Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Toralf Christian Skaali Frich

A real-world pallet loading problem - developing a MIP formulation with practical constraints

A real-world pallet loading problem

Master's thesis in Computer science
Supervisor: Magnus Lie Hetland
February 2022

NTNU
Norwegian University of
Science and Technology

Toralf Christian Skaali Frich

A real-world pallet loading problem - developing a MIP formulation with practical constraints

A real-world pallet loading problem

Master's thesis in Computer science
Supervisor: Magnus Lie Hetland
February 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Abstract

The aim of this thesis is to generate knowledge for creating better stacks of boxes in a robot company's pallet loading problem (PLP). To address this aim, an exact mixed-integer program (MIP) of the given PLP was formulated, tested, and compared with the robot company's state-of-the-art beam-search heuristic method.

The thesis suggests several new mathematical formulations of practical constraints and PLP research concepts, including a guillotine concept, a partial base load-bearing concept, a middle-incentive constraint, and an improved partial base support formulation using $\frac{1}{3}$ less space than an existing state-of-the-art formulation.

The results show that MIP models cannot, alone, with their current limitations, generate satisfactory box stacks for the given PLP within its runtime limit. However, the results also indicate that solution schemes combining a MIP model with a heuristic method achieve better stacks of boxes. In addition, it is found that it is easier to solve the given PLP if an instance of the problem contains tall boxes.

These findings have several implications for the PLP research field. Few researchers have proposed solution schemes that combine exact- and heuristic methods. This thesis suggests that higher-quality stacks of boxes may be generated by combining these methods. Further, this thesis indicates that better box stacks can be achieved if solution schemes are more considerate of boxes' characteristics. Finally, this thesis demonstrates that there are still many MIP formulations of practical constraints that warrant further investigation, with the potential to overcome the limitations of current MIP models.

Sammendrag

Målet med denne oppgaven er å utvikle kunnskap som kan gi bedre stabler med bokser i et robotfirmas pallelastingsproblem (PLP). For å undersøke dette, ble det utviklet et blandet heltallsprogram (MIP) for det gitte PLP. Dette ble testet og sammenlignet med robotfirmaets etablerte heuristiske metode basert på strålesøk.

Oppgaven fremsetter flere nye matematiske formuleringer av praktiske restriksjoner og PLP-tilnærminger som kan studeres videre, herunder et guillotin-konsept, et lasteevne-konsept, en midt-insentiv restriksjon, og en forbedret støtterestriksjonsformulering som bruker $\frac{1}{3}$ mindre plass enn en etablert formulering.

Resultatene viser at MIP-modellen alene, med dens eksisterende begrensninger, ikke gir tilfredsstillende stabler for PLP innen den gitte kjøretiden. Resultatene tyder imidlertid på at en løsningsmetode som kombinerer en MIP-modell med en heuristisk metode kan gi bedre stabler med bokser. Resultatene viser videre at det er lettere å løse den gitte PLP hvis en instans inneholder høye bokser.

Disse resultatene har flere implikasjoner for forskning på PLP. Få forskere har foreslått løsningsmetoder som kombinerer eksakte- og heuristiske metoder. Denne oppgaven antyder at bedre stabler med bokser kan oppnås ved å kombinere disse metodene. Oppgaven foreslår også at det kan lages bedre stabler med bokser hvis en løsningsmetode tar mer hensyn til boksenes karakteristika. Oppgaven peker i tillegg på at det fortsatt er mange MIP-formuleringer knyttet til praktiske restriksjoner som bør studeres nærmere for å avhjelpe de nåværende begrensningene ved eksisterende MIP-modeller.

Preface

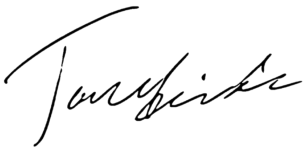
This thesis is the culmination of my M.Sc. degree in Computer Science at the Norwegian University of Technology and Science.

I was motivated to write a thesis about optimisation during a course about algorithmic construction. More specifically, I wanted to learn how to formulate mixed-integer programs, and Currence Robotics' problem presented itself as a good challenge that stimulated me to work creatively with mathematical formulation.

I want to acknowledge Currence Robotics and SINTEF for valuable discussions while writing this thesis.

A special thanks to Magnus Lie Hetland for being a great supervisor, and for the continuous inspiration and helpful feedback.

Finally, I want to thank my family for their support along the way.



Oslo, February, 2022

Contents

Acronyms

3KP The three-dimensional knapsack problem.

CLP Container loading problem.

CNN Convolutional Neural Network.

DS Dynamic stability.

G Guillotine.

GEO Geometric.

IP Integer programming.

KP Knapsack problem.

LB Load balancing.

LBP Load-bearing pressure.

LP Linear programming.

MB Multiple boxes.

MID Middle incentive.

MIP Mixed-integer programming.

NP Nondeterministic polynomial time.

NPC NP complete.

OT On top.

PBS Partial base support.

PLP Pallet loading problem.

RQ Research question.

SBSBPP Single Bin Size Bin Packing Problem.

SKP Single Knapsack Problem.

SLOPP Single Large Object Placement Problem.

SOS Special Ordered Set.

SS Static stability.

SSSCSP Single Stock Size Cutting Stock Problem.

Chapter 1

Introduction

The transportation sector changed dramatically with the introduction of containers and pallets in the 1930s. Not only did they streamline all aspects of transportation, but they reduced costs, protected goods from damage, and reduced shipping times [1].

Today, the transportation sector has become an integral part of our lives and accounts for 5 % of the GDP of EU countries [2]. However, in 2021, shipping costs soared, quadrupling prices of 2020 [3]. Thus, there is a need to propose more cost-efficient solutions.

Vargas-Osorio and Zúñiga [4] estimate that more efficient packing schemes can cut storage costs by 20-25 %, and transportation costs up to 30 %. Today, goods are usually placed into containers and onto pallets using manual labour. Hence, a solution to the rising costs may be autonomous agents, which have the potential to create efficient packaging at lower costs [4] [5]. And thus, we might be on the verge of the next dramatic change in transportation: a change of employment - from humans to robots.

Currence Robotics is a Norwegian robot company aimed at developing robots for the transportation sector. One of the robots they have created is "*Grab*": an autonomous robot capable of placing boxes onto a pallet.

Grab is currently used in a warehouse containing grocery shop items in Norway. Supermarkets can order items from the warehouse, which are delivered to them in

cardboard boxes placed on top of a pallet. Once the warehouse receives an order, it is transmitted to Grab. Grab then has access to each boxes' dimensions, weight, and warehouse location.

2400 seconds of processing time is allowed before Grab needs to start fetching the items. Grab fetches boxes using its robot arm, capable of placing boxes on top of an attached pallet. When a complete order is stacked on top of a pallet, plastic wrap is used to wrap the stack of boxes, ensuring that they do not fall over at a later point in time. Finally, the order is sent to the customer. A picture of Grab can be seen in figure 1.1.

Grab is able to move around the warehouse, and it can place boxes onto the pallet using its robot arm. However, some challenges remain until Grab is fully autonomous. In particular, it does not know where to place the boxes onto the pallet so that a stable and compact stack of boxes is achieved. This problem of placing boxes onto a pallet is known as the *Pallet loading problem* (PLP) [4, 6, 7]. The PLP of Currence Robotics has some characteristics that will be further explained in chapter 2, but essentially it revolves around placing a finite amount of boxes onto a pallet, ensuring that these are stable and able to withstand the forces of the real world.

1.1 Goals and objectives

The solution scheme used by Currence Robotics to solve their PLP is a state-of-the-art heuristic beam-search algorithm. At the time of writing, it generates stacks of boxes filling 70-80 % of the total pallet capacity. However, no guarantees can be given regarding real-world viability, as the stacks can be unstable due to real-world forces. With this foundation, a research aim for this thesis is established:

Research Aim The aim of this thesis is to generate knowledge that can be used to create better stacks of boxes in Currence Robotics' PLP

Currence Robotics are currently using the best heuristic method for solving the PLP [8]. But, as it is unable to provide sufficiently good stacks of boxes, alternative solution schemes are motivated. An exact method had not been looked into when the author and the supervisor were approached about Currence Robotics' problem. In addition, preliminary research showed that multiple authors were approaching the PLP and similar problems using *Mixed-integer programming* (MIP). Thus, to address

the aim of the thesis, the following goals are presented:

Goal 1 To develop a MIP formulation of Currence Robotics' PLP

Goal 2 To test the developed MIP formulation of Currence Robotics' PLP

Developing and testing a MIP model for Currence Robotics' PLP may provide for better box stacks, either by generating higher quality stacks or by introducing new insight. It can be that creating a MIP formulation of Currence Robotics' PLP is impossible. Alternatively, it may be that the developed MIP formulation generates lower quality box stacks. However, that information is, in itself, useful. *Research questions* (RQ) are proposed for each goal, making way for how they can be achieved. The research questions for the first goal are:

RQ 1.1 What sub-problems of Currence Robotics' PLP can be solved using MIP?

RQ 1.2 What does the MIP research literature propose as the best solutions to sub-problems of Currence Robotics' PLP?

RQ 1.3 If the MIP research literature does not propose any solutions to sub-problems of Currence Robotics' PLP, what possible solutions exist?

Creating a high-performing MIP formulation requires the best possible solutions of sub-problems to Currence Robotics' PLP to be established. These can later be combined into a MIP formulation of Currence Robotics' PLP. However, Frich [9] found only a limited number of articles proposing MIP formulations for sub-problems of Currence Robotics' PLP. Hence, to achieve Goal 1, solutions to sub-problems of Currence Robotics' PLP must be developed using solutions by researchers and by using solutions proposed by the author. The research questions for Goal 2 are:

RQ 2.1 How do MIP formulations to sub-problems of Currence Robotics' PLP perform?

RQ 2.2 How do MIP formulations of Currence Robotics' PLP perform?

RQ 2.3 How do the MIP formulations of Currence Robotics' PLP compare to Currence Robotics' heuristic method?

RQ 2.4 Are there instances of Currence Robotics' PLP that are more difficult to solve due to their set of boxes?

The developed MIP formulation of Currence Robotics' PLP will consist of a combination of MIP formulations to sub-problems of Currence Robotics' PLP. As there might exist multiple solutions and formulations to a single sub-problem, some will probably perform better than others. Thus, to choose the best combinations, individual solutions and formulations must be tested and compared. Once these combinations have been created, they will be tested against Currence Robotics' heuristic method.

The research aim of this thesis is to generate knowledge that can lead to better stacks of boxes. Better box stacks are, for the most part, generated by better solution schemes. Hence, one way to approach the aim is to explore alternative, possibly better, solution schemes. This thesis does so by developing a MIP formulation of Currence Robotics' PLP. A second way to approach the aim is to generate knowledge about the problem itself, which can be used to design better solution schemes. RQ 2.4 addresses this possibility as, to the best of the author's knowledge, little research has been conducted on how the set of boxes in a PLP impacts the difficulty of solving that PLP. Hence, experiments used during testing will be designed to reveal whether the characteristics of an instance's boxes impact the performance of tested solution schemes.

1.2 Contributions

Listed below are the main contributions of this thesis. These are further discussed in chapter 6 along with the other contributions of this thesis.

- Results showing that MIP models of Currence Robotics' PLP cannot, with their current limitations, generate satisfactory box stacks within the given runtime limit.

- Indications of higher-quality box stacks for Currence Robotics' PLP if the solution scheme combine a MIP model and a heuristic method.
- An improved static stability partial base support constraint formulation, $\frac{1}{3}$ smaller in size than the state-of-the-art formulation by Nascimento et al. [10], without compromise to performance.
- A discovery that a PLP instance may be easier to solve if it contains taller boxes.
- A MIP PLP implementation consisting of 4500 lines of code in Julia, tested for 700 CPU hours [11] [12].
- A comparison between a heuristic method currently used on an autonomous robot and a MIP formulation for solving a real-world application of the PLP.
- Proposal of mathematical formulations for new PLP practical constraints and concepts, including a guillotine static stability constraint concept, a partial base load-bearing constraint, and a middle-incentive dynamic stability constraint.

1.3 Research method

Recent literature and state-of-the-art solutions have been consulted to address the aim and achieve the goals of the thesis. In addition, conversations regarding different concepts and solutions about the PLP have been held with several people. In particular, the author's supervisor, Currence Robotics and their research partner SINTEF. They have been involved throughout the research process to discuss solutions and PLP concepts.

Multiple articles about the PLP and similar problems were used to understand Currence Robotics' PLP and to discover possible solutions. The approach that was used to find relevant articles is explained in a literature review protocol in section 2.7. Section 2.7 also displays an overview of relevant literature used in this thesis.

Experiments conducted in the thesis were designed to reveal how a set of boxes in a PLP instance can impact that PLP's difficulty. Moreover, statistical analysis was used to find the correlation between the obtained results for a PLP instance and the characteristics for that instance's boxes.

1.4 Thesis structure

This thesis is made up of six chapters. Chapter 1 serves as the basis for this thesis, introducing its motivation, aim, goals and research questions.

In chapter 2, relevant background theory is presented, explaining the prerequisites for understanding Currence Robotics' PLP. A brief summary of Currence Robotics' heuristic method is also presented.

For each sub-problem of Currence Robotics' PLP, its state-of-the-art solutions are discussed and presented in chapter 3. Then, a decision is made regarding what solutions are to be given a MIP formulation. Furthermore, mathematical formulations of these solutions are presented.

In chapter 4, the results of the experiments are presented along with hardware, software, parameter values, and instances used to conduct the experiments.

Then, in chapter 5, the development of the MIP formulation and the results that were obtained are discussed with consideration to the research questions and goals of the thesis.

Finally, conclusions regarding the aim, goals, and research questions, along with future work, are presented in chapter 6.

1.5 Notice

This master thesis is based on a specialisation project by Frich [9] written the second to last semester during the spring of 2021. Hence, the reader is made aware that certain parts of this thesis may be similar to that specialisation project. These parts and their material from the specialisation project are:

- Table 2.1: Using many of the same articles.
- Section 2.1, Section 2.2. Section 2.4, Section 2.5: Using many of the same background theory formulations.
- Chapter 3: Using the same overall structure. Some material is similar in sections regarding state of the art.
- Section 3.1: Using the same mathematical formulations.
- Figures initially created for the specialisation project: figure 2.1, figure 2.2, figure 3.1, figure 3.2, figure 3.3, figure 3.5, and figure 3.7.

Figure 1.1 and figure 3.26 have been used with permission from Currence Robotics.

Figure 1.1: Currence Robotics' Grab robot (Picture used with permission from Currence Robotics)

Chapter 2

Background Theory

This chapter introduces relevant background theory for this master thesis, gradually building an understanding of Currence Robotics' PLP. Insight is provided into the following broad topics: Optimisation, linear-, integer-, mixed-integer programming, and complexity theory. More specialised topics are also covered, such as the knapsack problem, the container loading problem and the pallet loading problem. Specific characteristics to the pallet loading problem and Currence Robotics' heuristic method are also introduced. And lastly, a literature review and an overview of the most relevant articles for this thesis are presented.

2.1 Optimisation

Optimisation is a mathematical problem-solving methodology for selecting a feasible, optimal value of some optimisation problem. An optimisation problem is, in general, made up of three parts:

- The objective function specifying the objective of the optimisation
- Constraints enclosing a *solution space*, defining what parameter values a feasible solution is allowed to take
- Input parameters deciding the specific problem instance

A solution scheme to an optimisation problem can generally be classified as a heuristic- or an exact method for optimisation. A heuristic method of optimisation finds what

it believes to be the best solution based on information retrieved while solving a problem. Heuristic methods are generally quicker at finding solutions but have trade-offs regarding solution quality. Exact methods of optimisation have the ability to produce an optimal solution to an optimisation problem. However, as optimisation problems may be large or complicated, this may take a long time.

There are many ways of expressing an optimisation problem exactly. However, some methods are preferred depending on the problem, its restrictions, and the desired outcome. *Linear Programming* (LP) is an exact method of optimisation and expresses an optimisation problem using a set of linear constraints across a solution space [13].

What is particular about linear programming is that its parameters, a vector x of the optimisation problem, are allowed to take any values $x \in \mathbb{R}$. A linear maximisation program can be formulated as follows [13]:

Maximise value of $c^T x$ among all vectors $x \in \mathbb{R}$ satisfying $Ax \leq b$

where:

$c^T x$:	objective function $c^T x = c_1 x_1 + \cdots + c_n x_n$ where $c \in \mathbb{R}^n$ is a given vector
A:	Given $m * n$ matrix
b:	Given $b \in \mathbb{R}^n$ vector

Integer Programming (IP) is similar to linear programming. However, an integer program only allows its parameters; a vector x, integer values: $x \in \mathbb{Z}$. Given an optimisation problem with an integer- and a linear program formulation, then all solutions to the integer program are attainable with the corresponding linear program. An integer programming problem is NP-hard, meaning it is probably not possible to find an optimal solution in polynomial time [14]. An integer programming maximisation problem can be formulated as follows [14]:

Maximise value of $c^T x$ among all vectors $x \in \mathbb{Z}$ satisfying $Ax \leq b$

The PLP can be expressed using a *Mixed Integer Linear Programming* (MIP) formulation. A MIP formulation of an optimisation problem requires that at least one of the parameters $x_i \in x$ has an integer constraint: $x_i \in \mathbb{Z}$. The rest of the parameters can have linear constraints: $x \setminus x_i \in \mathbb{R}$ [14]. As a MIP formulation has integer constraints, its complexity is equal to that of an IP problem; NP-hard. A MIP program

can be formulated similarly to how the LP and the IP were formulated, but instead with the constraints above.

To explain aspects of the PLP, many commonly known optimisation characteristics of research can be used. A *feasible solution* to an optimisation program is a solution where all constraints, $Ax \leq b$, are satisfied. This means that a program's solution vector $x \in \mathbb{R}^n$, used as a parameter to the objective function, does not violate any of its constraints [14]. In Currence Robotics' PLP, a feasible solution would imply that the boxes subject to placement had been placed onto the pallet. In addition, they would satisfy any constraints imposed on them. An example of a feasible solution in a simple optimisation program can be seen in figure 2.1. On the contrary, an *infeasible* program is a problem with no feasible solution [14]. An infeasible PLP could, e.g., be that the boxes had nowhere to be placed within the solution space due to space issues.

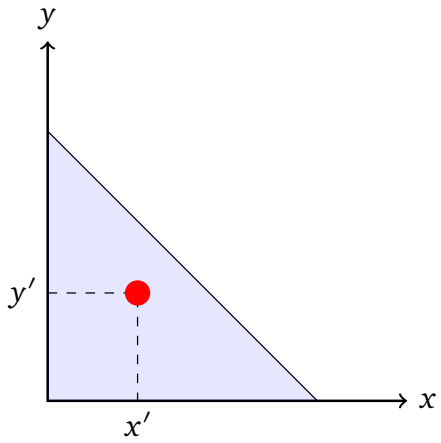


Figure 2.1: A feasible solution to a maximisation program at $x = x'$ and $y = y'$ within the solution space constrained by $y = -1.5x + 1.5$

LP Relaxation is later used by a MIP solver to create bounds regarding a possible best solution. Finding an LP relaxation can be done by reducing a problem from a more difficult IP- or MIP formulation to an LP formulation [14]. Reduction happens by relaxing the integer constraints of the parameters in the IP- or MIP program, x, to weaker linear constraints; enforcing $x \in \mathbb{R}$ instead of $x \in \mathbb{Z}$. The maximum ratio between an IP or MIP solution and an LP relaxation is known as an *integrality gap* [14]. An extended concept of the integrality gap, known as *gap* is used in chapter 4 to analyse results from a MIP solver. The gap is the difference between what the solver believes is the best possible solution and the best solution it has found so far.

The characteristics above are central aspects to any optimisation problem. However, there also exist more specific characteristics for MIP formulations. Particularly with regards to how MIP formulations are created.

Big-M constraints are used in IP and MIP to formulate more articulate inequalities. In particular, it models binary behaviour, allowing variables to be "turned on or off" [15]. Big-M is used throughout chapter 3, and in section 3.1, an in-depth explanation of big-M constraints is given as a constraint of Currence Robotics' PLP is formulated.

A *sentinel value* (dummy value) is also needed to formulate Currence Robotics' PLP. A sentinel value can be used to terminate an algorithm or program [16]. However, in the MIP formulations of chapter 3, it is used as a dummy value for the pallet floor. In particular, for the partial base support constraint seen in section 3.4.1.

2.2 Complexity classes

As mentioned in section 2.1, the PLP is an NP-hard problem. An NP-hard problem is a problem that is part of a complexity hierarchy, of which parts can be seen in figure 2.2. Complexity theory is relevant for the PLP as it explains why the PLP is so hard to solve.

The lowest placed complexity class in figure 2.2 is P. If a problem lies within P, then that problem is possible to solve deterministically in polynomial computation time [13].

The larger category of problems located above P is the *Non-deterministic polynomial* (NP) complexity class. NP is a greater category of problems that also contains the problems of P. For NP problems; it is possible to verify whether a problem solution has been found in polynomial time using a deterministic Turing machine. All NP problems are solvable in polynomial time by a non-deterministic Turing machine [13].

If a problem is NP-complete (NPC), it is recognised as one of the hardest problems in the NP complexity class. Furthermore, if an algorithm exists to solve an NPC problem in polynomial time, then all NP problems can be solved in polynomial time. IP- and MIP problems are part of NPC, given that they are formulated as a *decision problem*. If not, their complexity is *NP-hard* [17]. *NP-hard* problems are problems that are at least as hard to solve as the hardest NP problems. And since the MIP is used to formulate the PLP is NP-hard, this means that it can take a long

time to obtain optimal solutions.

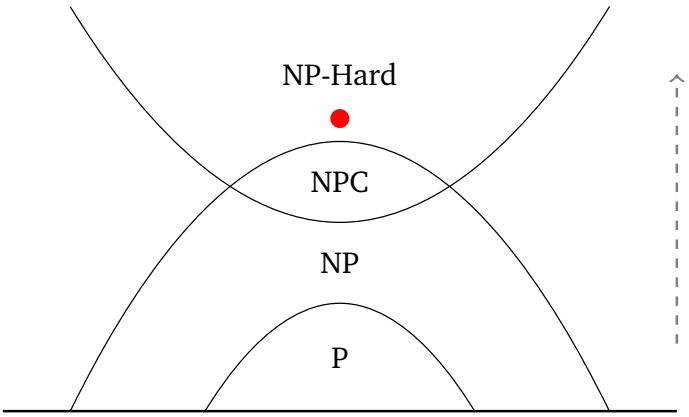


Figure 2.2: An overview of the discussed complexity classes. The complexity of the PLP is indicated by the red dot. The grey dashed arrow indicates how problems become harder further up in the hierarchy.

2.3 Elimination of products of variables

Products of variables are not allowed in an LP-, IP-, or MIP formulation, as a product makes a problem non-linear. *Non-linear optimisation problems* are generally harder to solve than linear optimisation problems [18]. Hence, if possible, a problem should remain linear.

In section 3.4.1, a product calculation is necessary to calculate the partial base support between two boxes. Thus, for the PLP to remain linear, that product is eliminated. Elimination of a product can be done by approximating that product's result. There are multiple ways to do this, but in this master thesis, the method described by Williams [19] and Bisschop [20] is used.

The product in section 3.4.1 is a product of two variables, but the method by Williams [19] and Bisschop [20] for eliminating a product requires a product of a single variable. Thus, the given product is transformed into a *separable function*. A separable function consists of a single variable or a sum of single variables. Equation (1) is a separable function as it consists of single variables summed together, while equation (2) is not as it is a fraction consisting of two variables. It is often possible to convert non-separable functions to separable functions, which is the method used in section 3.4.1.

$$x_1^2 + x_2 = 3 \tag{1}$$

$$\frac{x_1^2}{x_2} = 3 \tag{2}$$

Once a separable function is obtained, the linear approximation of the product is made by using a *piecewise linear function*. A piecewise linear function consists of straight-line segments used for approximating function values. An example by Williams [19] can be seen in figure 2.3.

The straight-line segments of the piecewise linear function are usually defined on an interval. In figure 2.3, the interval of the approximation is between $(0,0)$ and $(2.5,6.25)$. This interval needs to be pre-defined, as it is impossible to retrieve values of variables from a MIP solver while solving. However, as the dimensions of the pallet are used in the product in section 3.4.1, it is possible to pre-define the intervals.

Using the separable function, y-values are calculated for a given number of x-values. These make up the start- and endpoints of the straight-line segments. In figure 2.3, four black dots are used to indicate their position. It can be observed that shortening the intervals by adding more segments reduces the approximation error.

A *special ordered set 2 constraint* (SOS2) is needed for using the piecewise linear function in the approximation. The SOS2 constraint was first introduced as a concept by. It states that if one is given a list of n variables $[\lambda_1, \lambda_2, ..., \lambda_n]$, at most two consecutive variables in that list may be nonzero [20, 21].

The SOS2 constraint is used to only allow two consecutive variables to be non-zero. Moreover, only two consecutive black dots in figure 2.3 are allowed to be use in any approximation. E.g., activating λ_1 and λ_2 is allowed, while activating λ_1 and λ_3 is not allowed. Furthermore, the approximation requires a constraint enforcing $\lambda_1 + \lambda_2 + ... + \lambda_n = 1$. Together, these two constraints ensure that the approximated value of a product only relies on the function values of the nearest defined domain values. E.g., if asked to find the approximated value for $x = 1.5$ in figure 2.3, $0.5\lambda_2 + 0.5\lambda_1$ would be given as an answer $= 2.5$. This is 0.25 off from the correct answer: $1.5 \times 1.5 = 2.25$.

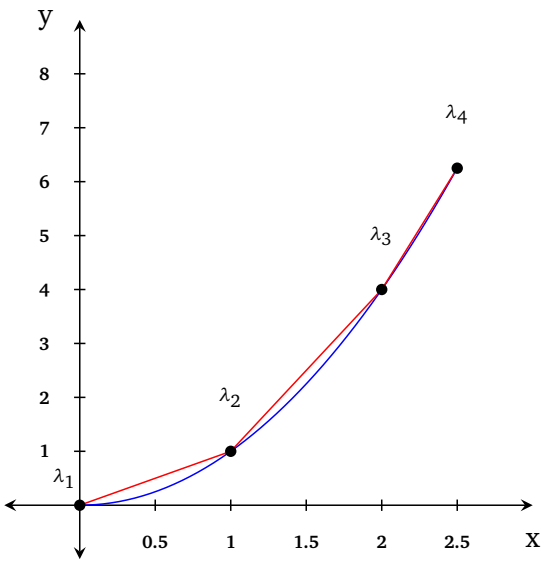


Figure 2.3: The separable x^2 function is plotted using a blue line, and the piecewise linear function is plotted using a red line. The black dots indicate the intervals of the piecewise linear function. The error between the separable function and its piecewise linear function for an x-value is given by function value difference [19].

2.4 Packing problems

The PLP is, along with many other problems, part of the packing problem family. And to fully understand the PLP and its concepts, knowledge is gradually built by exploring similar packing problems first.

The knapsack problem (KP) is an optimisation problem that tries to optimise the value V while ensuring that a given max cost C_{max} is not surpassed. Given a set of items $A = (x_1, \ldots, x_n)$ where each item x_i has an associated cost c_i and a value v_i, create a subset $B \subseteq A$ where the total cost of each item C_B is $C_B \leq C_{max}$ and the total item value of B (V_B) is maximised [13].

At it's core, the KP resembles the traits of the PLP. But, an even more closely related knapsack problem is the *The three-dimensional knapsack problem* (3KP). The 3KP is defined within a three-dimensional coordinate system (x, y, z) for $x, y, z \in \mathbb{R}^+$ [22]. The problem consists of maximising the value from placing three-dimensional items i of dimensions (p_i, q_i, r_i) within the bounds of X, Y, and Z [22]. These bounds can

be referred to as as *length* L, *width* W, and *height* H. Hence, if position (x_i, y_i, z_i) is the lower left back corner for an item i in the coordinate system, then the following constraint proposed by Egeblad and Pisinger [23] must be satisfied:

$$0 \leq x_i \leq L - p_i, \quad 0 \leq y_i \leq W - q_i, \quad 0 \leq z_i \leq H - r_i \tag{3}$$

In addition to boundary constraints, non-overlapping constraints are imposed on the items, ensuring that items cannot be placed inside each other. These non-overlapping constraints will be explored further in section 3.1.

The container loading problem (*CLP*) is similar to the 3KP in that it is defined within a container with a coordinate system $(x, y, z) \in \mathbb{R}^+$ bounded by L, W and H. The container can be loaded with three-dimensional items (usually boxes) i from one end, while the rest of the ends are encapsulated by walls. A usual objective to the CLP, similar to the 3KP, is to maximise the total volume utilised by the boxes placed in the container C_i (given that the total volume of all items C exceeds the volume of the container LWH) [7]. In the 3KP, there is always a desire to maximise item value within the solution space. Differently, the CLP may aim to minimise the height of the stack of boxes placed within the container, if the total volume of the boxes is less than the volume of the container. An illustration of a container and its initial solution space can be seen in figure 2.4.

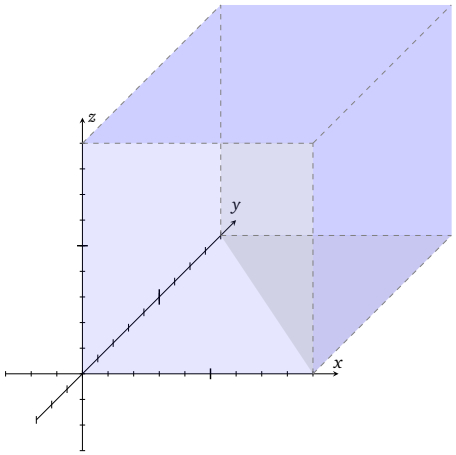


Figure 2.4: A container with the opening seen from the front. The solution space $(x, y, z) \in \mathbb{R}^+$ is constrained within the bounds of the container (L, W, H).

The CLP is the most similar problem to the PLP. The only difference is that, in the PLP, boxes are placed on a pallet approachable from all sides [7]. Consequently, there are no walls in the PLP that can provide side-support to boxes. As a research topic, the CLP is more popular than the PLP, but almost all concepts of the CLP can be applied to the PLP [24]. An illustration of a pallet and its initial solution space can be seen in figure 2.5.

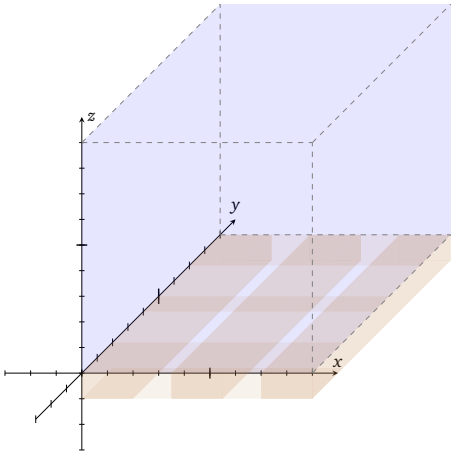


Figure 2.5: A pallet with the solution space $(x, y, z) \in \mathbb{R}^+$ seen above the pallet. The solution space is not bounded by physical walls, but rather by the dimensions of the pallet and a set height.

2.5 PLP Characteristics

The word "*instance*" is often used in PLP and CLP literature to describe a specific problem [10]. Moreover, an instance is a set of boxes to be optimised. An instance $\mathscr{I}$ to be placed on a pallet can either be *strongly heterogeneous* or *weakly heterogeneous* [7]. If an instance is strongly heterogeneous, then it consists of boxes (i) with dissimilar dimensions (p_i, q_i, r_i). But if an instance is weakly heterogeneous, then it consists of boxes with equal- or similar dimensions [7]. The average number of boxes with equal dimensions can be used to decide if a set is strongly- or weakly heterogeneous. In Currence Robotics' PLP, an average of 1.6 boxes have equal dimensions in a standard order. Considering that a typical order consists of 60 to 70 boxes, the instances of Currence Robotics' PLP can be classified as strongly heterogeneous.

There exist multiple ways of representing a CLP or PLP solution space using differ-

ent combinations of coordinate values [25]. In the MIP formulations proposed by Chen et al. [26], a continuous grid is used, meaning that a box can be placed on all continuous coordinates. Junqueira et al. [27] and Junqueira and Queiroz [28] formulate models using discretised grids. A discretised grid reduces the solution space by restricting what coordinates boxes can use. E.g., Silva et al. [25] formulate a discretised grid that combine multiples of each box's dimensions as possible coordinates of box position.

Many solution schemes use a *block-building* approach for placing boxes [29]. A block consists of multiple boxes placed into each other. Furthermore, a block-building approach places blocks instead of individual boxes onto a pallet or into a container. The goal of block-building approaches is to reduce the running time of programs [30]. In addition, it can be easier to enforce constraints and properties to a block rather than to a stack of boxes.

In CLP- and PLP publications, the goal is often to minimise the number of containers/pallets used for placing a large set of boxes [31]. Similar variations consist of maximising the volume (or minimising the unused volume) of the boxes placed in-/on a single container/pallet given that the total volume of all the boxes C exceeds the volume of the container/pallet $C \geq C_{max}$ [32]. Gimenez-Palacios et al. [33] maximises the total number of complete orders within a container, while Gajda et al. [34] maximises the profit of the boxes in the container. However, in Currence Robotics' PLP, a single pallet is used, and $C < C_{max}$. Thus, the goal is to create a viable stack of boxes on the pallet that can cope with real-world physics.

Elaborating on this, formal distinctions of CLPs and PLPs regarding their objective function and goal are introduced. Formulated by Wäscher et al. [7]; the objective of CLPs and PLPs can be one of two depending on the problem: *input minimising* or *output maximising*.

Input minimisation involves assigning boxes to one or more pallets/containers while minimising some measure. Typically, this measure is the height of the stack or the number of pallets/containers used [7]. This problem arises when the total volume of an instance is smaller than the pallet(s)/container(s). Output maximisation involves stacking an instance with a total volume larger than the volume of the pallet(s)/container(s) [7]. Hence, an output maximisation problem centres around maximising the value of the boxes that are stacked. E.g., to choose the boxes that have the highest price tag or similar. Currence Robotics' PLP is an input minimisation problem.

The CLP and PLP can, more specifically, be classified as *Single Stock Size Cutting*

Stock Problem (SSSCSP) for weakly heterogeneous boxes and *Single Bin Size Bin Packing Problem* (SBSBPP) for strongly heterogeneous boxes [7]. These classifications apply if the CLP and PLP are input minimisation problems - which they are in Currence Robotics' PLP. However, for output maximisation problems, the classification is *Single Large Object Placement Problem* (SLOPP) for weakly heterogeneous boxes, and *Single Knapsack Problem* (SKP) for strongly heterogeneous boxes [7]. Both of these classifications are provided so that the reader can distinguish between the problems in the research literature.

Hence, SBSBPP is the more specific classification of our PLP instance. However, Currence Robotics' PLP is used instead for simplicity.

2.6 Currence Robotics' heuristic solution

Currence Robotics' heuristic solution is based on the scheme from "The six elements to block-building approaches for the single container loading problem"; an article by Weng et al. [29]. In the article, Weng et al. [29] breaks the CLP into six separate sub-problems, which can be solved individually. The six sub-problems that need to be solved are [29]:

1. How to represent free space
2. How to generate a list of blocks
3. How to select a free space
4. How to select a block
5. How to place the selected block into the selected free space
6. Define an overarching search strategy

To solve these problems, Currence Robotics have opted for the following approach. Firstly, to represent free space, the maximal residual/unoccupied space is found [29]. This is the space that is eligible for a box placement.

Currence Robotics use a fixed order for fetching boxes. Hence, Grab only needs to traverse the warehouse once. This impacts the block generation of the heuristic method as blocks must be made up of boxes that are physically close to each other.

