20	5	2,500
29	A0 (BR0)	26	18	6	2,808
30	A1	25	18	8	3,600
31	A1 (BH0, BY0, BYO)	25	18	8	3,600
32	BP1 (V4)	34	24	6	4,896
33	BM2	30	24	6	4,320
34	A3	32	24	8	6,144
35	A3 (BH1, BY1)	25	18	13	5,850
36	1A3 (BY4)	32	25	8	6,400
37	1A1 (BY2)	29	22	11	7,018
38	BM5	30	23	10	6,900
39	Z1	30	23	10	6,900
40	1AD (BY8, BND)	34	24	9	7,344
41	W01 (BDA)	25	20	17	8,500
42	1B2	38	30	8	9,120
43	BM3	30	24	13	9,360
44	E1	41	23	10	9,430
45	1A5 (B45,BF5)	34	28	12	11,424
46	A4	30	22	18	11,880
47	E4	41	30	10	12,300
48	2B4	36	43	8	12,384
49	1AB (BFA)	36	23	16	13,248
50	N3	38	28	13	13,832
51	1A7 (B47,BF7)	37	20	18	13,320
52	1BF	49	36	8	14,112
53	S6A (BT8)	89	13	13	15,041
54	BT9 (S6A)	13	13	90	15,210
55	2A5	51	28	11	15,708
56	C1	48	33	10	15,840
57	N3 (B41, BF1, BJ0)	41	30	13	15,990
58	1AE (BY9-BNB)	33	25	21	17,325

59	BT2 (S7)	61	38	8	18,544
60	J7 (B7H)	103	13	13	17,407
61	1B9	46	23	18	19,044
62	W02 (BDB)	32	28	22	19,712
63	N6 (B40)	16	30	41	19,680
64	Z4	41	31	16	20,336
65	Q3	93	27	8	20,088
66	C2	48	33	15	23,760
67	J7A	133	13	13	22,477
68	1B9 (B4E, BFE)	46	23	23	24,334
69	1AC (B7N)	34	29	25	24,650
70	Z16	41	36	17	25,092
71	PC	60	41	10	24,600
72	P8	46	58	10	26,680
73	E6	41	30	20	24,600
74	K3 (B42-BF2)	48	34	16	26,112
75	S8 (B7M)	66	38	10	25,080
76	U3 (BFR)	102	27	10	27,540
77	2AA	61	41	11	27,511
78	1A9 (B48)	36	32	24	27,648
79	Z05	41	26	26	27,716
80	2A7	61	44	11	29,524
81	1B4	43	33	20	28,380
82	C3	48	33	20	31,680
83	C3	48	33	20	31,680
84	P1 (BS5)	54	39	16	33,696
85	1BG	46	55	13	32,890
86	B0	43	28	28	33,712
87	1B4 (B4D, BFD)	45	36	21	34,020
88	K4 (B5E)	50	33	21	34,650
89	U1 (BCS – B7T)	102	25	14	35,700
90	Q4	51	94	8	38,352
91	2A0	51	41	18	37,638
92	Z06	41	36	26	38,376
93	B0 (BS2)	44	30	29	38,280
94	2B5 (B31)	39	33	30	38,610
95	1BA (B4F, BFF)	48	31	29	43,152

96	B11	71	41	15	43,665
97	U4 (B76)	102	23	19	44,574
98	C4	48	34	30	48,960
99	P4 (BS8)	66	41	18	48,708
100	Q6	94	74	8	55,648
101	B05	56	38	13	27,664
102	Q5	76	94	8	57,152
103	PA (BS3)	51	35	31	55,335
104	B14	61	46	20	56,120
105	1BB (B4G, BFG)	49	37	31	56,203
106	2A6	53	43	27	61,533
107	PB (BCS)	58	38	30	66,120
108	F3	66	48	22	69,696
109	U0 (BSG)	102	30	23	70,380
110	2A8	66	48	23	72,864
111	K89	113	25	25	70,625
112	P7 (BSB)	71	50	21	74,550
113	CB4 (B58)	17	30	147	74,970
114	D4	56	46	30	77,280
115	P9 (B77-B87)	70	57	20	79,800
116	S5 (B44)	57	46	31	81,282
117	2BB	56	42	36	84,672
118	P2	53	45	39	93,015
119	3A1	79	52	27	110,916
120	P5	65	53	42	144,690
121	Q2	86	56	34	163,744
122	U5 (B80)	102	51	36	187,272
123	U2 (BFR)	102	57	41	238,374

APPENDIX

D

INSTANCE - BOX SET HB

Table 44 containing the 123 HB boxes generated as described in Subsection 3.3.2.

