Mathematical Models for a 3D Container Packing Problem

By

Valentina Ekui Ocloo
(10747417)

June 2019

THIS THESIS IS SUBMITTED TO THE UNIVERSITY OF GHANA, LEGON IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF MASTER OF PHILOSOPHY MATHEMATICS DEGREE.

DECLARATION

This thesis was written in the Department of Mathematics, University of Ghana, Legon from August 2018 to July 2019 in partial fulfilment of the requirements for the award of Master of Philosophy degree in Mathematics under the supervision of Professor Dr.Armin Fügenschuh and Professor Dr.Olivier M. Pamen of the University of Ghana.

I hereby declare that except where due acknowledgement is made, this work has never been presented wholly or in part for the award of a degree at the University of Ghana or any other University.

Signature:
Student: Valentina Ekui Ocloo
valentina@aims.edu.gh

We hereby declare that we supervised the work of our student Valentina Ekui Ocloo.

Signature:
Professor Dr. Armin Fügenschuh

Signature:
Professor Dr. Olivier M. Pamen

ABSTRACT

We address the single container packing problem of a company that has to serve its customers by first placing the products in boxes and then loading the boxes into a container. We approach the problem by developing and solving mixed-integer linear models. Our models consider geometric constraints which feature non-overlappping constraints, box orientation constraints, dimensionality constraints, relative packing position constraints and linearity constraints. We also consider extension of the models by integrating load balance as well as deviation of the center of gravity. The models have been tested on a large set of real instances involving up to 41 boxes and obtaining optimal solutions in most cases and very small gaps when optimality could not be proven.

DEDICATION

I dedicate this work to my husband, Collins Abeka.

ACKNOWLEDGEMENTS

Firstly, I would like to express my sincere gratitude to the Almighty God for helping me throughout this research.

I owe my special gratitude to Ghana Government Scholarship, AIMS-Ghana Research center and German Academic Exchange Service (DAAD) for their financial support in this research.

To my supervisors Prof. Dr. Armin Fügenschuh and Prof. Dr. Olivier M. Pamen for their continuous support, patience, motivation, and immense knowledge in this research. I could not have imagined having better advisors for this study.

I also owe my special thanks to Johannes and Fabian for their insightful ideas during this research. In addition, I acknowledge Rhoda Mahama for the support of all administrative needs at the AIMS-Ghana Research center.

I acknowledge the support and love of all research members at AIMS Research center particularly to Bernard, Cedric, Abigail and Elizabeth.

I also acknowledge the support and prayers of my parents and siblings, Priscilla Ocloo and Monaliza Ocloo in this research.

Contents

Chapter 1

Introduction

Logistics is the act of transporting materials, products or goods to customers. Most small businesses are interested in the design and production of their goods and services to their customer needs, however if these products are not able to reach the customers these business fail to satisfy the needs of their customers. Hence there is a need for a quick freight distribution to work easier for enterprises and to reduce cost of distribution of items purchased by customers. Recently evolution of logistic cost reveal that the share of transportation costs is increasing relative to inventory carrying cost [19].

The container packing problem (CPP) is a vital activity in transportation industries which needs to be investigated. With the use of transportation equipment like a container, it reduces the cost of transportation if the boxes are packed into the container in an optimal way. Thus an optimal packing of boxes into the container reduces the transportation cost and also increases balance or stability of load which is displayed in Figure 1.1. Therefore we define the container packing or 3D container packing problem as the optimal arrangement of goods and products into boxes or cartons and loaded into containers for delivery.

There have been a lot of studies conducted in the aspect of the container packing problem. Paquay et al. proposed a mixed-integer linear programming model that factors real-world encountered constraints such as stability, fragility of boxes, weight distribution and rotation of boxes in a three-dimensional container [18]. They tested the model on small instances.

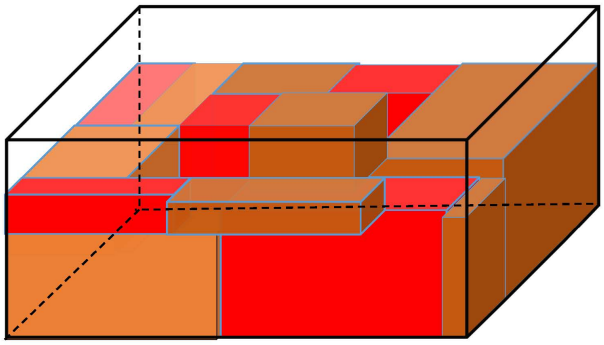


Figure 1.1: Optimal packing of boxes into the container.

Furthermore, a single container packing problem under practical constraints addressed by Lim et al. under US legal requirements stipulated in the California Vehicle Code (CVC) which is associated to truck axle weight distribution [14]. They proposed a heuristic solution method that entails a GRASP wall-building algorithm with linear integer programming models. Also they generated data from real cases and conducted experiments to show the effectiveness of their approach.

In addition, Kurpel et al. addresses multiple container loading problems that features placing rectangular boxes, non-overlapping and orthogonal packing inside containers with the goal to optimize a given objective function, that is either maximize the volume of the packed boxes or minimize the number of containers needed to load all available boxes [12]. They also propose new techniques to obtain bounds for these problems by improving an existing heuristic model. Furthermore, they considered box orientation, load stability and separation of boxes and tested their method on several existing benchmark sets. They proved optimality and improved the best known results for several instances.

Recently, a multi container packing problem of a company which serves customers by arranging the products on a pallet and then load the pallets into trucks was investigated by Alonso et al. where they developed integer linear models [2]. They consider three types of constraints in their models:

1) geometric constraints such that the pallets lie fully inside the trucks and must not overlap,

2) weight distribution and

3) position of the center of gravity of the cargo.

Furthermore, the model was extended so as to adapt to demands served to meet delivery dates over a set of time periods. In addition, a real instance made up of 44 trucks was used to test the model. In some instances optimal solutions were obtained and in other instances minimal gaps were obtained when optimality could not be proven.

1.1 Overview of Problem Statement

Our work considers household equipment which needs to be transported from a retailing shop to customers based on their orders. The products requested include televisions, laptops, refrigerators, phones and some cooking utensils. Since these products are to be delivered to customers, the products are loaded into boxes or cartons. The boxes are placed into a rectangular container which is then transported for distribution to customers.

1.2 Objective of the Study

The objectives of the study are as follows:

1. Design, model and implement a time effective packing method.
2. Maximize the volume of packed boxes in a large problem.
3. Improve the stability of load to avoid damage of products.
4. Reduce the cost of transportation by obtaining an optimal packing approach to the problem.

1.3 Organization of the Thesis

The following is the remaining outline of this thesis. Chapter 2 provides the basic definitions of linear programming and preliminaries of the methods in mixed-integer programming. We illustrate some examples using the exact method approach in mixed-integer programming.

Chapter 3 discusses the sets, parameters and decision variables required for mixed-integer programming. A mixed-integer programming model is formulated for a single container packing problem to maximize the volume of packed boxes in the container considering box orientation constraints, non-overlapping constraints, dimensionality constraints, relative packing position constraints and linear constraints.

Chapter 4 extends the model proposed in Chapter 3 and integrates weight distribution or load balance as well as deviation of the center of gravity into the model to explore the optimal height of the container for stability purposes. The objective is to minimize the height of the container such that the boxes will be balanced.

Chapter 5 discusses the results obtained using the model in Chapter 3 on small and large scale instances using an exact algorithm. Also we implement the model proposed by Józefowska et al. [11] and compare results obtained using similar data sets and an exact algorithm. We also implement the model in Chapter 4 and obtain results.

Chapter 6 concludes our research and provides future research direction.

Chapter 2

Basic Concepts and Solution Methods in Mixed-Integer Programming

This chapter presents basic definitions and terms in mixed-integer programming (MIP). We will also describe the methods used in solving mixed-integer programs and gives some examples of how the methods works. We begin by giving an introduction to a mixed-integer programming models.