To create the blocks, some criteria are imposed: ensure that sub-blocks combined with other sub-blocks have the same height, encourage similar-dimension boxes, and ensure internal block stability. The stability is calculated by placing the boxes

to withstand a maximum acceleration[1]. Sub-blocks are combined to create larger blocks with more boxes because they are preferred to the algorithm. The reason for preferring blocks with many boxes is because they are generally more compact than the rest of the stack while also having smoother surfaces [35].

Selecting a free space and a block is done by scoring every possible block/space pair. This scoring scheme is based on the VCS, a heuristic function for selecting boxes proposed by Araya et al. [35]. After finding a block/space pair, the block is placed into the selected free space in that space's backmost, leftmost corner. And, if there are no other blocks that can fit the residual space after placing that block, that block is spread out to fill the entire space it was assigned.

The overarching search strategy is based on an iterative beam search algorithm proposed by Araya et al. [8]. The search effort is doubled for each iteration by increasing the beam length by $\sqrt{2}$. And, while keeping track of the n best solutions thus far, branches are pruned if they return stacks of greater height than the ones found so far.

Finally, the solution with the highest compacity is returned, effectively meaning the stack of boxes that has the lowest height.

2.7 Literature Review

This section presents a literature review protocol and an overview of relevant articles to this thesis.

This thesis uses formulated mathematical constraint inequalities for solving Currence Robotics' PLP. However, as seen in table 2.1, the author was only able to find three articles regarding this topic. Thus, the scope was broadened to include heuristic methods for solving the PLP, in which only one was found. As a consequence, similar problems were investigated. And, in particular, the CLP; explained in-depth in section 2.4. By including articles that solve the CLP exactly, relevant formulations, written constraints, and concepts were discovered that also applied to the PLP.

It was not required that the author learn every aspect of Currence Robotics' method. Nevertheless, a brief insight into relevant heuristic concepts was needed to under-

[1]This stability criterion was developed independently by Currence Robotics during the same time period as this master thesis. Hence, it is not discussed in this master thesis, and thus, section 3.4 considers the previous stability criterion used by Currence Robotics' heuristic method. Currence Robotics' new stability criterion is used in the experiments with their heuristic method.

stand its general approach. Hence, some articles solving the CLP heuristically were included in the search. Even more so when it was discovered that many of the articles solving the CLP heuristically also provided valuable discussions about practical constraints or other central PLP concepts. Finally, some articles that were mentioned by Bortfeldt and Wäscher [24] regarding similar problems were included for insight and explanation purposes.

In Frich [9], the practical constraints to Currence Robotics' PLP were outlined. Furthermore, Frich [9] discovered that there exists a limited number of mathematical formulations for the relevant practical constraints. Hence, almost all articles considering the relevant practical constraints were included. The article by Bortfeldt and Wäscher [24] was particularly useful in finding material for the practical constraints, providing an almost exhaustive overview of all practical constraints discussed in research up until 2013.

Table 2.1 outlines the most important articles used in this thesis. The articles are listed in five categories: Exact PLP, Heuristic PLP, Exact CLP, Heuristic CLP, Other. The following abbreviations have been used to reduce the table size: E = Exact method, H = Heuristic method, RPC = Relevant practical constraints, SS = static stability, DS = dynamic stability, LB = Load balance, MO = Multiple orientations, LBP = Load bearing constraint, and R = Robot arm constraint. All of these concepts are explained in chapter 2 and chapter 3.

Table 2.1: Overview of the most important articles for this thesis

Authors	Problem	Method	Objective Function	RPC	Article Type	Relevant Summary
Exact PLP						
Carpenter et al. [6] (1985)	PLP	-	-	SS	Propose SS	Propose SS for PLP along with reflections and insight
Vargas-Osorio et al. [4] (2016) analysis	PLP	-	-	ALL	Lit. review	PLP economic
Frich [9] (2021)	PLP	-	-	ALL	Lit. review	Relevant specialisation project
Heuristic PLP						
Hemminki et al. [36] (1998)	PLP	Online: First fit + Best fit	Min. # pallets	SS	Propose H	SS insight
Exact CLP						
Chen et al. [26] (1995)	CLP	E: Chen	Min. unused space	MO	Propose E	E used in thesis

Table 2.1: Overview of the most important articles for this thesis

Authors	Problem	Method	Objective Function	RPC	Article Type	Relevant Summary
Junqueira et al. [27] (2012)	CLP	E: Junqueira	Max. box volume	SS DS	Propose E	E insight
Junqueira et al. [37] (2012)	CLP	E: Junqueira	Max. box volume	SS	Propose E	E insight
Paquay et al. [38] (2016)	CLP	E: Paquay	Min. # pallets	SS LB MO	Propose E	Introduces multiple concepts and proposes several prac. constraints
Alonso et al. [39] (2017)	CLP	E: Alonso	Min. # pallets	LB	Propose E	Alternative E Model
Silva et al. [25] (2019)	CLP	E: Chen, Hifi, Tsai, Paquay, Fasano, Junqueira	Min. empty space	-	Lit. review + Benchmark E's	Benchmark different E's

Table 2.1: Overview of the most important articles for this thesis

Authors	Problem	Method	Objective Function	RPC	Article Type	Relevant Summary
Alonso et al. [40] (2019)	CLP	E: Alonso	Min. height	-	Propose prac. constraints	Propose DS concepts and provide LB and DS insight
Junqueira et al. [28] (2019)	2CPP	E: Junqueira	Max. box volume	-	Benchmark SS in 2D	Insight into SS from 2D perspective
Silva et al. [41] (2020)	CLP	E: Chen + 3D -Knapsack	Max. box volume	SS DS LB MO	Proposes solution scheme with E	Splits CLP into smaller pieces
Nascimento et al. [10] (2021)	CLP	E: Junqueira	Max. box volume	SS DS LB MO LBP	Propose SS DS LB MO LBP	Multiple practical constraints used in thesis
Cavone et al. [32] (2021)	CLP	-	Min. unused space	SS DS LB MO	Propose E SS DS LB MO	Simple practical constraints
Heuristic CLP						

Table 2.1: Overview of the most important articles for this thesis

Authors	Problem	Method	Objective Function	RPC	Article Type	Relevant Summary
Bischoff et al. [42] (1995)	CLP	H: Layer building	Max. box volume	SS DS LB MO	Lit. review + Propose H	Introduces full base support
Pisinger [43] (2002)	CLP	H: Tree search + Wall building	Min. depth	-	Propose H	Proposes stability concept
Eley [30] (2002)	CLP	H: Tree search	Min. # containers	-	Propose H	Introduces block building used in H by Currence Robotics
Boef et al. [44] (2003)	CLP	H: Constraints Prog.	-	R	Propose R	Robot insight
Moura et al. [45] (2005)	CLP	H: GRASP	Max. volume	SS	Propose H	SS insight
Parreño et al. [46] (2010)	CLP	H: Rand. Cons. Heuristic	Best volume + Best fit	-	Propose H	SS insight

Table 2.1: Overview of the most important articles for this thesis

Authors	Problem	Method	Objective Function	RPC	Article Type	Relevant Summary
Ceschia et al. [47] (2010)	CLP	H: Local search	Multiple objectives	SS	Propose H	SS insight
Weng et al. [29] (2010)	CLP	H: Greedy	Max. box volume	-	Creates CLP sub-problems	Understand H method of Currence Robotics and aspects of solving CLP
Ramos et al. [48] (2016) and proposed SS	CLP	H: Genetic algorithm	Max. box volume	SS	Propose SS	SS insight,
Zhao et al. [49] (2016)	CLP	-	-	-	Lit. review of H for CLP	Insight into CLP solution schemes
Ramos et al. [50] (2018)	CLP	H: Genetic algorithm	Max. box volume	LB	Propose H	LB insight
Araya et al. [8] (2020)	CLP	H: Beam search	Max. box volume + profit	-	Propose H	Insight into H of Currence Robotics

Table 2.1: Overview of the most important articles for this thesis

Authors	Problem	Method	Objective Function	RPC	Article Type	Relevant Summary
G.-Palacios et al [33] (2020)	CLP	H: Variable Ngbh. Search	Max. box volume + # of complete shipments	-	Propose H	New objective function for complete shipments
Gajda et al. [34] (2022)	CLP:	H: Rand. Cons. Heuristic	Max. box profit	SS LB MO	Propose H	Insight regarding E's
Other						
Bischoff et al. [51] (1998)	Packing	H: Layer building	-	LBP	Insight into weight	LBP insight
Padberg [52] (2000)	CLP	-	-	-	Propose E	Inspired visualisations
Wäscher et al. [7] (2007)	CLP + PLP	-	-	-	Lit. review	Proposes typology of packing problems
Martello et al. [53] (2007)	Packing	H: Tree search	Max. box volume	R	Propose H + General Packings	Robot insight

Table 2.1: Overview of the most important articles for this thesis

Authors	Problem	Method	Objective Function	RPC	Article Type	Relevant Summary
Egeblad et al. [23] (2009)	3KP	-	-	-	Propose H	Similar problem insight
Baldi et al. [22] (2012)	3BKP	-	-	-	Introduces 3BKP	Similar problem insight
Bortfeldt et al. [24] (2013)	CLP + PLP	-	-	SS DS LB MO	Lit. review	Gives overview of all constraints formulated as of 2013
Iori et al. [31] (2020)	PBP	H: Layer building	Min. # pallets	-	Proposes H	Similar problem insight

Chapter 3

Model

Researchers have proposed many different practical constraints for PLPs, and CLPs [24]. However, not all are relevant to Currence Robotics' PLP. To identify the ones that were, articles outlined in section 2.7 were consulted. In particular, the article by Bortfeldt and Wäscher [24].

Each section in this chapter is about a relevant practical constraint to Currence Robotics' PLP. However, before introducing any practical constraints, a basis of initial constants, variables, and inequalities is required. This basis is also referred to as *geometric constraints* and are what all practical constraints are formulated on top of. In addition, the MIP formulation requires an objective function, for which several are explored. Then follows the practical constraints, starting with Currence Robotics' multiple box orientations. Static- and dynamic stability constraints are also introduced - ensuring that boxes do not fall over when the pallet stands still or is moving. These stability constraints do not consider the weight of the stack when finding stability. Hence, load balancing constraints restricting the centre of gravity of the stack are introduced. Furthermore, Grab's robot arm constraint is discussed, but it is found that few formulations are viable. As an alternative, load-bearing constraints are explored. Some of the mentioned constraints above require that boxes are placed on top of other boxes or the pallet floor to be effective. Thus, the author proposes an on-top constraint. Finally, if the reader is interested, the end of the chapter contains a brief summary of other existing non-relevant practical constraints.

All of the introduced constraints are formulated within a three-dimensional Cartesian coordinate system $(x, y, z) \in \mathbb{R}^+$, which is seen in figure 3.1.

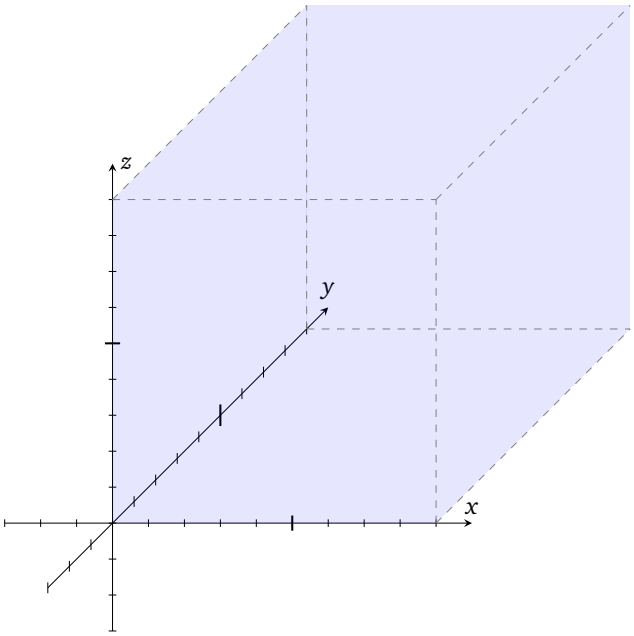


Figure 3.1: Initial solution space of Currence Robotics' PLP; $(x, y, z) \in \mathbb{R}^+$ indicated in blue.

Practical constraints must be formulated mathematically to be part of the MIP formulation of Currence Robotics' PLP. There may exist multiple mathematical formulations of a practical constraint. Ideally, mathematical formulations of practical constraints are precise and use little space. Hence, sections are structured so that the best formulations are obtained.

Initially, in a section, a practical constraint and its purpose are introduced. Then, recent literature and state of the art are reviewed for each constraint, as different implementations may exist. Time restrictions and constraint inviability made it impossible to formulate all of the reviewed implementations using mathematical inequalities. Thus, based on the state of the art review, a decision was made regarding what implementations to formulate mathematically. Consequently, formulations of mathematical inequalities are presented for the chosen constraints, either proposed by a researcher or the author. At least one concept is formulated for each identified relevant practical constraint.

The clarification between constraint inequality, worded constraint, and concept is vital for this thesis. *Constraint inequalities* are mathematical formulations necessary for adding a constraint to the MIP formulation of Currence Robotics' PLP. Many researchers only provide worded explanations to solutions. Thus, if a researcher has proposed a *worded constraint*, then no constraint inequalities of a practical constraint are proposed. Lastly, if a researcher has proposed a *concept*, then there exists an idea. Thus, both worded constraints and constraint inequalities have yet to be created.

Many of the concepts, worded constraints, and constraint inequalities of practical constraints used in the MIP formulation of Currence Robotics' PLP have previously been proposed by researchers. However, some are also proposed by the author. In table 3.1, an overview is of the practical constraints discussed in this thesis is provided. Additionally, three columns indicate who proposed the concept, the worded constraint, or the constraint inequalities for each practical constraint.

3.1 Geometric Constraints

The formulations of the geometric constraints provide a basis for other constraint- and objective function formulations. Three constraints are always enforced when researchers formulate geometric constraints for the CLP, or the PLP [24]:

1. The boxes must be placed within the boundaries of the pallet.
2. The boxes cannot overlap each other.
3. The boxes must be placed parallel to the "walls" of the pallet.

State of the art

Different formulations of geometric constraints are introduced by Chen et al. [26], Silva et al. [25], Paquay et al. [38], Alonso et al. [40], Junqueira et al. [37], Junqueira et al. [27], Cavone et al. [32], and Nascimento et al. [10] among others.

Silva et al. [25] compare and benchmark geometric constraints, all adapted to express the same input minimisation problem, against each other. Silva et al. [25] conclude that the best formulations of geometric constraints, regarding time and space, are the ones proposed by Chen et al. [26] and Paquay et al. [38], both originally intended for strongly heterogeneous boxes. However, the formulations proposed by Chen et al. [26] use slightly less space and find additional solutions to the formulations by Paquay et al. [38].

Practical constraints discussed in this thesis			
Category	Concept	Worded Constraint	Constraint Inequalities
Geometric	Geometric: Chen et al. [26]	-	Chen et al. [26]
Objective	Min. height: Alonso et al. [40] Pisinger [43]	-	The author
Static stability	Partial base support: Nascimento et al. [10]	Nascimento et al. [10]	The author
	Reduced partial base support: The author	The author	The author
	Multiple box constraint: Carpenter and Dowsland [6]	-	The author
Dynamic stability	Non-guillotine criterion: Carpenter and Dowsland [6]	-	The author
	Multiple box incentive: Carpenter and Dowsland [6] The author	-	The author
	Middle incentive: The author	-	The author
Load balancing	Load balancing: Nascimento et al. [10]	-	Nascimento et al. [10]
Robot arm	Robot-packable pattern: Boef et al. [44]	-	Boef et al. [44]
	Robot-packable pattern* Bortfeldt and Wäscher [24]	The author	-
Load-bearing	Number of boxes: Nascimento et al. [10]	Nascimento et al. [10]	The author
	Pressure per unit area: Nascimento et al. [10]	Nascimento et al. [10]	The author
	Partial base load-bearing: The author	-	The author
On-top	On top: The author	-	The author

Table 3.1: Practical constraints discussed in this thesis with the respective author of their concepts, worded constraints and constraint inequalities. (The author = Toralf Frich)

Different formulations also use different representations of solution space. Silva et al. [25] conclude that the discrete grid proposed by Junqueira et al. [27] is the best for weakly heterogeneous boxes. However, for strongly heterogeneous boxes, the continuous formulations of Chen et al. [26] and Paquay et al. [38] are deemed as the better.

Decision

Silva et al. [25] are the only researchers to benchmark MIP formulations of geometric constraints against each other. Thus, the geometric constraints formulated by Chen et al. [26] are chosen for implementation.

3.1.1 Formulation

The variables used to express the geometric constraints by Chen et al. [26] are listed in Table 3.2 and Table 3.3 along with their purpose. They are similar to the ones presented by Silva et al. [25]. However, to reduce the model size and eliminate redundancies, the variables that account for multiple pallets/containers are removed, as there is only one pallet in Currence Robotics' PLP. In addition, a variable for a box weight has been added.

$\mathscr{I}$	set of boxes
N	Total number of boxes in $\mathscr{I}$
p_i, q_i, r_i	length, width and height of box i $\forall i \in \mathscr{I}$
L, W, H	length, width and height of pallet
x_i, y_i, z_i	continuous variables indicating the coordinate of the front-left bottom corner of box i $\forall i \in \mathscr{I}$
w_i	the weight of box i $\forall i \in \mathscr{I}$

Table 3.2: List of input parameters used for geometric constraints

Equations (4) to (8) outline the domains of the variables introduced in Table 3.2 and Table 3.3.

l_{xi}, l_{yi}, l_{zi}	binary variables indicating whether the length of box i (dimension p_i) is parallel to the X-, Y- or Z-axis
w_{xi}, w_{yi}, w_{zi}	binary variables indicating whether the width of box i (dimension q_i) is parallel to the X-, Y- or Z-axis
h_{xi}, h_{yi}, h_{zi}	binary variables indicating whether the height of box i (dimension r_i) is parallel to the X-, Y- or Z-axis
$a_{ij}, b_{ij}, c_{ij}, d_{ij}, e_{ij}, f_{ij}$	binary variables indicating the relative position of boxes i and j (as seen from i, j can be placed a_{ij} = left, right, behind, in front, below, and above i)

Table 3.3: List of variables used for geometric constraints

$$x_i, y_i, z_i \geq 0 \ \forall i \in \mathscr{I} \tag{4}$$

$$l_{xi}, l_{yi}, l_{zi} \in \{0,1\} \ \forall i \in \mathscr{I} \tag{5}$$

$$w_{xi}, w_{yi}, w_{zi} \in \{0,1\} \ \forall i \in \mathscr{I} \tag{6}$$

$$h_{xi}, h_{yi}, h_{zi} \in \{0,1\} \ \forall i \in \mathscr{I} \tag{7}$$

$$a_{ij}, b_{ij}, c_{ij}, d_{ij}, e_{ij}, f_{ij} \in \{0,1\} \ \forall i, j \in \mathscr{I} \tag{8}$$

Using the parameters and variables defined in Table 3.2 and Table 3.3, geometric constraints are formulated in equations (9) to (17) [25, 26].

$$x_i + p_i l_{xi} + q_i w_{xi} + r_i h_{xi} \leq x_j + L(1 - a_{ij}) \; \forall i, j \in \mathscr{I} \tag{9}$$

$$x_k + p_k l_{xj} + q_k w_{xj} + r_k h_{xj} \leq x_i + L(1 - b_{ij}) \; \forall i, j \in \mathscr{I} \tag{10}$$

$$y_i + q_i w_{yi} + r_i h_{yi} + p_i l_{yi} \leq y_j + W(1 - c_{ij}) \; \forall i, j \in \mathscr{I} \tag{11}$$

$$y_k + q_k w_{yj} + r_k h_{yj} + p_k l_{yj} \leq y_i + W(1 - d_{ij}) \; \forall i, j \in \mathscr{I} \tag{12}$$

$$z_i + r_i h_{zi} + p_i l_{zi} + q_i w_{zi} \leq z_j + H(1 - e_{ij}) \; \forall i, j \in \mathscr{I} \tag{13}$$

$$z_k + r_k h_{zj} + p_k l_{zj} + q_k w_{zj} \leq z_i + H(1 - f_{ij}) \; \forall i, j \in \mathscr{I} \tag{14}$$

In equations (9) to (14), the non-overlapping constraints between boxes are formulated using *big-M* constraints for binary behaviour. Box i and box j are restricted to not overlap by ensuring that the space occupied by box i does not overlap with the space occupied by box j.

Depending on the placement of i in relation to j, the big-M constraints are activated. An equation is activated if the corresponding relative positional binary variable between box i and box j is *true* (a_{ij}, b_{ij} etc.). If a relative positional binary variable is true, then M is neglected from that equation. But if it is *false*, then M is activated, ensuring that a box is placed within the constant M coordinate-value.

Any box is also restricted by equations (15) to (17), ensuring that a box is not placed outside the pallet boundaries. Hence, the right side values of equations (9) to (14) cannot be greater than the dimensions of the pallet.

$$x_i + p_i l_{xi} + q_i w_{xi} + r_i h_{xi} \leq L \; \forall i \in \mathscr{I} \tag{15}$$

$$y_i + p_i l_{yi} + q_i w_{yi} + r_i h_{yi} \leq W \; \forall i \in \mathscr{I} \tag{16}$$

$$z_i + p_i l_{zi} + q_i w_{zi} + r_i h_{zi} \leq H \; \forall i \in \mathscr{I} \tag{17}$$

L and W are given by the pallet floor. However, as there is no roof on a pallet, H can be adjusted - allowing for flexibility in the implementation. Two boxes adhering to the constraints of equations (9) to (17) can be seen in Figure 3.2.

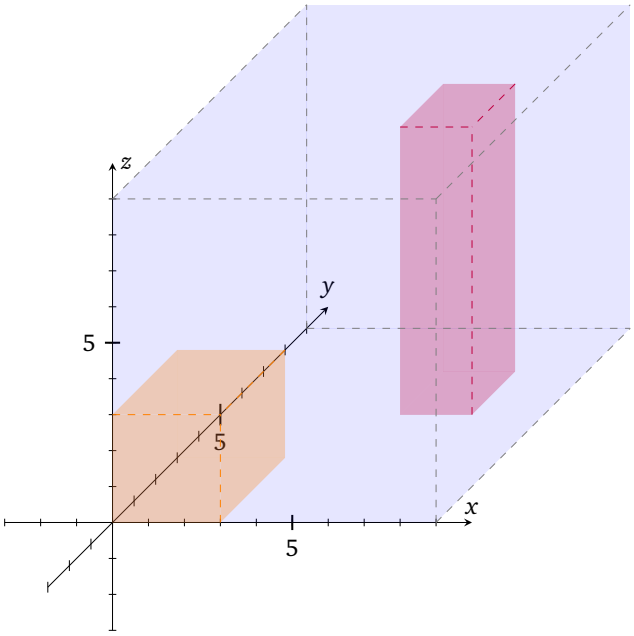


Figure 3.2: Two boxes placed onto a pallet adhering to the geometrical constraints of the model.

3.2 Objective Function

Multiple objective functions have been proposed for CLP- and PLP problems. However, to the best of the author's knowledge, there exists little research regarding what objective functions generate the best stacks. One of the reasons for this might be that it depends on the problem instance. In this section, the reader is presented with a brief introduction to the objective functions of CLP and PLP. In addition, an objective function is formulated as part of the MIP formulation of Currence Robotics' PLP.

State of the art

Alonso et al. [40] and Pisinger [43] propose another objective function for problem instances that only have one container: *minimise the height* of the stack. Furthermore, Silva et al. [25], Paquay et al. [38], and Chen et al. [26] propose to *minimising empty space*. These input minimisation problems minimise the number of containers used to load a set of given boxes. And, because introducing another container for stacking boxes also introduces more empty space, fewer containers results in

less empty space. They implement this objective using Equation 18.

$$min\ LWH - \sum_{i \in \mathscr{I}} p_i q_i r_i \tag{18}$$

Output maximisation problems are the most common in CLP- and PLP literature [24]. And, researchers focusing on ouput maximisation seem to agree on an objective function: *maximise the total volume of all the boxes*. This objective function is chosen by Junqueira et al. [27], Junqueira et al. [37], Junqueira and Queiroz [28] and Nascimento et al. [10] for exact methods and Araya et al. [8] (multi-objective), Bischoff and Ratcliff [42], Ramos et al. [48], and Weng et al. [29] amongst others for heuristic methods.

3.2.1 Minimising stack height

It would not be possible to minimise empty space in Currence Robotics' PLP. This is due to Currence Robotics' PLP only having one pallet and because the total volume of the boxes is smaller than the total volume of the pallet. Thus, the empty space is constant. The same argument applies to the output maximisation objective function. As the volume of the boxes is constant, it will not work to maximise the total box volume. Hence, minimising the height of the stack is the only viable objective function from literature, so it is chosen for our model.

Minimising the height of the stack is done by minimising the max $z_i + r_i$ value of box $i\ \forall i \in \mathscr{I}$. The MIP formulation can be seen in Equation 20 using the variable *max_height* as the objective variable.

$$H \geq max\_height \geq 0 \tag{19}$$

$$max\_height \geq [z_i + r_i]\ \forall i \in \mathscr{I} \tag{20}$$

3.3 Multiple Orientations

Orientation constraints ensure boxes can be rotated. However, boxes must be parallel to the walls of the pallet, and thus, boxes are only allowed 90 ° rotations [24]. In a significant study on practical constraints for the CLP and PLP, Bortfeldt and

Wäscher [24] found that of the 163 articles analysed, 115 of them mentioned orientation constraints.

For the multiple orientation constraint, there is no discussion of state of the art, as the rotational variables are based on the geometric constraints introduced in subsection 3.1.1.

In this section, constraint inequalities that allow *all rotations* are presented for this topic's completeness. In addition, orientation constraints for Grab are introduced.

3.3.1 All orientations

Allowing a box to be placed using all possible orientations could increase the number of possible solutions, as items could be placed in six different ways. However, as the boxes are rectangular, there would, in reality, only be three different options of orientation. Allowing for more rotations would make the problem harder to solve, as a larger model would be generated. To formulate a model allowing all rotations, equations (21) to (26) would need to be implemented [10].

$$l_{xi} + l_{yi} + l_{zi} = 1, \forall i \in \mathscr{I} \tag{21}$$

$$w_{xi} + w_{yi} + w_{zi} = 1, \forall i \in \mathscr{I} \tag{22}$$

$$h_{xi} + h_{yi} + h_{zi} = 1, \forall i \in \mathscr{I} \tag{23}$$

$$l_{xi} + w_{xi} + h_{xi} = 1, \forall i \in \mathscr{I} \tag{24}$$

$$l_{yi} + w_{yi} + h_{yi} = 1, \forall i \in \mathscr{I} \tag{25}$$

$$l_{zi} + w_{zi} + h_{zi} = 1, \forall i \in \mathscr{I} \tag{26}$$

3.3.2 Currence Robotics' orientations

In Currence Robotics' PLP, Grab picks up boxes standing on shelves in the warehouse. Then, it can rotate those boxes 90 ° about the z-axis, implying that a box has two distinct dimensions sets. The formulation of these orientation constraints can be seen in equations (27) to (32), while a rotation of a box in the z-axis can be seen in Figure 3.3.

$$l_{xi} + l_{yi} = 1, \forall i \in \mathscr{I} \tag{27}$$

$$w_{xi} + w_{yi} = 1, \forall i \in \mathscr{I} \tag{28}$$

$$h_{zi} = 1, \forall i \in \mathscr{I} \tag{29}$$

$$l_{xi} + w_{xi} = 1, \forall i \in \mathscr{I} \tag{30}$$

$$l_{yi} + w_{yi} = 1, \forall i \in \mathscr{I} \tag{31}$$

$$h_{xi}, h_{yi}, l_{zi}, w_{zi} = 0, \forall i \in \mathscr{I} \tag{32}$$

In equations (27) to (31) a box's length or width must be parallel to either the x- or y-axis, while the height must be parallel to the z-axis. Equation (32) reduces the size of the generated MIP models, as its variables will always equal 0 due to the other orientation constraints.

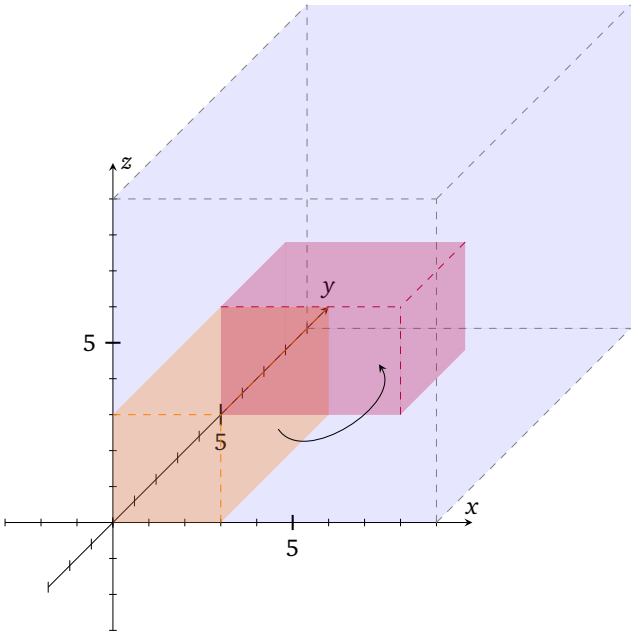


Figure 3.3: A rotation of a box in the z-axis.

3.4 Static stability

An ideal *static stability* constraint (also referred to as vertical stability [24]) ensures that boxes placed in a stack do not fall while the pallet is at a standstill. More specifically, a static stability implementation for any box i attempts to suffice sufficient underlying support from other boxes or the pallet floor, depending on i's location in the stack.

A static stability constraint causes gravity to be established within a MIP model. Moreover, a box adhering to a static stability constraint cannot be placed anywhere without sufficient static support. If there are no feasible solutions to a static stability constraint, then no solutions in which the boxes would not fall to the ground exist.

In the review presented by Bortfeldt and Wäscher [24], static stability constraints and *dynamic stability* (see section 3.5) constraints are considered in 61 articles (37.4 % of all reviewed). Hence, stability constraints are highly relevant for current problem formulations.

State of the art

Pisinger [43] argues that stability is guaranteed when a high space utilisation of the available volume of the pallet, C_{max}, is achieved. However, Bortfeldt and Wäscher [24] explains how that argument is, in general, only valid when the problem is an output maximisation problem. As in an output maximisation problem instance, the sum of the box volumes, C, is usually greater than the pallet's available volume: $C > C_{max}$. In an input minimisation problem instance, $C \ll C_{max}$ can occur. Hence, further requirements than a high volume utilisation are necessary to guarantee static stability.

There have been multiple attempts at modelling static stability [26]. One static stability constraint concept introduced by Carpenter and Dowsland [6] require that a percentage of a box' base is supported by multiple boxes. E.g., require that at least 5 % of a box' base is supported by two boxes below. There has been little research done in exploring the effects of this constraint. But, it is not impossible that an optimal solution can contain such a configuration.

Bischoff and Ratcliff [42] require *full base support*. Full base support occurs when 100 % of the base of a box is supported by other boxes or by the pallet floor [24] [42]. An alternative full base support concept is defined by Ceschia and Schaerf [47]; requiring that a box i's dimensions must be smaller than or equal to the dimensions of box j for i to be placed on top of j. Thus full base support is

ensured. Ramos et al. [48] argue that full base support does not guarantee a statically stable stack of boxes, but rather a stack where all boxes have full base support. In addition, they argue that full base support reduces algorithmic efficiency for problems with strongly heterogeneous boxes [48].

As an alternative to full base support, researchers have proposed a *partial base support* criterion [6, 10]. Partial base support requires that a percentage of a box' base is supported by underlying boxes. In addition, if a box is placed on the pallet floor, it is regarded as having 100 % base support. Boxes that adhere to a partial base support constraint can be seen in figure 3.4.

Ramos et al. [48] outline partial base support requirements by researchers for ensuring sufficient static stability. They find that the percentage of base support required in different implementations range between 55 % and 95 % [48]. In an article not considered by Ramos et al. [48]; Hemminki et al. [36] suggest that 70 % partial base support ensures sufficient static support for PLPs, given that the boxes are wrapped in plastic before being shipped. However, different percentages of base support could be preferred depending on specific problem instances. E.g., a higher percentage of base support could be used when placing *fragile boxes* (boxes that contain items that easily break [38]).

Ramos et al. [48] outline a *static mechanical equilibrium* constraint to ensure static stability. The static mechanical equilibrium constraint requires that a box is either placed on the pallet floor or: "*That the sum of external forces acting on the box is zero and; that the sum of torque exerted by the external forces is zero*" - [48]. To enforce this constraint, Ramos et al. [48] introduce a *support polygon*. A support polygon is an area made up of boxes located beneath box i, in which i's centre of gravity must lie inside. Its use ensures that the static mechanical equilibrium constraint is satisfied. However, there is yet to be proposed constraint inequalities for the support polygon, as the solution presented by Ramos et al. [48] is heuristic. A support polygon constraint can be seen in figure 3.5.

Based on a similar theory to that of Ramos et al. [48], Paquay et al. [38] propose constraint inequalities for a static stability criterion. In the criterion, Paquay et al. [38] formulate constraints ensuring that three of the four vertices of the base of any box i are covered by other boxes. Hence, the criterion ensures that the centre of gravity of a box i is located within i's *support basis*, meaning the convex hull of all i's contact points with boxes below. However, Paquay et al. [38] do not conclude regarding its effectiveness. In figure 3.6, a support basis for a box is shown.

In Currence Robotics heuristic method, a more advanced static stability requirement

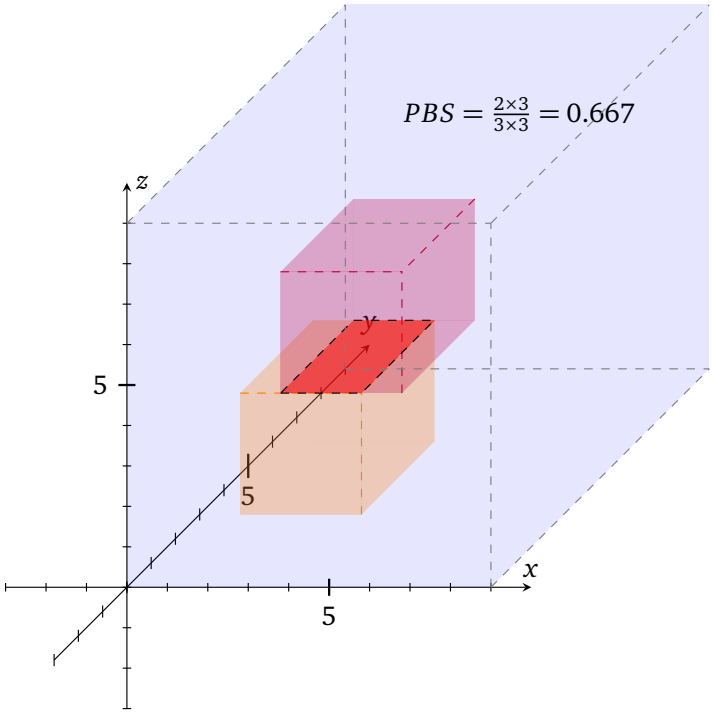


Figure 3.4: Purple box placed on top of the orange box. The partial base support (PBS) is calculated using the area of contact between the two boxes as seen in the equation. The area of contact is coloured in red.

is used. In addition to preferring boxes with much base support, they constrain a box's maximum allowed tilt angle in both x- and y-direction. An example can be seen in figure 3.7.