Table 44 – Table containing HB dimensions.

Box ID	Length	Height	Width	Volume
1	24	6	6	864
2	24	8	5	960
3	15	9	9	1,215
4	14	12	8	1,344
5	28	15	6	2,520
6	43	12	5	2,580
7	22	13	12	3,432
8	27	25	6	4,050
9	20	15	15	4,500
10	19	17	14	4,522
11	27	14	12	4,536
12	43	19	6	4,902
13	38	31	5	5,890
14	45	29	5	6,525
15	47	29	5	6,815
16	25	23	13	7,475
17	44	35	5	7,700
18	30	27	10	8,100
19	44	37	5	8,140
20	45	16	12	8,640
21	36	19	13	8,892
22	34	18	17	10,404

23	37	21	14	10,878
24	33	31	11	11,253
25	46	42	6	11,592
26	43	18	15	11,610
27	27	23	19	11,799
28	34	24	15	12,240
29	32	22	18	12,672
30	44	31	10	13,640
31	47	35	9	14,805
32	32	31	15	14,880
33	32	31	15	14,880
34	39	20	20	15,600
35	31	23	22	15,686
36	46	44	8	16,192
37	33	25	20	16,500
38	40	35	12	16,800
39	47	50	8	18,800
40	35	32	18	20,160
41	45	27	17	20,655
42	44	24	20	21,120
43	46	46	10	21,160
44	46	42	11	21,252
45	29	29	27	22,707
46	31	31	24	23,064
47	30	28	28	23,520
48	40	31	19	23,560
49	40	30	21	25,200
50	41	28	22	25,256
51	34	34	22	25,432
52	44	40	15	26,400
53	46	36	16	26,496
54	32	30	28	26,880
55	44	44	14	27,104
56	43	34	19	27,778
57	46	32	19	27,968
58	42	26	26	28,392
59	34	34	26	30,056

60	46	41	16	30,176
61	44	35	20	30,800
62	45	37	19	31,635
63	40	35	23	32,200
64	36	36	25	32,400
65	47	50	14	32,900
66	46	46	16	33,856
67	46	37	20	34,040
68	39	35	25	34,125
69	42	39	21	34,398
70	39	35	26	35,490
71	39	37	25	36,075
72	47	48	16	36,096
73	43	37	23	36,593
74	47	47	17	37,553
75	39	38	26	38,532
76	38	37	28	39,368
77	46	46	19	40,204
78	43	43	22	40,678
79	46	37	24	40,848
80	40	38	27	41,040
81	43	40	24	41,280
82	45	42	22	41,580
83	36	34	34	41,616
84	36	36	33	42,768
85	46	31	30	42,780
86	41	37	29	43,993
87	41	40	28	45,920
88	45	43	24	46,440
89	46	44	23	46,552
90	46	44	23	46,552
91	45	45	23	46,575
92	40	38	31	47,120
93	45	35	30	47,250
94	47	36	28	47,376
95	45	44	24	47,520
96	46	43	25	49,450

97	43	34	34	49,708
98	45	43	26	50,310
99	37	37	37	50,653
100	43	38	31	50,654
101	42	39	31	50,778
102	42	37	33	51,282
103	45	44	26	51,480
104	47	38	29	51,794
105	45	36	32	51,840
106	46	38	30	52,440
107	45	42	29	54,810
108	44	39	32	54,912
109	44	41	31	55,924
110	45	39	32	56,160
111	45	44	29	57,420
112	45	44	29	57,420
113	42	37	37	57,498
114	46	37	34	57,868
115	41	39	37	59,163
116	45	44	30	59,400
117	47	46	28	60,536
118	44	43	32	60,544
119	45	41	33	60,885
120	47	45	29	61,335
121	44	41	34	61,336
122	46	42	32	61,824
123	47	49	27	62,181

APPENDIX

E

INSTANCE - BOX SET HBND

Table 45 containing the 123 HBnd boxes generated as described in Subsection 3.3.2.