2.1 Introduction to Mixed-Integer Programming

We provide some basic foundations in linear programming [7] and then introduce the concept of mixed-integer programming [20].

Definition 2.1.1. [7] A *linear program* (LP) is a mathematical optimization problem in which a linear objective function is to be maximized or minimize subject to a finite number of linear constraints (linear equations and/or linear inequalities).

Hence, the general form of a LP-Model is given as

$$\begin{aligned} &\max \quad \sum_{j=1}^{n} c_j x_j \quad \left[\text{or} \quad \min \quad \sum_{j=1}^{n} c_j x_j\right] \\ &\text{subject to} \quad \sum_{j=1}^{n} a_{ij} x_j \leq b_i, \ \forall \quad i = 1, 2, \ldots, m, \\ &x_j \geq 0, \ \forall \quad j = 1, 2, \ldots, n, \quad \text{(non-negativity constraints)} \end{aligned} \tag{2.1}$$

where c_j are the objective function coefficients (constant), x_j are the decision variables, a_{ij} are the technical coefficients (constant) and b_i are the resource endowments (constant). We refer to $x_1, x_2, \ldots, x_n$ as the unknowns or variables or activities. Furthermore, the LP model can be revised in the form of matrix notation as follows:

$$\begin{aligned} \max \quad & cx \\ \text{subject to} \quad & Ax \leq b \\ & x \geq 0, \end{aligned} \tag{2.2}$$

given that c is a row vector of the form $c = (c_1, \ldots, c_n) \in \mathbb{R}^n$, column vectors $x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$, $b = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}$ given that $x \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$, respectively, and $A = (a_{ij}) \in \mathbb{R}^{m \times n}$.

An optimal solution denoted x^* of (2.1) gives the values for the activities that optimize the objective function. When the variables satisfy all constraints of the problem we have a *feasible solution*. An *infeasible solution* occurs when there are values for the variables such that at least one constraint is violated.

Now, an optimal solution to a linear programming model may take fractional values because the decision variables are not constrained to take integer solutions. There are cases of problems where the fractional solutions can be allowed, but in some real applications these are not realistic solutions. For instance, a transportation problem of moving goods from warehouses to retailers with the aim of minimizing the transportation costs may have fractional solutions. In other cases such as assignment problems, packing problems, and others, fractional solutions are not realistic, hence there is a need to constrain some or all of the decision variables to take integer solutions. We consider the optimization problem:

$$\begin{aligned} \max \quad & c_1x_1 + c_2x_2 \\ \text{subject to} \quad & A_1x_1 + A_2x_2 \leq b \\ & x_1 \geq 0, \quad [x_1 \in \mathbb{Z}] \\ & x_2 \geq 0. \end{aligned} \tag{2.3}$$

The aforementioned optimization problem is a *mixed-integer linear program* and for brevity, we denote it as a MILP model. Also by ignoring integer conditions (or

integrality restrictions) in (2.3) we have a *LP relaxation*. In addition, if all the x and y variables are integers in (2.3) then we have an *integer program* (IP). There is also a special case where the variables must either take 0 or 1 values. These problems are called *binary integer program* and can be used to model "yes" or "no" decisions, for instance whether to pack a box into a container or not to pack a box into a container. In most cases, real-world problems involve a mix of continuous and integer-value decision variables, hence we will base our model on a mixed-integer linear program since we are not dealing with non-linear functions in this work.

Definition 2.1.2. [20] A *mixed-integer programming (MIP)* problem is an optimization problem where some variables take non-integer optimal solutions while other variables are restricted to take integer value solutions.

2.2 Methods for Solving Mixed-Integer Programming Problems

The following are currently the most successful methods used in solving mixed-integer programming models:

1. Exact algorithm: In this method we obtain a guaranteed optimal solution, but it may take an exponential number of iterations to obtain it. Examples of the exact algorithms are cutting plane method [8], branch-and-bound method [13], branch-and-cut method [17] and dynamic programming [3].

2. Heuristics algorithm: This method finds a suboptimal solution but its quality of solution is not guaranteed. Meanwhile, the running time is not necessarily polynomial. Empirical evidence suggests that these algorithms tend to find a good solution quickly. Examples are the greedy method [6], local search algorithms [1] and others.

3. Approximation algorithms: Approximation algorithms [10] involve a polynomial time suboptimal solution obtained together with a bound on the degree of suboptimality.

In this work we will use an exact algorithm for solving our problem.

2.2.1 Cutting Plane Algorithm

We introduce an algorithmic principle that is at the heart of state of the art software for mixed-integer programming. The cutting plane algorithm was the first method introduced by Gomory [8] in 1958 for solving IP and MIP problems.

Consider the problem:

$$\text{MIP}: \quad \max \ \{c_1x_1 + c_2x_2 : (x_1, x_2) \in G\},$$

where $G := \{(x_1, x_2) \in \mathbb{Z}^n_+ \times \mathbb{R}^n_+ : A_1x_1 + A_2x_2 \leq b\}$. We denote by (x_1^*, x_2^*) and z^* an optimal solution and value of MIP respectively. We say (x_1^0, x_2^0) and z^0 is the optimal solution and value of the LP relaxation obtained as

$$\max \ \{c_1x_1 + c_2x_2 : (x_1, x_2) \in P\}, \tag{2.4}$$

where P is the LP relaxation (as the LP problem obtained when we allow non-integer solutions) of G. In addition, suppose that (x_1^0, x_2^0) is a basic optimal solution of (2.4) which we compute using the simplex algorithm. Now when $G \subseteq P$, it follows that $z^* \leq z^0$. In addition, if (x_1^0, x_2^0) is an integral vector then $(x_1^0, x_2^0) \in G$ and therefore $z^* = z^0$; in this case the MIP is solved. However, if (x_1^0, x_2^0) is not an integral vector then we need to find a valid inequality that satisfies every point in G. There are many valid inequalities developed for linear integer and mixed integer sets some of which are Gomory mixed-integer cuts [21], Chvatal-Gomory cuts [21], Cover inequalities [9], Mixed integer rounding inequalities [15] and many more. The quality of the cuts generated by separation algorithm is mostly related to the time spent on the cut generation.

In this algorithm, we will use the simplex method to solve the LP relaxation and use Chvatal-Gomory cuts to generate valid cuts as follows:

1. Consider the optimization problem in (2.3).

2. Solve the relaxation LP to optimality (by the simplex method):

$$\begin{aligned} x_{j+1} &= b_i^* - \sum_{j \in \mathbb{N}} a_{ij}^* x_j, \quad i \in \mathbb{N} \\ z &= z^* + \sum_{j \in \mathbb{N}} c_j^* x_j, \end{aligned}$$

where z^* is the optimal value of the LP relaxation.

3 If basic optimal solution to LP relaxation is non integer then there exits some row where $b_i^* \notin \mathbb{Z}$. The Chvatal-Gomory cut applied to this row is:

$$x_{j+1} + \sum_{j \in \mathbb{N}} \lfloor a_{ij}^* \rfloor x_j \leq \lfloor b_i^* \rfloor,$$

where $\lfloor a_{ij}^* \rfloor$ is $\max \{m \in \mathbb{Z} |\ m \leq a_{ij}^*\}$, and similarly for $\lfloor b_i^* \rfloor$.

4. Eliminating $x_{j+1} = b_i^* - \sum_{j \in \mathbb{N}} a_{ij}^* x_j$ in the Chvatal-Gomory cut we obtain:

$$\begin{aligned} \sum_{j \in \mathbb{N}} \left(a_{ij}^* - \lfloor a_{ij}^* \rfloor\right) x_j &\geq b_i^* - \lfloor b_i^* \rfloor, \\ \sum_{j \in \mathbb{N}} f_{ij} x_j &\geq f_i, \end{aligned}$$

where $0 \leq f_{ij} \leq 1$ and $0 < f_i < 1$.

5. We add a slack variable which is the new constraint as

$$s = -f_i + \sum_{j \in \mathbb{N}} f_{ij} x_j,$$

to the problem and solve using the simplex method until all b_i^* are integers.