Decision

The support polygon proposed by Ramos et al. [48] and Currence Robotics' angle-solution are both promising static stability constraints. However, no constraint inequalities exist for either. Currence Robotics' solution also violates the geometric constraints ensuring boxes are parallel to the walls of the pallet. Thus, its use would require a reformulation of the geometric constraints by Chen et al. [26]. Paquay et al. [38] propose constraint inequalities for the support basis, but no concluding remarks regarding its viability. There also does not exist any conclusions regarding the multiple box constraint by Carpenter and Dowsland [6]. However, it has

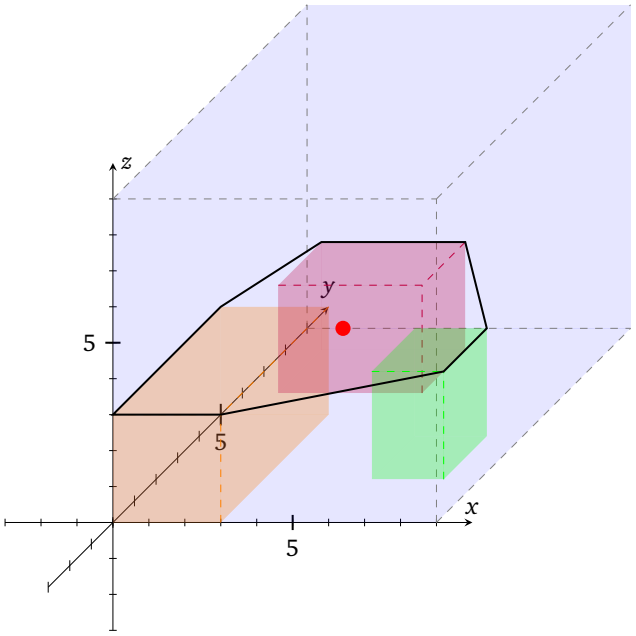


Figure 3.5: The support polygon proposed by Ramos et al. [48], where the centre of gravity (COG) of a box must be placed within the area outlined by the black continuous line enveloping all boxes. The red dot at coordinate $(4, 3, 4)$ indicate a possible position of a box' COG.

the advantage of being simpler to formulate. Thus, the multiple box constraint by Carpenter and Dowsland [6] is explored further in section 3.4.3.

Most researchers using static stability constraints are looking to partial-, or full base support [24]. One reason for this might be that researchers have reached some conclusions regarding its usefulness. In addition, few other constraint inequalities exist for static stability constraints. Thus, partial base support is chosen for implementation. Junqueira et al. [27] propose constraint inequalities for a partial base support calculation using geometric constraints proposed by Junqueira et al. [27] as a basis. Reformulating the constraints by Junqueira et al. [27] to the geometric constraints proposed by Chen et al. [26] would require a significant adaptation. Thus, the worded constraints to the partial base support by Nascimento et al. [10] were preferred, as they would be easier to formulate on top of the basis by Chen et al. [26]. Mathematical formulations of constraints inequalities for the partial base support static stability criterion by Nascimento et al. [10] are presented in section 3.4.1.

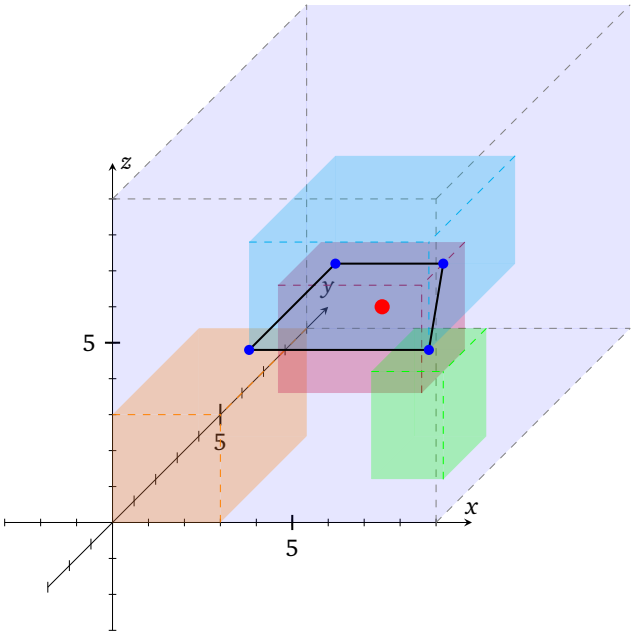


Figure 3.6: The support basis proposed by Paquay et al. [38], where the centre of gravity (COG) of a box must be placed within the convex hull made up of the black continuous lines. The contact points enveloping the convex hull are indicated by the small blue dots. The blue box's COG is located at $(4.5, 3, 5)$, and is indicated by the red dot. In this figure, three of the four vertices of the base of the blue box are covered by boxes below.

While formulating the partial base support constraint by Nascimento et al. [10], it became apparent that it was possible to reduce its size. Hence, this new size-reducing formulation is presented in section 3.4.2.

3.4.1 Partial Base Support

The worded partial base support constraint proposed by Nascimento et al. [10] included no constraint inequalities. Thus, the constraint inequalities formulated in this subsection are created from the ground up.

Two continuous variables are needed to introduce the worded partial base support constraint Nascimento et al. [10]. These can be seen in table 3.4, indicating the space used by any box in the x- and y-dimension.

The worded partial base support constraint by Nascimento et al. [10] is introduced

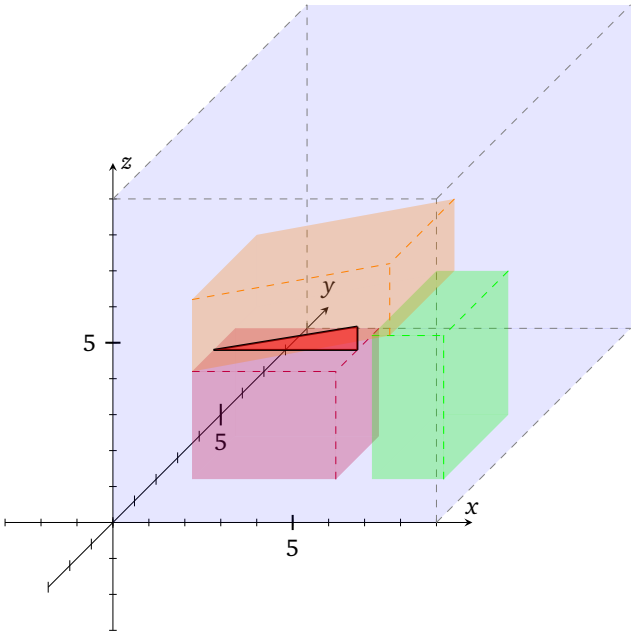


Figure 3.7: Currence Robotics' solution restricting the angle of boxes lying on top of other boxes visualised. The area marked in red represent the angle on the orange box.

in Code listing 3.1. It first checks whether a box i is placed above another box j. Then, the overlaps between boxes i, j in the x- and y-axis are found and multiplied, giving the partial base support from j onto i. This partial base support from j is added to the total partial base support of i. Finally, the partial base support constraint ensures that the total base support for i is greater than a given percentage of its base size.

x_cover_i	$l_{xi} * p_i + w_{xi} * q_i \ \forall i \in \mathscr{I}$
y_cover_i	$l_{yi} * p_i + w_{yi} * q_i \ \forall i \in \mathscr{I}$

Table 3.4: Variables in worded partial base support constraint by Nascimento et al. [10]

$$x\_cover_i, y\_cover_i \geq 0 \ \forall i \in \mathscr{I} \tag{33}$$

```
if (z[i] == z[j] + r[j] &&
    x[j] < x[i] + x_cover[i] &&
    x[i] < x[j] + x_cover[j] &&
    y[j] < y[i] + y_cover[i] &&
    y[i] < y[j] + y_cover[j])

    l_ij = min(x[i] + x_cover[i], x[j] + x_cover[j]) - max(x[i], x[j])
    w_ij = min(y[i] + y_cover[i], y[j] + y_cover[j]) - max(y[i], y[j])

    tot_pbs_box_i += l_ij * w_ij
end

@constraint(tot_pbs_box_i >= pbs_var * p_i * q_i)
```

Code listing 3.1: Worded partial base support constraint by Nascimento et al. [10]

Conditions

The conditional statements are formulated first. These are realised by introducing a binary variable for each, indicating whether the conditional is satisfied or not. In addition, one binary variable (if_pbs_{ij}) keeps track of the status of the five conditionals, returning *true* if all of the others return *true*. The binary variables can be seen in table 3.5.

$if1_{pbs_{ij}}$	indicating if $z_i == z_j + r_j \; \forall i,j \in \mathscr{I}$
$if2_{pbs_{ij}}$	indicating if $x_j < x_i + x\_cover_i \; \forall i,j \in \mathscr{I}$
$if3_{pbs_{ij}}$	indicating if $x_i < x_j + x\_cover_j \; \forall i,j \in \mathscr{I}$
$if4_{pbs_{ij}}$	indicating if $y_j < y_i + y\_cover_i \; \forall i,j \in \mathscr{I}$
$if5_{pbs_{ij}}$	indicating if $y_i < y_j + y\_cover_j \; \forall i,j \in \mathscr{I}$
if_pbs_{ij}	*AND*-ing the variables above

Table 3.5: Binary variables indicating the conditions' in the pseudo code's status

$$if1_{pbs_{ij}}, if2_{pbs_{ij}}, if3_{pbs_{ij}}, if4_{pbs_{ij}}, if5_{pbs_{ij}}, if\_pbs_{ij} \in 0,1 \; \forall i,j \in \mathscr{I} \tag{34}$$

Finding $if\,1_{pbs_{ij}}$ is not trivial, as it requires checking whether two variables; z_i, z_j+r_j, are equal: $z_i == z_j + r_j$. This is not possible in a MIP, as it relies on knowing the value of z_i and $z_j + r_j$ before the solver terminates. Instead we check for $z_j + r_j - \epsilon \leq z_i \leq z_j + r_j \epsilon$ for a small ϵ using big-M constraints. These new inequalities force two new variables; $if\,1_{pbs_{ij}}\_leq$, and $if\,1_{pbs_{ij}}\_geq$ (one for each inequality) to be *true* if z_i lies within the given interval. For Currence Robotics' PLP, the big-M is realised as H, as it is the upper bound for z_i and $z_j + r_j$. The two inequalities are enforced using equations (35) to (38).

$$H \times if\,1_{pbs_{ij}}\_leq \geq z_i + \epsilon - z_j - r_j \;\forall i, j \in \mathscr{I} \tag{35}$$

$$H \times (1 - if\,1_{pbs_{ij}}\_leq) \geq z_j + r_j - z_i - \epsilon \;\forall i, j \in \mathscr{I} \tag{36}$$

$$H \times if\,1_{pbs_{ij}}\_geq \geq z_j + r_j - z_i + \epsilon \;\forall i, j \in \mathscr{I} \tag{37}$$

$$H \times (1 - if\,1_{pbs_{ij}}\_geq) \geq z_i - \epsilon - z_j - r_j \;\forall i, j \in \mathscr{I} \tag{38}$$

If both $if\,1_{pbs_{ij}}\_leq$ and $if\,1_{pbs_{ij}}\_geq$ are *true*, then the inequality; $z_j + r_j - \epsilon \leq z_i \leq z_j + r_j \epsilon$ returns *true*. However, it is not allowed to multiply the binary variables directly into other equations, as this would make the program non-linear. Hence, $if\,1_{pbs_{ij}}$ is used to *AND* $if\,1_{pbs_{ij}}\_leq$ and $if\,1_{pbs_{ij}}\_geq$. Equation (39) is formulated to express this *AND*.

$$if\,1_{pbs_{ij}}\_leq + if\,1_{pbs_{ij}}\_geq \geq 2 \times if\,1_{pbs_{ij}} \;\forall i, j \in \mathscr{I} \tag{39}$$

Constraints for finding $if\,2_{pbs_{ij}}$ to $if\,5_{pbs_{ij}}$ also use big-M constraints. As an inequality like $<$ or $>$ can't be formulated using MIP, a similar trick to the one above is used: introduce a small ϵ to modify the inequalities. $<$ becomes $\leq -\epsilon$ and $>$ becomes $\geq +\epsilon$. Using this trick, we can formulate the conditional constraints for $if\,2_{pbs_{ij}}$ to $if\,5_{pbs_{ij}}$ with equations (40) to (47).

$$L \times (1 - if2_{pbs_{ij}}) \geq x_j + \epsilon - x_i - x\_cover_i \ \forall i, j \in \mathscr{I} \tag{40}$$

$$L \times if2_{pbs_{ij}} \geq x_i + x\_cover_i - x_j - \epsilon \ \forall i, j \in \mathscr{I} \tag{41}$$

$$L \times (1 - if3_{pbs_{ij}}) \geq x_i + \epsilon - x_j - x\_cover_j \ \forall i, j \in \mathscr{I} \tag{42}$$

$$L \times if3_{pbs_{ij}} \geq x_j + x\_cover_j - x_i - \epsilon \ \forall i, j \in \mathscr{I} \tag{43}$$

$$W \times (1 - if4_{pbs_{ij}}) \geq y_j + \epsilon - y_i - y\_cover_i \ \forall i, j \in \mathscr{I} \tag{44}$$

$$W \times if4_{pbs_{ij}} \geq y_i + y\_cover_i - y_j - \epsilon \ \forall i, j \in \mathscr{I} \tag{45}$$

$$W \times (1 - if5_{pbs_{ij}}) \geq y_i + \epsilon - y_j - y\_cover_j \ \forall i, j \in \mathscr{I} \tag{46}$$

$$W \times if5_{pbs_{ij}} \geq y_j + y\_cover_j - y_i - \epsilon \ \forall i, j \in \mathscr{I} \tag{47}$$

A single variable; if_pbs_{ij}, is used to *AND* all of the conditionals from the worded constraints in Code listing 3.1, similar to what was done in equation (39). The formulation of this *AND* is done in equation (48).

$$if1_{pbs_{ij}} + if2_{pbs_{ij}} + if3_{pbs_{ij}} + if4_{pbs_{ij}} + if5_{pbs_{ij}} \geq 5 \times if\_pbs_{ij} \ \forall i, j \in \mathscr{I} \tag{48}$$

Thus, any pair of boxes (i, j) are subject to partial base support calculation given that the conditions from table 3.5 are true, indicated by if_pbs_{ij}.

Overlap

The next step in the partial base support calculation is to find the overlap between boxes that satisfy the conditions from above. And given a pair of boxes (i, j), the overlap between them in the x- and y-axis can be calculated. A visualisation of three boxes and the overlap between the pairs of them can be seen in figure 3.8.

Two continuous variables introduced in table 3.6 are used to calculate the size of the overlaps: $l_{pbs_{ij}}$ and $w_{pbs_{ij}}$. These represent the size of the overlap in each axis

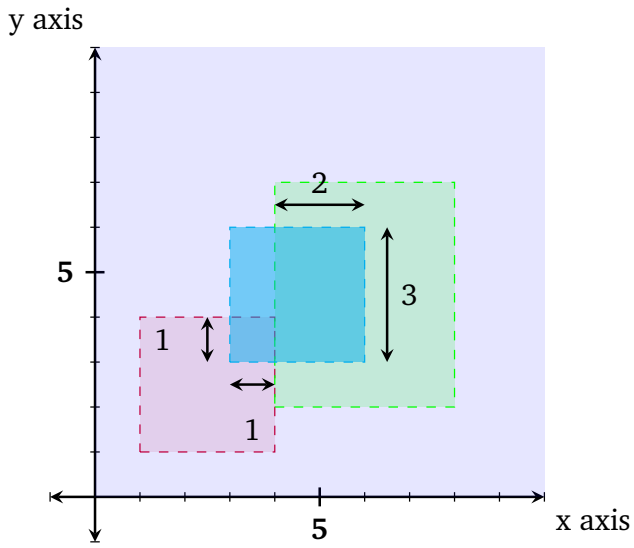


Figure 3.8: The blue box is standing on top of the green and purple box. The size of the overlaps between the boxes in the x- and y-axis is are marked.

$l_{pbs_{ij}}$	The overlap between box j and i in the $x-axis$ $\forall i,j \in \mathscr{I}$
$w_{pbs_{ij}}$	The overlap between box j and i in the $y-axis$ $\forall i,j \in \mathscr{I}$

Table 3.6: Variables indicating overlap between two boxes in each axis

for any two boxes. In Code listing 3.2, the pseudocode for finding overlap can be seen.

$$l_{pbs_{ij}}, w_{pbs_{ij}} \geq 0 \; \forall i,j \in \mathscr{I} \tag{49}$$

```
l_ij = min(x[i] + x_cover[i], x[j] + x_cover[j]) - max(x[i], x[j])
w_ij = min(y[i] + y_cover[i], y[j] + y_cover[j]) - max(y[i], y[j])
```

Code listing 3.2: Overlap in each axes

As seen in Code listing 3.2, a formulation of the pseudo code requires a max- and a min-function for each axis. Hence, four new continu-

ous variables representing max- and min function values are introduced: $max\_x_{pbs_{ij}}, max\_y_{pbs_{ij}}, min\_x_{pbs_{ij}}, min\_y_{pbs_{ij}}$. Equations (50) to (54) are used to find the value of each of those variables.

$$max\_x_{ij}, max\_y_{ij}, min\_x_{ij}, min\_y_{ij} \geq 0 \; \forall i, j \in \mathscr{I} \tag{50}$$

$$max\_x_{ij} >= [x_i, x_j] \; \forall i, j \in \mathscr{I} \tag{51}$$

$$max\_y_{ij} >= [y_i, y_j] \; \forall i, j \in \mathscr{I} \tag{52}$$

$$[x_i + x\_cover[i], x_j + x\_cover[j]] >= min\_x_{ij} \; \forall i, j \in \mathscr{I} \tag{53}$$

$$[y_i + y\_cover[i], y_j + y\_cover[j]] >= min\_y_{ij} \; \forall i, j \in \mathscr{I} \tag{54}$$

In equations (50) to (54), the constraints provide a lower bound for the max-variables and an upper bound for the min-variables. Hence, the solver can choose greater values for the max-variables and smaller ones for the min-variables. But in doing so, the resulting overlap becomes smaller, leading to less partial base support.

$l_{pbs_{ij}}$ and $w_{pbs_{ij}}$ are calculated in equation (55) and equation (56) using the formulation described in the pseudo code from Code listing 3.2. These represent the overlap between two boxes in the x- and the y-dimension.

$$l_{pbs_{ij}} = min\_x_{ij} - max\_x_{ij} \; \forall i, j \in \mathscr{I} \tag{55}$$

$$w_{pbs_{ij}} = min\_y_{ij} - max\_y_{ij} \; \forall i, j \in \mathscr{I} \tag{56}$$

Multiplication

To obtain the partial base support of box i from box j, the overlap in both axes need to be multiplied: $l_{pbs_{ij}} \times w_{pbs_{ij}} \; \forall i, j \in \mathscr{I}$. However, multiplication is impossible for a linear MIP model as it would introduce non-linear constraints. Instead, a linear approximation of the product of $l_{pbs_{ij}} \times w_{pbs_{ij}}$ is performed. Certain theoretical aspects of this linearisation have been covered in section 2.3 and are based of the work by Williams [19] and Bisschop [20].

The method by Williams [19] and Bisschop [20] require that a linearisation of a product can only be done to separable functions, and not directly on $l_{pbs_{ij}} \times w_{pbs_{ij}}$.

So, the first step is to separate the variables. This is done by introducing two new expressions in equation (57) and equation (58): $u1_{ij}$, $u2_{ij}$. As seen in equation (59), the difference between each squared represents the value of $l_{pbs_{ij}} \times w_{pbs_{ij}}$.

$$u_{1ij} = \frac{1}{2}(l_{pbs_{ij}} + w_{pbs_{ij}}) \; \forall i, j \in \mathscr{I} \tag{57}$$

$$u_{2ij} = \frac{1}{2}(l_{pbs_{ij}} - w_{pbs_{ij}}) \; \forall i, j \in \mathscr{I} \tag{58}$$

$$u_{1ij}^2 - u_{2ij}^2 = l_{pbs_{ij}} \times w_{pbs_{ij}}, \forall i, j \in \mathscr{I} \tag{59}$$

No expressions can be squared in a MIP model, as that would introduce non-linear constraints. Hence, two new continuous linear variables are introduced in table 3.7 representing the piecewise linear function values of u_{1ij}^2 and u_{2ij}^2.

z_{1ij}	A linear variable used to approximate u_{1ij}^2
z_{2ij}	A linear variable used to approximate u_{2ij}^2

Table 3.7: New variables to approximate u_{1ij}^2 and u_{2ij}^2

$$z_{1ij}, z_{2ij} \geq 0 \; \forall i, j \in \mathscr{I} \tag{60}$$

Hence, instead of using equation (59) for finding the partial base support onto i from j, equation (61) is used.

$$z_{1ij} - z_{2ij} = l_{pbs_{ij}} \times w_{pbs_{ij}}, \forall i, j \in \mathscr{I} \tag{61}$$

As discussed in section 2.3, the segments of the piecewise linear function are defined on some domain in an x-axis. And, as the maximum possible overlap is restricted by the *max* of L and W (a box can be no greater in size than the pallet floor), the piecewise linear function domain is defined from 0 to $max(L, W)$. In between these domain bounds, equally spaced intermediate x-values are added. Together, they define the domain of each segment in the piecewise linear function (the black dots in figure 2.3).

The approximation error is kept sufficiently low by adding 8 equally spaced intermediate values (10 domain values in total). The numerical value of this error is later discussed in section 3.4.1.

A constant representing a function value (y-value) is introduced for each corresponding domain value (x-value) in table 3.8. These function values are calculated using equations (57) to (59). The obtained function values are constants defined prior to solving the MIP, and so they can be squared ($f^2_{u1_{ijk}}$, $f^2_{u2_{ijk}}$) to get the piecewise linear function value of z_{1ij} and z_{2ij}.

In equations (62) to (69), function values are calculated for the domain's bounds as an example. The intermediary domain values' function values are calculated equally but with different domain values.

$f_{u_{1ijk}}$	Constant function value of u_{1ij} for a given intermediate x-value in the domain 0 to $max(L,W)$ $\forall i,j \in \mathscr{I}, k \in 1..10$
$f_{u_{2ijk}}$	Constant function value of u_{2ij} for a given intermediate x-value in the domain 0 to $max(L,W)$ $\forall i,j \in \mathscr{I}, k \in 1..10$
$f_{z_{1ijk}}$	Constant function value of z_{1ij} for a given intermediate x-value in the domain 0 to $max(L,W)$ $\forall i,j \in \mathscr{I}, k \in 1..10$
$f_{z_{2ijk}}$	Constant function value of z_{2ij} for a given intermediate x-value in the domain 0 to $max(L,W)$ $\forall i,j \in \mathscr{I}, k \in 1..10$

Table 3.8: Constants representing the pre-defined function values in the piecewise linear function

$$f_{u_{1ij1}} = \frac{1}{2}(0+0) \ \forall i,j \in \mathscr{I} \tag{62}$$

$$f_{u_{1ij10}} = \frac{1}{2}(L+W) \ \forall i,j \in \mathscr{I} \tag{63}$$

$$f_{u_{2ij1}} = \frac{1}{2}(0-max(L,W)) \ \forall i,j \in \mathscr{I} \tag{64}$$

$$f_{u_{2ij10}} = \frac{1}{2}(max(L,W)-0) \ \forall i,j \in \mathscr{I} \tag{65}$$

$$f_{z_{1ij1}}lower = (u_{1ij}lower)^2 \ \forall i,j \in \mathscr{I} \tag{66}$$

$$f_{z_{1ij10}}upper = (u_{1ij}upper)^2 \ \forall i,j \in \mathscr{I} \tag{67}$$

$$f_{z_{2ij1}}lower = (u_{2ij}lower)^2 \ \forall i,j \in \mathscr{I} \tag{68}$$

$$f_{z_{2ij10}}upper = (u_{2ij}upper)^2 \ \forall i,j \in \mathscr{I} \tag{69}$$

Weights are introduced to ensure that the approximated function values lie on a segment line (see section 2.3). There is one weight per domain value, and there are two ranges of weights for each pair of boxes: one for u_{1ij} and z_{1ij}, and one for u_{2ij} and z_{2ij}. To ensure the approximated function values lie on a segment line, equation (70) is enforced.

λ_{1ijk}	Weights used in the approximation of u_{1ij} and z_{1ij} $\forall i,j \in \mathscr{I}, k \in 1..10$
λ_{2ijk}	Weights used in the approximation of u_{2ij} and z_{1ij} $\forall i,j \in \mathscr{I}, k \in 1..10$

Table 3.9: Constants needed for linear approximation.

$$1 \geq \lambda_{1ijk} \geq 0 \ \forall i,j \in \mathscr{I}, k \in 1..10 \tag{70}$$

$$1 \geq \lambda_{2ijk} \geq 0 \ \forall i,j \in \mathscr{I}, k \in 1..10 \tag{71}$$

Each of the ranges of weights $[\lambda_{ij1}...\lambda_{ij10}]$ are constrained by an SOS2 constraint. The SOS2 is built-in for most optimisation programming languages. Thus, equation (72) is implemented for all of the ranges defined in table 3.9.

$$[\lambda_{1ij1}, \lambda_{1ij2}, ..., \lambda_{1ij10}] \in SOS2 \; \forall i, j \in \mathscr{I} \tag{72}$$

$$[\lambda_{2ij1}, \lambda_{2ij2}, ..., \lambda_{2ij10}] \in SOS2 \; \forall i, j \in \mathscr{I} \tag{73}$$

$$\sum_{k=1}^{10} \lambda_{1ijk} = 1 \; \forall i, j \in \mathscr{I} \tag{74}$$

$$\sum_{k=1}^{10} \lambda_{2ijk} = 1 \; \forall i, j \in \mathscr{I} \tag{75}$$

The piecewise linear function constraints are formulated in equations (76) to (79). In these, two domain values and their corresponding function values are activated by the SOS2 constraint. The same weights are used for u_{1ij} and z_{1ij}, and equally, for u_{2ij} and z_{1ij}. u_{1ij} and u_{2ij} are already known. Hence, the weights' values can be found in equation (76) and equation (78). However, as the same weights are used in equation (77) and equation (79), z_{1ij} and z_{2ij} are also found.

$$-u_{1ij} + (f_{u_{1ij1}} \times \lambda_{1ij1} + ... + f_{u_{1ij10}} \times \lambda_{1ij10}) == 0 \; \forall i, j \in \mathscr{I} \tag{76}$$

$$-z_{1ij} + (f_{z_{1ij1}} \times \lambda_{1ij1} + ... + f_{z_{1ij10}} \times \lambda_{1ij10}) == 0 \; \forall i, j \in \mathscr{I} \tag{77}$$

$$-u_{2ij} + (f_{u_{2ij1}} \times \lambda_{2ij1} + ... + f_{u_{2ij10}} \times \lambda_{2ij10}) == 0 \; \forall i, j \in \mathscr{I} \tag{78}$$

$$-z_{2ij} + (f_{u_{2ij11}} \times \lambda_{2ij1} + ... + f_{z_{2ij10}} \times \lambda_{2ij10}) == 0 \; \forall i, j \in \mathscr{I} \tag{79}$$

The multiplication is finalised in equation (80), by subtracting z_{2ij} from z_{1ij}. The if_pbs_{ij} variable defined in table 3.5 is multiplied with $(z_{1ij} - z_{2ij})$, ensuring that only pairs of boxes subject to partial base support calculation are added to the tally.

$$pbs\_ij_{ij} = if\_pbs_{ij} \times (z_{1ij} - z_{2ij}) \; \forall i, j \in \mathscr{I} \tag{80}$$

Constraint

The partial base support constraint applies to each individual box. Hence, the total partial base support for each box is found. This is done by finding the sum of the

partial base support from each pair of boxes, formulated in equation (81).

$$pbs\_i_i = \sum_{j=1}^{\mathscr{I}} pbs\_ij_{ij} \; \forall i \in \mathscr{I} \tag{81}$$

Two constants are introduced to create the partial base support constraint. The $area\_of\_box_i$ is the length of a box multiplied by its width: $p_i \times q_i$. The second constant: *pbs_constraint_variable* is specified by the user, determining how many percentages of a box's base must be supported from below. Both constants are formally introduced in table 3.10.

$area\_of\_box_i$	Constant equal to $p_i \times q_i \; \forall i \in \mathscr{I}$
pbs_constraint_variable	Constant set by the user indicating the required partial base support for each box

Table 3.10: Constants used for final partial base support constraint

$$L \times W \geq area\_of\_box_i \geq 0 \; \forall i, j \in \mathscr{I} \tag{82}$$

$$1 \geq pbs\_constraint\_variable \geq 0 \; \forall i, j \in \mathscr{I} \tag{83}$$

Finally, the partial base support constraint proposed by Nascimento et al. [10] is formulated in equation (84).

$$pbs\_i_i \geq pbs\_constraint\_variable \times area\_of\_box_i \; \forall i \in \mathscr{I} \tag{84}$$

However, there is one challenge regarding this formulation: boxes on top of the pallet floor do not qualify for partial base support, as they are not standing on top of other boxes. Hence, a sentinel value is used (see section 2.1). The sentinel value is a box i with dimensions $p_i = L$, $q_i = W$ and $r_i = 0$, and coordinates $x_i = 0$, $y_i = 0$ and $z_i = 0$. In addition, $l_{xi} = 1$, $w_{yi} = 1$, and $h_{zi} = 1$. By using a sentinel value as the pallet floor, all boxes standing on top of it achieve 100 % partial base support.

Approximation error

As the calculation of partial base support is an approximation, an error is incurred. This error can be found by looking at the worst-case input values to the approximation. Moreover, the worst approximation occurs by choosing input values just in the middle of two domain values.

In the formulation above, the number of domain values is set to 10, and $L = 1.2\,m$ and $W = 0.8\,m$. Hence, by choosing input values in the middle of two domain values, a worst-case error of no more than $0.004\,m^2$ is found. For an average box with dimensions between $0.1\,m - 0.3\,m$, an error of no more than $0.004\,m^2$ makes little difference. However, for smaller boxes, the error is relatively higher.

Multiplication is performed for all boxes. Thus, if two boxes support a box i, the total error might add up for i. Hence, it is possible for stacks to be rejected, even though they would have had sufficient partial base support in the real world.

3.4.2 partial base support size reduction

In section 3.4.1, constraint inequalities for the worded partial base support constraint proposed by Nascimento et al. [10] were formulated. It was later observed that its model size could be reduced by eliminating constraint inequalities.

The reduction is made by observing that if two boxes do not overlap in the x- and y-axis, their calculated overlap equals 0. Hence, the constraint inequality conditionals for the x and y-axis, or equations (40) to (47) can be removed. Thus, in this subsection, a new worded constraint for the partial base support criterion is proposed, seen in Code listing 3.3.

```
if (z[i] == z[j] + r[j])

    l_ij = min(x[i] + x_cover[i], x[j] + x_cover[j]) - max(x[i], x[j])
    w_ij = min(y[i] + y_cover[i], y[j] + y_cover[j]) - max(y[i], y[j])

    tot_pbs_box_i += l_ij * w_ij
end

@constraint(tot_pbs_box_i >= pbs_var * p_i * q_i)
```

Code listing 3.3: Worded partial base support reduced size constraint

3.4.3 Multiple boxes

The multiple box constraint requires that for any box i placed above the pallet floor, at least x % of its base must be supported by n boxes [6]. To formulate this concept by Carpenter and Dowsland [6], two constants and two variables are introduced in table 3.11.

The two constants in table 3.11 indicate the number of boxes required to support a box i and the percentage of i's base that each supporting box must provide. These are specified by the user. The two binary variables indicate whether any box j provide the necessary base support to i and whether a box i is placed above the pallet floor. $number_{mb}$ must be at least 1, and the value of $support_{mb}$ depends on the value of $number_{mb}$, as the total base support can be no more than 100 %.

$number_{mb}$	Constant indicating how many boxes that must provide base support to any box i above the pallet floor
$support_{mb}$	Constant indicating the minimum base support required from each supporting box
$bin\_support_{mb_{ij}}$	Binary variable indicating whether a box j provide the required base support to a box i $\forall i, j \in \mathscr{I}$
$above\_floor_{mb_i}$	Binary variable indicating whether $z_i > 0$ $\forall i \in \mathscr{I}$

Table 3.11: Variables used for calculating the multiple boxes constraint

$$number_{mb} \geq 1 \tag{85}$$

$$\frac{1}{number_{mb}} \geq support_{mb} \geq 0 \tag{86}$$

$$bin\_support_{mb_{ij}}, above\_floor_{mb_i} \in \{0,1\} \ \forall i, j \in \mathscr{I} \tag{87}$$

To find the base support each box j provides to a box i, the partial base support calculations from section 3.4.1 are used. In equation (88) and equation (89), $bin\_support_{mb_{ij}}$ is found using big-M constraints by checking the partial base support, pbs_ij_{ij}, against the support requirement for each box: $area\_of\_box_i \times support_{mb}$.

$$area\_of\_box_i \times (1 - bin\_support_{mb_{ij}}) \geq area\_of\_box_i \times support_{mb} - pbs\_ij_{ij} \; \forall i, j \in \mathscr{I} \quad (88)$$

$$area\_of\_box_i \times bin\_support_{mb_{ij}} \geq pbs\_ij_{ij} - area\_of\_box_i \times support_{mb} \; \forall i, j \in \mathscr{I} \quad (89)$$

To check whether a box i is placed above the pallet floor, equation (90) and equation (91) are formulated. Then, to finalise the multiple box constraint, equation (92) and equation (93) are formulated. These ensure that the number of boxes providing sufficient underlying support is greater than or equal to the number of boxes required.

$$H \times (1 - above\_floor_{mb_i}) \times \geq \epsilon - z_i \; \forall i \in \mathscr{I} \quad (90)$$

$$H \times above\_floor_{mb_i} \geq z_i - \epsilon \; \forall i \in \mathscr{I} \quad (91)$$

$$number_{mb} \times (1 - above\_floor_{mb_i}) \geq number_{mb} - \sum_{j!=i}^{\mathscr{I}} bin\_support_{mb_{ij}} \; \forall i \in \mathscr{I} \quad (92)$$

$$number_{mb} \times above\_floor_{mb_i} \geq \sum_{j!=i}^{\mathscr{I}} bin\_support_{mb_{ij}} - number_{mb} \; \forall i \in \mathscr{I} \quad (93)$$

By turning the constraint formulated in equation (84) off, a standalone multiple box constraint is achieved.

3.5 Dynamic stability

A *dynamic stability* constraint (also referred to as a horizontal stability [24]) ensures that boxes placed on a pallet are stable when the pallet is moving [24]. In the PLP of

Currence Robotics, the pallet moves when, e.g, the robot travels to retrieve another box in the warehouse. As mentioned in section 3.4, stability is discussed in 61 of the 163 papers analysed in Bortfeldt and Wäscher [24]. However, the majority of the discussed stability constraints are static.

To the best of the author's knowledge, there are almost no mathematical formulations of constraint inequalities for dynamic stability. Moreover, Nascimento et al. [10] mention how "*we are not aware of any formulation in the literature for handling horizontal stability to its full extent concerning container loading problems*". One of the reasons for this may be that researchers have relied on the dynamic stability provided by the container walls in the CLP, thus not proposing any additional criteria.

State of the art

Bischoff and Ratcliff [42] propose restricting the percentage of items that are not in contact with other boxes or the walls of the container. Similarly, Nascimento et al. [10] propose a lateral support factor ensuring the same. These criteria use the advantage of walls for support, and thus, they can only be used for the CLP.

Alonso et al. [40] observe the behaviour of pallets placed in a truck and formulate constraint inequalities with potential for providing dynamic stability. The constraints proposed by Alonso et al. [40] are: restricting the height difference between consecutive pallets (see figure 3.9), imposing a uniform stack height, and imposing a minimum pallet height.

Implementing these constraints, Alonso et al. [40] found that restricting the height difference between consecutive pallets was too time-consuming. Furthermore, they found that imposing a uniform stack height was effective for output maximisation problems. However, this would require boxes to be left out of the stack, which is not an option in Currence Robotics' input minimising PLP. Imposing a minimum height was found to yield better stacks within a reasonable time limit [40]. However, this criterion is similar to the already chosen objective function of section 3.2.