Table 45 – Table containing HBnd dimensions.

Box Id	Length	Height	Width	Volume
1	26	26	17	11,492
2	34	48	18	29,376
3	12	37	9	3,996
4	20	31	5	3,100
5	17	35	6	3,570
6	32	42	9	12,096
7	39	39	17	25,857
8	16	18	11	3,168
9	23	49	14	15,778
10	32	32	13	13,312
11	14	43	11	6,622
12	32	36	32	36,864
13	32	33	19	20,064
14	14	42	14	8,232
15	31	47	10	14,570
16	31	41	10	12,710
17	36	45	29	46,980
18	35	46	6	9,660
19	21	21	10	4,410
20	23	33	10	7,590
21	32	32	6	6,144
22	5	30	39	5,850

23	12	29	8	2,784
24	21	28	16	9,408
25	28	44	18	22,176
26	32	48	9	13,824
27	25	29	16	11,600
28	34	40	5	6,800
29	25	29	14	10,150
30	36	44	28	44,352
31	16	20	9	2,880
32	35	41	12	17,220
33	38	41	9	14,022
34	34	35	14	16,660
35	18	39	11	7,722
36	33	38	25	31,350
37	34	40	6	8,160
38	26	33	18	15,444
39	33	44	32	46,464
40	39	42	5	8,190
41	16	20	15	4,800
42	21	47	10	9,870
43	17	35	12	7,140
44	22	44	5	4,840
45	27	29	19	14,877
46	31	39	31	37,479
47	27	40	15	16,200
48	34	40	9	12,240
49	23	27	13	8,073
50	33	43	15	21,285
51	27	39	26	27,378
52	28	28	17	13,328
53	32	46	22	32,384
54	31	31	18	17,298
55	35	37	18	23,310
56	20	48	10	9,600
57	22	41	22	19,844
58	22	43	5	4,730
59	38	41	9	14,022

60	21	33	15	10,395
61	19	36	5	3,420
62	14	23	10	3,220
63	25	26	10	6,500
64	11	20	11	2,420
65	25	33	8	6,600
66	17	27	12	5,508
67	11	24	6	1,584
68	17	27	12	5,508
69	29	49	18	25,578
70	39	47	18	32,994
71	39	41	10	15,990
72	28	28	10	7,840
73	13	24	10	3,120
74	33	33	13	14,157
75	34	42	6	8,568
76	31	36	14	15,624
77	16	31	16	7,936
78	18	32	39	22,464
79	33	36	27	32,076
80	39	40	23	35,880
81	14	17	14	3,332
82	37	42	19	29,526
83	30	30	22	19,800
84	15	25	10	3,750
85	33	39	18	23,166
86	33	37	5	6,105
87	30	38	10	11,400
88	6	22	6	792
89	29	41	24	28,536
90	35	35	13	15,925
91	35	38	34	45,220
92	34	38	8	10,336
93	36	46	10	16,560
94	27	31	19	15,903
95	24	42	9	9,072
96	38	39	9	13,338

97	30	33	17	16,830
98	35	37	8	10,360
99	18	47	16	13,536
100	22	47	16	16,544
101	25	26	5	3,250
102	19	34	15	9,690
103	29	33	28	26,796
104	33	47	9	13,959
105	19	49	14	13,034
106	23	42	16	15,456
107	17	17	13	3,757
108	30	39	28	32,760
109	33	33	15	16,335
110	35	37	8	10,360
111	13	23	6	1,794
112	21	36	20	15,120
113	16	21	12	4,032
114	16	31	9	4,464
115	30	42	16	20,160
116	28	40	14	15,680
117	23	30	21	14,490
118	26	30	16	12,480
119	28	36	28	28,224
120	31	47	10	14,570
121	18	44	17	13,464
122	29	31	23	20,677
123	27	46	25	31,050