Example 2.2.2. Let us consider the following mixed-integer programming problem and solve it using the cutting plane method:

$$\begin{aligned} \max \quad & x_2 \\ \text{subject to} \quad & 3x_1 + 2x_2 \leq 6 \\ & -3x_1 + 2x_2 \leq 0 \\ & x_1 \geq 0, \quad [x_1 \in \mathbb{Z}] \\ & x_2 \geq 0. \end{aligned} \tag{2.5}$$

We apply the simplex method to solve the LP relaxation problem by introducing x_3 and x_4 as the slack variables and we obtain the first dictionary as

$$\begin{aligned} x_3 &= 6 - 3x_1 - 2x_2 \\ x_4 &= \quad 3x_1 - 2x_2 \\ z &= \quad\quad\ x_2. \end{aligned} \tag{2.6}$$

Next x_2 is entering and x_4 is leaving and we have the second dictionary as

$$\begin{aligned} x_2 &= \frac{3}{2}x_1 - \frac{1}{2}x_4 \\ x_3 &= 6 - 6x_1 + x_4 \\ z &= \frac{3}{2}x_1 - \frac{1}{2}x_4. \end{aligned} \tag{2.7}$$

Furthermore, x_1 is entering and x_3 is leaving and we obtain the third dictionary as

$$\begin{aligned} x_1 &= 1 - \frac{1}{6}x_3 + \frac{1}{6}x_4 \\ x_2 &= \frac{3}{2} - \frac{1}{4}x_3 - \frac{1}{4}x_4 \\ z &= \frac{3}{2} - \frac{1}{4}x_3 - \frac{1}{4}x_4. \end{aligned} \tag{2.8}$$

At this point we have obtained an optimal solution for the LP relaxation problem and from the integrality constraint from (2.5) we have also obtain the optimal solution of the MIP problem where $x_1^* = 1, x_2^* = 3/2$ and $z^* = 3/2$. Let us remark that in the case both x_1 and x_2 are constrained to take integer value then we have not obtained an optimal solution, because x_2 is not an integer. We show how to obtain integer solutions for the variables by the cutting plane method. We first generate a constraint row cut from (2.8), where we take the constraint which has the fractional solution as

$$x_2 + \frac{1}{4}x_3 + \frac{1}{4}x_4 \leq \frac{3}{2} \tag{2.9}$$

and for the valid inequality we have

$$x_2 + \left\lfloor \frac{1}{4}x_3 \right\rfloor + \left\lfloor \frac{1}{4}x_4 \right\rfloor \leq \left\lfloor \frac{3}{2} \right\rfloor. \tag{2.10}$$

Now, subtracting (2.9) from (2.10) we have

$$-\frac{1}{4}x_3 - \frac{1}{4}x_4 \leq -\frac{1}{2}. \tag{2.11}$$

The first cutting plane is obtained by substituting the slack variables x_3 and x_4 into (2.11) and we obtain $x_2 \leq 1$. Then by adding a new slack variable to (2.11) we have an additional constraint as $x_5 = -\frac{1}{2} + \frac{1}{4}x_3 + \frac{1}{4}x_4$ which is added to the last dictionary.

Thus we have

$$\begin{aligned} x_1 &= 1 - \frac{1}{6}x_3 + \frac{1}{6}x_4 \\ x_2 &= \frac{3}{2} - \frac{1}{4}x_3 - \frac{1}{4}x_4 \\ x_5 &= -\frac{1}{2} + \frac{1}{4}x_3 + \frac{1}{4}x_4 \\ z &= \frac{3}{2} - \frac{1}{4}x_3 - \frac{1}{4}x_4. \end{aligned} \tag{2.12}$$

Re-optimizing using the dual simplex method where x_3 is entering and x_5 is leaving we have

$$\begin{aligned} x_1 &= \frac{2}{3} - \frac{2}{3}x_5 + \frac{1}{3}x_4 \\ x_2 &= 1 - x_5 \\ x_3 &= 2 + 4x_5 - x_4 \\ z &= 1 - x_5. \end{aligned} \tag{2.13}$$

A new fractional solution has been found which $x_1^* = 2/3, x_2^* = 1$ and $z^* = 1$ and so we generate another constraint row cut from only (2.13) which has the fractional part as

$$x_1 + \frac{2}{3}x_5 - \frac{1}{3}x_4 \leq \frac{2}{3} \tag{2.14}$$

and for the valid inequality we have

$$x_1 + \left\lfloor \frac{2}{3}x_5 \right\rfloor + \left\lfloor \frac{-1}{3}x_4 \right\rfloor \leq \left\lfloor \frac{2}{3} \right\rfloor. \tag{2.15}$$

At this point we subtract (2.14) from (2.15) which gives

$$-\frac{2}{3}x_5 - \frac{2}{3}x_4 \leq -\frac{2}{3} \tag{2.16}$$

and we obtain the second cutting plane as $x_1 - x_2 \geq 0$. Also repeating the same

procedure we have the next dictionary as

$$\begin{aligned} x_1 &= \frac{2}{3} - \frac{2}{3}x_5 + \frac{1}{3}x_4 \\ x_2 &= 1 - x_5 \\ x_3 &= 2 + 4x_5 - x_4 \\ x_6 &= -\frac{2}{3} + \frac{2}{3}x_5 + \frac{2}{3}x_4 \\ z &= 1 - x_5. \end{aligned} \tag{2.17}$$

Re-optimizing using the dual simplex method we have the new dictionary in which x_4 is entering and x_6 leaving as shown below:

$$\begin{aligned} x_1 &= 1 - x_5 + \frac{1}{2}x_6 \\ x_2 &= 1 - x_5 \\ x_3 &= 1 + 5x_5 - \frac{3}{2}x_6 \\ x_4 &= 1 - x_5 + \frac{2}{3}x_6 \\ z &= 1 - x_5. \end{aligned} \tag{2.18}$$

Finally the optimal solution is integral thus $x_1^* = 1, x_2^* = 1$ and $z^* = 1$.

2.2.3 Branch-and-Bound Algorithm

Land and Doig [13] were the first to propose the branch and bound method in 1960. The branch-and-bound method also uses the simplex algorithm along with an iterative process which follows a decision tree to solve an integer programming problem. Thus the solution approach involves partitions of the set of all the feasible solutions into smaller subproblems and solving systemically until the best solution is found. The branch-and-bound method uses a tree diagram of nodes and branches to organize the solution partitioning. The initial node of the branch-and-bound is the LP relaxation.

Definition 2.2.4. [20] A subproblem is *fathomed* if any of the following conditions are satisfied:

- The relaxation of the subproblem has an optimal solution with $z \leq z^*$, where z^* is the current best solution;
- The relaxation of the subproblem has no feasible solution;

- The relaxation of the subproblem has an optimal solution that has all integers.

The following are the steps needed in the branch-and-bound algorithm:

1) Consider the LP relaxation in (2.3).

2) If the optimal solution of the LP relaxation are all integers then an optimal solution is obtained.

3) Else, let x_j be the integer variable and $x_j^* \in \mathbb{R}$ the current fractional solution, then $x_j \leq \lfloor x_j^* \rfloor$ is the bounding inequality (only for the left branch) and in the right branch the new inequality is $x_j \geq \lceil x_j^* \rceil$.

4) Repeat until a global optimal IP or MIP solution is found whenever the iteration results in an upper bound that equals the LP.

5) Otherwise, if a particular branch is infeasible (or fathomed) then discard and only branch on a feasible node which will result in an upper bound that equals the LP.

Example 2.2.5. Consider the optimization problem in Example 2.2.2 and assume also that both variables are integers and use the branch-and-bound method to solve it.