Carpenter and Dowsland [6] introduce several stability concepts. One of them - a *non-guillotine criterion*, requires that there are no guillotine cuts longer than z % of the stack's height. Carpenter and Dowsland [6] suggest that combining a non-guillotine criterion with some requirement for multiple box support (see section 3.4.3) could introduce stacks with an "interlocking of boxes property". A stack with no interlocking of boxes is shown in figure 3.10.

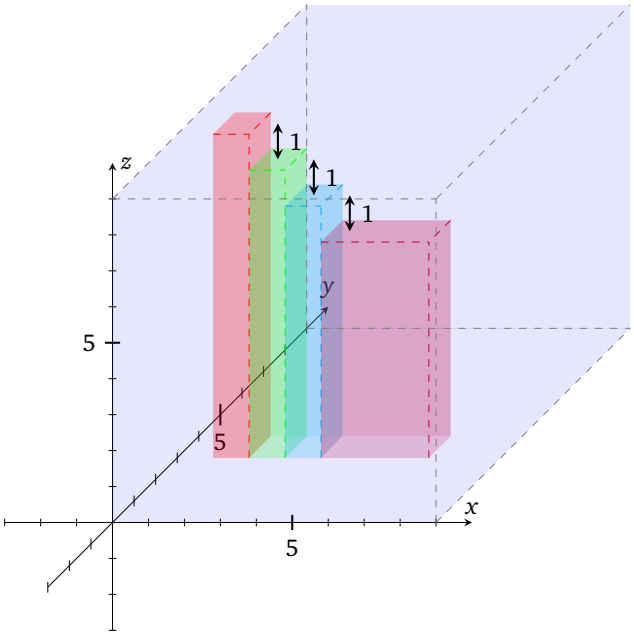


Figure 3.9: The constraint on consecutive boxes' height difference by Alonso et al. [40]. The difference in the z-axis between consecutive boxes is restricted to 1.

Decision

No constraint inequalities exist for the non-guillotine criterion concept introduced by Carpenter and Dowsland [6]. But, as almost no other dynamic stability formulations exist, it was decided to explore this criterion in section 3.5.1. Creating a working non-guillotine criterion was found to be challenging. Hence, a *multiple box incentive* and a *middle incentive* were formulated based on the work by Carpenter and Dowsland [6] and Alonso et al. [40], to emulate the non-guillotine criterion. These constraints are shown in section 3.5.2 and section 3.5.3.

3.5.1 Non-guillotine criterion

The non-guillotine criterion could not be formulated as intended. The expression power of MIP formulations is the reason why. A *guillotine cut* in a box stack is the non-interrupted difference between that stack's lowest- and highest z-axis coordinates. The non-guillotine criterion attempted to restrict the size of these guillotine cuts by placing a constraint on the maximum allowed guillotine cut size. However,

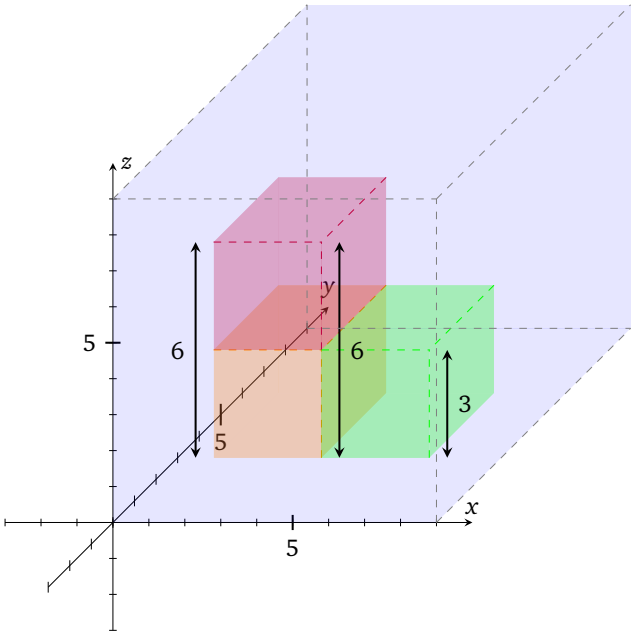


Figure 3.10: A stack of boxes that has no interlocking. If there was interlocking of boxes, the red box might have been standing on top and in the middle of the orange and green box. This would ensure that cuts in the stack were short. The cuts in this stack are marked using arrows, and in this case, all the cuts are guillotine cuts.

this did not stop the MIP solver from choosing a higher value for that box stack's lowest z-axis coordinate. E.g., even though the real lowest z-axis coordinate for a stack was equal to 1, the solver could set it equal 1.5. Hence, making it impossible to restrict the size of guillotine cuts. This explanation may be why no researchers have previously formulated and published constraint inequalities for the non-guillotine constraint.

The above insight was only discovered after much of the criterion had been formulated. Thus, to reduce the workload and still obtain some criterion, a novel naive approach to a non-guillotine criterion was pursued, able to find the *maximum possible cut* in the z-axis. A *maximum possible cut* is the size-difference between the lowest- and highest z-axis coordinates of box walls placed at equal x- or y-coordinates. A stack's maximum possible cuts usually equal its guillotine cuts for smaller instances. However, they are usually greater than guillotine cuts for larger instances. An illustration of a maximum possible cut can be seen in figure 3.11 A

proof of concept callback scheme was also created for eliminating too large cuts.

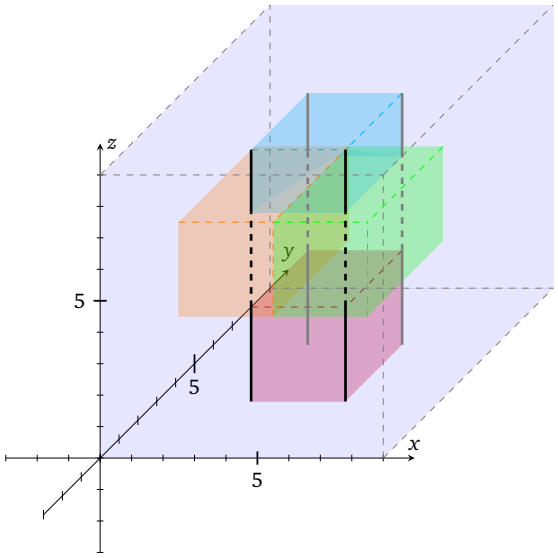


Figure 3.11: The maximum possible cuts of the blue and red box in this stack would be equal to 8. However, there exists no guillotine cuts, and the longest uninterrupted cut in the stack has size 3.

Finding a stack's maximum possible cuts requires multiple steps. Firstly, for each pair of boxes (i, j), it is checked whether any of i's and j's walls share x-coordinates. Then, it is checked whether i and j have any overlapping y-coordinate values. If any boxes, (i, j), satisfy the above conditions, their maximum possible cut size is calculated. In figure 3.12, the size of the maximum possible cut between the two boxes at position $x_i == x_j$ is equal to six.

In the formulations, a maximum possible cut is referred to a as a *cut*. In addition, maximum possible cuts are only formulated for the x-axis. However, in the coded implementation, maximum possible cuts are found for the x- and y-axis.

Checking for x-axis

Four different cuts can occur between two boxes (i, j) depending on boxes' x-coordinates. These four x-coordinate value equalities can be seen in figures 3.12 to 3.15. Binary variables are introduced to indicate if boxes' walls have an equal x-coordinate value. These can be seen in table 3.12.

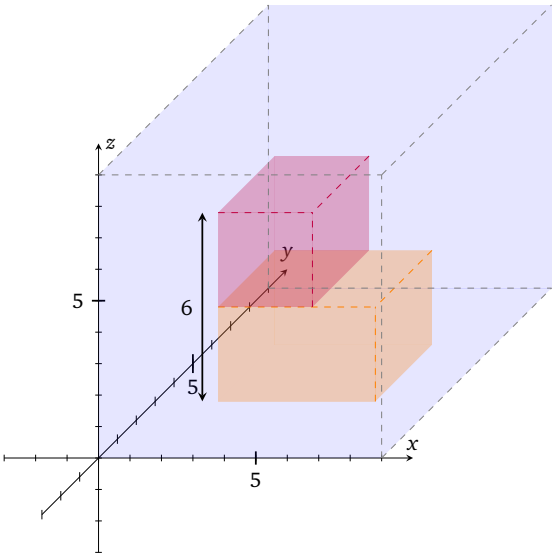


Figure 3.12: $if1_{gxij}$: A guillotine cut of size 6 at $x_i == x_j$.

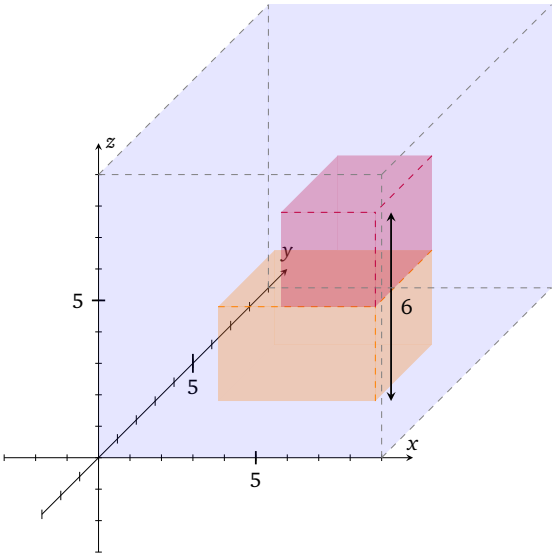


Figure 3.13: $if2_{gx\_ij_{ij}}$: A guillotine cut of size 6 at $x_i + x\_cover_i == x_j + x\_cover_j$.

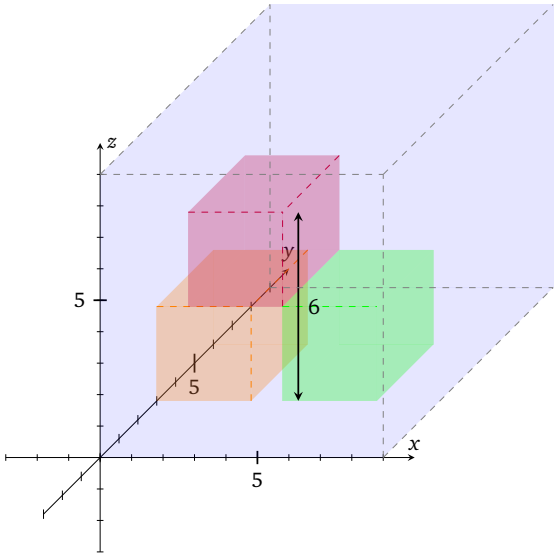


Figure 3.14: $if3_{gx}\_i_{ij}$: A guillotine cut of size 6 between purple (i) and green (j) box at $x_i + x\_cover_i == x_j$

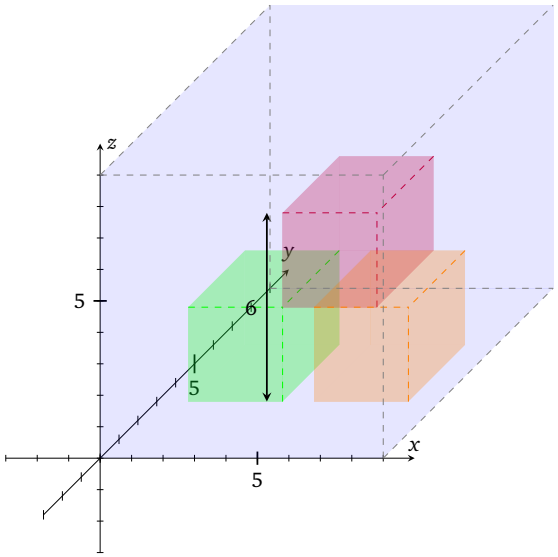


Figure 3.15: $if4_{gx}\_j_{ij}$: A guillotine cut of size 6 between purple (i) and green (j) at $x_i == x_j + x\_cover_j$

$if1_{cxij}$	$x_i == x_j \ \ \forall i,j \in \mathscr{I}$
$if2_{cx\_ij_{ij}}$	$x_i + x\_cover_i == x_j + x\_cover_j \ \ \forall i,j \in \mathscr{I}$
$if3_{cx\_i_{ij}}$	$x_i + x\_cover_i == x_j \ \ \forall i,j \in \mathscr{I}$
$if4_{cx\_j_{ij}}$	$x_i == x_j + x\_cover_j \ \ \forall i,j \in \mathscr{I}$

Table 3.12: Variables indicating whether two boxes' walls share an x-coordinate

$$if1_{cxij}, if2_{cx\_ij_{ij}}, if3_{cx\_i_{ij}}, if4_{cx\_j_{ij}} \in \{0,1\} \ \forall i \in \mathscr{I} \tag{94}$$

Big-M constraints are formulated to find whether there is an x-coordinate equality between two boxes' walls. These are seen equations (95) to (114).

$$L \times if11_{cxij} \geq x_i - x_j + \epsilon \ \forall i,j \in \mathscr{I} \tag{95}$$

$$L \times (1 - if11_{cxij}) \geq x_j - x_i - \epsilon \ \forall i,j \in \mathscr{I} \tag{96}$$

$$L \times if12_{cxij} \geq x_j - x_i + \epsilon \ \forall i,j \in \mathscr{I} \tag{97}$$

$$L \times (1 - if12_{cxij}) \geq x_i - x_j + \epsilon \ \forall i,j \in \mathscr{I} \tag{98}$$

$$if11_{cxij} + if12_{cxij} \geq 2 \times if1_{cxij} \ \forall i,j \in \mathscr{I} \tag{99}$$

$$L \times if21_{cx\_ij_{ij}} \geq (x_i + x\_cover_i) - (x_j + x\_cover_j) + \epsilon \ \forall i,j \in \mathscr{I} \tag{100}$$

$$L \times (1 - if21_{cx\_ij_{ij}}) \geq (x_j + x\_cover_j) - (x_i + x\_cover_i) - \epsilon \ \forall i,j \in \mathscr{I} \tag{101}$$

$$L \times if22_{cx\_ij_{ij}} \geq (x_j + x\_cover_j) - (x_i + x\_cover_i) + \epsilon \ \forall i,j \in \mathscr{I} \tag{102}$$

$$L \times (1 - if22_{cx\_ij_{ij}}) \geq (x_i + x\_cover_i) - (x_j + x\_cover_j) - \epsilon \ \forall i,j \in \mathscr{I} \tag{103}$$

$$if21_{cx}\_ij_{ij} + if22_{cx}\_ij_{ij} \geq 2 \times if2_{cx}\_ij_{ij} \ \forall i, j \in \mathscr{I} \tag{104}$$

$$L \times if31_{cx}\_i_{ij} \geq (x_i + x\_cover_i) - x_j + \epsilon \ \forall i, j \in \mathscr{I} \tag{105}$$

$$L \times (1 - if31_{cx}\_i_{ij}) \geq x_j - (x_i + x\_cover_i) - \epsilon \ \forall i, j \in \mathscr{I} \tag{106}$$

$$L \times if32_{cx}\_i_{ij} \geq x_j - (x_i + x\_cover_i) + \epsilon \ \forall i, j \in \mathscr{I} \tag{107}$$

$$L \times (1 - if32_{cx}\_i_{ij}) \geq (x_i + x\_cover_i) - x_j - \epsilon \ \forall i, j \in \mathscr{I} \tag{108}$$

$$if31_{cx}\_i_{ij} + if32_{cx}\_i_{ij} \geq 2 \times if3_{cx}\_i_{ij} \ \forall i, j \in \mathscr{I} \tag{109}$$

$$L \times if41_{cx}\_j_{ij} \geq x_i - (x_j + x\_cover_j) + \epsilon \ \forall i, j \in \mathscr{I} \tag{110}$$

$$L \times (1 - if41_{cx}\_j_{ij}) \geq (x_j + x\_cover_j) - x_i - \epsilon \ \forall i, j \in \mathscr{I} \tag{111}$$

$$L \times if42_{cx}\_j_{ij} \geq (x_j + x\_cover_j) - x_i + \epsilon \ \forall i, j \in \mathscr{I} \tag{112}$$

$$L \times (1 - if42_{cx}\_j_{ij}) \geq x_i - (x_j + x\_cover_j) - \epsilon \ \forall i, j \in \mathscr{I} \tag{113}$$

$$if41_{cx}\_j_{ij} + if42_{cx}\_j_{ij} \geq 2 \times if4_{cx}\_j_{ij} \ \forall i, j \in \mathscr{I} \tag{114}$$

Two new binary variables are introduced for each possible x-coordinate equality between two boxes' walls. This is similar to the method in section 3.4.1. The binary variables indicate whether each side of an x-coordinate equality satisfies a small ϵ ($\geq$ and $\leq$). If they do, then the corresponding x-coordinate equality binary variable is set to $true$.

Checking for y-axis

Figure 3.16 illustrates why it is necessary to check for an overlap in the y-axis. Four overlaps in the y-axis are possible between a pair of boxes (i, j). These are shown in figures 3.17 to 3.20.

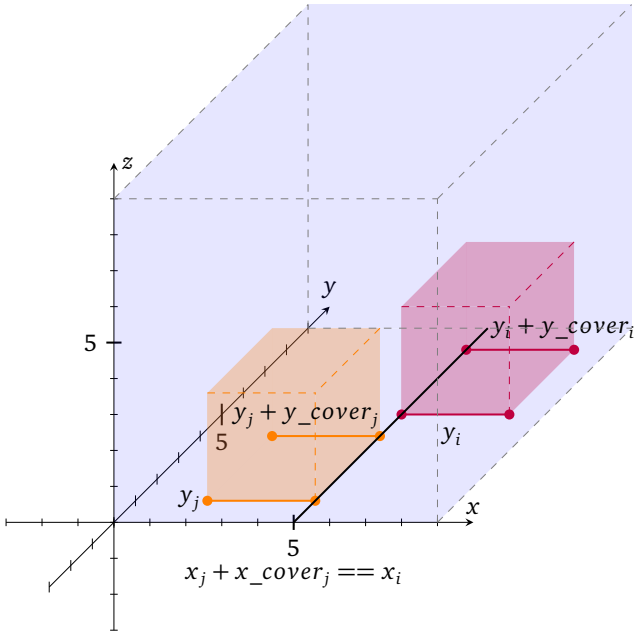


Figure 3.16: The black line indicates: $x_j + x\_cover_j == x_i$. But the y-coordinates don't overlap, and so there is no potential cut.

Binary variables are introduced to indicate the existence of overlap in the y-axis between a pair of boxes (i, j). These binary variables can be seen in table 3.13.

$if5_{cy_{ij}}$	$y_j < y_i + y\_cover_i \ \forall i, j \in \mathscr{I}$
$if6_{cy_{ij}}$	$y_i < y_j + y\_cover_j \ \forall i, j \in \mathscr{I}$
$overlap_{cy_{ij}}$	$if5_{cy}\_left_{ij}$ AND $if6_{cy}\_right_{ij} \ \ \forall i, j \in \mathscr{I}$

Table 3.13: Variables indicating y-axis overlap between two boxes

$$if5_{cy_{ij}}, if6_{cy_{ij}} \in \{0, 1\} \ \forall i, j \in \mathscr{I} \tag{115}$$

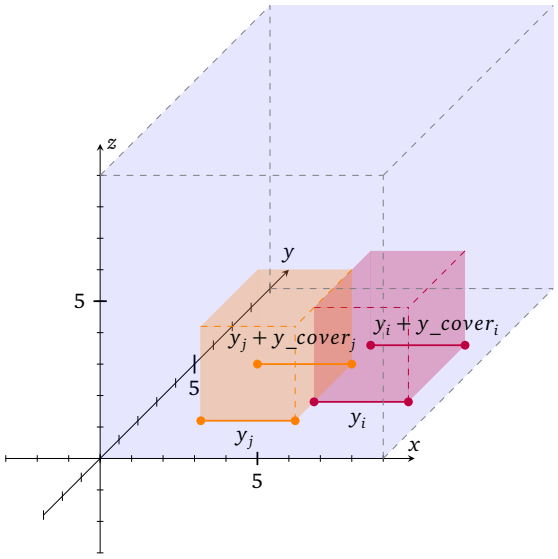


Figure 3.17: $y_j \leq y_i$ AND $y_i \leq y_j + y\_cover_j$

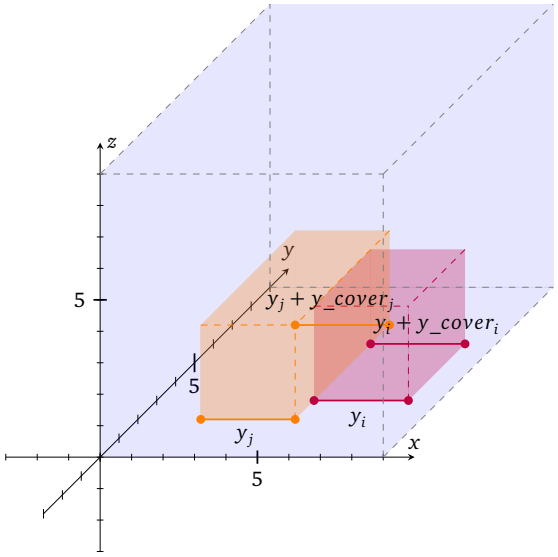


Figure 3.18: $y_j \leq y_i$ AND $y_i + y\_cover_i \leq y_j + y\_cover_j$

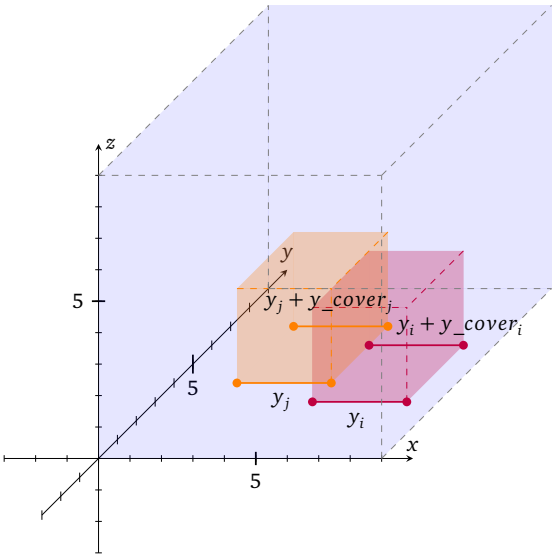


Figure 3.19: $y_j + \epsilon \leq y_i + y\_cover_i$ AND $y_i + y\_cover_i \leq y_j + y\_cover_j$

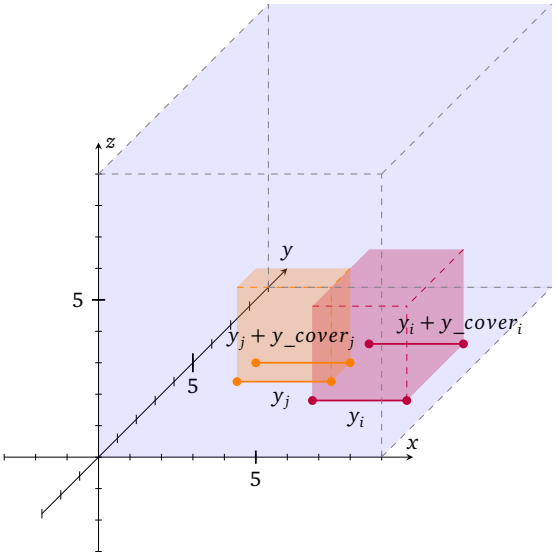


Figure 3.20: $y_i \leq y_j$ AND $y_j + y\_cover_j \leq y_i + y\_cover_i$

The y-axis overlap binary variables are found using big-M constraints in equations (116) to (120). These are similar to the constraint inequalities handling $>$ and $<$ in section 3.4.1.

$$W \times (1 - if5_{cy_{ij}}) \geq y_j + \epsilon - y_i - y\_cover_i \ \forall i, j \in \mathscr{I} \tag{116}$$

$$W \times if5_{cy_{ij}} \geq y_i + y\_cover_i - y_j - \epsilon \ \forall i, j \in \mathscr{I} \tag{117}$$

$$W \times (1 - if6_{cy_{ij}}) \geq y_i + \epsilon - y_j - y\_cover_j \ \forall i, j \in \mathscr{I} \tag{118}$$

$$W \times if6_{cy_{ij}} \geq y_j + y\_cover_j - y_i - \epsilon \ \forall i, j \in \mathscr{I} \tag{119}$$

$$if5_{cy_{ij}} + if6_{cy_{ij}} \geq 2 \times overlap_{cy_{ij}} \ \forall i, j \in \mathscr{I} \tag{120}$$

Checking for cut

For each pair of boxes (i, j), it is now checked whether a cut exists. This is done by checking for an x-coordinate equality for the walls of i and j, and for a y-axis overlap for the same two boxes i, j. Four new binary variables are introduced to indicate whether a cut exists or not. These can be seen in table 3.14.

$cut_{x_{bin}ij}$	cut at $x_i == x_j \ \ \forall i, j \in \mathscr{I}$
$cut_{x_{bin}}\_ij_{ij}$	cut at $x_i + x\_cover_i == x_j + x\_cover_j \ \ \forall i, j \in \mathscr{I}$
$cut_{x_{bin}}\_i_{ij}$	cut at $x_i + x\_cover_i == x_j \ \ \forall i, j \in \mathscr{I}$
$cut_{x_{bin}}\_j_{ij}$	cut at $x_i == x_j + x\_cover_j \ \ \forall i, j \in \mathscr{I}$

Table 3.14: Binary variables indicating whether a cut is present

$$cut_{x_{bin}ij}, cut_{x_{bin}}\_ij_{ij}, cut_{x_{bin}}\_i_{ij}, cut_{x_{bin}}\_j_{ij} \in \{0, 1\} \ \forall i \in \mathscr{I} \tag{121}$$

Equations (122) to (125) finds the values of the variables from table 3.14. Calculating cuts is an incentive. Thus, the variables from table 3.14 must be added to the objective function.

$$if\, 1_{cxij} + overlap_{cy_{ij}} \geq 2 \times cut_{x_{bin}ij}\ \forall i,j \in \mathscr{I} \tag{122}$$

$$if\, 2_{cx\_ij_{ij}} + overlap_{cy_{ij}} \geq 2 \times cut_{x_{bin}\_ij_{ij}}\ \forall i,j \in \mathscr{I} \tag{123}$$

$$if\, 3_{cx\_i_{ij}} + overlap_{cy_{ij}} \geq 2 \times cut_{x_{bin}\_i_{ij}}\ \forall i,j \in \mathscr{I} \tag{124}$$

$$if\, 4_{cx\_j_{ij}} + overlap_{cy_{ij}} \geq 2 \times cut_{x_{bin}\_j_{ij}}\ \forall i,j \in \mathscr{I} \tag{125}$$

Calculating the size of a cut

If either of the constraints in equations (122) to (125) are satisfied, then there exists a cut between boxes i, j. Then, the difference between the lowest- and highest z-axis coordinates of i and j must be calculated.

Continuous variables are introduced in table 3.15 indicating the lowest- and highest z-axis coordinates of i and j. Moreover, equations (127) to (150) finds the values of these variables.

$max\_cut_{x_{ij}}$	max height of cut located at $x_i == x_j\ \forall i,j \in \mathscr{I}$
$max\_cut_{x\_ij_{ij}}$	max height of cut located at $x_i + x\_cover_i == x_j + x\_cover_j\ \forall i,j \in \mathscr{I}$
$max\_cut_{x\_i_{ij}}$	max height of cut located at $x_i + x\_cover_i == x_j\ \forall i,j \in \mathscr{I}$
$max\_cut_{x\_j_{ij}}$	max height of cut located at $x_i == x_j + x\_cover_j\ \forall i,j \in \mathscr{I}$
$min\_cut_{x_{ij}}$	min height of cut located at $x_i == x_j\ \forall i,j \in \mathscr{I}$
$min\_cut_{x\_ij_{ij}}$	min height of cut located at $x_i + x\_cover_i == x_j + x\_cover_j\ \forall i,j \in \mathscr{I}$
$min\_cut_{x\_i_{ij}}$	min height of cut located at $x_i + x\_cover_i == x_j\ \forall i,j \in \mathscr{I}$
$min\_cut_{x\_j_{ij}}$	min height of cut located at $x_i == x_j + x\_cover_j\ \forall i,j \in \mathscr{I}$

Table 3.15: Variables indicating max and min cut height between boxes i, j

$$max\_cut_{x_{ij}}, max\_cut\_ij_{ij}, max\_cut\_i_{ij}, max\_cut\_j_{ij},$$
$$min\_cut_{x_{ij}}, min\_cut\_ij_{ij}, min\_cut\_i_{ij}, min\_cut\_j_{ij} \geq 0 \; \forall i, j \in \mathscr{I} \quad (126)$$

The maximum height of a cut is found first. A simple method would be to calculate maximum cut height between two boxes i, j. This would be the greatest value of $z_i + r_i$ and $z_j + r_j$. Then, if boxes i, j do not share a cut, then the maximum cut height between them is 0. However, doing this means cut sizes will only be calculated using two boxes. A more advanced method finds the maximum cut height that a box is part of. This is done in equations (127) to (138).

$$max\_cut_{x_{ij}} \geq cut_{x_{bin}ij} \times (z_j + r_j) \; \forall i, j \in \mathscr{I} \quad (127)$$

$$max\_cut_{x_i} \geq max\_cut_{x_{ij}} \; \forall j \in \mathscr{I} \quad (128)$$

$$max\_cut_{x_i} \geq z_i + r_i \; \forall i \in \mathscr{I} \quad (129)$$

$$max\_cut\_ij_{ij} \geq cut_{x_{bin}}\_ij_{ij} \times (z_j + r_j) \; \forall i, j \in \mathscr{I} \quad (130)$$

$$max\_cut\_ij_i \geq max\_cut\_ij_{ij} \; \forall j \in \mathscr{I} \quad (131)$$

$$max\_cut\_ij_i \geq z_i + r_i \; \forall i \in \mathscr{I} \quad (132)$$

$$max\_cut\_i_{ij} \geq cut_{x_{bin}}\_i_{ij} \times (z_j + r_j) \; \forall i, j \in \mathscr{I} \quad (133)$$

$$max\_cut\_i_i \geq max\_cut\_i_i \; \forall j \in \mathscr{I} \quad (134)$$

$$max\_cut\_i_i \geq z_i + r_i \; \forall i \in \mathscr{I} \quad (135)$$

$$max\_cut_x\_j_{ij} \geq cut_{x_{bin}}\_j_{ij} \times (z_j + r_j) \; \forall i, j \in \mathscr{I} \tag{136}$$

$$max\_cut_x\_j_{ij} \geq max\_cut_x\_j_i \; \forall j \in \mathscr{I} \tag{137}$$

$$max\_cut_x\_j_i \geq z_i + r_i \; \forall i \in \mathscr{I} \tag{138}$$

Equations (127) to (138) find the maximum cut height that a box i is part of. Firstly, if a box i, j share a cut, then $max\_cut_{x\_ij}$ is set to $z_j + r_j$. Hence, for a box i, $max\_cut_{x\_ij}$ indicate the heights of all the other boxes.

Then, the greatest cut height that i is part of is found. This is done by minimising a new variable $max\_cut_{x\_i}$ across all of j's heights ($max\_cut_{x\_ij}$). Thus, finding the greatest height of a box j that i share a cut with.

Finally, the greatest cut height that i is part of cannot be lower than the height of i. Hence, $max\_cut_{x\_i}$ must be greater than $z_i + r_i$.

$$min\_cut_{x_{ij}}. \geq [cut_{x_{bin}ij} \times z_j, (1 - cut_{x_{bin}ij}) \times z_i] \; \forall i, j \in \mathscr{I} \tag{139}$$

$$min\_cut_{x_{ij}} \geq min\_cut_{x_i} \; \forall j \in \mathscr{I} \tag{140}$$

$$z_i \geq min\_cut_{x_i} \; \forall i \in \mathscr{I} \tag{141}$$

$$min\_cut_x\_ij_{ij}. \geq [cut_{x_{bin}}\_ij_{ij} \times z_j, (1 - cut_{x_{bin}}\_ij_{ij}) \times z_i] \; \forall i, j \in \mathscr{I} \tag{142}$$

$$min\_cut_x\_ij_{ij} \geq min\_cut_x\_ij_i \; \forall j \in \mathscr{I} \tag{143}$$

$$z_i \geq cut_x\_ij_i \; \forall i \in \mathscr{I} \tag{144}$$

$$min\_cut_x\_i_{ij}. \geq [cut_{x_{bin}}\_i_{ij} \times z_j, (1 - cut_{x_{bin}}\_i_{ij}) \times z_i] \; \forall i, j \in \mathscr{I} \tag{145}$$

$$min\_cut_x\_i_{ij} \geq min\_cut_x\_i_i \; \forall j \in \mathscr{I} \tag{146}$$

$$z_i \geq cut_x\_i_i \; \forall i \in \mathscr{I} \tag{147}$$

$$min\_cut_x\_j_{ij}. \geq [cut_{x_{bin}}\_j_{ij} \times z_j, (1 - cut_{x_{bin}}\_j_{ij}) \times z_i] \; \forall i, j \in \mathscr{I} \tag{148}$$

$$min\_cut_x\_j_{ij} \geq min\_cut_x\_j_i \; \forall j \in \mathscr{I} \tag{149}$$

$$z_i \geq cut_x\_j_i \; \forall i \in \mathscr{I} \tag{150}$$

Equations (139) to (150) find the minimum cut height that a box i is part of. Firstly, if i, j share a cut, then $min\_cut_{x\_ij}$ is set to z_j. If not, $min\_cut_{x\_ij}$ is set to z_i.

Then, a new variable $min\_cut_{x\_i}$ is maximised across all $min\_cut_{x\_ij}$, finding the greatest minimum value of any box that i share a cut with.

Finally, the minimum cut height of a box i cannot be lower than its own height z_i.

All of the max and min variables mentioned above are added to the objective function for minimisation, except the new minimum variable, which is added to the objective function for maximisation.

New variables are introduced to indicate the size of a cut. These can be seen in table 3.16.

cut_{xij}	cut size at $x_i == x_j \;\; \forall i, j \in \mathscr{I}$
$cut_x\_ij_{ij}$	cut size at $x_i + x\_cover_i == x_j + x\_cover_j \;\; \forall i, j \in \mathscr{I}$
$cut_x\_i_{ij}$	cut size at $x_i + x\_cover_i == x_j \;\; \forall i, j \in \mathscr{I}$
$cut_x\_j_{ij}$	cut size at $x_i == x_j + x\_cover_j \;\; \forall i, j \in \mathscr{I}$

Table 3.16: Variables indicating cut sizes

$$cut_{xij}, cut_x\_ij_{ij}, cut_x\_i_{ij}, cut_x\_j_{ij} \geq 0 \; \forall i, j \in \mathscr{I} \tag{151}$$

The cut sizes are calculated by finding the difference between the maximum and the minimum height of a type of cut in a given location. These calculations are formulated in equations (152) to (155).

$$cut_{xij} = max\_cut_{xij} - min\_cut_{xij} \; \forall i, j \in \mathscr{I} \tag{152}$$

$$cut_x\_ij_{ij} = max\_cut_x\_ij_{ij} - min\_cut_x\_ij_{ij} \; \forall i,j \in \mathscr{I} \tag{153}$$

$$cut_x\_i_{ij} = max\_cut_x\_i_{ij} - min\_cut_x\_i_{ij} \; \forall i,j \in \mathscr{I} \tag{154}$$

$$cut_x\_j_{ij} = max\_cut_x\_j_{ij} - min\_cut_x\_j_{ij} \; \forall i,j \in \mathscr{I} \tag{155}$$

At this point, it would be desirable to add a constraint on the maximum allowed cut size as a percentage of the stack height. However, this is not possible due to reasons explained initially.