APPENDIX

F

TABLES - RELAXATION AND TIMES RESULTS FOR Q=1

Appendix containing tables with results for the relaxed models constraints comparison for $Q = 1$, mentioned in Subsection 3.3.5. Table 46 contains summary data with mean, median and standard deviation (SD) of the percentage improvement of the relaxed objective value of the two different constraints by item quantity, calculated as $\frac{Z^{Rel}_{TMKa2} - Z^{Rel}_{TMKa1}}{Z^{Rel}_{TMKa2}} \cdot 100$. Tables 47 and 48, contains summary of runtimes for mix-integer models, respectively Z_{TMKa} [Tsai, Malstrom and Kuo (1993)] and for Z_{CLSwr} [Chen, Lee and Shen (1995)].

Table 46 – Percentage difference between linear relaxations ($Q = 1$).

	Improvement		
Items	Mean	Median	SD
1	0.00%	0.00%	0.00%
2	0.13%	0.11%	0.09%
3	0.17%	0.16%	0.10%
4	0.20%	0.19%	0.10%
5	0.27%	0.27%	0.12%
6	0.25%	0.25%	0.09%
7	0.29%	0.30%	0.08%
8	0.29%	0.26%	0.11%
9	0.30%	0.30%	0.10%
10	0.33%	0.32%	0.11%

Table 47 – Run times for the two formulations of Z_{TMKa} (without rotation) by the number of items considering AMB box set and $Q = 1$.

	Z_{TMKa1}				Z_{TMKa2}				
Items	Mean	Median	SD	Total	Mean	Median	SD	Total	Instances
1	<1	<1	<1	2	<1	<1	<1	2	251
2	<1	<1	<1	5	<1	<1	<1	6	380
3	<1	<1	<1	3	<1	<1	<1	3	155
4	<1	<1	<1	2	<1	<1	<1	2	71
5	<1	<1	<1	1	<1	<1	<1	1	18
6	<1	<1	<1	2	<1	<1	<1	2	25
7	<1	<1	<1	4	<1	<1	<1	4	31
8	<1	<1	<1	7	<1	<1	<1	7	31
9	6	<1	15	90	5	<1	13	76	16
10	4	1	7	87	3	1	4	62	22
Total	1	<1	2	201	1	<1	2	165	1000

Table 48 – Run times for the two formulations of Z_{CSLwr} (without rotation) by the number of items considering AMB box set and $Q = 1$.

	Z_{CLSwr1}				Z_{CLSwr2}				
Items	Mean	Median	SD	Total	Mean	Median	SD	Total	Instances
1	0	0	0	2	0	0	0	2	251
2	0	0	0	4	0	0	0	5	380
3	0	0	0	2	0	0	0	3	155
4	0	0	0	2	0	0	0	2	71
5	0	0	0	1	0	0	0	1	18
6	0	0	0	2	0	0	0	3	25
7	0	0	0	4	0	0	0	5	31
8	0	0	0	8	0	0	0	9	31
9	1	0	2	16	1	0	2	18	16
10	1	1	2	29	1	1	1	26	22
Total	0	0	0	70	0	0	0	73	1000

APPENDIX

G

ALGORITHMS - LOCAL SEARCH USING N2

Algorithms for local search with *Dimension Decrease (N2)* are presented here, as mentioned in Subsection 4.3.3. The main difference for Algorithm 10 is that the HB_t set is created with dimensions smaller than the explored box down to tier t. Algorithm 11 has an extra condition that requires the tested box volume to be less than the current solution box (*currBox*) volume. Last, Algorithm 12 main difference is that orders from the current explored partition have to be packed either into the new box or into a larger already existing box (lines 25 - 34). This occurs because when a box's dimensions are reduced, some orders will not fit, so they must be packed into larger partitions.