From the previous iteration using the simplex method we obtained $x_1^* = 1, x_2^* = 3/2$ and $z^* = 3/2$ which is an optimal solution for the MIP problem given and not optimal for the IP problem, hence we apply the branch-and-bound method to obtain the IP optimal solution. Since $x_2 = 3/2$ we branch on x_2 where the left branch is $x_2 \leq 1$ and the right branch is $x_2 \geq 2$ are our new constraints added to the first dictionary and are solved separately using the simplex method. Starting with the left branch we have the initial dictionary as:

$$\begin{aligned} x_3 &= 6 - 3x_1 - 2x_2 \\ x_4 &= \quad 3x_1 - 2x_2 \\ x_5 &= 1 \quad\quad - x_2 \\ z &= \quad\quad\quad x_2. \end{aligned} \tag{2.19}$$

The entering variable is x_2 and leaving variable is x_5 and we have the next dictionary

as

$$\begin{aligned} x_2 &= 1 \quad - x_5 \\ x_3 &= 4 \quad - 3x_1 - 2x_5 \\ x_4 &= -2 + 3x_1 + 2x_5 \\ z &= 1 \quad - x_5. \end{aligned} \tag{2.20}$$

Using the dual simplex method we have x_1 entering and x_4 leaving as

$$\begin{aligned} x_1 &= \frac{2}{3} + \frac{1}{3}x_4 + \frac{2}{3}x_5 \\ x_2 &= 1 \quad - x_5 \\ x_3 &= 2 \quad - x_4 \\ z &= 1 \quad - x_5. \end{aligned} \tag{2.21}$$

From this dictionary we have an optimal solution where $x_1 = 2/3, x_2 = 1$ and $z = 1$. Now we solve the right branch by adding a new constraint ($x_2 \geq 2$) to (2.6) which gives the initial dictionary

$$\begin{aligned} x_3 &= 6 - 3x_1 - 2x_2 \\ x_4 &= \quad 3x_1 - 2x_2 \\ x_5 &= 2 \quad - x_2 \\ z &= \quad x_2. \end{aligned} \tag{2.22}$$

Solving problem (2.22) results to an infeasible solution. Next we branch again on the node which had a feasible solution (left branch). Hence, we have additional two sub-problems where their constraints are $x_1 \leq 0$ (left-left branch) and $x_1 \geq 1$ (right-left branch) and we solve them differently. For $x_1 \leq 0$, we have $x_1 = 0, x_2 = 0$ and $z = 0$ and for $x_1 \geq 1$, we have $x_1 = 1, x_2 = 1$ and $z = 1$. Finally, we obtain the optimal integer solution where $x_1^* = 1, x_2^* = 1$ and $z^* = 1$.

2.2.6 Branch-and-Cut Algorithm

Padberg and Rinaldi [17] proposed the branch-and-cut algorithm in 1987. The branch-and-cut method involves running a branch-and-bound and using cutting planes to tighten the LP relaxations. On the other hand, if bounds are not used to tighten the LP relaxation then we have cut-and-branch algorithm. The cutting plane method is fast, but unreliable and branch-and-bound method is reliable but slow. Therefore,

the branch-and-cut method combines the advantages from these two methods and improves the defects in both algorithms. It has proven to be a very successful approach for solving a wide variety of integer programming problems. We can solve the MIP problem by taking some cutting planes before applying the branch-and-bound to the resulting system. The branch-and-cut method is not only reliable, but also faster than branch-and-bound method alone. The procedure of the branch-and-cut algorithm is as follows:

Input: An MIP problem in (2.3).
Output: An optimal solution $x^* \in X_{MIP}$ and its objective value $z^* = cx^*$ or if $X_{MIP} = \emptyset$, denote $z^* = +\infty$.

1. Initialize a queue list of active problem $Q := \{P_{LP}\}$, $z^* := +\infty$.
2. If $Q = \emptyset$, exit by returning the optimal solution x^* with value z^*.
3. Select a problem P from the queue Q.
4. Solve the linear program $z(P) = \max\{cx : x \in P\}$ with optimal solution $\bar{x}(P)$.
5 If cuts should be generated, go to step 6, otherwise go to step 7.
6. Generate a cut, $Tx \geq u$, where $T \in \mathbb{R}^{k\times n}$, $u \in \mathbb{R}^k$, and k is the generated cuts. Add them to the formulation $P := P \cup \{x : Tx \geq u\}$. Go to step 4.
7. If $z(P) \geq z^*$, go to step 3.
8. If $\bar{x}(P) \in X_{\text{MIP}}$, update the incumbent $x^* = \bar{x}(P)$ and $z^* = z(P)$.
9. Branch to split P into subproblems and add them to Q and return to step 3.

Let us remark that the unsolved pending subproblems are called the active problems which are kept in a pool denoted by Q.

Chapter 3

A Mixed-Integer Programming Model for Single Container Packing

In this chapter, we present the problem formulation and give the mathematical model description of the single container packing problem.

3.1 Problem Formulation

We consider household equipment needed to be transported from a retailing shop to customers based on their orders. The products requested include televisions, laptops, refrigerators, phones, and cooking utensils. Since these product are to be delivered to customers, the products are packed into boxes or cartons. The boxes are placed into a rectangular container which is transported by a vehicle for distribution to customers so as to we maximize the volume of the packed boxes and thus reduce the cost of transportation. In the next section, we model this problem as a 0-1 mixed-integer programming (MIP) model of a single container packing problem (SCPP).

3.2 Mathematical Model Description

The following assumptions are needed in the packing problem:

Assumption 1. *Any two boxes packed cannot overlap within the container.*

Assumption 2. *The placement of boxes must be orthogonal, i.e. the sides or edges of the boxes have to be parallel to the container side.*

Assumption 3. *Each box can freely be rotated in the container.*

Assumption 4. *Boxes are allowed to be placed on top of each other without restrictions.*

3.2.1 Sets and Parameters of the Model

The following sets describe an instance of the problem. A set of rectangular boxes $B := \{1, \ldots, n\}$ and a set of orientations $O := \{1, \ldots, 6\}$ are given.

We give the following data set to the problem: the number of boxes to pack, the size of each box base on its six orientations and the size of the container. These describe the technical packing of the boxes into the container. The SCPP technical parameters are given by the following:

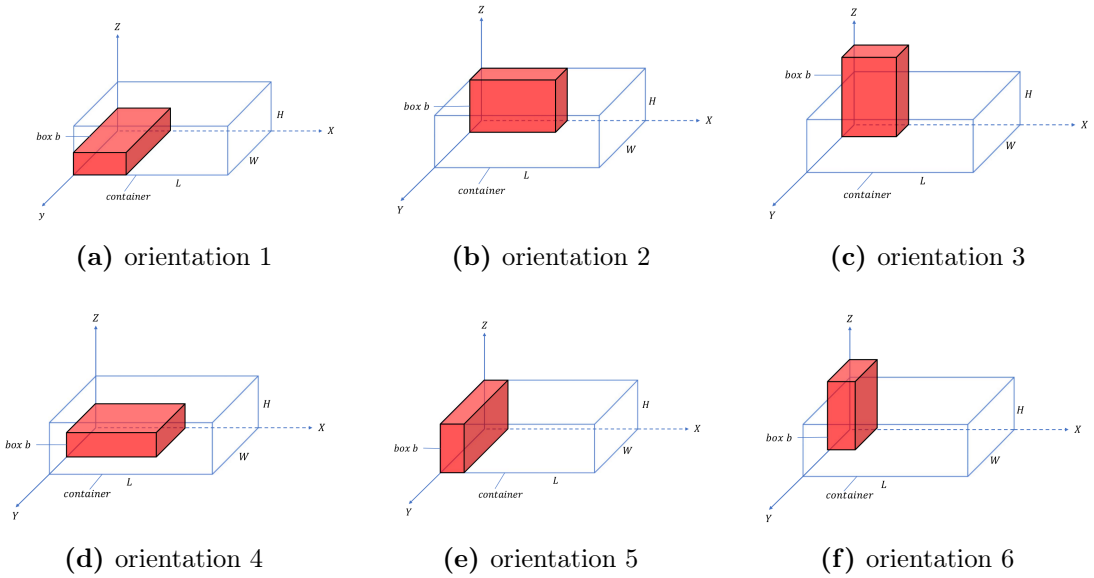
1. The length of the container is given as $L \in \mathbb{R}_+$, the width of the container is represented as $W \in \mathbb{R}_+$ and the height of the container is represented as $H \in \mathbb{R}_+$.