Figure 3.21 shows the cuts in the x-axis while figure 3.22 shows the cuts in the y-axis for the same stack. The arrows indicate where and what size the different cuts would have been.

It can be observed that two cut sizes are calculated for each location. More specifically, $cut_{x_{bin}ij}$ and $cut_{x_{bin}}\_j_{ij}$ share location, or $cut_{x_{bin}}\_ij_{ij}$ and $cut_{x_{bin}}\_i_{ij}$ share location. Hence, the two cut sizes at a location may be combined to find that location's greatest cut. However, if cuts are combined, information may get lost. Figure 3.23 shows that different cut types may have unequal sizes, even though they are located in the same place. And, a combined cut would only indicate the largest of the two combined cuts.

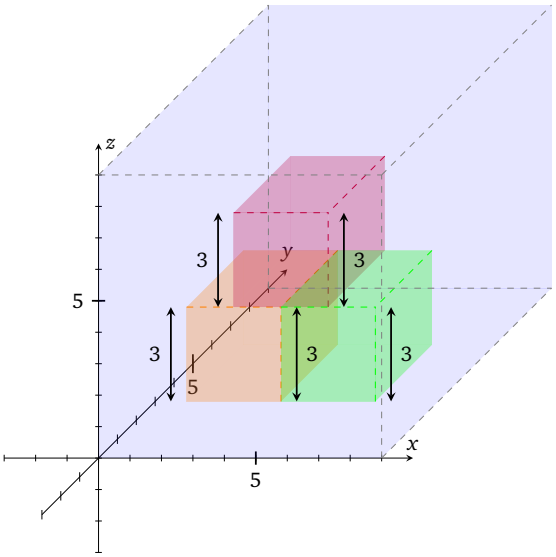


Figure 3.21: The cuts of the stack as seen from the x-axis.

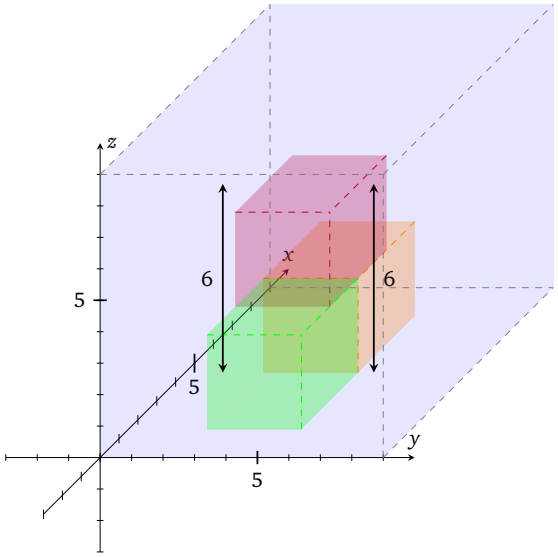


Figure 3.22: y-axis cuts of the same stack, but with the coordinate system is rotated. Hence, the cuts in the y-axis are seen.

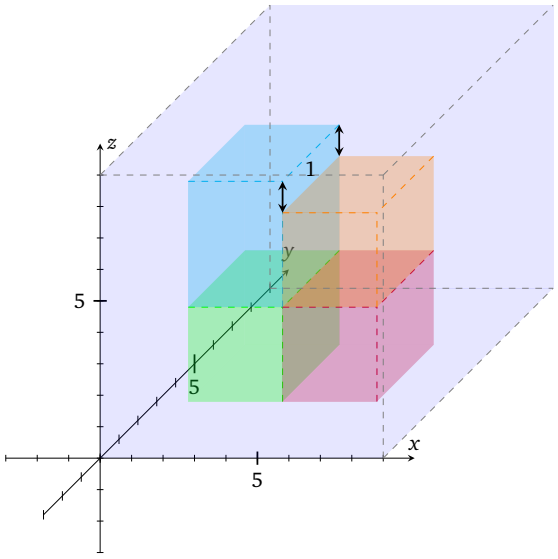


Figure 3.23: Given that i is the purple box, then $max\_cut_{x_{ij}} == 6$, but $max\_cut_x\_j_{ij} == 7$

Callback

A simple method for constraining cuts was implemented using callbacks. As a proof of concept, it cannot produce viable real-world stacks. The callback function in Code listing 3.4 finds whether a cut's size is greater than the *allowed_cut_size*. If so, it is fixed by imposing a constraint on the top box part of that cut, forcing it to be at least *cut_fix_var* away from the second-highest placed box. If a cut is fixed, it is added to a fixed cut list. The callback function checks a cut prior to fixing it, whether it has been fixed or not. If it has, then no more constraints are imposed on that cut.

The main drawback with this cut fixing scheme is that a constraint cannot be undone once imposed. Hence, if two boxes are told to be cut_fix_var units away from each other, then they cannot not be it. There are also other flaws with this solution. However, it remains as a concept for further exploration.

3.5.2 Multiple boxes incentive

Other criteria were explored as the non-guillotine criterion could not provide dynamic stability. However, few alternatives existed. Thus it was attempted to emulate the behaviour of a non-guillotine criterion by using incentives. An incentive urges the solver to look for some quality without specifying a hard constraint on that quality. The disadvantage with incentives is that they do not give result guarantees.

The *multiple boxes incentive* is an extension of the multiple box constraint introduced in section 3.4 [6]. The multiple box incentive reduces an objective function penalty for each box j any box i is placed on top of. The goal of the incentive is to eliminate cuts in the stack by placing boxes on top of intersections between multiple boxes. The difference between the multiple box incentive and the multiple box constraint is that the incentive does not disallow boxes standing on top of only one box.

Equation (156) and equation (157) formulate the multiple box incentive. Equation (156) counts the number of boxes j that are placed below any box i using if_pbs_{ij} from section 3.4.1. Then, equation (157) subtracts the number of boxes below any box i from the number of boxes in an instance; $|\mathscr{I}|$. Finally, this total is added to the objective function.

$$sum\_of\_boxes\_below_i = \sum_{j=1;j!=i}^{J} if\_pbs_{ij}, \forall i \in \mathscr{I} \tag{156}$$

$$|\mathscr{I}| - sum\_of\_boxes\_below_i, \forall i \in \mathscr{I} \tag{157}$$

```
for each box i
    positions_of_fixed_cuts = []
    for each cut
        if (cut[size] >= allowed_cut_size &&
            check_if_cut_is_fixed(positions_of_fixed_cuts, i))
            get_involved_boxes_in_cut(cut[type], i)
            push (x_i, positions_of_fixed_cuts)
        end
    end
end

function check_if_cut_is_fixed(positions, i)
    fixed = false
    for each pos in positions
        if (pos == x_i)
            fixed = true
        end
    end
    return not fixed
end

function get_involved_boxes_in_cut(cut_type, i)
    involved_boxes = [i]
    for each box j
        if (cut_type[i, j] == true)
            push (j, involved_boxes)
        end
    end
    create_constraint(involved_boxes)
end

function create_constraint(involved_boxes)
    box1 = box of max height in involved boxes
    box2 = box of second max height in involved boxes
    if (dimensions of box1 < dimensions of sentinel - cut_fix_var)
        @build_constraint(box1_x >= box_2_x + cut_fix_var)
        #inequality would depend on type of cut
    end
end
```

Code listing 3.4: Callback function for correcting cuts that violate a cut size constraint

3.5.3 Middle-incentive

The middle incentive incentivises a box i to be placed in the middle of the boxes placed directly below. Thus, eliminating cuts in the stack. Neither the multiple box incentive nor the middle incentive has previously been mentioned in the research literature.

The middle-incentive finds the maximum and minimum x- and y coordinates the boxes located below a box i. Then, i is given an incentive to be placed in the middle of those coordinates. As the middle incentive is an incentive, boxes are allowed to be placed away from the middle. However, this results in a higher objective function penalty. The middle incentive be seen in figure 3.24.

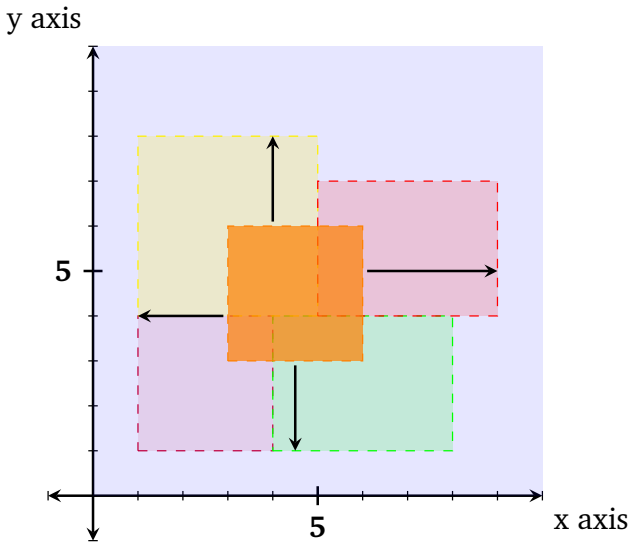


Figure 3.24: The middle incentive tries to ensure that the x-axis arrows, and the y-axis arrows are of equal length. Here, five boxes are seen from above, and the arrows mark the distance from the box i to the edges of i's supporting boxes.

Continuous variables are introduced in table 3.17 to find the middle coordinate of boxes that box i is placed on top of. These indicate the maximum and minimum x- and y-coordinates to boxes placed below a box i. If a box j is not placed directly below a box i, variables equal 0.

$max_x\_supporting_{ij}$	max x-coordinate of box j below box i $\forall i, j \in \mathscr{I}$
$min_x\_supporting_{ij}$	min x-coordinate of box j below box i $\forall i, j \in \mathscr{I}$
$max_y\_supporting_{ij}$	max y-coordinate of box j below box i $\forall i, j \in \mathscr{I}$
$min_y\_supporting_{ij}$	min y-coordinate of box j below box i $\forall i, j \in \mathscr{I}$

Table 3.17: Variables representing the maximum or minimum x-/y-coordinates of j below box i

$$max_x\_supporting_{ij}, min_x\_supporting_{ij}, max_y\_supporting_{ij}, min_y\_supporting_{ij} \geq 0 \; \forall i, j \in \mathscr{I} \quad (158)$$

Equations (159) to (162) are formulated to find the values of the variables introduced in table 3.17. The maximum constraints find the maximum $x_j + x\_cover_j / y_j + y\_cover_j$ value, given that i is placed on top of j. Similarly, the minimum constraints finds x_j / y_j if i is placed on top of j.

$$max_x\_supporting_{ij} \geq if\_pbs_{ij} \times (x_j + x\_cover_j) \; \forall i, j \in \mathscr{I} \quad (159)$$

$$min_x\_supporting_{ij}. \geq if\_pbs_{ij} \times x_j \; \forall i, j \in \mathscr{I} \quad (160)$$

$$max_y\_supporting_{ij} \geq if\_pbs_{ij} \times (y_j + y\_cover_j) \; \forall i, j \in \mathscr{I} \quad (161)$$

$$min_y\_supporting_{ij}. \geq if\_pbs_{ij} \times y_j \; \forall i, j \in \mathscr{I} \quad (162)$$

Similar to section 3.5.1, new variables for each box i ($\forall i \in \mathscr{I}$) are introduced that indicate the maximum and minimum x- and y-coordinate value of boxes j placed below i. The value of these new value of is found in equations (163) to (166).

All variables mentioned thus far are added to the objective function for minimisation except for the newly introduced minimum variables in equation (164) and equation (166).

$$max_x\_supporting_i \geq max_x\_supporting_{ij} \ \forall i, j \in \mathscr{I} \tag{163}$$

$$min_x\_supporting_{ij} \geq min_x\_supporting_i \ \forall i, j \in \mathscr{I} \tag{164}$$

$$max_y\_supporting_i \geq max_y\_supporting_{ij} \ \forall i, j \in \mathscr{I} \tag{165}$$

$$min_y\_supporting_{ij} \geq min_y\_supporting_i \ \forall i, j \in \mathscr{I} \tag{166}$$

Continuous variables representing the dimensional middle of boxes j placed below a box i are introduced in table 3.18. equation (168) and equation (169) finds the midpoints in each axis.

$midpoint\_x_i$	x-axis midpoint of boxes j placed below box i
$midpoint\_y_i$	y-axis midpoint of boxes j placed below box i

Table 3.18: Axes midpoints of boxes j placed below a box i

$$midpoint\_x_i, midpoint\_y_i \geq 0 \ \forall i \in \mathscr{I} \tag{167}$$

$$midpoint\_x_i = \frac{max_x\_supporting_i + min_x\_supporting_i}{2} \ \forall i \in \mathscr{I} \tag{168}$$

$$midpoint\_y_i = \frac{max_y\_supporting_i + min_y\_supporting_i}{2} \ \forall i \in \mathscr{I} \tag{169}$$

The incentive is only intended to work on boxes placed above the pallet floor, as boxes placed on the pallet floor are considered stable. Hence, a binary variable is introduced, indicating whether a box is placed above the floor or not. In addition,

new incentive variables are introduced to indicate how much penalty ought to be given to a box's position. If a box is placed in its ideal position (in the middle of the boxes below), it receives no penalty. However, if a box is placed above the pallet floor, then the further away from the middle, the higher the penalty. These variables are introduced in table 3.19. Equation (172) and Equation (173) find, similar to section 3.4.3, whether a box is placed above the pallet floor or not.

$above\_ground_{ds_i}$	indicates whether i is placed on top of other boxes $\forall i \in \mathscr{I}$
$incentive\_x_i$	represents the amount of penalty a box i is to be given for it's position on the x-axis $\forall i \in \mathscr{I}$
$incentive\_y_i$	represents the amount of penalty a box i is to be given for it's position on the y-axis $\forall i \in \mathscr{I}$

Table 3.19: Variables used for calculating the penalty for each box position.

$$above\_ground_{ds_i} \in \{0,1\}, \forall i \in \mathscr{I} \tag{170}$$

$$incentive\_x_i, incentive\_y_i \geq 0 \ \forall i \in \mathscr{I} \tag{171}$$

$$H \times (1 - above\_ground_{ds_i}) \geq \epsilon - z_i \ \forall i \in \mathscr{I} \tag{172}$$

$$H \times above\_ground_{ds_i} \geq z_i - \epsilon \ \forall i \in \mathscr{I} \tag{173}$$

The constraints that are formulated for $incentive\_x_i$ and $incentive\_y_i$ are absolute value constraints. These minimise the absolute value distance from the $midpoint\_x_i$ and $midpoint\_y_i$ for each box. Then, that value is multiplied by the binary variable indicating whether a box is placed above the pallet floor or not, as no penalty is imposed if a box is placed on the pallet floor. The relevant constraints can be seen in equations (174) to (177).

$$incentive\_x_i \geq above\_ground_{ds_i} \times (midpoint\_x_i - (\frac{xi + x\_cover_i}{2})), \forall i \in \mathscr{I} \tag{174}$$

$$incentive\_x_i \geq above\_ground_{ds_i} \times ((\frac{xi + x\_cover_i}{2}) - midpoint\_x_i), \forall i \in \mathscr{I} \quad (175)$$

$$incentive\_y_i \geq above\_ground_{ds_i} \times (midpoint\_y_i - (\frac{yi + y\_cover_i}{2})), \forall i \in \mathscr{I} \quad (176)$$

$$incentive\_y_i \geq above\_ground_{ds_i} \times ((\frac{yi + y\_cover_i}{2}) - midpoint\_y_i), \forall i \in \mathscr{I} \quad (177)$$

The incentive is finalised by adding the sum of the middle incentives, formulated in equation (178), to the objective function.

$$\sum_{i=1}^{\mathscr{I}} incentive\_x_i + \sum_{i=1}^{\mathscr{I}} incentive\_y_i \quad (178)$$

3.6 Load Balancing

A load balancing/weight distribution constraint ensures that the centre of gravity of the stack is located within the dimensional centre of the pallet [24]. Thus, when a pallet is picked up or moved, its stack of boxes does not sway or fall.

State of the art

Chen et al. [26] formulate load balancing constraints for the chosen basis MIP formulation by Chen et al. [26]. Moreover, Nascimento et al. [10] and Alonso et al. [39] formulate constraints requiring the centre of gravity of the stack to be within restricted intervals in each axis. The constraint inequalities proposed by Nascimento et al. [10] are used due to their simplicity.

3.6.1 Load Balancing

The load balancing constraint by Nascimento et al. [10] uses pre-defined constants defining the intervals that the centre of gravity of the stack may lie in. These constants are introduced in Table 3.20. The constants for the z-axis are not mentioned in the formulation by Nascimento et al. [10] but are included as well.

lb_start_x	Start of the load balance x-axis interval
lb_end_x	End of the load balance x-axis interval
lb_start_y	Start of the load balance y-axis interval
lb_end_y	End of the load balance y-axis interval
lb_start_z	Start of the load balance z-axis interval
lb_end_z	End of the load balance z-axis interval

Table 3.20: Pre-defined constants indicating the start and end of load balancing intervals in each axis.

Equations (179) to (187) make up the load balancing constraints. By using big-M constraints, they ensure that the centre of gravity of the stack lies inside the intervals defined by the constants in table 3.20. In figure 3.25, the load balancing constraints of the x- and y-axis are shown.

$$load\_balance\_x_i = w_i \times (x_i + \frac{x\_cover_i}{2}) \; \forall i \in \mathscr{I} \tag{179}$$

$$\frac{\sum_{i=1}^{\mathscr{I}} load\_balance\_x_i}{sum\_of\_masses} \geq lb\_start_x \; \forall i \in \mathscr{I} \tag{180}$$

$$\frac{\sum_{i=1}^{\mathscr{I}} load\_balance\_x_i}{sum\_of\_masses} \leq lb\_end_x \; \forall i \in \mathscr{I} \tag{181}$$

$$load\_balance\_y_i = w_i \times (y_i + \frac{y\_cover_i}{2}) \; \forall i \in \mathscr{I} \tag{182}$$

$$\frac{\sum_{i=1}^{\mathscr{I}} load\_balance\_y_i}{sum\_of\_masses} \geq lb\_start_y \; \forall i \in \mathscr{I} \tag{183}$$

$$\frac{\sum_{i=1}^{\mathscr{I}} load\_balance\_y_i}{sum\_of\_masses} \leq lb\_end_y \; \forall i \in \mathscr{I} \tag{184}$$

$$load\_balance\_z_i = w_i \times (z_i + \frac{r_i}{2}) \; \forall i \in \mathscr{I} \tag{185}$$

$$\frac{\sum_{i=1}^{\mathscr{I}} load\_balance\_z_i}{sum\_of\_masses} \geq lb\_start_z \; \forall i \in \mathscr{I} \tag{186}$$

$$\frac{\sum_{i=1}^{\mathscr{I}} load\_balance\_z_i}{sum\_of\_masses} \leq lb\_end_z \; \forall i \in \mathscr{I} \tag{187}$$

Depending on the problem instance, load balancing constraints could place the centre of gravity anywhere on the pallet. This could be done by changing the values of the constants in table 3.20.

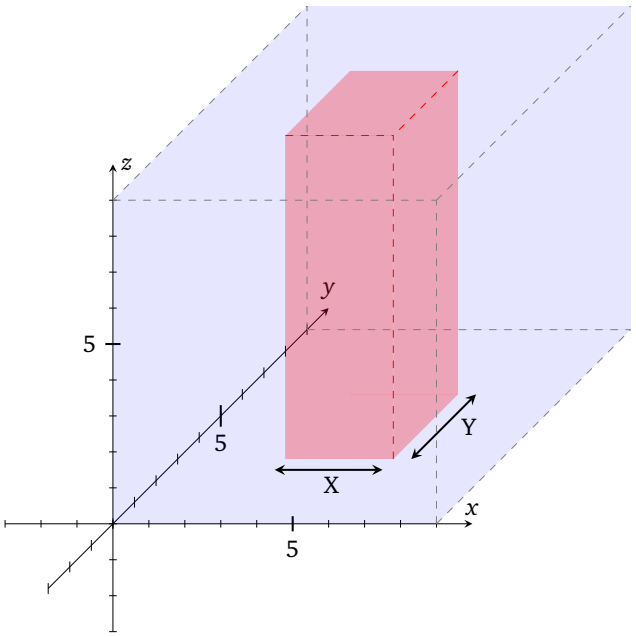


Figure 3.25: Load balancing constraints in the x- and y-axis ensure that the centre of gravity of the stack is located within the red area. The interval's can be adjusted by changing the size of X and Y; the difference between the constants in table 3.20.

3.7 Robot Arm

Grab's robot arm can be seen in figure 3.26. It operates the following way: firstly, the robot arm picks up a box. Then, it can choose to rotate that box 90 ° about the z-axis or not. Continuing, the robot arm places the box in the location decided by the stacking algorithm, approaching the pallet from one side.

The robot arm is, initially, able to place boxes anywhere on the pallet. However, if a tall wall of boxes is placed in the middle of the pallet, the robot arm's physical constraint disallows it from reaching behind that wall. A visualisation of such an event can be seen in figure 3.27. Hence, the motivation behind the robot arm constraint is to ensure that the robot arm can place all boxes in their solved for position.

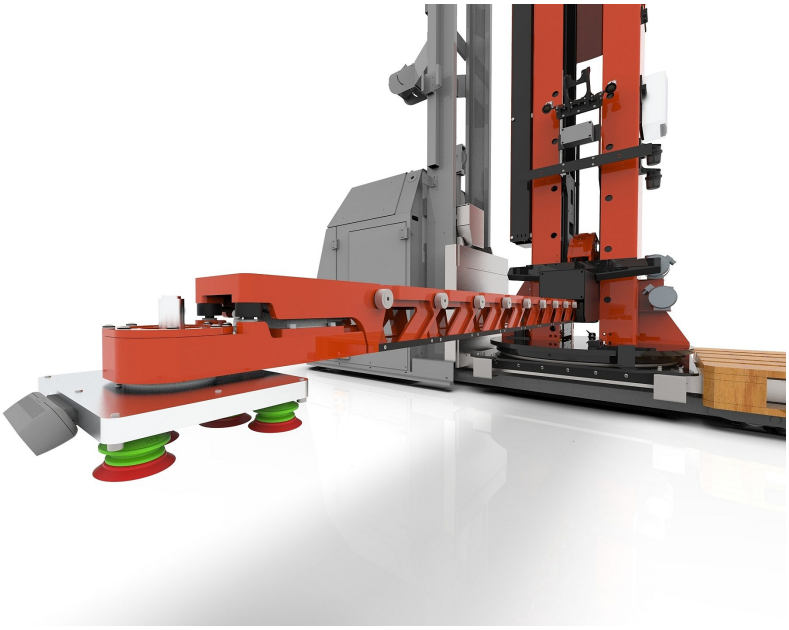


Figure 3.26: The robot arm of Grab (Picture used with permission from Currence Robotics)

The order boxes are placed in is currently fixed by Currence Robotics and is decided before any stacking algorithm has decided where boxes are to be placed. This is done for two reasons. Firstly, by having a fixed order, Grab only needs to traverse the warehouse once. It can start at one end of the warehouse and finish at the other. Secondly, the warehouse is arranged to accommodate the heaviest boxes first. Hence, if Grab starts at the beginning, the heaviest boxes are placed on the

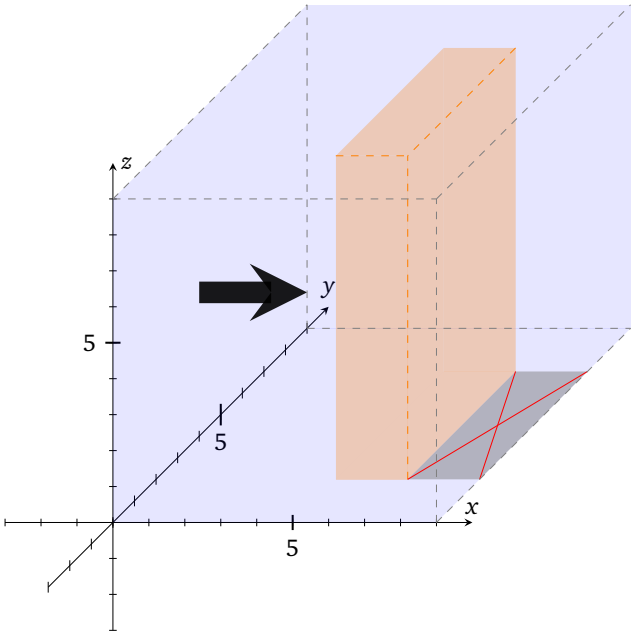


Figure 3.27: The unavailable area for the robot arm is marked with a red "X" behind the orange box. The arm approaches with the arrow.

bottom part of the stack. Moreover, having such a placement scheme facilitates an indirect *load bearing constraint* (see section 3.6).

If the box placement order is ignored, then it is likely that an event such as in figure 3.27 would occur. In addition, not following the order would mean multiple traversals of the warehouse, seeing as Grab would never follow any convention for retrieving boxes.

One way to ensure a robot arm constraint would be to build horizontal layers of boxes while starting from the back left. Similar packing patterns are mentioned by Bortfeldt and Wäscher [24] as a *robot-packable pattern*. Bortfeldt and Wäscher [24] states that a robot-packable pattern occurs when the first box is placed in the back left bottom corner of the pallet, and all successive boxes are either placed in front, to the right, or on top of the previously placed boxes. Martello et al. [53] propose a heuristic method for finding a robot-packable pattern, while Boef et al. [44] propose constraint inequalities requiring the next box to be placed to have one of its coordinates greater than that of the previously placed boxes. Other than that, few authors have investigated robot-packable patterns. The robot patterns formulated

by Bortfeldt and Wäscher [24] and Boef et al. [44] are explored.

Binary variables are introduced to formulate the robot-packable pattern by Boef et al. [44], indicating whether a consecutive box has a greater than or equal coordinate to that of the previously placed box. These can be seen in table 3.21.

$robot_{x_{ij}}$	indicates whether $x_{i+1} \geq x_i + x\_cover_i \ \forall i \in \mathscr{I}$
$robot_{y_{ij}}$	indicates whether $y_{i+1} \geq y_i + y\_cover_i \ \forall i \in \mathscr{I}$
$robot_{z_{ij}}$	indicates whether $z_{i+1} \geq z_i + r_i \ \forall i \in \mathscr{I}$
$robot\_var_{ij}$	$robot_{x_{ij}}$ OR $robot_{y_{ij}}$ OR $robot_{z_{ij}} \ \forall i \in \mathscr{I}$

Table 3.21: Binary variables for the simple robot-packable pattern proposed by Boef et al. [44]

$$robot_{x_{i+1}}, robot_{y_{i+1}}, robot_{z_{i+1}}, robot\_var_{i+1} \in \{0,1\} \ \forall i \in \mathscr{I} \tag{188}$$

Equations (189) to (192) formulate constraint inequalities for ensuring a robot-packable pattern, using the binary variables introduced in table 3.21. Firstly, big-M constraints are used to compare the coordinates of consecutive boxes in the box order. Then, for any given box i, it is required to have one x, y or z-coordinate greater than that of all previously placed boxes.

$$L(1 - robot_{x_{i+1}}) + x_{i+1} \geq [x_j + x\_cover_j \, for \, j \in \{1,..,i\}] \ \forall i \in \mathscr{I} \tag{189}$$

$$W(1 - robot_{y_{i+1}}) + y_{i+1} \geq [y_j + y\_cover_j \, for \, j \in \{1,..,i\}] \ \forall i \in \mathscr{I} \tag{190}$$

$$H(1 - robot_{z_{i+1}}) + z_{i+1} \geq [z_j + r_j \, for \, j \in \{1,..,i\}] \ \forall i \in \mathscr{I} \tag{191}$$

$$robot_{x_{i+1}} + robot_{y_{i+1}} + robot_{z_{i+1}} \geq robot\_var_{i+1} \tag{192}$$

$$robot\_var_{i+1} \geq 1 \tag{193}$$

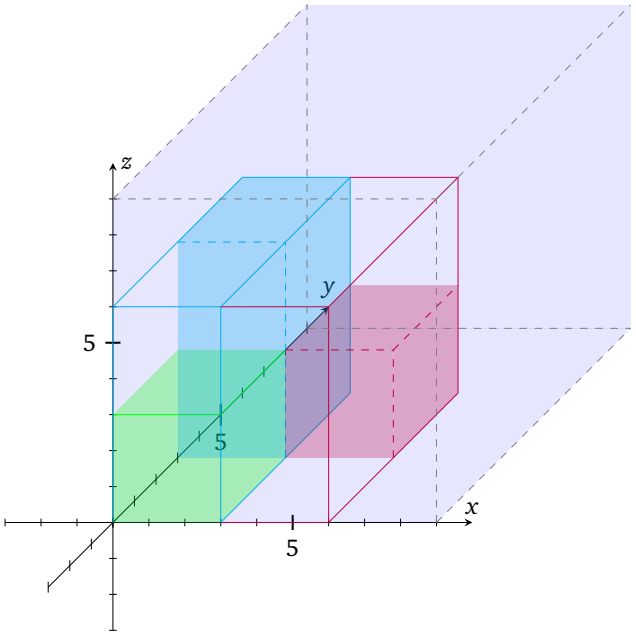


Figure 3.28: The robot-packable pattern as defined by Boef et al. [44]. The order of placement is: green, blue, and purple. The lines encapsulating the placed boxes and empty space indicate the robot constraints that each box activated.

The problem with the solution by Boef et al. [44] is that it restricts box placement a lot. This can be seen in figure 3.28.

Bortfeldt and Wäscher [24] define a robot-packable pattern to be:

"*A robot-packable pattern is a pattern which can be implemented by successively placing boxes, starting from the left corner in the back of the container and placing each further box either in front, on the right or on top of the previously placed ones.*" - Bortfeldt and Wäscher [24]

From the definition above, it can appear as if a robot-packable pattern is ensured by restricting boxes to be placed in front, on the right or on top of either box already placed. However, such a pattern is not a robot-packable pattern, as a box can be placed where the robot arm cannot reach. An example of such a scenario can be seen in figure 3.29. The definition by Bortfeldt and Wäscher [24] generates a robot-packable pattern if boxes are thought of as an entity. And if so, then robot-packable patterns by Bortfeldt and Wäscher [24] and Boef et al. [44] are alike.

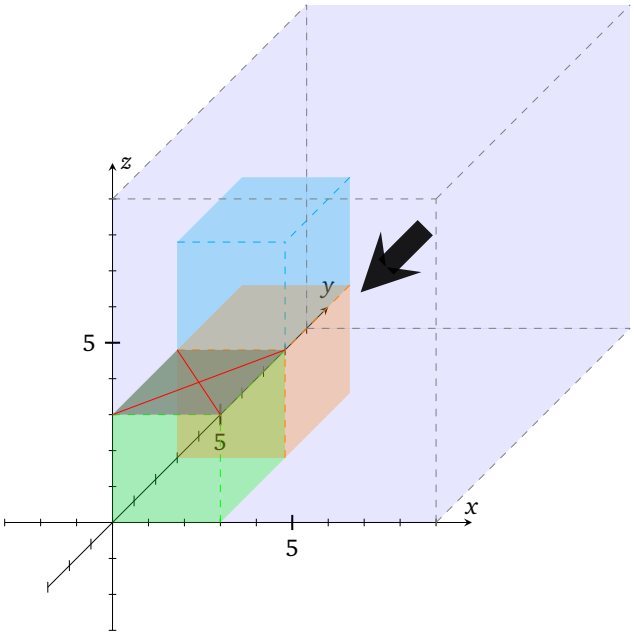


Figure 3.29: The robot arm places the green box first, then the orange box in front, and lastly the blue box on top of the orange box. The robot approaches with the arrow, but it will be unable to place a new box on top of the green box (on top of any previously placed box).

Worded constraints to the pattern by Bortfeldt and Wäscher [24] where boxes are not thought of as an entity have been implemented. As this pattern could potentially be used for something else, its written constraint is added in appendix A.

3.8 Load Bearing

None of the explored robot-arm constraints are sustainable for large instances, as their restrictions on the solution space are too strict. The purpose of the robot-arm constraints was to partially ensure indirect load-bearing constraints. Hence, direct load-bearing constraints are explored as an alternative to the robot-arm constraints.

If a box has too much weight on top, it might get deformed, or its content may be destroyed. Thus, a *load-bearing constraint* ensures that the maximum pressure that a box experience from boxes placed above is not exceeded [24].

State of the art

Two different worded load-bearing constraints are proposed by Nascimento et al. [10]. The first one is based on the work by Junqueira et al. [37] and involves restricting the number of boxes placed on top of any box i. In its formulation, each box i is given a maximum number of boxes that can be placed on top of it: f_i.

Boxes can be classified as *fragile*, meaning that other boxes cannot be placed on top of them [38]. Hence, a fragile box would be the same as if box i had $f_i = 0$.

The second load-bearing constraint by Nascimento et al. [10] restricts the maximum pressure per unit area. For execution, if a box j is placed on top of i, it calculates the pressure exerted onto i from j using j's maximum weight per unit area: $\frac{w_j}{p_j q_j}$. Then, the total pressure exerted on i to a maximum pressure constant η_i ($\forall i \in \mathscr{S}$) is restricted. A similar constraint is also mentioned in Bortfeldt and Wäscher [24].

Ratcliff and Bischoff [51] mention how box load-bearing ability may vary based on a box' contents and location within the stack. In addition, Ratcliff and Bischoff [51] suggest how a cardboard box may be stronger at its edges due to how it is constructed. Accommodating these notions is difficult as they require many MIP formulation edge-cases and are thus, not pursued.

Decision

There exists few constraint inequalities for load-bearing constraints [24]. Hence, the two load-bearing constraints by Nascimento et al. [10] are formulated in section 3.8.1 and section 3.8.2. In addition, the author proposes a load-bearing constraint concept based on partial base stability in section 3.8.3.

3.8.1 Number of boxes on top of a box

Worded constraints to the first load-bearing constraint by Nascimento et al. [10] are seen in Code listing 3.5. The conditionals check whether a box j is placed above box i. If all the conditionals are *true*, then $on\_top_{ij} = 1$. Lastly, the constraint ensures that the number of boxes placed above box i is no greater than f_i.