Algorithm 10 – Function to create a HB_t set of boxes from HB, with smaller boxes.

```
 1: Inputs:
 2: box as $(x,y,z)$ coordinates for current box
 3: $HBx, HBy, HBz$ as vector of Herz point for $x, y$ and $z$ axis, respectively
 4: $t$ as tier size
 5: function CREATETIERDOWN(box, HBx, HBy, HBz,t)
 6:     $HBx_t \leftarrow t$ coordinates from $HBx$ that are smaller than the $x$ coordinate from box starting from the last
 7:     $HBy_t \leftarrow t$ coordinates from $HBy$ that are smaller than the $y$ coordinate from box starting from the last
 8:     $HBz_t \leftarrow t$ coordinates from $HBz$ that are smaller than the $z$ coordinate from box starting from the last
 9:     $HB_t \leftarrow$ boxes combining all dimensions from $(HBx_t, HBy_t, HBz_t)$ respecting $(L \geq W \geq H)$
10:     return $HB_t$
11: end function
```

Algorithm 11 – Skip boxes for that do not yield better results or are infeasible, comparing with the order.

```
1: Inputs:
2: order as an order with its items dimensions and volume
3: newBox as $(x, y, z)$ coordinates of neighbour box
4: currBox as $(x, y, z)$ coordinates of current box solution
5: function PREPACKINGN2(order, newBox, currBox)
6:     // Test newBox feasibility
7:     // Ensure that smallest item dimensions fits the box
8:     $itL \leftarrow$ largest item dimension from order
9:     $nboxL \leftarrow$ largest dimension form newBox
10:    if $nboxL \leq itL$ then
11:        return $False$
12:    end if
13:    // Ensure that the order volume is smaller than box volume
14:    if $volume(order) \leq volume(newBox)$ then
15:        return $True$
16:    end if
17:    // Ensure that the new box volume is smaller than the current box volume
18:    if $volume(newBox) \leq volume(currBox)$ then
19:        return $True$
20:    end if
21: end function
```

Algorithm 12 – Pseudocode for Local Search with N2

```
1: Inputs:
2: $\mathscr{O}$ as the set of orders
3: $solution$ as a list with the boxes of an initial solution
4: $HBx, HBy, HBz$ vector of Herz point for $x, y$ and $z$ axis, respectively
5: $t$ as the tier size
6: $m$ as the number of partitions
7: function DimensionReductionN2($\mathscr{O}$, $solution$, $HBx$, $HBy$, $HBz$, $t$, $m$)
8:     $bestSolution \leftarrow solution$
9:     $p \leftarrow m-1$
10:    while $partition \geq 1$ do
11:        $HB_t \leftarrow$ createTierDown($solution[n]$, $HBx$, $HBy$, $HBz$,$t$)
12:        for $b \leftarrow 1$ to $|HB_t|$ do
13:            $reduction \leftarrow 0$
14:            $increase \leftarrow 0$
15:            if skipBox($HB_t[b]$, $solution[p]$) then
16:                for $order \in \mathscr{O}_{p-1}$ do
17:                    if prepackingN2($order$, $HB_t[b]$, $solution[p-1]$) then
18:                        $box \leftarrow$ packingModel($order$, $HB_t[b]$)
19:                        if $box \neq \emptyset$ then
20:                            $reduction \leftarrow reduction//$
21:                            $+volume(solution[p-1]) - volume(HB_t[b])$
22:                        end if
23:                    end if
24:                end for
25:                for $order \in \mathscr{O}_p$ do
26:                    $box \leftarrow$ packingModel($order$, $HB_t[b] + solution[p+1, m]$)
27:                    if $box = HB_t[b]$ then
28:                        $reduction \leftarrow reduction//$
29:                        $+volume(solution[p-1]) - volume(HB_t[b])$
30:                    else
31:                        $increase \leftarrow increase//$
32:                        $+volume(box) - volume(solution[p])$
33:                    end if
34:                end for
35:                if $reduction > increase$ then
36:                    $bestSolution[partition] \leftarrow HB_t[b]$
37:                    Reorganize $\mathscr{O}$ for partitions $p$ and $p-1$
38:                    if $p = (m-1)$ then
39:                        $p \leftarrow m$
40:                    else
41:                        $p \leftarrow p+2$
42:                    end if
43:                    Break
44:                end if
45:            end if
46:            $p \leftarrow p-1$
47:        end for
48:    end while
49:    return $bestSolution$
50: end function
```

ICMC USP
SÃO CARLOS