2. We assume that the boxes can be rotated freely in the container. For this, we define the length of each box b rotated with orientation o as denoted $l_{b,o} \in \mathbb{R}_+$, similarly for the width and height of each box b rotated with orientations o denoted as $w_{b,o} \in \mathbb{R}_+$ and $h_{b,o} \in \mathbb{R}_+$ respectively. Figure 3.1 represent six possible ways that a box can rotate in the container.

3.2.2 Decision Variables of the Model

We construct the following decision variables below and without loss of generality, the axes of the coordinate system are assumed to be placed so that the length of the container L (respectively width W, height H) lies on the X-axis (respectively Y-axis, Z-axis). Thus, the origin of the coordinate system lies on the bottom-left corner (BLC) of the box. Hence, we define a continuous variable location of the BLC of the box b as $(X_b, Y_b, Z_b) \in \mathbb{R}_+, \forall\ b \in B$. Next we define a binary variable, $\alpha_{b,o}$ indicates the orientation of each box as

$$\alpha_{b,o} = \begin{cases} 1, & \text{if box } b \text{ is rotated with orientation } o; \\ 0, & \text{otherwise} \end{cases}$$



(a) orientation 1 **(b)** orientation 2 **(c)** orientation 3

(d) orientation 4 **(e)** orientation 5 **(f)** orientation 6

Figure 3.1: Six possible orientations of a box.

For example the variable is equal to $\alpha_{1,1} = 1$ assuming that box 1 is placed into the container in the first orientation as shown in Figure 3.1a. We define a continuous variable that gives the orientation of the placement of box b in the x-axis. That is,

$$x_b = \sum_{o \in O} l_{b,o} \cdot \alpha_{b,o}, \quad \forall \quad b \in B. \tag{3.1}$$

Similarly, we denote a variable that gives the orientation of the placement of the box b in the y-axis. Thus,

$$y_b = \sum_{o \in O} w_{b,o} \cdot \alpha_{b,o}, \quad \forall \quad b \in B. \tag{3.2}$$

Lastly, in the z-axis of the orientation placement of box b is denoted as

$$z_b = \sum_{o \in O} h_{b,o} \cdot \alpha_{b,o}, \quad \forall \quad b \in B. \tag{3.3}$$

Since the boxes can be rotated orthogonally, the variables x_b, y_b and z_b are introduced to describe its length, width and height orientations of each box along the axis.

Example 3.2.3. Suppose that box 2 has dimensions of 17.40mm long, 26.88mm wide and 39.99mm high where the dimensions of the container is 63.99mm long, 42.37mm wide and 27.44mm high. If box 2 is placed into the container of orientation 4 then $x_b = l_{2,1} \cdot \alpha_{2,1} = 39.99(1) = 39.99$mm, $y_b = w_{2,1} \cdot \alpha_{2,1} = 26.88(1) = 26.88$mm and $z_b = h_{2,1} \cdot \alpha_{2,1} = 17.40(1) = 17.40$mm. Assuming box 2 is placed in orientation 3 or

6, this will be an infeasible packing of the box since the height of the box is higher than the container.

Also, we define a binary variable that determines if a box is packed into the container given as:

$$\Omega_b = \begin{cases} 1, & \text{if box } b \text{ is packed;} \\ 0, & \text{otherwise.} \end{cases}$$

Now, the following binary variables describe the relative packing position of box b and k inside the container. That is,

$$x'_{b,k} = \begin{cases} 1, & \text{if box } b \text{ is placed to the left or right of box } k; \\ 0, & \text{otherwise.} \end{cases}$$

$$y'_{b,k} = \begin{cases} 1, & \text{if box } b \text{ is placed in front or behind of box } k; \\ 0, & \text{otherwise.} \end{cases}$$

$$z'_{b,k} = \begin{cases} 1, & \text{if box } b \text{ is placed above or below of box } k; \\ 0, & \text{otherwise.} \end{cases}$$

To guarantee that there is no overlapping of two boxes, we need to know the relative position of two boxes and these variables describe all the situations. For example, if the box b is on the right of box k it means that box k is on the left of the box b, then $x'_{b,k} = 1$, $y'_{b,k} = 0$ and $z'_{b,k} = 0$ as shown below:

$$x'_{1,2} = \begin{array}{c} \\ 1 \\ 2 \end{array}\!\!\begin{array}{c} \begin{array}{cc} 1 & 2 \end{array} \\ \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \end{array}, \quad y'_{1,2} = \begin{array}{c} \\ 1 \\ 2 \end{array}\!\!\begin{array}{c} \begin{array}{cc} 1 & 2 \end{array} \\ \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \end{array} \quad \text{and} \quad z'_{1,2} = \begin{array}{c} \\ 1 \\ 2 \end{array}\!\!\begin{array}{c} \begin{array}{cc} 1 & 2 \end{array} \\ \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \end{array}.$$

Even if the definition of $z'_{b,k}$ is the same as $x'_{b,k}$ and $y'_{b,k}$, we will see in the next subsection that these variables are not fully determined. Indeed, part of the definition will be ensured by the constraints: if $z'_{b,k} = 1$ then we are sure that $Z_b + z_b \leq Z_k$. If $z'_{b,k} = 0$, then we have no information.

3.2.4 Constraints of the Model

Some constraints in Bortfeldt and Wäscher [4] are called basic constraints (the geometric ones) and others are specific constraints. The specific constraints comprise stability, fragility, weight distribution or load balance, load priorities of the boxes placed in the container and others.

Some Geometric Constraints

The following are the geometric constraints of the model:

$$\sum_{o \in O} \alpha_{b,o} = \Omega_b, \qquad \forall\, b \in B. \tag{3.4}$$

$$X_b + x_b \leq L \cdot \Omega_b, \quad \forall\, b \in B. \tag{3.5a}$$

$$Y_b + y_b \leq W \cdot \Omega_b, \quad \forall\, b \in B. \tag{3.5b}$$

$$Z_b + z_b \leq H \cdot \Omega_b, \quad \forall\, b \in B. \tag{3.5c}$$

The constraint (3.4) ensures that each box packed can rotate orthogonally in exactly one orientation at a time in the container. Constraints (3.5) ensure that all boxes are packed within the physical dimension of the container known as the dimensional bounds constraints and thus does not exceed the container size. The following constraints guarantee non-overlapping of two boxes and they cannot occupy the same portion of the space in the container:

$$x'_{b,k} + x'_{k,b} \leq \Omega_b, \quad \forall\, b < k \in B. \tag{3.6a}$$

$$y'_{b,k} + y'_{k,b} \leq \Omega_b, \quad \forall\, b < k \in B. \tag{3.6b}$$

$$z'_{b,k} + z'_{k,b} \leq \Omega_b, \quad \forall\, b < k \in B. \tag{3.6c}$$

$$x'_{b,k} + x'_{k,b} + y'_{b,k} + y'_{k,b} + z'_{b,k} + z'_{k,b} \geq \Omega_b + \Omega_k - 1, \quad \forall\, b < k \in B. \tag{3.7}$$

$$X_b + x_b \leq X_k + L \cdot (1 - x'_{b,k}), \quad \forall\, b \neq k \in B. \tag{3.8a}$$

$$Y_b + y_b \leq Y_k + W \cdot (1 - y'_{b,k}), \quad \forall\, b \neq k \in B. \tag{3.8b}$$

$$Z_b + z_b \leq Z_k + H \cdot (1 - z'_{b,k}), \quad \forall\, b \neq k \in B. \tag{3.8c}$$

When the variables $x'_{b,k}, x'_{k,b}, y'_{b,k}, y'_{k,b}, z'_{b,k}$ or $z'_{k,b}$ equal to 1, the two boxes b and k must not overlap along any of the coordinates. To avoid two boxes occupying the same portion of space, it is sufficient for no overlapping along at least one of the relative packing position. Thus, at most one of these variables must be equal 1 and this describes the linearity constraints in (3.6). Similarly in constraints (3.7) ensures that each pair b, k it must hold at least one of the six relative packing position: b is left to k or b is right to k or b is in front of k or b is behind k or b is above k or b is below k which is given as the relative constraint. Constraints (3.8) help to fully determine these variables $x'_{b,k}, y'_{b,k}$ and $z'_{b,k}$ to ensure that there is no overlapping of two boxes in the container.