Six binary variables are introduced in table 3.22 for constraint inequalities to the worded constraints in Code listing 3.5. Five binary variables indicate the conditionals, while if_lb1_{ij} is used for *AND*-ing the five conditionals together.

```
if (z[j] >= z[i] + r[i] &&
    x[j] < x[i] + x_cover[i] &&
    x[i] < x[j] + x_cover[j] &&
    y[j] < y[i] + y_cover[i] &&
    y[i] < y[j] + y_cover[j])

    on_top_ij = 1
end

@constraint(sum(on_top_ij) <= f_i)
```

Code listing 3.5: Pseudo code for the first load-bearing constraint presented by Nascimento et al. [10]

$if_{lb1}\_1_{ij}$	indicating if $z_j >= z_i + r_i \; \forall i, j \in \mathscr{I}$
$if_{lb1}\_2_{ij}$	indicating if $x_j < x_i + x\_cover_i \; \forall i, j \in \mathscr{I}$
$if_{lb1}\_3_{ij}$	indicating if $x_i < x_j + x\_cover_j \; \forall i, j \in \mathscr{I}$
$if_{lb1}\_4_{ij}$	indicating if $y_j < y_i - y\_cover_i \; \forall i, j \in \mathscr{I}$
$if_{lb1}\_5_{ij}$	indicating if $y_i < y_j + y\_cover_j \; \forall i, j \in \mathscr{I}$
if_lb1_{ij}	*AND*-ing the binary variables above $\forall i, j \in \mathscr{I}$

Table 3.22: Binary variables indicating conditionals from Code listing 3.5

Equations (194) to (203) finds the values of the variables of table 3.22 similarly to how was done in section 3.4.1.

$$H(1 - if_{lb1}\_1_{ij}) \geq z_i + r_i - z_j \; \forall i, j \in \mathscr{I} \tag{194}$$

$$H \times if_{lb1}\_1_{ij} \geq z_j - z_i + r_i + \epsilon \; \forall i, j \in \mathscr{I} \tag{195}$$

$$L(1-if_{lb1}\_2_{ij}) \geq x_j + \epsilon - (x_i - x\_cover_i)\ \forall i,j \in \mathscr{I} \tag{196}$$

$$L \times if_{lb1}\_2_{ij} \geq (x_i - x\_cover_i) - x_j - \epsilon\ \forall i,j \in \mathscr{I} \tag{197}$$

$$L(1-if_{lb1}\_3_{ij}) \geq x_i + \epsilon - (x_j - x\_cover_j)\ \forall i,j \in \mathscr{I} \tag{198}$$

$$L \times if_{lb1}\_3_{ij} \geq (x_j - x\_cover_j) - x_i - \epsilon\ \forall i,j \in \mathscr{I} \tag{199}$$

$$W(1-if_{lb1}\_4_{ij}) \geq y_j + \epsilon - (y_i - y\_cover_i)\ \forall i,j \in \mathscr{I} \tag{200}$$

$$W \times if_{lb1}\_4_{ij} \geq (y_i - y\_cover_i) - y_j - \epsilon\ \forall i,j \in \mathscr{I} \tag{201}$$

$$W(1-if_{lb1}\_5_{ij}) \geq y_i + \epsilon - (y_j - y\_cover_j)\ \forall i,j \in \mathscr{I} \tag{202}$$

$$W \times if_{lb1}\_5_{ij} \geq (y_j - y\_cover_j) - y_i - \epsilon\ \forall i,j \in \mathscr{I} \tag{203}$$

The *AND*-constraint from section 3.4.1 could not be used this time. The reason for this is due to the inequality sign in the load-bearing constraint, as seen in equation (205). If the *AND*-constraint from section 3.4.1 was used, then the solver could just set if_lb1_{ij} equal to 0, and so the load-bearing constraint in equation (205) would always be satisfied. Hence, another *AND*-constraint is formulated in equation (204). An example of the presented load-bearing constraint being satisfied can be seen in figure 3.30.

$$if\_lb1_{ij} \geq if_{lb1}\_1_{ij} + if_{lb1}\_2_{ij} + if_{lb1}\_3_{ij} + if_{lb1}\_4_{ij} + if_{lb1}\_5_{ij} - 4\ \forall i,j \in \mathscr{I} \tag{204}$$

$$\sum_{i!=j}^{\mathscr{I}} if\_lb1_{ij} \leq f_i\ \forall i \in \mathscr{I} \tag{205}$$

3.8.2 Pressure per unit area

The constraint inequalities for the pressure per unit area load-bearing constraint proposed by Nascimento et al. [10] are very similar to the load-bearing constraint in section 3.8.1. The worded constraints for the pressure per unit area load-bearing constraint can be seen in Code listing 3.6.

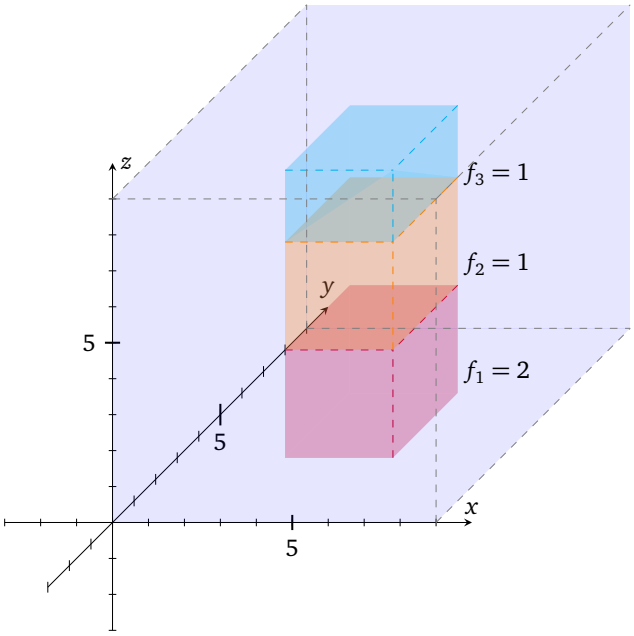


Figure 3.30: The number of boxes placed above box i is never greater than the maximum number allowed f_i. Hence, the load bearing constraint is satisfied.

```
if (z[j] >= z[i] + r[i] &&
    x[j] < x[i] + x_cover[i] &&
    x[i] < x[j] + x_cover[j] &&
    y[j] < y[i] + y_cover[i] &&
    y[i] < y[j] + y_cover[j])

    p_ij = w_j/(l_j*w_j)
end

@constraint(sum(p_ij) <= n_i)
```

Code listing 3.6: Load bearing constraint 2 by Nascimento et al. [10]

Six binary variables are introduced to check whether a box j is placed above box i. The six new binary variables are introduced in table 3.23. However, as the constraints inequalities for finding these variables are the same as for section 3.8.1, the reader is referred to equations (194) to (203) for their formulations.

$if_{lb2\_}1_{ij}$	indicating if $z_j >= z_i + r_i \ \forall i, j \in \mathscr{I}$
$if_{lb2\_}2_{ij}$	indicating if $x_j < x_i + x\_cover_i \ \forall i, j \in \mathscr{I}$
$if_{lb2\_}3_{ij}$	indicating if $x_i < x_j + x\_cover_j \ \forall i, j \in \mathscr{I}$
$if_{lb2\_}4_{ij}$	indicating if $y_j < y_i - y\_cover_i \ \forall i, j \in \mathscr{I}$
$if_{lb2\_}5_{ij}$	indicating if $y_i < y_j + y\_cover_j \ \forall i, j \in \mathscr{I}$
if_lb2_{ij}	*AND*-ing the binary variables above $\forall i, j \in \mathscr{I}$

Table 3.23: Binary variables indicating conditionals from Code listing 3.6

The constraint inequality for the pressure per unit area load-bearing constraint multiplies if_lb2_{ij} with a weight per unit area constant for each box j. Then, the sum of pressure per unit area for each box i must be less than a maximum pressure per unit area constant η_i. The pressure per unit area constraint can be seen in figure 3.31.

$$\sum_{i!=j}^{\mathscr{I}} if\_lb2_{ij} \times \frac{w_j}{p_j q_j} \leq \eta_i \ \forall i \in \mathscr{I} \tag{206}$$

3.8.3 Partial base load bearing

This last load-bearing constraint was developed by the author. Currently, it is not an effective load-bearing constraint but could potentially be developed further.

There is a drawback with the load-bearing constraints proposed by Nascimento et al. [10]: if a box j is placed just a tiny fraction on top of i, then j's entire mass is counted towards the load-bearing constraint of i. As seen in figure 3.31, the blue box' φ_4 counts towards both the green and the orange box' η_i. Thus, the load-bearing constraints by Nascimento et al. [10] over-estimate the pressure each box must handle.

A load-bearing constraint based on partial base stability is proposed to address this drawback. By finding the partial base support between two boxes i and j, the correct overlap between i and j is found. Thus, it is likely that a more accurate weight on top of the lower placed box can be found.

Equation (207) only counts the weight from boxes placed directly on top of a box i. However, further work with this constraint could mean that the correct weight

from all the boxes above i could be found. The partial base support load-bearing constraint can be seen in figure 3.32.

$$\sum_{i!=j}^{\mathscr{I}} \frac{pbs\_ij_{ij} \frac{w_i}{p_i q_i}}{area\_of\_box_i} \leq \eta_i \tag{207}$$

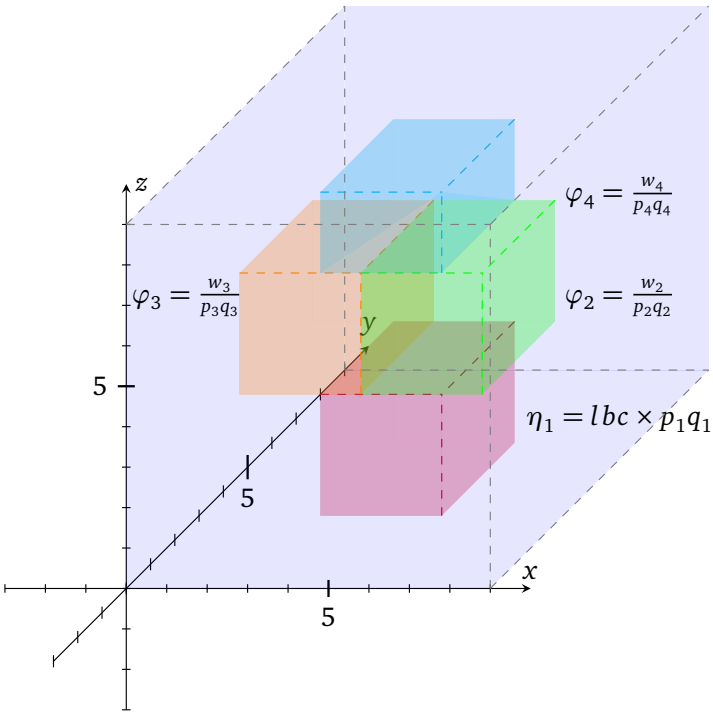


Figure 3.31: The pressure per unit area load-bearing constraint. Above, η_1 is defined as some constant lbc multiplied by the area of the box $p_i q_i$.

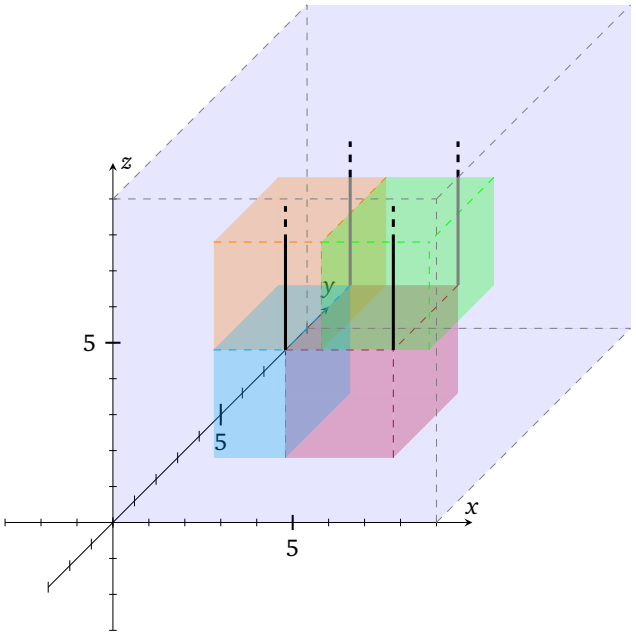


Figure 3.32: The partial base load-bearing constraint. Only the weight directly above a box i is counted towards i's load-bearing constraint. This is indicated by the black lines for the red box. In this case, the partial base load-bearing constraint is not entirely accurate, but more precise than either of the load-bearing constraints by Nascimento et al. [10].

3.9 Other Practical Constraints

On top

Some of the constraints mentioned in the previous sections are ineffective unless placed on top of other boxes. An example is a load-bearing constraint: if boxes placed above the pallet floor are not placed in contact with any boxes below, then load-bearing constraints are always satisfied. Hence, an *on-top* constraint is introduced to be used in combination with other constraints, restricting boxes to be placed on the pallet floor or being in contact with boxes placed below. The implementation of the on-top constraint uses equations equations (35) to (47) previously defined for the partial base stability constraint in section 3.4.1. As the pallet floor is defined as a box, boxes placed on the pallet floor are technically speaking standing on top of another box. Thus, the on top constraint is satisfied.

Summary of other constraints

Research literature has proposed numerous constraints for the CLP and PLP. As it could be interesting for the reader to gain insight into what constraints exist, this section provides a brief summary. In-depth overviews of practical constraints can be found in the articles by Bortfeldt and Wäscher [24] and Nascimento et al. [10].

Many authors consider PLPs and CLPs that have *multiple containers/pallets*. E.g., the basis MIP formulation by Chen et al. [26] was initially formulated for problems with multiple containers. In addition, there exist constraints only relevant to problems considering multiple containers/pallets. An example of this would be an *allocation constraint*, restricting a subset of boxes to be placed in specific containers/pallets [24].

Multi-drop constraints are also considered by many authors [24]. These types of constraints consider situations when there are multiple stops for offloading. Hence, multi-drop constraint ensures that boxes are placed according to their offloading stop.

Positional constraints ensure that boxes of certain types cannot be placed close to each other within the same container/pallet [24]. E.g., if food and washing powder were to be loaded into the same container, it would be ideal if they were placed apart.

With output maximisation problems, other constraints emerge such as *complete-shipment constraints* and *loading priorities constraints* [24]. If the total volume of the boxes exceeds that of the container/pallet, there can be priorities regarding what should be loaded. E.g., if two parts have to be screwed together at the destination, then ideally, it would be best if they arrived together.

3.10 Model space complexity

Many variables and constraints have been introduced in the MIP formulations of chapter 3. In table 3.24, the space complexity for each formulated constraint is shown. All rely on an instance's number of boxes, n, for complexity calculation.

The constraints noted with "*" are dependent on other constraints to be effective. However, space usage is shown independent of any required additional constraints.

Constraint	Variables	Constraints
Geometric	$6n^2 + 12n$	$7n^2 + 3n$
All orientations	0	$6n$
Currence Robotics' orientations	0	$9n$
Partial base stability	$16n^2$	$26n^2 + n$
Partial base stability reduced	$11n^2$	$17n^2 + n$
Multiple boxes constraint*	$n^2 + n$	$2n^2 + 3n$
Non-guillotine criterion	$54n^2 + 20n$	$74n^2 + 32n$
Multiple boxes incentive*	0	0
Middle incentive	$4n^2 + 7n$	$8n^2 + 5n$
Load balancing	0	4
On top	$8n^2 + n$	$14n^2 + 3n$
Robot arm	$4n$	$(\frac{3n}{2} + 2)n$
Robot arm #2	$26n^2$	$(39 + \frac{n}{2})n^2 + 3n$
Load bearing number	$6n^2$	$11n^2 + n$
Load bearing pressure	$6n^2$	$11n^2 + n$
Load bearing PBS*	0	1

Table 3.24: Space complexity of the implemented constraints

Chapter 4

Results

In this chapter, different compositions of the MIP formulations developed in chapter 3 are tested. The experiments that were carried out needed to resemble Currence Robotics' PLP. Hence, the approach that was used to design the experiments is presented. In addition, hardware, software, and constant- and parameter values used when the experiments were carried out are outlined. In total, over 4500 lines of code in Julia was written, and over 700 CPU hours of experiments were run [12]. The source code is public at Frich [11].

Furthermore, the results of the experiments are presented, along with the strategy for how the results were produced. And finally, the results of the developed MIP models are compared to the results of Currence Robotics' heuristic solution scheme for Currence Robotics' PLP.

4.1 Setup

In this section, the configuration for the conducted experiments is discussed. This includes hardware, software, parameter values, and instances.

4.1.1 Computing

There exist many different MIP solvers. *CPLEX* and *Gurobi* are among the most used for the CLP and the PLP [25]. CPLEX is used is used for the CLP and the PLP by,

amongst others: Nascimento et al. [10], Junqueira et al. [27], Alonso et al. [39], and Paquay et al. [38]. Gurobi is used by Junqueira and Queiroz [28]. Silva et al. [25] use both CPLEX and Gurobi and performs a comparative study between the two.

Silva et al. [25] conclude, using the MIP formulation by Chen et al. [26] as a basis, that Gurobi yields higher quality solutions than CPLEX when a runtime limit is imposed on a problem. The runtime limit in Currence Robotics' PLP is 2400 seconds, as this is the processing time Currence Robotics has given themselves before Grab needs to start fetching boxes. Hence, Gurobi was chosen as the MIP solver for the conducted experiments. More specifically, Gurobi v0.9.14 with default settings [54]. The MIP formulations were implemented in Julia v1.6.4 using the JuMP v22.1 mathematical optimisation package [12, 55].

No experiments were conducted using Grab's internal processing unit due to inconvenience. Instead, the NTNU IDUN computing cluster was used [56]. "*IDUN*" has more than 70 nodes, each containing two Intel Xeon cores with at least 128 GB of main memory. However, no more than 32 GB of main memory was ever used in an experiment.

The heuristic method of Currence Robotics was used as given to the author from Currence Robotics, with all parameter values set to default. To ensure an equal basis for comparison, the heuristic method was run on IDUN using the same instances as the MIP model. To visualise the resulting stack of boxes from both the optimised MIP models and the heuristic method, a visualisation tool by Currence Robotics was used, able to draw stacks of boxes in a point cloud.

4.1.2 Instances

The instances that were used in the experiments had to be created. Hence, they had to be designed to produce robust results with the potential to answer RQ 2.4 regarding instances' impact on problem difficulty. The approach that was used to design the instances is described below.

Researchers conduct experiments using weakly- or strongly heterogeneous instances depending on their research goals. The instances proposed by Bischoff and Ratcliff [42] employ 15 different sets of instances varying in heterogeneity and have been much used by researchers. However, many of the instances proposed by Bischoff and Ratcliff [42] are irrelevant to Currence Robotics' PLP, as only strongly heterogeneous boxes are motivated. Thus, to ensure the most relevant instances, boxes of real orders from the customers of Currence Robotics were used (see section 2.5).

A customer's real order contains between 60 to 70 boxes. Hence, preliminary testing concluded real-order instances were too hard for the MIP models, as they contained too many boxes. In addition, a limited number of real orders were available for use. A strategy was developed to create instances using real orders that were solvable for the MIP models. Firstly, it was decided to use fewer boxes; with instances containing 3, 5, 7, 10, 15, 20, or 30 boxes. For these numbers of boxes, preliminary testing showed that the solver could produce results for all of them. Thus, different constraints could be observed at increasing difficulty.

The available real-order boxes were added to a pool of 200 boxes from which random draws were made to create the N-box instances. Ten instances were created for each N-box instance group and were most likely different, as there is 1 313 400 ways of choosing 3 boxes from 200, and 4.1×10^{35} ways of choosing 30 boxes from 200.

In table 4.1, the characteristics of the boxes in the pool are shown. Most of the characteristics should be self-explanatory, except "Avg. box dimensional difference". This was found by averaging the average absolute value difference between the dimensions of each box in an instance.

4.1.3 Problem parameters

Many of the practical constraints from chapter 3 had parameter values, usually deciding how strict the constraint ought to be. Hence, these constraints' relevant literature was consulted to see what parameter values different researchers had proposed.

The initial solution space of the experiments was defined to be equal to the real-world PLP of Currence Robotics. A pallet has the following dimensions: $L = 1.2$ and $W = 0.8$. Hence, L and W in the MIP formulations were set to those values. The maximum stack height H was set equal to the H used in Currence Robotics' heuristic method: $H = 2.0$. The reason for using $H = 2.0$ is because it allows the robot arm a comfortable distance between the top of the pallet and its maximum height reach. In addition, very few stacks become taller than 2.0 meters.

The load balancing constraint and its parameters are explored by Ramos et al. [50]; investigating how a truck behaves when containers are placed on top of it. Ramos et al. [50] found that restricting the centre of gravity between 0 and L in the x-axis, $0.4W$ and $0.6W$ in the y-axis, and 0 and $0.9H$ in the z-axis ensures for a stable truck [10, 50]. In Currence Robotics' PLP, Grab is driving along with its pallet along the floor (as seen in figure 1.1). Hence, both longitudinal (start and stop) and lateral

Instance characteristics	
Characteristic	Value
Average weight (units)	9.807
Average length (m)	0.270
Average width (m)	0.255
Average height (m)	0.162
Maximum length (m)	0.600
Maximum width (m)	0.520
Maximum height (m)	0.351
Minimum length (m)	0.080
Minimum width (m)	0.098
Minimum height (m)	0.010
Avg. number of each box (#)	1.786
Avg. box dimensional diff. (m)	0.116

Table 4.1: Characteristics of boxes in the pool of boxes

(corners) forces act on the stack. As no walls are countering these forces, having a centre of gravity located at one of the sides of the pallet might cause an angle on the pallet. Moreover, boxes may fall over. To prevent this, the parameter values by Ramos et al. [50] for both the x-axis and the y-axis are employed. No z-axis constraint is imposed, as preliminary tests showed the stack's height did not exceed $0.9H$ for any instances.

Constraints must be deemed relevant. E.g., a load-bearing constraint allowing 100 boxes to be placed on top of any box would always be satisfied. On the other hand, a too strict measure does not allow any solutions. Both scenarios leave the researcher with little information about the constraints. In literature, most researchers set the weights of the boxes according to some strategy defined by themselves. E.g. Junqueira et al. [37] set each box weight proportional to its volume.

Load-bearing constraints correlate with the weights of the boxes. Hence, researchers can set the weights of the boxes so that the load-bearing constraint in focus can be deemed relevant. Researchers also use randomness to define the load-bearing

constraint of each box. Junqueira et al. [37] and Ramos et al. [50] use a random number multiplied with some property of a box such as its volume or height to decide its load-bearing property.

In Currence Robotics' PLP, the weight of each box is given by the real orders. Hence, a relevant load-bearing constraint had to be created. After some preliminary testing, it was decided to implement the load-bearing constraint proposed by Nascimento et al. [10]: make the pressure any box i can withstand proportional to i's volume. Hence, a large box would be able to withstand much pressure. In turn, this meant that the other load-bearing constraints defined in section 3.8.1 and section 3.8.3 were not included due to irrelevancy.

Researchers have proposed many different support factors for the partial base support criterion, ranging between 55% to 95% [48]. However, as mentioned in section 3.4, Hemminki et al. [36] claim that a support factor of 70% is sufficient for the PLP if plastic is wrapped around the stack of boxes after they have been placed. In Currence Robotics' PLP, this is the case. Hence, a support factor of 70% is imposed. For the multiple box static stability implementation, the initial figure proposed by Carpenter and Dowsland [6] of 5% partial base support from two or more boxes below was established.

It was decided not to include the non-guillotine criterion's callback function in the experiments. This decision was made based on preliminary testing that proved it non-sustainable for larger instances due to the reasons mentioned in section 3.5.1. However, the non-guillotine criterion is still included in the experiments but is referred to as the *guillotine* constraint.

The orientation constraint allowing for all rotations and the robot arm #2 constraint were omitted from the experiments. This was due to them being irrelevant for the PLP of Currence Robotics. A table of all the constraints used in the experiments can be seen in table 4.3.

Table 4.2 summarises the discussed constants and constraints' parameter values.

4.2 Results

An overarching strategy for producing results was created to answer the research questions related to the goal of testing the developed MIP. The chosen strategy split the experimenting phase into two parts. The first part of the experiments tested the individual constraints introduced in chapter 3 by themselves. Based on these res-

Constant	Value
L	1.2
W	0.8
H	2.0
LB lower_x	0.4L
LB higher_x	0.6L
LB lower_y	0.4W
LB lower_y	0.6W
PBS factor	0.7

Table 4.2: Summary of constant values used in experiements

ults, a discussion regarding what combinations of individual constraints may best fit Currence Robotics' PLP was held. The two measures used to evaluate constraints were their contribution to solving a sub-problem of Currence Robotics' PLP and their performance. The second part of the experiments tested the chosen combinations of individual constraints and evaluated their effectiveness in solving Currence Robotics' problem. Lastly, to address the third research question, the chosen combinations of individual constraints were compared with the results of Currence Robotics' heuristic method.

4.2.1 Individual constraints

The results of the individual constraints are shown as averages of the ten instances used for each N-box instance in table 4.3. The top row indicates what N-box instance the results are valid for. In the row below, T_s indicates the average time in seconds spent to obtain a solution, while $G_{\%}$ indicates the average solution gap from optimum that was achieved in percentages (see section 2.1). If a gap $G_{\%}$: $G_{\%} \leq 0.01\%$, the result was declared as an optimum, as this is the default value for optimum provided by Gurobi [54]. On the other hand, a $G_{\%} = 100\%$ indicates that no solution was found. Each T_s is rounded off to the closest one thousands of a second to avoid losing information, while each $G_{\%}$ is rounded up to its nearest 0.01%. Hence, no gaps are recorded as lower than their actual value.

In table 4.4, the standard deviation for each time and gap result is shown. For both σ_T and σ_G, the numbers are rounded off to the closest decimal. In table 4.5, the number of optimal solutions found for each individual constraint is shown.

The constraints are listed using the following abbreviations: GEO: Geometric, LB: Load Balancing, Robot: Robot arm, OT: On top, PBS 1: Partial base stability proposed by Nascimento et al. [10], PBS 2: Partial base stability with reduced space usage, G: Guillotine, G OT: Guillotine On top, LBP OT: Load-bearing On top, MB DS OT: Multiple box dynamic stability incentive On top, MID OT: Middle incentive On top, MB SS OT: Multiple box static stability constraint On top.

Instances	3 boxes		5 boxes		7 boxes		10 boxes		15 boxes		20 boxes		30 boxes	
MIP\T+G	T_s	$G_{\%}$	T_s	$G_{\%}$	T_s	$G_{\%}$	T_s	$G_{\%}$	T_s	$G_{\%}$	T_s	$G_{\%}$	T_s	$G_{\%}$
GEO	0.130	0	0.133	0	0.046	0	0.088	0	2.932	0	1921.484	17.12	2400	24.24
LB	0.018	0	0.004	0	0.008	0	0.044	0	1.083	0	1920.411	17.78	2400	24.53
Robot	0.003	0	0.004	0	0.060	0	3.211	0	15.520	0	155.137	0	2400	100
OT	0.005	0	0.010	0	0.021	0	0.124	0	10.763	0	1941.298	18.46	2400	32.57
PBS 1	0.044	0	1.210	0	128.209	0	2235.061	37.53	2400	100	2400	100	2400	100
PBS 2	0.026	0	0.828	0	144.927	0	2040.272	36.99	2400	100	2400	100	2400	100
G	0.088	≤ 0.01	23.519	≤ 0.01	1949.098	0.02	2400	0.12	2400	21.65	2146.390	19.76	2400	39.38
G OT	0.135	≤ 0.01	25.279	≤ 0.01	2060.763	0.02	2400	0.07	2400	29.29	2222.450	22.94	2400	68.68
LBP OT	0.007	0	0.020	0	0.040	0	0.231	0	61.225	0	2046.628	20.30	2400	100
MB DS OT	0.005	≤ 0.01	0.039	0	0.859	0	264.005	0.02	1259.041	0.11	2272.409	19.84	2400	72.42
MID OT	0.004	0	0.009	0	0.021	0	0.126	0	8.933	0	1936.687	18.46	2400	32.71
MB SS OT	0.010	0	0.741	0	154.340	0	2400	100	2400	100	2400	100	2400	100

Table 4.3: The average box instances results for each number of boxes

Instances	3 boxes		5 boxes		7 boxes		10 boxes		15 boxes		20 boxes		30 boxes	
MIP\σ_T, σ_G	σ_T	σ_G	σ_T	σ_G	σ_T	σ_G	σ_T	σ_G	σ_T	σ_G	σ_T	σ_G	σ_T	σ_G
GEO	0.172	0	0.204	0	0.111	0	0.171	0	5.556	0	1008.805	12.74	0	11.58
LB	0.042	0	0.000	0	0.006	0	0.100	0	1.451	0	1011.063	12.56	0	11.02
Robot	0.000	0	0.000	0	0.079	0	4.734	0	7.053	0	79.006	0	0	0
OT	0.001	0	0.001	0	0.004	0	0.212	0	15.172	0	967.303	13.03	0	1.12
PBS 1	0.029	0	0.800	0	192.753	0	521.582	27.64	0	0	0	0	0	0
PBS 2	0.012	0	0.738	0	268.141	0	763.338	28.43	0	0	0	0	0	0
G	0.029	0	19.319	0	669.686	0	0	0.14	0	13.19	690.385	13.17	0	13.59
G OT	0.095	0	17.314	0	609.191	0.01	0	0.02	0	28.49	561.463	13.17	0	22.63
LBP OT	0.003	0	0.011	0	0.009	0	0.412	0	98.232	0	762.189	13.76	0	0
MB DS OT	0.002	0	0.049	0	1.287	0	753,896	0.04	1087,714	0.18	403.478	11.39	0	29.41
MID OT	0.001	0	0.001	0	0.002	0	0.230	0	12.645	0	977.044	13.03	0	11.05
MB SS OT	0.001	0	0.502	0	153.489	0	0	0	0	0	0	0	0	0

Table 4.4: The standard deviations for the results obtained in table 4.3

The geometric constraint is the basis for every other implemented constraint, making it very unlikely that any other constraint can perform better than the geometric

MIP\Instances	3	5	7	10	15	20	30
GEO	10	10	10	10	10	2	0
LB	10	10	10	10	10	2	0
Robot	10	10	10	10	10	10	0
OT	10	10	10	10	10	2	0
PBS 1	10	10	10	1	0	0	0
PBS 2	10	10	10	2	0	0	0
G	10	10	4	0	0	2	0
G OT	10	10	3	0	0	1	0
LBP OT	10	10	10	10	10	2	0
MB DS OT	10	10	10	9	6	1	0
MID OT	10	10	10	10	10	2	0
MB SS OT	10	10	10	0	0	0	0

Table 4.5: The number of optimal instances found for each constraint from the 10 box instances for each number of boxes

constraint. This reaction can be seen in table 4.3 and table 4.5, as the geometric constraint is the best performing constraint for all instance-groups. The only exception is the robot arm constraint, whose results are presented shortly. For 30 boxes, the geometric constraint's gap is limited to an average of 24.24%, with its σ_G equalling 11.58. Following, in terms of performance, is the load balancing constraint, with an average gap of 24.53% and its σ_G equalling 11.02. The reason for the high σ_G is most likely due to the instances that have been used. In figure 4.1, a 20-box instance solved with geometric constraints is shown. And even though no additional constraints are applied, a compact stack is achieved.

The differences between instances and what makes them difficult or easy to solve will be discussed in section 5.1.2. However, it can be observed that the other constraints, for the most part, have an even higher σ_G.

The robot constraint is, as mentioned in section 3.7, not sustainable for a large number of boxes. An example of this can be seen in figure 4.2. Furthermore, it can be observed in table 4.3 that the robot constraint's results do not align with the rest of the constraints. E.g., it has an T_s of 155.137 for the 20-box instances, much lower than the other constraints. However, the robot arm constraint achieved no results for 30-box instances, as there was not enough space in the pallet to accommodate

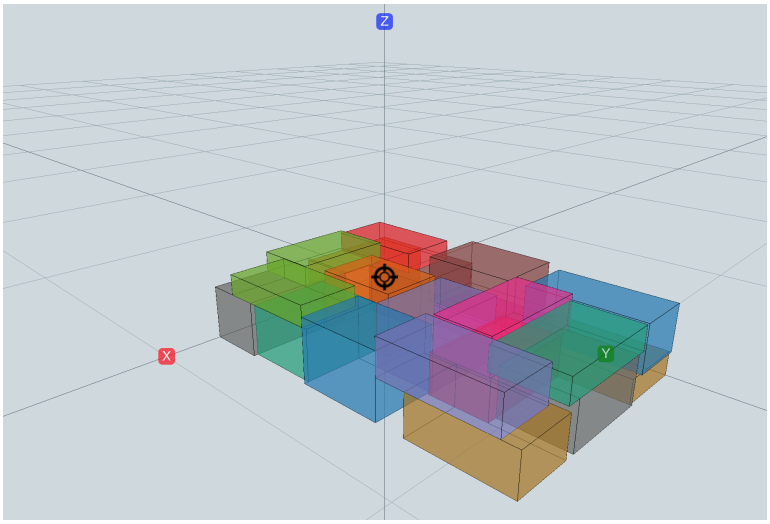


Figure 4.1: Geometric constraint with 20 boxes

it. The robot constraint could have been combined with the on top constraint. However, this would have caused the resulting stack to be one horizontal line of boxes at the bottom and one vertical line of boxes advancing towards the top of the pallet: H.

As seen in table 4.3, the implemented on top constraint obtained worse results than the geometric- and the load balancing constraint. In particular for 30-box instances. However, it was only 1.34% off the geometric constraint for 20-box instances. Lastly, its σ_G was significantly lower than the other constraints' σ_G for 30-box instances.

Both the partial base stability criterion by Nascimento et al. [10] (PBS 1) and the reduced space partial base stability criterion introduced in section 3.4.2 (PBS 2) performed almost equally good. The only real difference was how PBS 2 was able to find another optimal solution, in addition to spending approximately 200 seconds less on average for 10-box instances. σ_T and σ_G were very high for both PBS 1 and PBS 2.

In figure 4.3, the PBS 2 constraint is used on a 10-box instance that was not solved optimally. Still, it can be observed that the resulting stack appears to be a stack that could have been used in real life. In particular, if plastic was wrapped around the stack as suggested by Hemminki et al. [36] and as done by Currence Robotics.

As mentioned in section 3.5.1 and section 4.2.1, the guillotine constraint could be

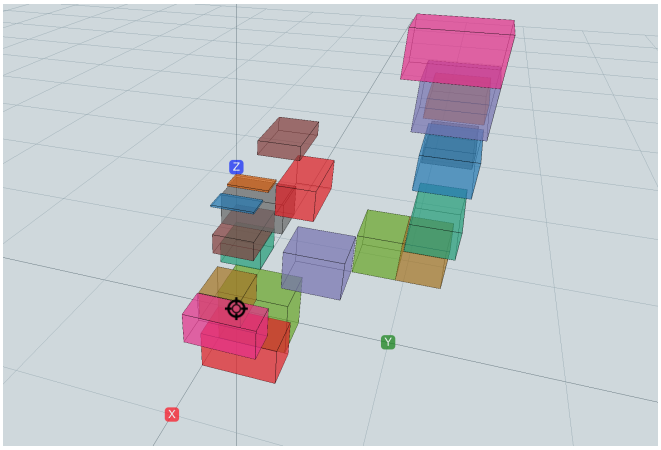


Figure 4.2: Robot arm constraint with 20 boxes

used for something else than ensuring the non-guillotine criterion. As an incentive is given for guillotine cuts, boxes are placed as such. This did, as seen in figure 4.4, cause compact stacks. The guillotine constraint was, as seen in table 4.3, also able to find solutions for 30-box instances. However, as the guillotine constraint is based on incentives with small adjustments, few optimal solutions could be found.

The guillotine constraint was also combined with the on-top constraint in an attempt to create an alternative partial base support constraint. In figure 4.5, the on top guillotine constraint is used on a 7-box instance in what appears to be a stack generated by the partial base support constraint. However, there was a limit to how much compacting work the solver could manage in 2400 seconds for larger instances, causing less compact stacks for larger instances.

The load-bearing constraint needed an on-top constraint, or else it would always be satisfied. For smaller instances where few boxes had to be placed on top of each other, the load-bearing constraint performed well. However, for 15-box instances and up, more boxes would need to have one or two boxes on top of them. Hence, the load-bearing constraint emerged and started to use more time. For 30-box instances, no results were achieved.

Figure 4.6 shows a 20-box instance solved using the load-bearing- and the on top constraint. It can be observed how the resulting stack is not as compact as geometric constraints' stack from a similar 20-box instance in figure 4.1.

The multiple boxes incentive with the on top constraint (MB DS OT) performed

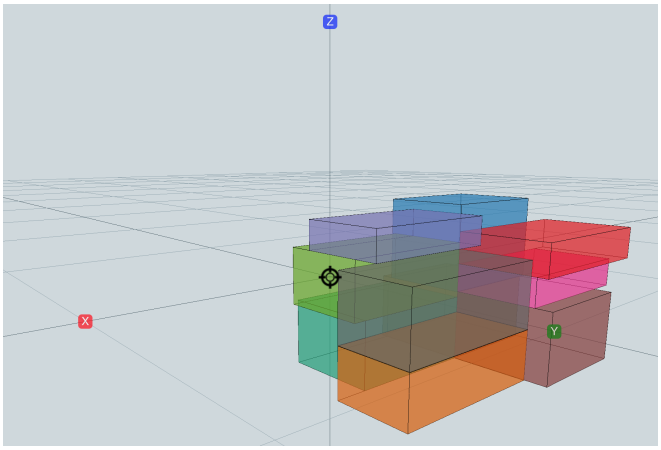


Figure 4.3: PBS 2 with 10 boxes

well for all, except the 30-box instances. This was in contrast to the multiple box constraint and the on-top constraint (MB SS OT), which could not yield solutions for 10-box instances and up. Seemingly alike, they produced very different results. Reasons for this will be discussed in section 5.2.1. Thus, concluding the results of the individual constraints.