3.2.5 Objective of the Model

There are a lot of objectives which have been addressed in the container packing problem. Wu et al. formulate a MIP model of a single container with the goal of minimizing the height of the container [22]. Although various objectives may be considered, the one which is often relevant in logistic applications is to maximize the volume of the packed boxes. This will help to know the full utilization of the packed boxes in the container so as to reduce cost of transporting the products. Hence, our primary goal is to maximize the volume of the packed boxes. This reflects the following objective function:

$$\max \sum_{b \in B} (l_{b,1} \cdot w_{b,1} \cdot h_{b,1}) \Omega_b. \tag{3.9}$$

Other MIP models of the basic container packing problem have been proposed by Paquay et al. [18], Chen et al. [5], Józefowska et al. [11] and many others.

Chapter 4

Extended MIP Model of the Single Container

Although the single container packing problem has been modelled, there are still deficiencies in the single container problem since it does not really conform into the real-life issues concerning the packing problem. In this chapter we will extend the single container packing problem to adapt to real-world applications in order to evaluate the quality of our results.

4.1 Practical Issues for Container Packing

There are several constraints encountered during container loading. The following are some typical constraints encountered in CPP:

1. Weight Distribution or Load Balance Constraints
2. Fragility Constraints
3. Priority Packing Constraints

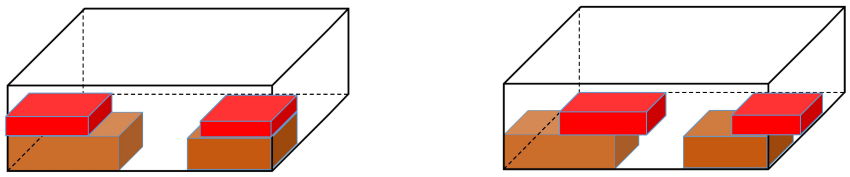
In this work we will focus on weight distribution or load balance constraints.

4.1.1 Weight Distribution Constraints under Single CPP

Moon et al. [16] propose an algorithm with balance constraints and trade off between weight balance and volume utilization and also propose a MIP model that groups upper and lower bounds for a 3D-CPP. The balance of load in the container is very important in CPP. When the load in the container is not well balanced (stable), the container shifts while it is moved. Unstable packing happens when there is uneven distribution of weight. This leads to breakages and damages of the products in the boxes. To achieve an even weight distribution, we need to consider the center of gravity of the load near to the geometrical center of the container. We assume that the weight (mass) of each box is evenly distributed. Usually, the center of gravity of the container must lie around a specific area. In the horizontal form of the container, the area is defined around the geometric midpoint of the container floor. However, vertically the center of gravity lies below a given level. Given boxes of different masses placed in a container, we define the center of gravity of boxes along the x axis as

$$x_{CG} = \frac{\sum_b m_b x_b}{\sum_b m_b},$$

where x_b is the x-coordinate of the box b and m_b its mass. Figure 4.1 displays an example of a balanced load and an unbalanced load.



(a) stable packed boxes **(b)** unstable packed boxes

Figure 4.1: Example of balanced packing.

Now, in the extended Single Container Packing Problem (SCPP) we assume the following:

a. All boxes are packed into the container, hence Ω_b is set to 1.

b. All other sets, parameters, variables and constraints are maintained.

We add new parameters, new decision variables and new constraints to the model and propose a new objective to the problem. The following new parameters are added:

- We introduce m_b which is the mass of each box.
- We add
$$\text{totalMass} = \sum_{b \in B} m_b$$
that refers to the total mass of all the boxes.
- We add $\theta \in [0, 1)$ which is a value that balances the contribution height versus the deviation of the x, y-center of mass from the middle of the container.

Next we propose the following decision variables to the problem. We introduce decision variables for the x-axis center of mass of the box as

$$\xi_b = X_b + \frac{1}{2}x_b. \tag{4.1}$$

Similarly, we add the y-axis and z-axis center of mass of the box as follows;

$$\delta_b = Y_b + \frac{1}{2}y_b \tag{4.2}$$

and

$$\psi_b = Z_b + \frac{1}{2}z_b. \tag{4.3}$$

Also, we introduce additional decision variables, $X_{\text{cent}} \geq 0$ and $Y_{\text{cent}} \geq 0$ which is the x and y center of gravity direction respectively. Now, concerning the centre of mass in z direction is quite different since we want the gravity to lie as low as possible. Hence, we introduce a decision variable $Z_{\text{height}} \geq 0$ which refers to the contribution height in terms of the center of mass.

Furthermore, we introduce additional decision variables $\text{diff}_X \geq 0, \text{diff}_Y \geq 0$ which refer to the value of deviation of the x and y center of mass direction, respectively. Also we introduce a decision variable

$$\text{Diff} = \text{diff}_X + \text{diff}_Y, \tag{4.4}$$

which is the sum of all the deviation in the x and y center of gravity position.

Now the extended model:

$$\text{minimize} \quad \theta \cdot \text{Diff} + (1 - \theta) \cdot Z_{\text{height}} \tag{4.5a}$$

$$\text{subject to} \quad \text{totalMass} \cdot X_{\text{cent}} = \sum_{b \in B} m_b \cdot \xi_b, \tag{4.5b}$$

$$\text{totalMass} \cdot Y_{\text{cent}} = \sum_{b \in B} m_b \cdot \delta_b, \tag{4.5c}$$

$$\text{totalMass} \cdot Z_{\text{height}} = \sum_{b \in B} m_b \cdot \psi_b, \tag{4.5d}$$

$$X_{\text{cent}} - \frac{1}{2} \cdot L \leq \text{diff}_X, \tag{4.5e}$$

$$\frac{1}{2} \cdot L - X_{\text{cent}} \leq \text{diff}_X, \tag{4.5f}$$

$$Y_{\text{cent}} - \frac{1}{2} \cdot W \leq \text{diff}_Y, \tag{4.5g}$$

$$\frac{1}{2} \cdot W - Y_{\text{cent}} \leq \text{diff}_Y. \tag{4.5h}$$

Constraints (4.5b), (4.5c) and (4.5d) are imposed to ensure that we fully determine the center of gravity along the x, y and z axis. We also model constraints (4.5e) and (4.5f) to check the deviation in the x axis center of gravity position. Similarly, we model that in the y axis. In the basic model we considered maximization of volume of packed boxes, since we want the center of gravity to fall nearer to the ground of the container, we minimize the height of container (4.5a).

Chapter 5

Discussion of Results

5.1 Computational Experiments

5.1.1 Test Bed

The model was implemented in the AMPL programming language on a quad core Intel Core i5 with 1.70 GHz and 8 MByte cache, running on a Linux laptop, and an IBM ILOG CPLEX 12.9 as a linear programming based mixed-integer branch-and-cut framework. The set of test instances proposed in literature most often does not contain orientations in our case. Notwithstanding, our first experiments were performed on the set of instances with $3, 5, 8, 10, 12$ and 15 boxes. The result were on average 70% better in terms of the computational time for volume utilization as compared to the model by Józefowska et al. [11]. These results were promising enough to proceed the experiments on large instances.

Even though the motivation for our model comes from a practical application, unfortunately, no real order test scenarios are available. To evaluate the model performance, we prepared a set of instances on a list of real products. This include the data for the length, width and height of boxes of 28 different types and also the size of the container. We generated instances by randomly choosing boxes from the list to set up a shipment order. The experiment was performed on a set of 369 instances where 9 instances were randomly generated from each 41 groups. The groups of instances are defined by providing the number of box types with its sizes. The container size of 130.87 mm length, 110.64 mm width, and 120.40 mm height was assumed.

5.1.2 Results

The computational execution time for different instances is presented in Figure 5.1. We set a time limit of 3600s (1h) for every instance given. A score shown in the graph reflects each instances solve within the time limit and if not solved shown as a gap. Thus,

$$i = \begin{cases} \frac{t_i}{3600}, & \text{if instance solved} \\ 1 + \frac{\text{gap}}{100}, & \text{if instance not solved.} \end{cases}$$

Here, the gap is computed as:

$$\text{gap} = \frac{\text{LP}_{\text{opt}} - \text{Best integer}}{\text{LP}_{\text{opt}}} \times 100\%.$$

Figure 5.1 shows that from 24 boxes onwards not all boxes can be packed within the given time limit due to a medium container. In addition, the total number of 41 boxes could not be packed and had a gap of 35%. We also computed the median, arithmetic average, and geometric average of the scores which is depicted in the Figure 5.1.

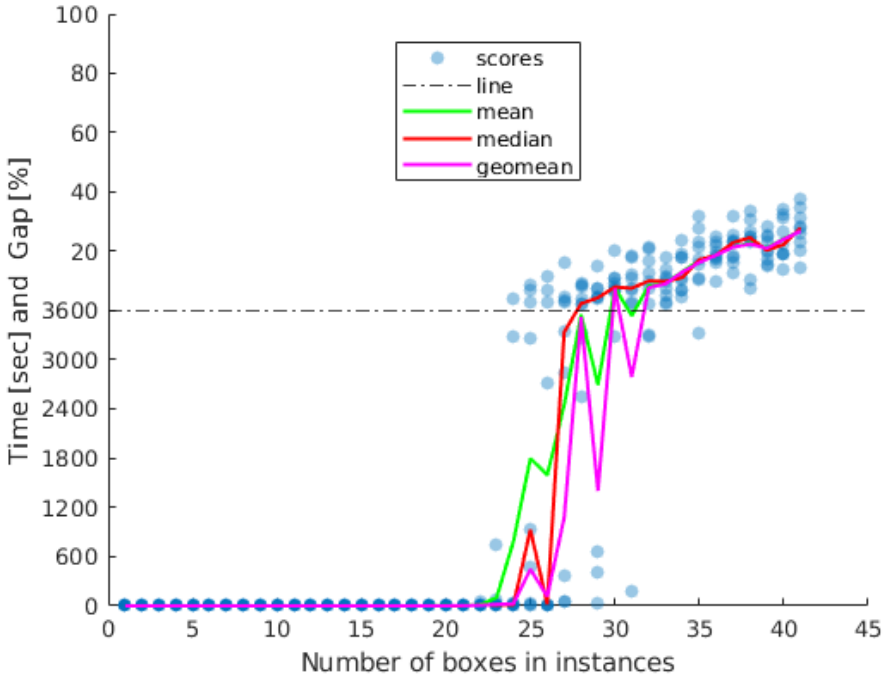


Figure 5.1: Number of boxes packed against time and gap of our model

For comparison, we implemented the model of Józefowska et al. [11] to check the computational execution time with the same data set generated in our instances.

We denote the model of Józefowska et al. [11] as "Jppmk" model, and from Figure 5.2 we see that 17 boxes and above could not be packed within the given time limit. Eventually, some instances had recorded a high gap of 80% showing that such number of boxes could not be packed. Overall, the packing of boxes into the container becomes more difficult compared to a sole maximization of the volume of packed boxes when we have the following:

- A small container is given to pack the boxes
- A higher number of different type of boxes to pack
- The number of boxes to pack increases

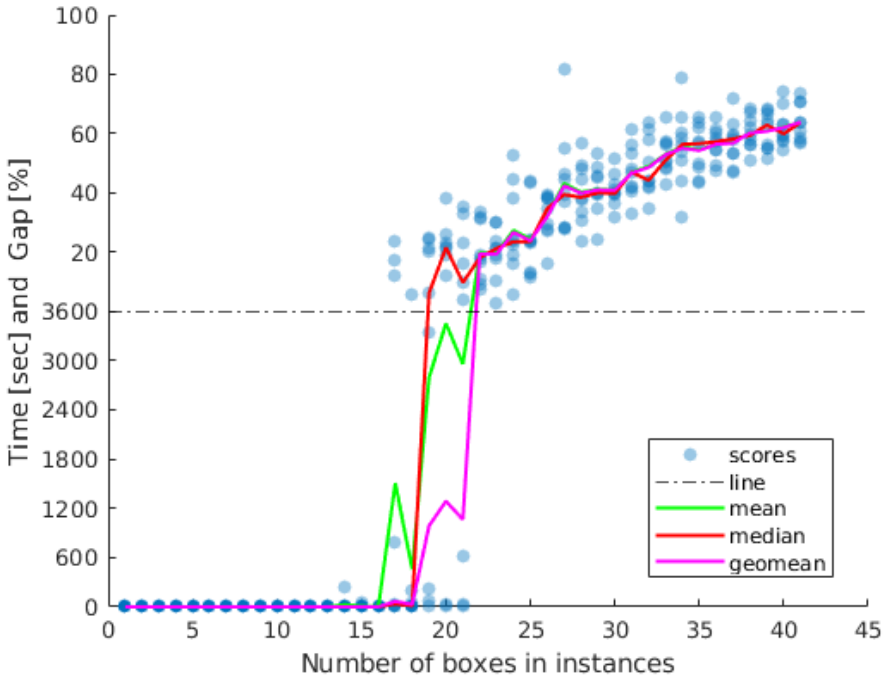


Figure 5.2: Number of boxes packed against time and gap of Jppmk model

Furthermore to check on the quality of our solution, we compared both models together on the same graph to detect which is faster than the other and this is depicted in Figure 5.3. At the end of the experiment, the solver CPLEX was able to find better solutions in shorter time on instances of our model compared to Jppmk model.

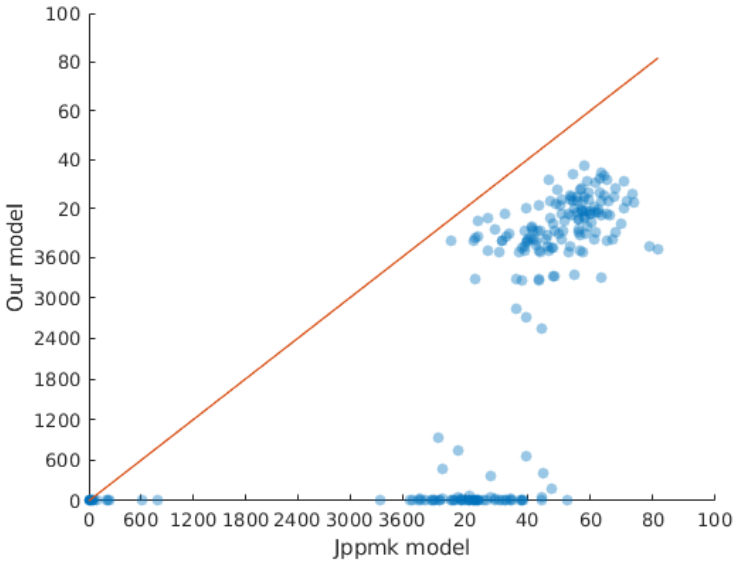


Figure 5.3: Comparison between our model and Jppmk model.

We explore an experiment on the extended model for 10 boxes of 7 different types. We assume the container length, width and height as 50mm, 45mm and 200mm respectively. Next, we varied θ and obtained three unique solutions for a range of θ value. We obtained the following:

- For $\theta = 0.001$ to 0.007, we have Diff $= 1.58398$ and $Z_{\text{height}} = 7.9118$.
- For $\theta = 0.008$ to 0.189, we have Diff $= 0.432482$ and $Z_{\text{height}} = 7.99849$.
- For $\theta = 0.190$ to 0.999, we have Diff $= 0$ and $Z_{\text{height}} = 8.11293$.