4.2.2 Combined constraints

The second part of the experiment phase revolved around creating a MIP formulation of Currence Robotics' PLP using combinations of individual constraints. Hence, each individual constraint had to be evaluated regarding their performance and efficacy in solving a sub-problem of Currence Robotics' PLP. Multiple combinations of individual constraints were created for a clearer image of the best possible MIP formulation.

All of the constraints evaluated in section 4.2.1 are relevant to Currence Robotics' PLP. However, many of them enforce equal stack qualities. Those that do not are the geometric constraints and the load balancing constraint. Hence, the geometric constraint and the load balancing constraint were included in all combinations.

Some robot arm constraint was also needed. However, none of the implemented robot arm constraints are viable for large instances. Preliminary testing showed that 30-box instances would become less relevant for the combinations of individual constraints. Thus, the load-bearing constraint, performing poorly for 30-box instances,

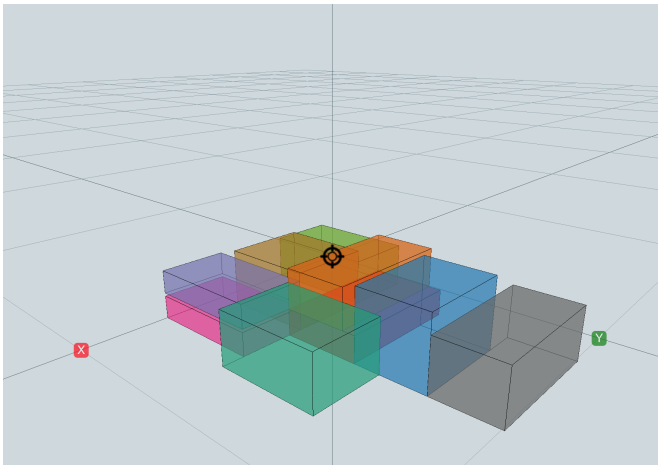


Figure 4.4: A compact stack using the guillotine constraint with 9 boxes

was included as a substitute.

A static stability constraint was needed so that boxes did not fall to the ground. Ideally, a partial base support constraint could be used to provide stack stability guarantees. However, as seen in table 4.3 it lacked performance, as no stacks could be generated for 15-box instances and up. Hence, the guillotine- and the on-top constraint, providing static stability and able to solve larger instances, were also included. The multiple box static stability constraint did not perform well enough to be considered.

The multiple box and middle incentives are used to ensure dynamic stability. However, they cannot be used alongside the guillotine constraint, as their incentives oppose each other. While the middle incentive and the multiple box incentive emulate the non-guillotine criterion, the guillotine constraint does the opposite. Hence, the middle and the multiple box incentive needed to be used with the partial base support constraint.

In total, four combinations of constraints were created. The first was a combination of guillotine, on top, load balancing and load-bearing (G OT LB LBP). Then, a similar combination, but with partial base support instead: PBS 2 LB LBP. PBS 2 G LB LBP was created to look at potential synergy effects between G and PBS 2. OT was omitted from PBS 2 G LB LBP as PBS 2 ensures an on-top constraint. Lastly, a combination was created as an attempt to ensure static- and dynamic stability: PBS 1 LB LBP MID MB.

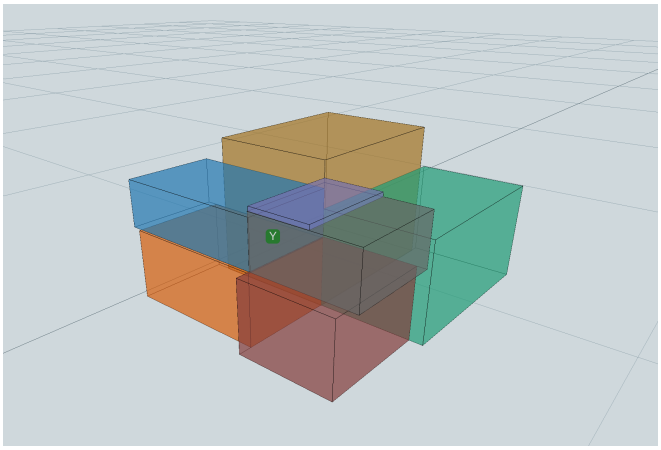


Figure 4.5: Guillotine constraint with 7 boxes

The results of these combinations are shown in table 4.6. Furthermore, the standard deviations for the results are given by table 4.7, and the number of optimal instances found for each combination can be seen in table 4.12.

Instances	3 boxes		5 boxes		7 boxes		10 boxes		15 boxes		20 boxes		30 boxes	
MIP\T+G	T_s	$G_{\%}$	T_s	$G_{\%}$	T_s	$G_{\%}$	T_s	$G_{\%}$	T_s	$G_{\%}$	T_s	$G_{\%}$	T_s	$G_{\%}$
G OT LB LBP	0.226	$\geq$ 0.01	27.445	$\geq$ 0.01	2313.861	0.02	2400	0.08	2400	35.63	2353.410	42.59	2400	100
PBS 2 LB LBP	0.038	0	1.337	0	167.049	0	2238.559	90	2400	100	2400	100	2400	100
PBS 2 G LB LBP	2.052	$\geq$ 0.01	102.035	$\geq$ 0.01	2304.973	49.39	2400	100	2400	100	2400	100	2400	100
PBS 1 LB LBP MID MB	0.419	0.02	20.577	$\geq$ 0.01	1789.656	2.14	2400	100	2400	100	2400	100	2400	100

Table 4.6: Results to the combination of constraints

From the results in table 4.6, it can be observed that the G OT LB LBP was the only combination able to generate stacks for 15- and 20-box instances. However, G OT LB LBP provides no guarantees of static stability. An example of why can be seen in figure 4.7, in which the top red box only has a tiny fraction of its base supported by the large dark-brown box. In the real world, the top red box and the light-brown box on top of it would have fallen down. Still, G OT LB LBP can generate viable real-world stacks sometimes. An example can be seen in figure 4.8, where all boxes are compact and have sufficient base support.

PBS 2 LB LBP is the only combination other than G OT LB LBP that can generate stacks for 10-box instances. However, as can be seen in table 4.12 (showing both

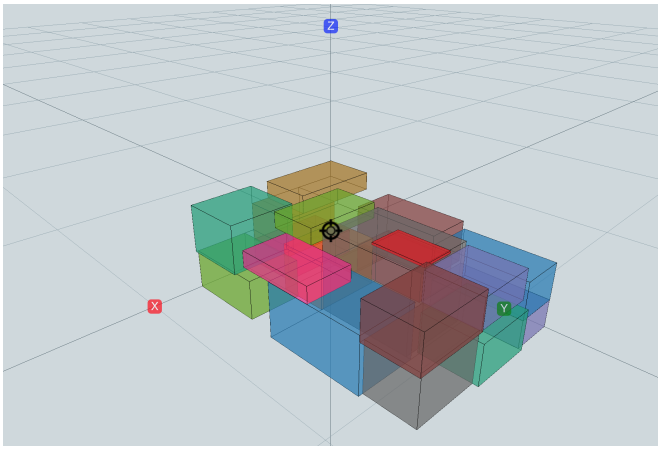


Figure 4.6: LBP On Top with 20 boxes

Instances	3 boxes		5 boxes		7 boxes		10 boxes		15 boxes		20 boxes		30 boxes	
MIP\σ_T, σ_G	σ_T	σ_G	σ_T	σ_G	σ_T	σ_G	σ_T	σ_G	σ_T	σ_G	σ_T	σ_G	σ_T	σ_G
G OT LB LBP	0.177	0	18.516	0	272.396	0	0	0.01	0	16.70	147.332	32.68	0	0
PBS 2 LB LBP	0.014	0	1.435	0	292.978	0	510.523	31.62	0	0	0	0	0	0
PBS 2 G LB LBP	1.287	0	43.015	0	300.502	47.15	0	0	0	0	0	0	0	0
PBS 1 LB LBP MID MB	0.202	0.03	13.661	0	736.613	0.35	0	0	0	0	0	0	0	0

Table 4.7: The standard deviations for the results obtained in table 4.6

optimal and non-optimal solutions), it was only able to solve one 10-box instance. PBS 2 LB LBP is a smaller version of the PBS 1 LB LBP MID MB combination. Hence, the reason why PBS 1 LB LBP MID MB cannot generate results for 10-box instances.

The only difference between PBS 2 LB LBP and PBS 1 LB LBP MID MB is that the latter has an incentive to place boxes on top of multiple boxes and in the middle of the underlying boxes. Other than that, they are equal. In figure 4.9, a stack from PBS 2 LB LBP is shown. However, it could have been a stack created by both solutions.

The last combination to be experimented with was PBS 2 G LB LBP, exploring the synergy effects between PBS 2 and G. This combination was difficult for the solver. And as can be seen in table 4.12, it was only able to solve seven of the ten 7-box instances, obtaining an average gap of 49.39%.

However, PBS 2 G LB LBP generated compact and statically stable stacks despite having large average gaps for the 7-box instances. In figure 4.10, a non-optimal

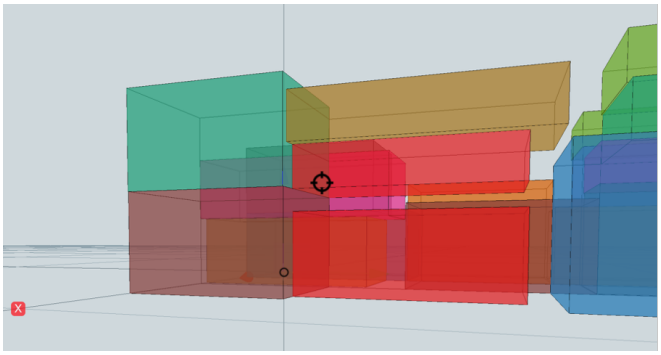


Figure 4.7: Non-sufficient static support using G OT LB LBP

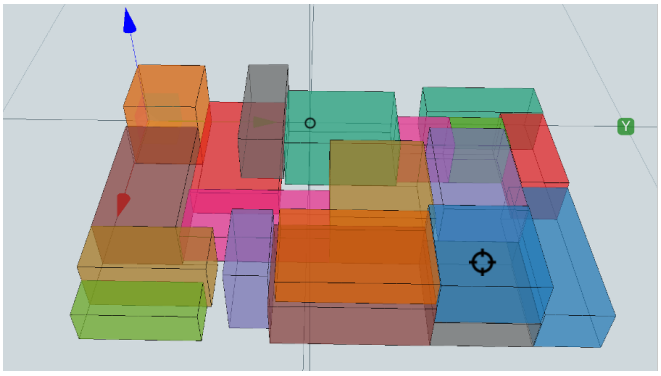


Figure 4.8: A compact, statically stable stack of G OT LB LBP

solution of a 7-box instance of PBS2 G LB LBP is shown. And even though the stack is non-optimal, it is still compact and statically stable.

Figure 4.10 is a good example of the load-bearing constraint in action, as the large brown box is placed on the bottom with the lighter orange and black box on top of it.

4.2.3 Heuristic- vs MIP results

Currence Robotics' heuristic method results were obtained using the same instances as for the MIP models. The heuristic method's results and can be seen in table 4.8. In table 4.8, *Time* indicates average running times for each N-box instance. *Compacity* indicates the height of the stack divided by the pallet area and is a measure used by Currence Robotics to measure the compactness of a stack.

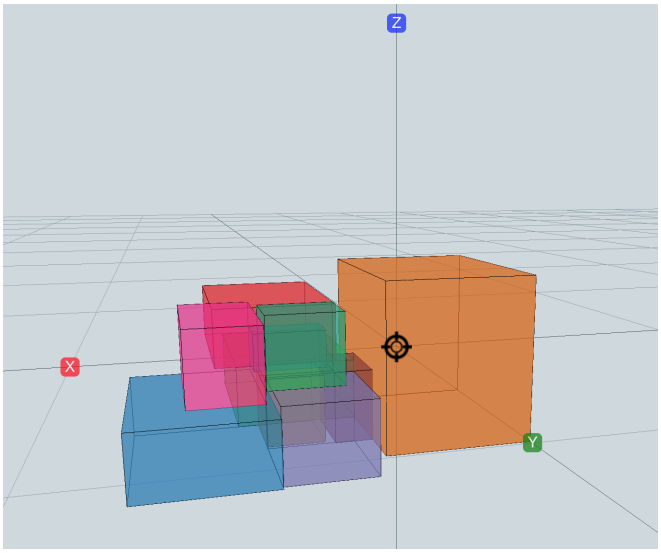


Figure 4.9: PBS 2 LB LBP

Meas.\Instances	3	5	7	10	15	20	30
Time	0.664	0.080	0.050	0.030	1.572	1014.714	2400
Height	0.256	0.224	0.258	0.244	0.252	0.302	0.404
Compacity	0.216	0.261	0.358	0.485	0.647	0.710	0.780

Table 4.8: Currence Robotics' heuristic method results

In table 4.9, a time comparison between the heuristic method and the MIP combinations is shown. It can be observed that the heuristic method spends less time solving each instance than the MIP combinations.

In table 4.10, it can be observed that the σ_{Time} of the heuristic method on the 20-box instances is very high. This is probably related to the instance difficulty differences, later discussed in section 5.2.1. The heuristic method's standard deviations for the other instances were low.

No one metric determines what makes a good stack. Hence, it is difficult to compare the stacks of the heuristic method and the MIP models. In addition, the two solutions optimise for many different qualities, making it even more challenging. The one major characteristic that both solutions optimise for is a low stack, as a

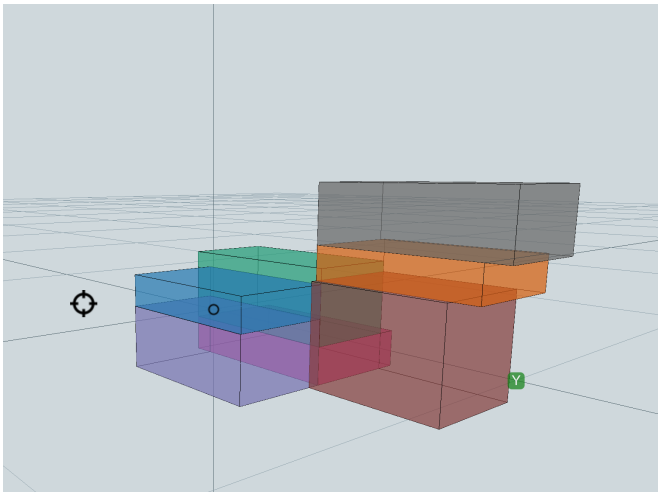


Figure 4.10: PBS2 Guillotine LB LBP with non-optimal solution

Time\Instances	3	5	7	10	15	20	30
Time currence	0.664	0.080	0.050	0.030	1.572	1014.714	2400
Time MIP	0.684	37.848	1643.882	2359.640	2400	2388.352	2400

Table 4.9: Time comparison between Currence Robotics' heuristic method and the average time spent of the MIP combined-constraints

low stack is likely to result in a compact stack. In table 4.11, the heights achieved by the different exact combined-constraints solutions are compared to the heuristic methods' resulting heights. A height was only recorded in table 4.11 if a solution was found. In general, Currence Robotics achieved equal- or lower stack heights to the combinations of individual constraints.

In figures 4.11 to 4.13, visualisations of stacks created by the heuristic method of Currence Robotics can be seen. In figure 4.11, it can be observed that, for the most part, the heuristic method can create compact solutions to ensure that the height of the stack is kept low. Then, in figure 4.12, a 30-box instance containing multiple larger boxes can be seen, causing the stack to become relatively tall. In figure 4.13, a solution using the same instance as in figure 4.10 is shown.

Another comparison metric for the heuristic method and the combinations of individual constraints is the number of solutions each managed to find for all instances.

σ\Instances	3	5	7	10	15	20	30
σ_{Time}	2.080	0.009	0.004	0.030	3.783	946.349	0
σ_{Height}	0.087	0.046	0.074	0.065	0.075	0.064	0.097
$\sigma_{Compacity}$	0.079	0.101	0.150	0.119	0.072	0.052	0.011

Table 4.10: The standard deviation of the different measures to the heuristic method of Currence Robotics

Method\Instances	3	5	7	10	15	20	30
Guillotine OT LB LBP	0.256	0.224	0.258	0.244	0.583	0.304	-
PBS 2 LB LBP	0.256	0.224	0.264	0.351	-	-	-
PBS 2 Guillotine LB LBP	0.256	0.224	0.322	-	-	-	-
PBS 1 LB LBP MID MB	0.256	0.224	0.264	-	-	-	-
Currence Robotics	0.256	0.224	0.258	0.244	0.252	0.302	0.404

Table 4.11: The average height of the generated stacks by the different solution schemes

In table 4.12, it can be observed that the heuristic method can solve all instances within the given runtime. However, the MIP models can not.

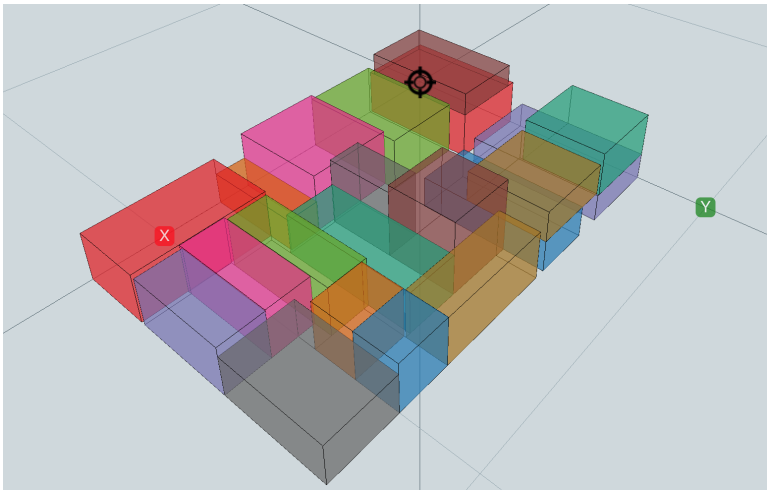


Figure 4.11: 20-box heuristically solved instance that is kept low in height

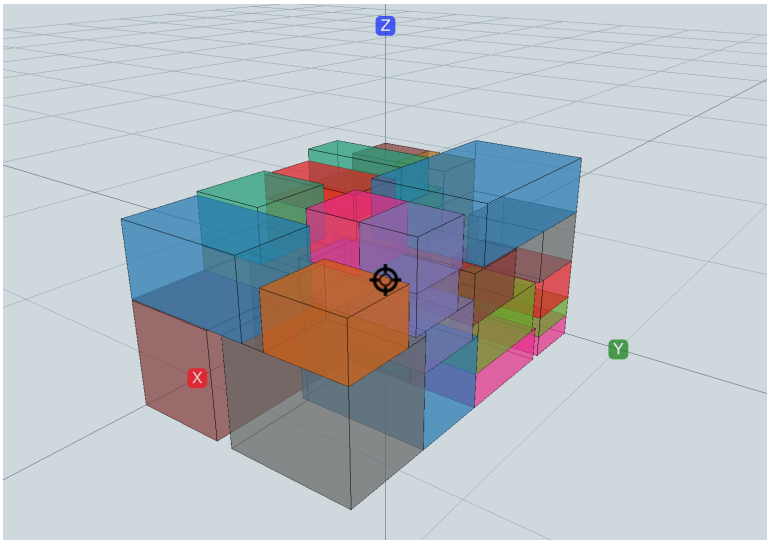


Figure 4.12: 30-box heuristically solved instance that is taller due to larger boxes

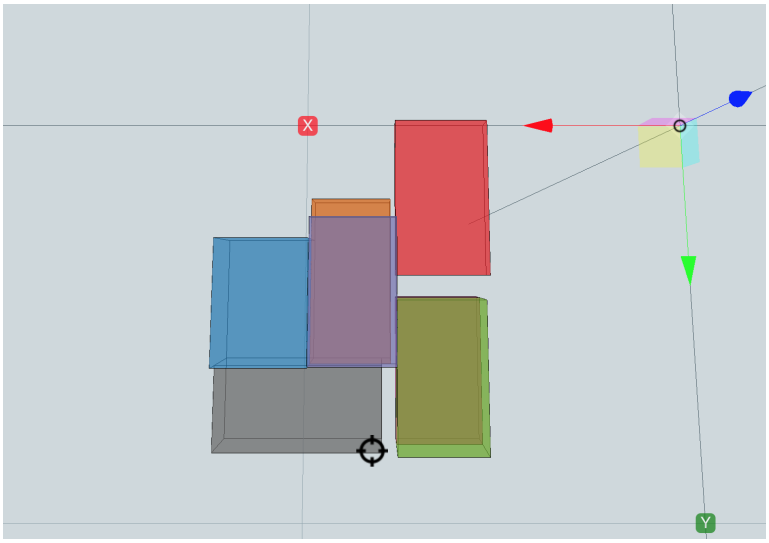


Figure 4.13: 7-box heuristically solved instance using the same instance as in figure 4.10

Method\Instances	3	5	7	10	15	20	30
Guillotine OT LB LBP	10	10	10	10	8	6	0
PBS 2 LB LBP	10	10	10	1	0	0	0
PBS 2 Guillotine LB LBP	10	10	6	0	0	0	0
PBS 1 LB LBP MID MB	10	10	10	0	0	0	0
Currence Robotics	10	10	10	10	10	10	10

Table 4.12: The number of solutions found for each instance for each number of boxes by the combined constraints

Chapter 5

Discussion

This chapter investigates the methodology, the findings, and the aim of the thesis. Firstly, the methodologies used in this thesis are examined. In particular, the development of the MIP formulation of Currence Robotics' PLP and the setup used in experiments. Then, the results and findings of the experiments are analysed in consideration to the relevant research questions. In addition, certain characteristics of MIP constraints are highlighted, such as constraint performance and efficacy. In closing, a more general discussion about the research aim is presented, considering methodology and the results. Moreover, based on this thesis' work, possibilities for creating better box stacks for Currence Robotics' PLP are presented.

5.1 Discussion of methodology

This section evaluates the development of the formulated practical constraints taking relevant research questions and results into consideration. Then, the solver and the instances used in the experiments are analysed.

5.1.1 Formulating MIP constraints

It was possible to formulate MIP constraints for many of the practical constraints in Currence Robotics' PLP. Either by using existing research or by proposing new constraints and improvements.

The guillotine constraint of this thesis is somewhat similar to the dynamic stability criterion presented by Nascimento et al. [10]. Intended for the CLP, the dynamic stability criterion by Nascimento et al. [10] ensures that a sufficient percentage of the walls of any box lie into other boxes or the container walls. As the dynamic stability criterion relies on container walls, it cannot be implemented for the PLP unless changed, either by eliminating the walls from the criterion or changing the constraint into an incentive. A dynamic stability incentive could have rewarded a box if its walls lay into other boxes. Such an incentive would be similar to the guillotine incentive, but possibly better as it would ensure that a more significant proportion of boxes' walls lay into other boxes, not only ensuring that boxes would be placed in cuts. Moreover, the guillotine constraint compromises dynamic stability (interlocking of boxes) for static stability, while the dynamic stability criterion explained above does not necessarily compromise anything.

The proposed load-bearing constraints were constant regardless of the number of boxes in an instance. Thus, less accurate load-bearing constraint results were achieved, which is why the load-bearing constraint could not produce any results for 30-box instances. Given more time, it is likely that a dynamic load-bearing constraint could have been formulated. However, the ideal load-bearing constraint is based on physical tests of the cardboard box strength. Therefore, spending too much time creating a dynamic load-bearing constraint may not have been worth it.

The load-bearing pressure- and number constraints of section 3.8.1 and section 3.8.2 were the only constraints of estimates that were formulated. They both overestimated the weight on top of a box. Estimates can be helpful as they may be simpler to formulate than 100 % accurate constraints. Furthermore, having a constraint estimate is a lot better than nothing.

The formulated MIP constraints of chapter 3 are very logical. They ensure sufficient support of boxes, balanced stacks, or similar. It could have been insightful to look at more creative formulations of constraints. Additionally, to look for alternative applications. E.g., the guillotine constraint of section 3.5.1 ensured different stack qualities from its intention and can, in hindsight, be thought of as a creative application for generating static stability.

It is possible that the size of the formulated constraints could have been reduced further by optimising variable and constraint inequality space usage. However, as seen from the performance of PBS 1 and PBS 2, doing so would not necessarily result in higher-performing constraints. Besides, for a certain number of boxes, the generated model sizes become large no matter what. Optimising and choosing smaller big-M constants would be difficult, as M was usually based on L, W or H.

5.1.2 Experimental setup

Solver

Solver hardware matters. Gurobi states on their website how high speed, low latency, and high-bandwidth memory are usually the best CPU characteristics for running their solver [57]. The CPUs used in Idun are Intel XEON scalable processors, which are not the best-performing CPUs with regards to the qualities mentioned by Gurobi [56, 58, 59]. Hence, it is likely that better results could have been achieved by using a different CPU. Investing in new hardware for this master thesis would have been too expensive. However, some companies are dependent on the best results that money can buy. And if they can save money on investing in new hardware, that should be considered. Still, the PLP is an NP-hard problem, and thus, better hardware can only do so much.

In Gurobi, a default optimal solution has a gap less than 0.01% off that solution's objective value [54]. This impacts the returned optimal solution stacks, in particular with regards to stack height. The reason is that 0.01% of a stack that is 1.2 meters high is greater than 0.01% of a stack that is 0.1 meters high. Consequently, constraints or incentives may be more or less relevant depending on the height of the stack. E.g., the guillotine constraint's incentive to ensure guillotine cuts becomes relatively smaller to the objective value if a stack is tall. A constant optimal solution threshold could be introduced to avoid this. Moreover, in doing so, the relationship between the quality of a stack and its height could be eliminated.

Are some instances more difficult to solve?

The ten different instances for each N-box instance were used to ensure robust results and reveal whether an instance's characteristics impacted its solving difficulty. If the difficulty of instances with the same number of boxes was equal, then σ_T and σ_G of table 4.4 and table 4.7 would be close to 0. However, high standard deviations were recorded for many of the N-box instances. E.g., Currence Robotics' heuristic method spent 2400 seconds to solve one 20-box instance while spending 0.648 seconds to solve a different 20-box instance. For the same instances, MID OT spent 2400 seconds on the first instance and 32.26 seconds on the latter. Consequently, the reasons for these differences are explored.

Statistical analysis was used to explore instance difficulty. Moreover, the correlation between characteristics of instances and average solving times and average gaps was found. The different characteristics that were analysed and their corresponding correlation can be seen in table 5.1. Most of the characteristics were also shown

in table 4.1, except "Tallest boxes diff.". The tallest boxes difference was found by subtracting the second tallest box height from the tallest box height in an instance.

Finding the correlation between instances and the time spent and gap achieved was done in two ways: firstly, for all the instances pooled together (indicated by "All"), and secondly as an average of the correlation found for the ten instances of each N-box instance group (indicated by "Avg.").

Multiple steps were taken to calculate the correlations in the *Average*-category. Firstly, the correlation within each N-box instance group was found. Secondly, those correlations were normalised using the Fisher transformation [60]. Then, the average correlation was found across all of the N-box instance groups, and lastly, that average was inversely transformed back to Pearson's correlation coefficient [61]. The Pearson correlation coefficient (r) can be seen in equation (208), the Fisher transformation (z) in equation (209), and the inverse Fisher transformation in equation (210).

$$r = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \tag{208}$$

$$z = \frac{1}{2}\ln\left(\frac{1+r}{1-r}\right) = \text{artanh}(r) \tag{209}$$

$$r = \frac{e^{2z}-1}{e^{2z}+1} = \tanh(z) \tag{210}$$

Having multiple variables of correlation can trigger the multiple comparisons problem of false positives [62]. Because of this, hypothesis tests were conducted for each category in table 5.1, checking whether the correlations were significant or not. A two-tailed z-test was conducted, using the strict Bonferroni correction to counteract the multiple comparisons problem [63]. Hence, for a number in table 5.1 to show correlation, it would need to be indicated as significant with regards to the Bonferroni correction and be a sufficiently large absolute correlation value. The correlations discussed below are Bonferroni correction significant.

As can be observed in the *All*-categories of correlations in table 5.1; the number of boxes and the total volume of boxes in an instance has a large positive correlation with both the average time spent solving an instance and the average gap achieved for a solved instance. This makes sense, as an increase in the number of boxes in

Correlation				
Measures	Time (All)	Time (Avg.)	Gap (All)	Gap (Avg.)
Number of boxes	0.937	n/a	0.963	n/a
Total weight of box instances	−0.301	−0.223	−0.288	−0.109
Avg. length of boxes	−0.219	0.263	−0.225	−0.094
Avg. width of boxes	−0.184	−0.055	−0.174	0.334
Avg. height of boxes	−0.358	−0.138	−0.339	−0.357
Total volume of boxes	0.836	−0.137	0.838	−0.370
Max length of a box	0.193	−0.097	0.148	−0.214
Max width of a box	0.348	−0.228	0.339	0.288
Max height of a box	0.125	−0.181	0.148	−0.589
Min length of a box	−0.510	0.279	−0.511	0.245
Min width of a box	−0.596	0.058	−0.585	0.088
Min height of a box	−0.546	0.051	−0.521	0.366
Avg. box dimensional diff.	0.038	0.232	0.016	0.393
Max volume of a box	0.159	−0.234	0.099	−0.532
Min volume of a box	−0.596	0.178	−0.544	0.472
Tallest boxes diff.	−0.080	−0.283	−0.153	−0.498
Avg. number of each box	0.366	0.163	0.351	0.191

Table 5.1: Characteristics of instances and their correlation between time spent solving the instance or resulting gap

a box instance also results in greater model sizes, again leading to an increased difficulty for the solver. Moreover, adding more boxes usually leads to an increased total box volume.

The correlations found for the All-category gives an impression that many box characteristics impact instance difficulty. However, this is not necessarily the case. We explain how by looking at the characteristics involving the minimum dimensions of a box (also includes the "Minimum volume of a box"). Larger instances have a greater probability of containing a box with small dimensions, as there are more

boxes to choose from. Hence, the All-category's minimum- and maximum dimensions correlations cannot be attached too much importance.

The correlations involving maximum- and minimum dimensions are more relevant for the Average-categories. This is because the Average-category disregards the number of boxes in an instance. In table 5.1, it can be observed that "Maximum height of a box" has a moderate negative correlation between it and the average achieved gaps. A possible explanation for this correlation may be due to the decision processes of Gurobi. If an instance contains tall boxes, then the solver can quickly determine that they should be placed on the pallet floor. In addition, having tall boxes may allow Gurobi to more easily see whether or not other boxes can combine to surpass the taller boxes in terms of height. Thus, reducing the gap. On the other hand, if an instance only contains short boxes, it may not necessarily be that clear what boxes should be placed where. Another reason for this correlation may be the objective function. And, it is uncertain whether a problem with a different objective function would have the same correlation between instances and their results. A scatter plot of the maximum-height-of-a-box correlation can be seen in figure 5.1.

Another moderately negative correlation is the correlation between the average gap and the "Maximum volume of a box". This correlation and the maximum-height-of-a-box correlation can most likely help explain each other, as the correlation between these two characteristics is 0.823, indicating a strong positive correlation. This may be because a tall box is more likely to have a greater volume.

In figure 5.1, the moderate negative correlation of -0.589 in the Avg-category of table 5.1 between the Maximum height of a box and the average gap achieved can be observed. No real trend is established. However, many of the instances with boxes that contain a small height have been solved optimally. On the other hand, many instances containing taller boxes have not been solved optimally.

There are few other notable correlations between box characteristics and the average solving time or the average gap achieved. Nevertheless, it is likely that the aggregated correlations of the other box characteristics can assist in explaining why some instances are harder to solve than others.

5.2 Discussion of results

In this section, the experimental results of the thesis are analysed with regards to relevant research questions. Firstly, the performance and efficacy of each individual constraint are examined. Then, the box stacks generated by the combinations of

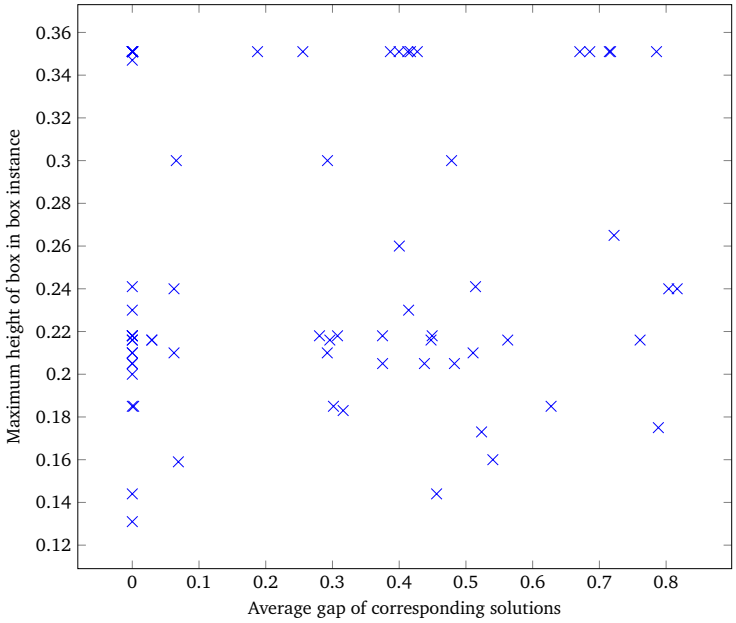


Figure 5.1: Moderate correlation between the maximum height of a box and the average achieved solution gaps

individual constraints are investigated. Is either able to solve Currence Robotics' PLP? Furthermore, the similarities and differences between the box stacks generated by these combinations of individual constraints and Currence Robotics' heuristic method are examined.

5.2.1 How did individual constraints perform?

The overall stack compactness from the geometric constraints was very high. This follows the argument by Pisinger [43], claiming that high space utilisation can ensure stable stacks without the need for a separate stability constraint. However, as seen in figure 4.7, the geometric constraints do not generate viable real-world box stacks, as boxes may be placed in the air without support.

The load balancing constraint did, as seen in table 4.3, perform almost equally as good as the geometric constraint. The reason is likely because the load balancing constraint requires few additional constraint inequalities. In addition, it can appear as if it is not the most challenging constraint to satisfy, which is also the conclusion to Nascimento et al. [10].

Currently, there does not exist sustainable robot constraints for large instances. Hence, a load-bearing constraint was explored as an alternative. It proved effective, except for 30-box instances. This is likely because the load-bearing implementation was too strict for that number of boxes and not because load-bearing constraints do not work for 30-box instances. However, in the real world, the actual load-bearing strength of cardboard boxes would be used (after physical tests were done).

If Currence Robotics were to implement a load-bearing constraint rather than using robot patterns, they would most likely achieve better stacks, as each box would have more placement options. However, this dilemma is for Currence Robotics to decide, as there are clear economic benefits with only requiring one traversal of the warehouse. E.g., multiple stacking robots can more easily be used concurrently.

An on-top constraint was introduced to ensure the effectiveness of other constraints. Seemingly successful, it compromised little with regards to performance. However, as seen in figure 4.7, additional constraints are needed to produce sufficient stability. Hence, the on top constraint may be redundant, as some static stability constraints also enforce an on-top constraint.

Table 4.3 shows that the multiple box static stability constraint concept proposed by Carpenter and Dowsland [6] is not suited for problems with strongly heterogeneous boxes. This is because there is a lack of equal-sized boxes to produce continuous surfaces for boxes to be placed on top of. Consequently, the static stability constraints proposed by Paquay et al. [38] and Ramos et al. [48] are also deemed unfit for problems with strongly heterogeneous boxes. Paquay et al. [38] required that three of the four vertices of a box must be covered by boxes below, making it a similar but stricter constraint than that of Carpenter and Dowsland [6]. Ramos et al. [48] restricted a box above the pallet floor to be placed within a support polygon. A support polygon can be made up of one box, but given a support polygon of multiple boxes, its boxes would need to be the same height. These arguments also apply, albeit to a lesser degree, to the middle incentive and the multiple-box incentive introduced in section 3.5 for the same reasons. It is even more challenging to satisfy these constraints in Currence Robotics' PLP, as orientations are limited. Moreover, limiting the number of orientations also limits the number of combinations of boxes that can be used to create continuous surfaces.

The implementation of PBS 2 requires less space than PBS 1. As a result, Gurobi explored 22 828 nodes on average for 30-box instances with PBS 1 and 41 114 nodes on average for PBS 2. Hence, it might be a surprise that they produced such similar results. A possible reason is that for both constraints to produce partial base support, all of the conditions removed from PBS 2 need to be satisfied. In PBS 2, these

conditions are indirectly integrated into the later inequalities, allowing Gurobi to explore solutions more freely when solving an instance with PBS 2. On the other hand, PBS 1 needs to satisfy its initial conditions before exploring possible solutions. Thus, PBS 2 uses more attempts at getting to the result as it has no guiding constraints, while PBS 1 starts off strict and then proceeds to look at solutions more likely to produce partial base support.

Another observation was made about the partial base stability criterion: if more time is left before a final result is needed, then higher quality solutions can be ensured by adding the partial base support constraint to the objective function. Thus, a solution does not have to be restricted by its minimum partial base support requirement.

No effective dynamic stability constraint was formulated, as neither the multiple box incentive nor the middle incentive provided sufficient interlocking of boxes. In addition, as both dynamic stability criteria were implemented as incentives, no guarantees for interlocking of boxes could be established.

The guillotine incentive proved to produce static stability, particularly for smaller instances. However, another application for which the guillotine concept may be a better fit is dynamic stability evaluation. Carpenter and Dowsland [6] states how interlocking boxes in a stack is desirable to ensure dynamic stability. Hence, by using a method similar to the guillotine incentive, all cut sizes in a stack could be found. Moreover, by measuring these cut sizes, the degree of stack interlocking of boxes could be decided. This method could also help measure the effectiveness of other dynamic stability constraints.

The efficacy of the implemented MIP constraints varied. However, it was apparent that new constraint inequalities for practical constraints are needed, as the proposed practical constraints could not solve all sub-problems of Currence Robotics' PLP.

5.2.2 How did combinations of constraints perform?

It is challenging to combine constraints, as a stack fulfilling one constraint might compromise another constraint. E.g. Parreño et al. [46] mention how strategies for ensuring dynamic stability can reduce a stack's static stability. The guillotine incentive of section 3.5.1 does the opposite, compromising dynamic stability for static stability. Nevertheless, as seen in the visualisations of section 4.2.2, combinations of constraints generated higher quality stacks than each individual constraint alone.

However, the combinations of constraints lacked performance. And as seen in sec-

tion 4.2.2, Gurobi was only able to solve smaller instances when solving using combinations of constraints. One of the reasons for this was the sheer size of the MIP models generated by the combinations. E.g., the solver was able to find a solution for the guillotine constraint for a 30-box instance. That model contained 71 773 rows, 53 506 columns and 289 734 nonzeroes. In comparison, the PBS 2 G LB LBP combination for a 30-box instance generated a MIP model with 99 707 rows, 47 289 columns and 442 554 nonzeroes. The second reason for low performance was the difficulty of individual constraints. As seen in table 4.3, the PBS constraints did not perform too well on their own. However, to ensure viable real-world stacks, a static stability constraint had to be included in the combinations of constraints. Furthermore, the combinations included additional practical constraints, which made the combinations even harder to solve.

Many of the combinations did provide for viable real-world stacks. However, what individual constraints contributed to that? PBS 2 LB LBP and PBS 1 LB LBP MID MB generated very similar stacks, but the latter had an incentive to place boxes that were above the pallet floor on top of multiple boxes and in the dimensional middle of the box(es) below. In section 5.2.1, an argument was made that constraints encouraging continuous surfaces made up of multiple boxes can be redundant in a problem with strongly heterogeneous boxes. Hence, it is likely that MID and MB did not add much to the stacks created by PBS 1 LB LBP MID MB.

PBS 2 G LB LBP did appear to have the most stable stacks (see figure 4.10). This can likely be attributed to the synergy-effect of PBS 2 and G. PBS 2 LB LBP and PBS 1 LB LBP MID MB achieved, along with PBS 2 G LB LBP, what appeared to be sufficient static support for all boxes. This was not the case for G OT LB LBP, as some of its generated stacks had boxes floating in the air (see figure 4.7). Thus, it can be concluded that the static support was mainly produced by the PBS constraints.

The combinations of constraints managed to generate high-quality box stacks - particularly PBS 2 G LB LBP. However, overall performance was lacking, and only small instances were solved within the runtime limit.

5.2.3 Heuristic vs MIP

In section 4.2.3, Currence Robotics' heuristic method is compared to the combinations of constraints. The heuristic method is better for almost all instances, spending less time finding better box stacks. Even for larger instances, it can produce compact and viable real-world stacks.

For smaller instances, the heuristic method spends very little time compared to the

combinations of constraints. However, they both produced similar stack quality. In figure 4.10 and figure 4.13 the same instance is used, but the first figure shows a non-optimal PBS 2 G LB LBP stack while the latter shows a stack generated by the heuristic method. From the figures, it can be argued that the solution schemes possess some different qualities. PBS 2 G LB LBP appears to generate more compact stacks, as all the boxes are placed into each other. On the other hand, as seen in table 4.11, the heuristic method creates stacks marginally lower in height. This is not necessarily crucial, as height is primarily a means to achieve a compact stack. To illustrate why, imagine a scenario in which there existed one very tall box. Then, the height of the stack could be equal to that tall box, but it would not indicate anything about the other boxes. However, a low stack is usually a compact stack.

PBS 2 G LB LBP generates compact stacks with little compromise in average stack height. Thus, for small instances, it can be argued that the average stack quality of PBS 2 G LB LBP is higher than the stack quality of Currence Robotics' heuristic method. Nonetheless, both methods generate viable real-life stacks.

Besides the comparisons made above, it is difficult to compare the stack quality of different methods. This is because different solution schemes focus on separate qualities or may fit specific contexts. In addition, to the best of the author's knowledge, little research has been conducted into what makes a good stack of boxes.

5.3 Creating better box stacks

In this section, the research aim is addressed by assessing the solution schemes explored in this thesis. Then, based on previous discussions, Currence Robotics' possibilities for creating better box stacks are outlined.

Using a MIP model

Researchers have not concluded what the best solution scheme for the CLP and PLP is. Alonso et al. [40] state that: "*The exact methods produce better quality solutions. And, as a real-world packing problem requires a certain amount of quality to be viable it is important to investigate exact methods further.*" This follows along the lines of a statement by Nascimento et al. [10]: "*These results show that there is still room for improvement regarding models and exact methods for container loading problems, especially if practical constraints are taken into consideration.*" By these statements, it can be extrapolated that development is needed before exact methods with practical constraints can produce real-world solutions. Using the results and insights from

this thesis, it is possible to get a clearer image of the shortcomings of exact methods, and more specifically, MIP models.

Alonso et al. [40] state that real-world packing problems require solutions that are of a certain quality. Heuristic methods can, most likely, produce the quality of solutions that are needed. However, they cannot guarantee stack qualities. These guarantees are sought after and are why researchers explore exact methods. MIP models cannot, presently, provide these guarantees mainly due to two reasons: there does not exist implementations of practical constraints that provide good enough guarantees in the real world. In addition, model sizes of MIP models become too difficult and too big to solve when practical constraints are enforced. These problems must be overcome before MIP models can provide viable real-world stacks for large instances. In section 4.2.2, it is noticed how MIP models can provide satisfactory stacks for smaller instances. Considering this, Currence Robotics can try to optimise their solution scheme.

2400 seconds can be spent before a solution is needed. Hence, Currence Robotics' goal should be to obtain the best possible solution within these 2400 seconds. Thus, it could be attempted to use a MIP model to solve the first couple of boxes for a large instance. This could provide a good foundation for a heuristic method to build upon. Such a solution scheme could potentially lead to the highest quality stack, as a MIP model could be modified to create the best possible foundations for heuristic methods.

However, modifying MIP models to create stack foundations for heuristic methods introduces another problem. And that is the problem of understanding what makes for a good foundation/intermediary stack. An *intermediary stack* is a stack of boxes where all boxes of an instance have not been placed. SINTEF, the researcher partner of Currence Robotics, is currently struggling with the intermediary stack problem for their convolutional neural network (CNN) of Currence Robotics' PLP, as it needs to know what intermediary stacks are likely to become viable real-world stacks.

Exact methods may also be used to indirectly solve the PLP or CLP, as there may be some sub-problems to heuristic methods that must be solved precisely. To the best of the author's knowledge, few researchers have proposed solution schemes for the PLP and the CLP that contain both exact- and heuristic components. However, this is probably not because it is ineffective, but because researchers are highly specialised in one research field.

Using a heuristic method

Gajda et al. [34] makes a case for heuristic methods, stating in a recent article that: "*...exact methods remain far from meeting industry standards where large instances with multiple constraints must be solved quickly.*". This could and probably should be considered by Currence Robotics when choosing what to do next. Currence Robotics' heuristic method is state of the art. Hence, few alternative heuristic methods can produce equally good results. However, there might be other ways of improving stack quality.

Silva et al. [25] found that small differences between the total box volume and the total container volume make it increasingly difficult to find optimal MIP solutions. This finding probably applies to heuristic methods as well. In the experiments of this thesis, the total box volume was much smaller than the total pallet volume. However, in a real order to Currence Robotics, the difference may be a lot smaller. Hence, a possibility could be to remove some boxes from an instance. E.g., restricting the heuristic method only to solve 50 boxes while placing the remaining boxes manually. In doing this, higher quality solutions are likely achievable. This approach does not solve Currence Robotics' PLP entirely, but the number of manual labour hours would be dramatically reduced without compromising stack quality.

Another possibility is to preprocess the instances more. As was discovered in section 5.1.2, a box' characteristic may make an instance harder to solve. Hence, solution schemes could be designed to only include boxes that make for a high probability of yielding a viable real-world stack. In addition, there may be other adjustments or changes to solution schemes that could come from further analysis and physical testing of boxes.

Currence Robotics' PLP

Currence Robotics' problem is that they cannot generate viable real-world stacks of boxes. And even though Currence Robotics' heuristic method generally achieves much higher-quality stacks of boxes than the tested MIP models, they are still not good enough. To overcome this, Currence Robotics can continue the development of their heuristic method while also waiting for new research to be proposed. However, generating a high-quality stack of boxes for a 70-box instance is difficult, and it will probably remain difficult for some time.

In the grand scheme, Currence Robotics's problem is an economic one. Can Grab reduce the costs of running a warehouse and create higher-quality stacks of boxes,

reducing transportation costs? This is likely possible with the solutions that exist today, however, not by using a solution scheme generating a mediocre 70-box stack.

It may be that Grab can generate very high-quality stacks containing 50 % of the boxes in an order using Currence Robotics' heuristic method. And for many companies, using a robot to create such stacks may dramatically reduce labour costs. Thus, high-quality stacks of boxes of smaller instances should not be neglected when Currence Robotics' are considering future solutions.

Chapter 6

Conclusion

This chapter answers the research questions, highlights the thesis' contributions and implications, and propose future work in the research field.

6.1 Conclusion

The aim of this thesis was to generate knowledge that can be used to create better stacks of boxes for Currence Robotics' PLP. This aim has been addressed by formulating a MIP model for Currence Robotics' PLP and by testing it and comparing it with Currence Robotics' heuristic method.

State of the art was consulted to identify the best solutions to sub-problems of Currence Robotics' PLP. Thus, based on work by different researchers, the following MIP constraints were formulated: geometric, multiple orientations, partial base support, multiple box static stability, load balancing, robot arm, load-bearing number, and load-bearing pressure. However, several requirements of Currence Robotics' PLP could not be satisfied by the state-of-the-art MIP constraints. Thus, the author proposed the following constraints and concepts: non-guillotine, multiple box incentive, middle incentive, partial base support, partial base support load-bearing, and on-top.

Each constraint was implemented in Julia using JuMP. Furthermore, constraint experiments were run for 2400 seconds with Gurobi using seven groups of N-box

instances, each containing ten different sets of strongly heterogeneous boxes. Constraints were chosen for another round of experiments, using combinations of individual constraints, based on performance and ability to solve sub-problems of Currence Robotics' PLP. Thus, four MIP models of Currence Robotics' PLP were proposed: PBS 2 LB LBP, PBS 1 LB LBP MID MB, G OT LB LBP, and PBS 2 G LB LBP. The latter MIP model (PBS 2 G LB LBP) produced the most viable real-world box stacks, providing compactness and sufficient static stability. However, the MIP model lacked performance and was only able to solve one of ten instances containing 10 boxes.

Currence Robotics' heuristic method was tested with the same instances and hardware as the MIP implementations. The method was able to solve instances with a large number of boxes and performed better than any of the four alternative MIP models proposed for Currence Robotics' PLP. However, PBS 2 G LB LBP tended to create more compact box stacks for smaller instances.

By developing and testing MIP models of Currence Robotics' PLP, it was observed that MIP models could not, with their current limitations, generate satisfactory box stacks for Currence Robotics' PLP. In addition, the results indicated that combining a MIP model and a heuristic method may result in better box stacks.

Due to the identified research gaps, the author proposed constraint inequalities to many practical constraints of Currence Robotics' PLP. In particular, an improved static stability partial base support constraint formulation, $\frac{1}{3}$ smaller in size than the state-of-the-art formulation by Nascimento et al. [10]. This finding may be vital for problems demanding small MIP models.

It was found that an instance tends to be harder to solve it if contains tall boxes. This finding may have implications for the development of solution schemes. Not only because researchers can ignore taller boxes to obtain better stacks of boxes, but because there may be more to learn from the boxes used in CLPs and PLPs.

6.2 Future work

There may be further ways to improve box stacks in Currence Robotics' PLP. And based on the work of this thesis, multiple directions for future research are proposed.

Setup

The parameter variables of the different practical constraints were chosen based on other researchers work [10, 36, 50]. However, as mentioned in section 4.1, researchers do not always agree on parameter values, and preferred values may vary by a lot. Thus, conducting physical tests of parameter values could be insightful for validation. In addition, doing so may result in more precise parameter values to practical constraints for Currence Robotics' PLP.

Model

There is a lack of MIP formulations for practical constraints, particularly for strongly heterogeneous boxes. There is also a need to create MIP formulations that use little space, as many of the existing formulations become too big, for instances with many boxes.

Silva et al. [25] conclude that an area of improvement for MIP formulations is to represent the relative position between boxes indirectly, as it could potentially reduce the number of problem variables. Researchers have yet, to the best of the author's knowledge, to propose such formulations. However, it may be a solution to the current limitations of MIP formulations.

Much PLP and CLP research revolve around weakly heterogeneous boxes. Weakly heterogeneous boxes are simpler to work with, as strategies can be developed for equal box dimensions. However, practical constraints for strongly heterogeneous boxes are effective for both strongly- and weakly heterogeneous boxes. Researchers should consider this in the future.

Most solution schemes focus on solving an entire instance. However, higher-quality box stacks are likely achievable by applying solution schemes to smaller instances. Either by designing solution schemes that focus on generating good intermediary box stacks or placing challenging boxes manually after a solution scheme has finished.

Testing

This thesis found a moderate negative correlation between tall boxes and the average gap achieved. In addition, multiple other characteristics of the instances were checked for correlation. However, there may be other methods for testing the relationship between instances and problem difficulty.

All of the experiments conducted in this thesis were directly relevant to the aim and goals. However, more insight could have been obtained by introducing experiments with significant variations in the MIP model or the instances. E.g., using instances with low or high values for some dimension to observe further correlations and constraints' behaviour with outlier boxes. Another option could be to introduce changes in the MIP model, like consistently placing the first box in coordinate $(0, 0, 0)$. Such experiments would not directly benefit Currence Robotics but could provide insight or validation for improving solution schemes later on.

Few stacking algorithms have been implemented and executed in the real world. And, to the best of the author's knowledge, no testing has been conducted with an autonomous system. Currence Robotics has an autonomous stacking robot, for which future real-world testing could be done. Consequently, it may have been possible to obtain alternative insights that few other researchers would have access to.

Another possible testing method not used in this thesis is simulator testing. Currence Robotics currently uses a simulator to test how much acceleration box stacks can resist without falling. Hence, dynamic stability criteria could be designed with the insight obtained from these simulator tests. It may also be possible to propose other simulator tests designed to test other stack qualities.

There exist few concrete measures for rating box stacks - both finished and intermediary. Having more precise rating measures of stacks could be valuable as it would mean that solution schemes could be designed to be more precise. Moreover, comparing stacks of different solution schemes would become a lot easier, as currently, some comparisons must be made by visual inspection.

Bibliography

[1] Wikipedia. Containerization, 2022. URL https://en.wikipedia.org/wiki/Containerization. Last accessed 13-February-2022.

[2] EU Science Hub. Transport sector economic analysis, 2021. URL https://ec.europa.eu/jrc/en/research-topic/transport-sector-economic-analysis. Last accessed 13-February-2022.

[3] The Economist. A perfect storm for container shipping, 2021. URL https://www.economist.com/finance-and-economics/a-perfect-storm-for-container-shipping/21804500. Last accessed 13-February-2022.

[4] S. Vargas-Osorio and C. Zúñiga. A literature review on the pallet loading problem. *Lámpsakos*, 15:69–80, 2016. doi: 10.21501/21454086.1790. URL http://www.funlam.edu.co/revistas/index.php/lampsakos/article/view/1790.

[5] J. Remes and S. Saxon. What's going on with shipping rates?, 2021. URL https://www.mckinsey.com/industries/travel-logistics-and-infrastructure/our-insights/whats-going-on-with-shipping-rates. Last accessed 13-February-2022.

[6] H. Carpenter and W. B. Dowsland. Practical considerations of the pallet-loading problem. *The Journal of the Operational Research Society*, 36(6):489–497, 1985. doi: 10/2582821. URL http://www.jstor.org/stable/2582821.

[7] G. Wäscher, H. Haußner, and H. Schumann. An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183(3): 1109–1130, 2007. doi: 10.1016/j.ejor.2005.12.047. URL https://www.sciencedirect.com/science/article/pii/S037722170600292X.

[8] I. Araya, M. Moyano, and C. Sanchez. A beam search algorithm for the biobjective container loading problem. *European Journal of Operational Research*, 286(2):417–431, 2020. doi: 10.1016/j.ejor.2020.03.040. URL https://www.sciencedirect.com/science/article/pii/S037722172030254X.

[9] T. Frich. Pallet loading problem. *Specialisastion project at the Norwegian University of Science and Technology (NTNU)*, 2021.

[10] O. Nascimento, T. A. Queiroz, and L. Junqueira. Practical constraints in the container loading problem: Comprehensive formulations and exact algorithm. *Computers & Operations Research*, 128:105186, 2021. doi: 10.1016/j.cor.2020.105186. URL https://www.sciencedirect.com/science/article/pii/S0305054820303038.

[11] T. Frich. Github repository to master thesis, 2022. URL https://github.com/ToralfFrich/Master_Thesis.

[12] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017. doi: 10.1137/141000671. URL https://epubs.siam.org/doi/10.1137/141000671.

[13] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009. ISBN 978-0-262-03384-8. URL http://mitpress.mit.edu/books/introduction-algorithms.

[14] J. Matouek and B. Gärtner. *Understanding and Using Linear Programming (Universitext)*. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 3540306978. URL https://link.springer.com/book/10.1007/978-3-540-30717-4.

[15] I. Griva, S. Nash, and A. Sofer. *Linear and Nonlinear Optimization (2. ed.)*. Society for Industrial and Applied Mathematics, 2008. ISBN 978-0-89871-661-0. URL https://www.cambridge.org/no/academic/subjects/statistics-probability/optimization-or-and-risk/linear-and-nonlinear-optimization-2nd-edition?format=HB&isbn=9780898716610.

[16] J. Farrell. *Programming logic and design, comprehensive, 9th Edition*. Cengage Learning, 2017. ISBN 978-1337102070. URL http://public.ebookcentral.proquest.com/choice/publicfullrecord.aspx?p=4794147.

[17] C. H. Papadimitriou. On the complexity of integer programming. *Journal of the ACM*, 28(4):765–768, 1981. doi: 10.1145/322276.322287. URL https://dl.acm.org/doi/10.1145/322276.322287.

[18] D. S. Hochbaum. Complexity and algorithms for nonlinear optimization problems. *Annals of Operations Research*, 153:257–296, 2007. doi: 10.1007/s10479-007-0172-6. URL https://link.springer.com/article/10.1007/s10479-007-0172-6.

[19] H. P. Williams. *Model Building in Mathematical Programming, 5th Edition*. Wiley, 2013. ISBN 9781118506189. URL https://www.wiley.com/en-be/Model+Building+in+Mathematical+Programming%2C+5th+Edition-p-9781118443330.

[20] J. Bisschop. *AIMMS - Optimization Modeling*. AIMMS B.V, 2006. ISBN 1411698991. URL https://documentation.aimms.com/_downloads/AIMMS_modeling.pdf.

[21] E. M. L. Beale and J. A. Tomlin. Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables. In *Proceedings 5th IFORS Conference, Tavistock, London & Wiley, New York,*, volume 69, pages 447–454, 1970. URL https://www.researchgate.net/publication/313166553_Special_facilities_in_a_general_mathematical_programming_system_for_nonconvex_problems_using_ordered_sets_of_variables.

[22] M. M. Baldi, G Perboli, and R. Tadei. The three-dimensional knapsack problem with balancing constraints. *Applied Mathematics and Computation*, 218(19):9802–9818, 2012. doi: 10.1016/j.amc.2012.03.052. URL https://www.sciencedirect.com/science/article/pii/S0096300312002913.

[23] J. Egeblad and D. Pisinger. Heuristic approaches for the two- and three-dimensional knapsack packing problem. *Computers & Operations Research*, 36:1026–1049, 2009. doi: 10.1016/j.cor.2007.12.004. URL https://www.sciencedirect.com/science/article/pii/S030505480700264X.

[24] A. Bortfeldt and G. Wäscher. Constraints in container loading – a state-of-the-art review. *European Journal of Operational Research*, 229(1):1–20, 2013. doi: 10.1016/j.ejor.2012.12.006. URL https://www.sciencedirect.com/science/article/pii/S037722171200937X.

[25] E. F. Silva, T. A. M. Toffolo, and T. Wauters. Exact methods for three-dimensional cutting and packing: A comparative study concerning single container problems. *Computers and Operations Research*, 109:12–27, 2019. doi: 10.1016/j.cor.2019.04.020. URL https://www.sciencedirect.com/science/article/pii/S0305054819301030?via%3Dihub.

[26] C. S. Chen, S. M. Lee, and Q.S. Shen. An analytical model for the container loading problem. *European Journal of Operational Research*, 80(1):68–76, 1995. doi: 10.1016/0377-2217(94)00002-T. URL https://www.sciencedirect.com/science/article/pii/037722179400002T.

[27] L. Junqueira, R. Morabito, and D. Sato Yamashita. Three-dimensional container loading models with cargo stability and load bearing constraints. *Computers & Operations Research*, 39(1):74–85, 2012. doi: 10.1016/j.cor.2010.07.017. URL https://www.sciencedirect.com/science/article/pii/S0305054810001486?via%3Dihub.

[28] L. Junqueira and T. A. Queiroz. The static stability of support factor-based rectangular packings: an assessment by regression analysis. *International Transactions in Operational Research*, 0:1–26, 2019. doi: 10.1111/itor.12750. URL https://onlinelibrary.wiley.com/doi/10.1111/itor.12750.

[29] Y. Weng, S. Guo, W. Zhu, A. Lim, and W. Oon. The 6 key elements to sclp block building approaches. In *2010 International Conference on Educational and Information Technology*, volume 1, pages V1–402–V1–407, 2010. doi: 10.1109/ICEIT.2010.5607657. URL https://ieeexplore.ieee.org/document/5607657.

[30] M. Eley. Solving container loading problems by block arrangement. *European Journal of Operational Research*, 141(2):393–409, 2002. doi: 10.1016/S0377-2217(02)00133-9. URL https://www.sciencedirect.com/science/article/pii/S0377221702001339.

[31] M. Iori, M. Locatelli, M. Moreira, and T. Silveira. A mixed approach for pallet building problem with practical constraints. In ICEIS 2020, editor, *Enterprise Information Systems, 22nd International Conference*, pages 122–139, 05 2021. doi: 10.1007/978-3-030-75418-1_7. URL https://link.springer.com/10.1007/978-3-030-75418-1_7.

[32] G. Cavone, R. Carli, G. Troccoli, G. Tresca, and M. Dotoli. A milp approach for the multi-drop container loading problem resolution in logistics 4.0. In *2021 29th Mediterranean Conference on Control and Automation (MED)*, pages 687–692. IEEE, 2021. doi: 10.1109/MED51440.2021.9480359. URL https://ieeexplore.ieee.org/document/9480359.

[33] I. Gimenez-Palacios, M. T. A. Martínez, R. Alvarez-Valdes, and F. Parreño. Logistic constraints in container loading problems: the impact of complete ship-

ment conditions. *TOP*, 29:177–203, 07 2020. doi: 10.1007/s11750-020-00577-8. URL https://link.springer.com/10.1007/s11750-020-00577-8.

[34] M. Gajda, A Trivella, R Mansini, and D Pisinger. An optimization approach for a complex real-life container loading problem. *Omega*, 107:102559, 2022. doi: 10.1016/j.omega.2021.102559. URL https://www.sciencedirect.com/science/article/pii/S0305048321001687.

[35] I. A. Araya, K. Guerrero, and E. Nuñez. Vcs: A new heuristic function for selecting boxes in the single container loading problem. *Computers & Operations Research*, 82:27–35, 2017. doi: 10.1016/j.cor.2017.01.002. URL https://www.sciencedirect.com/science/article/pii/S0305054817300023.

[36] J. Hemminki, T. Leipala, and O. Nevalainen. On-line packing with boxes of different sizes. *International Journal of Production Research*, 36(8):2225–2245, 1998. doi: 10.1080/002075498192869. URL https://www.tandfonline.com/doi/abs/10.1080/002075498192869.

[37] L Junqueira, R Morabito, and D Sato Yamashita. Mip-based approaches for the container loading problem with multi-drop constraints. *Annals of Operations Research*, 199:51–75, 2012. doi: 10.1007/s10479-011-0942-z. URL https://link.springer.com/article/10.1007/s10479-011-0942-z.

[38] C. Paquay, M. Schyns, and S. Limbourg. A mixed integer programming formulation for the three-dimensional bin packing problem deriving from an air cargo application. *International Transactions in Operational Research*, 23:187–213, 2016. doi: 10.1111/itor.12111. URL https://onlinelibrary.wiley.com/doi/10.1111/itor.12111.

[39] M. T. Alonso, R. Alvarez-Valdes, M. Iori, F. Parreño, and J.M. Tamarit. Mathematical models for multicontainer loading problems. *Omega*, 66:106–117, 2017. doi: 10.1016/j.omega.2016.02.002. URL https://www.sciencedirect.com/science/article/pii/S0305048316000335.

[40] M.T. Alonso, R. Alvarez-Valdes, M. Iori, and F. Parreño. Mathematical models for multi container loading problems with practical constraints. *Computers & Engineering*, 127:722–733, 2019. doi: 10.1016/j.cie.2018.11.012. URL https://www.sciencedirect.com/science/article/pii/S0360835218305527?via%3Dihub.

[41] E. F. Silva, A. A. S. Leão, F.M.B. Toledo, and T. Wauters. A matheuristic framework for the three-dimensional single large object placement problem with practical constraints. *Computers & Operations Research*, 124:105058, 2020.

doi: 10.1016/j.cor.2020.105058. URL https://www.sciencedirect.com/science/article/pii/S0305054820301751.

[42] E. E. Bischoff and M. S. W. Ratcliff. Issues in the development of approaches to container loading. *Omega*, 23(4):377–390, 1995. doi: 10.1016/0305-0483(95)00015-G. URL https://www.sciencedirect.com/science/article/pii/030504839500015G?via%3Dihub.

[43] D. Pisinger. Heuristics for the container loading problem. *European Journal of Operational Research*, 141(2):382–392, 2002. doi: 10.1016/S0377-2217(02)00132-7. URL https://www.sciencedirect.com/science/article/pii/S0377221702001327?via%3Dihub.

[44] E. Boef, J. Korst, S. Martello, D. Pisinger, and D. Vigo. A note on robot-packable and orthogonal variants of the three-dimensional bin packing problem. In *Datalogisk Institut - Københavns Universitet*, 2003. URL https://di.ku.dk/forskning/Publikationer/tekniske_rapporter/tekniske-rapporter-2003/03-02.pdf.

[45] A. Moura and J. F. Oliveira. A grasp approach to the container-loading problem. *IEEE Intelligent Systems*, 20(4):50–57, 2005. doi: 10.1109/MIS.2005.57. URL https://ieeexplore.ieee.org/document/1492318.

[46] F. Parreño, R. Alvarez-Valdes, J. F. Oliveira, and J. M. Tamarit. Neighborhood structures for the container loading problem: a vns implementation. *Journal of Heuristics*, 16(1), 2010. doi: 10.1007/s10732-008-9081-3. URL https://link.springer.com/article/10.1007%2Fs10732-008-9081-3.

[47] S. Ceschia and A. Schaerf. Local search for a multi-drop multi-container loading problem. *Journal of Heuristics*, 19:275–294, 01 2010. doi: 10.1007/s10732-011-9162-6. URL https://link.springer.com/article/10.1007%2Fs10732-011-9162-6.

[48] A. G. Ramos, J. F. Oliveira, J. F. Gonçalves, and M. P. Lopes. A container loading algorithm with static mechanical equilibrium stability constraints. *Transportation Research Part B: Methodological*, 91:565–581, 2016. doi: 10.1016/j.trb.2016.06.003. URL https://www.sciencedirect.com/science/article/pii/S0191261515302022?via%3Dihub.

[49] X. Zhao, J. A. Bennell, T. Bektaş, and K. Dowsland. A comparative review of 3d container loading algorithms. *International Transactions in Operational Research*, 23(1-2):287–320, 2016. doi: 10.1111/itor.12094. URL https://onlinelibrary.wiley.com/doi/10.1111/itor.12094.

[50] A. G. Ramos, E. Silva, and J. F. Oliveira. A new load balance methodology for container loading problem in road transportation. *European Journal of Operational Research*, 266(3):1140–1152, 2018. doi: 10.1016/j.ejor.2017.10.050. URL https://www.sciencedirect.com/science/article/pii/S0377221717309633.

[51] M. S. W. Ratcliff and Eberhard E. Bischoff. Allowing for weight considerations in container loading. *Operations-Research-Spektrum*, 20:65–71, 1998. doi: 10.1007/BF01545534. URL https://link.springer.com/article/10.1007/BF01545534.

[52] M. Padberg. Packing small boxes into a big box. *Mathematical Methods of Operations Research*, 52:1–21, 09 2000. doi: 10.1007/s001860000066. URL https://link.springer.com/article/10.1007/s001860000066.

[53] S. Martello, D. Pisinger, D. Vigo, E. Boef, and J. Korst. Algorithm 864: General and robot-packable variants of the three-dimensional bin packing problem. *ACM Transactions on Mathematical Software*, 33:7, 01 2007. doi: 10.1145/1206040.1206047. URL https://dl.acm.org/doi/10.1145/1206040.1206047.

[54] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2021. URL https://www.gurobi.com. Last accessed 13-February-2022.

[55] I. Dunning, J. Huchette, and M. Lubin. Jump: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017. doi: 10.1137/15M1020575. URL https://dl.acm.org/doi/10.1137/15M1020575.

[56] Magnus Själander, Magnus Jahre, Gunnar Tufte, and Nico Reissmann. EPIC: An energy-efficient, high-performance GPGPU computing research infrastructure, 2019.

[57] LLC Gurobi Optimization. How does gurobi perform on different computer hardware?, 2021. URL https://support.gurobi.com/hc/en-us/articles/360013196532-How-does-Gurobi-perform-on-different-computer-hardware-. Last accessed 13-February-2022.

[58] Intel corporation. Intel® xeon® scalable processors, 2022. URL https://ark.intel.com/content/www/us/en/ark/products/series/125191/intel-xeon-scalable-processors.html. Last accessed 13-February-2022.

[59] Passmark Software. Single thread performance, 2022. URL https://www.cpubenchmark.net/singleThread.html. Last accessed 13-February-2022.

[60] David M. Corey, William P. Dunlap, and Michael J. Burke. Averaging correlations: Expected values and bias in combined pearson rs and fisher's z transformations. *The Journal of General Psychology*, 125(3):245–261, 1998. doi: 10.1080/00221309809595548. URL https://www.tandfonline.com/doi/abs/10.1080/00221309809595548.

[61] D. Freedman, R. Pisani, and R. Purves. Statistics (international student edition), 4th edition. *Pisani, R. Purves, 4th edn. WW Norton & Company, New York*, 2007. URL https://www.wiley.com/en-au/Statistics%2C+4th+Edition+International+Student+Edition-p-9780393930436.

[62] R. G. Miller. *Simultaneous statistical inference, 2nd Edition*. Springer-Verlag, 1981. ISBN 978-0-387-90548-8. URL https://link.springer.com/book/10.1007/978-1-4613-8122-8.

[63] W. Haynes. *Bonferroni Correction*. Springer New York, 2013. ISBN 978-1-4419-9863-7. doi: 10.1007/978-1-4419-9863-7_1213. URL https://link.springer.com/referenceworkentry/10.1007/978-1-4419-9863-7_1213.

Appendix A

Non-robot-packable pattern

```
for k in 2:length(boxes)-1
    positions = []
    for j in k:k
         for i in 1:j
            if i!=j
                push!(positions, [i, j])
            end
        end
    end
    expressions = []
    for l in 1:length(positions)
        push!(expressions, [robot_pos[l][1], robot_pos[l][2]])
    end
    @constraint(model, sum(expressions[n] for n in 1:length(expressions)) == 1)
end
```

Code listing A.1: Non-robot-packable pattern written constraints

```
on_top, right, behind = 0

if (zi + ri == zj)
    if ((xi < xj < xi + xcoveri ||
        xi < xj + xcoverj < xi + xcoveri) &&
        (yi < yj < yi + ycoveri ||
        yi < yj + ycoverj < yi + ycoveri))
        on_top[i, j] = 1
    end

elseif (xi + xcoveri == xj)
    if ((zi < zj < zi + ri ||
        zi < zj + rj < zi + ri) &&
        (yi < yj < yi + ycoveri ||
        yi < yj + ycoverj < yi + ycoveri))
        right[i, j] = 1
    end

else (yi + ycoveri == yj)
    if ((zi < zj < zi + ri ||
        zi < zj + rj < zi + ri) &&
        ((xi < xj < xi + xcoveri ||
        xi < xj + xcoverj < xi + xcoveri)
        behind[i, j] = 1
    end
end

robot_pos[i, j] = on_top[i, j] + right[i, j] + behind[i, j]
```

Code listing A.2: Written constraints for finding *robot_pos* used in non-packable pattern constraint

Toralf Christian Skaali Frich

A real-world pallet loading problem - developing a MIP formulation with practical constraints

NTNU
Norwegian University of
Science and Technology