All these instances were solved to optimality and a derivation of Pareto curve is obtained for aforementioned bi-criterion optimization model shown in Figure 5.4. Though all these solutions are Pareto-optimal, we identify a "good choice" solution where $\theta = 0.190$ to 0.999, we have no deviation of the center of mass and the container height must have 8.11293 to obtain a stable packing arrangement of the boxes in the container.

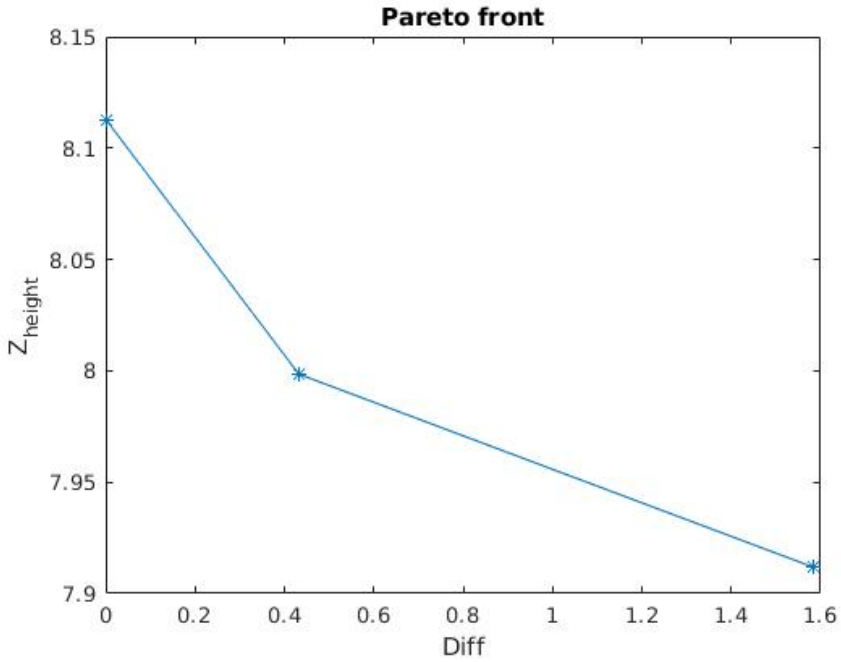


Figure 5.4: Tradeoff between Container height and deviation of center of mass.

Chapter 6

Conclusion and Future Work

We have addressed a real-world 3D container packing problem in this work. The container packing problem is an essential tool in logistics management, hence some practical constraints were considered to ensure that our the proposed results can be implemented by industries faced with some issues in packing plans. In order to assess a satisfactory solution, a constructive model based on a mixed-integer programming approach has been developed with regards to the problems encountered by companies.

Also due to current packing problems that companies face, we considered the container weight distribution or load balance issue in this research. Our model develops a strategy of showing the packing arrangement of boxes packed to the left or right, in front or behind and below or above in the container without overlapping each other and combining boxes of different orientations trying to maximize the volume of packed boxes. Randomly generated instances from a list of shippment order from 41 groups which is the number of types boxes and its size was used to evaluate the performance of the model. The solver CPLEX was able to compute a maximum volume of packed boxes, while satisfying all the constraints of the problem. Most instances could be solved to proven optimality within 1 hour for up to 16 boxes and some instances could not be solved within the time limit from 17 boxes onwards having at most 35% gap due to small container size, high number of different types of boxes and increased number of boxes.

Furthermore, some comparative tests have been conducted to literature related problems with the available data sets showing that the solver CPLEX was able to find better solutions in shorter time on instances of our model compared to instances of the Jppmk model. The good results raised the interest to consider the weight distri-

bution in the container to enhance stability of the packed boxes. Hence, we factored such constraints in the extended model. We obtained three unique solutions from the range θ value which was displayed as a Pareto front cure. A good choice of solution where $\theta = 0.190$ to 0.999, which have no deviation of the center of mass and the container height has 8.11293 to obtain a stable packing arrangement of the boxes in the container.

Now, our future work is to consider the case where boxes have different destinations to be unloaded to customers (priority constraints or multi-drops constraints). In this case, we will address that in the extended model to factor the loading of boxes by dividing the container into sub-containers and arranging the boxes per the destination order for easier off loading. Another case to consider is to factor in the fragility of certain boxes which cannot be packed on one another since this will lead to breakages of items like wine, laptops, etc. in the boxes loaded into the container. Lastly, there are certain boxes that cannot be oriented in all the six orientations (up side down) due to the nature of the product in the box (no orientation for such boxes). These extra constraints can be implemented in our future work.

References

[1] Emile Aarts, Emile HL Aarts, and Jan Karel Lenstra. *Local search in combinatorial optimization*. Princeton University Press, 2003.

[2] MT Alonso, R Alvarez-Valdes, M Iori, and F Parreño. Mathematical models for multi container loading problems with practical constraints. *Computers & Industrial Engineering*, 127:722–733, 2019.

[3] Richard Bellman and Robert E Kalaba. *Dynamic programming and modern control theory*, volume 81. Citeseer, 1965.

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

[5] CS Chen, Shen-Ming Lee, and QS Shen. An analytical model for the container loading problem. *European Journal of Operational Research*, 80(1):68–76, 1995.

[6] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. Greedy algorithms. *Introduction to algorithms*, 1:329–355, 2001.

[7] Richard Cottle and Mukund N Thapa. *Linear and Nonlinear Optimization*, volume 253. Springer, 2017.

[8] Ralph E Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64(5):275–278, 1958.

[9] Zonghao Gu, George L Nemhauser, and Martin WP Savelsbergh. Lifted cover inequalities for 0-1 integer programs: Computation. *INFORMS Journal on Computing*, 10(4):427–437, 1998.

[10] David S Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9(3):256–278, 1974.

[11] Joanna Józefowska, Grzegorz Pawlak, Erwin Pesch, Michał Morze, and Dawid Kowalski. Fast truck-packing of 3d boxes. *Engineering Management in Production and Services*, 10(2):29–40, 2018.

[12] Deidson Vitorio Kurpel, Cleder Marcos Schenekemberg, Cassius Tadeu Scarpin, José Eduardo Pécora Junior, and Leandro C Coelho. *The Exact Solutions of Several Classes of Container Loading Problems*. CIRRELT, Centre interuniversitaire de recherche sur les réseaux d'entreprise, 2018.

[13] AH Land and AG Doig. An automatic method of solving discrete programming problems, econométrica, 1998.

[14] Andrew Lim, Hong Ma, Chaoyang Qiu, and Wenbin Zhu. The single container loading problem with axle weight constraints. *International Journal of Production Economics*, 144(1):358–369, 2013.

[15] Hugues Marchand and Laurence A Wolsey. Aggregation and mixed integer rounding to solve mips. *Operations research*, 49(3):363–371, 2001.

[16] Ilkyeong Moon and Thi Viet Ly Nguyen. Container packing problem with balance constraints. *OR Spectrum*, 36(4):837–878, 2014.

[17] Manfred Padberg and Giovanni Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM review*, 33(1):60–100, 1991.

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

[19] Jean-Paul Rodrigue, Claude Comtois, and Brian Slack. *The geography of transport systems*. Routledge, 2016.

[20] J Cole Smith and Z Caner Taskin. A tutorial guide to mixed-integer programming models and solution techniques. *Optimization in Medicine and Biology*, pages 521–548, 2008.

[21] Laurence A Wolsey and George L Nemhauser. *Integer and combinatorial optimization*. John Wiley & Sons, 2014.

[22] Yong Wu, Wenkai Li, Mark Goh, and Robert de Souza. Three-dimensional bin packing problem with variable bin height. *European Journal of Operational Research*, 202(2):347–355, 2010.