UNIVERSITÀ DI PARMA

UNIVERSITÀ DEGLI STUDI DI PARMA

DOTTORATO DI RICERCA IN

“TECNOLOGIE DELL’INFORMAZIONE”

CICLO XXXIV

Algorithms for Cutting and Packing Problems

Coordinatore e relatore:

Chiar.mo Prof. Marco Locatelli

Correlatore:

Chiar.mo Prof. Manuel Iori

Dottorando: *Tiago Silveira*

Anni Accademici 2018/2019 – 2020/2021

To the two great loves of my life,
Sabrina and Bento.

Abstract

This thesis analyzes two fundamental problems in the “Cutting and Packing” (C&P) field: first, a specific problem called Pallet Building Problem (PBP) which has practical constraints is approached; then base variants of the so-called Rectangular Cutting Problem (RCP) are addressed.

In C&P problems, there are sets of small objects (items) that must be cut/packed from/onto large objects (containers). We address the case where containers are identical one another: in terms of PBP, the geometric shapes for both the items and containers are 3-dimensional boxes; on the other hand, in the case of RCP, the geometric shapes are 2-dimensional rectangles, somewhat simplifying the problem, but not making it easy per se.

In the PBP, the aim is to pack a given set of items into layers and then build pallets by stacking layers one on top of the other, by minimizing the number of pallets used. In the RCP, the aim is to cut a set of items from a container, while maximizing a profit function. We are considering a base set of constraints for both problems. They are as follows: (*i*) items must fit entirely within the container; (*ii*) items cannot overlap.

When addressing PBP, we take into account a larger set of constraints. In this case, some non-trivial operational constraints that originate from a real-world automated application are also included: in practice, items are grouped into families and must be packed into horizontal layers. To facilitate loading/unloading operations, items from the same family packed into the same layer should be contiguous with one another and at least one of them must be visible from the outside. In addition to these, we consider stackability and fill factor constraints.

The techniques we develop to solve the target problems are heuristic, exact, metaheuristic and matheuristic algorithms.

Due to a more complex set of constraints, we divide the PBP into two phases: (*i*) layer building; (*ii*) pallet building. In simple terms, items are first grouped into horizontal layers, and then layers are stacked one over the other to form pallets. To solve this problem, we propose heuristics and matheuristics based on heuristics and integer linear models. The main heuristic we highlight is the adaptation of the Extreme Points Heuristic (Crainic et al. [22]) to meet the new constraints. In regards to the mathematical models, we propose them for solving specific parts of the problem, since a single model for the entire problem might be unfeasible. We then propose matheuristic algorithms by taking advantage of these efficient heuristics and the mathematical models. In addition to that, a significant improvement in the solution was noticed when adapting the PBP to the GRASP metaheuristic with reactive method.

When it comes to RCP, we propose a new technique to generate both the guillotine

and first-order non-guillotine (G5 pattern) cuts. Based on the original and innovative idea of floating cuts, this model is a tree search where branching occurs by successive cuts (detailed in Chapter 8). However, unlike all known models, the exact position of the cuts is not fixed, instead it remains floating until a concrete small rectangle (also known as item) is assigned to a child node. This model does not include decision variables neither for the position coordinates of the items nor for the coordinates of the cuts. Under this framework, it is possible to address problems where: items have fixed orientations or can be rotated by 90 degrees; the value of each item is either proportional to its area or arbitrary; the demand of each item type is either equal to one or has a multiplicity greater than one. We present the entire formulation, along with algorithms that support the management of the variables' indices, required when tree search procedures are modeled as mixed-integer problems. In addition to that, we add a set of valid inequalities to the model to reduce symmetry and to strengthen the formulation.

Extensive computational experiments were performed using real life instances from an Italian company as well as benchmark instances taken from literature in order to evaluate the algorithms effectiveness. We carried out a comprehensive analysis of the results using the proposed techniques, detailing their pros and cons. In a general analysis, the results confirm the power and flexibility of the algorithms and models. However, even more importantly, this is a new way of looking at these problems which could serve as a catalyst to even better approaches, with the consequent economic and environmental benefits.

Table of Contents

List of Figures

List of Tables

Acronyms

2D 2-dimensional.

2DCSP 2-dimensional Cutting Stock Problem.

2DSKP 2-dimensional Single Knapsack Problem.

2DSLOPP 2-dimensional Single Large Object Placement Problem.

3D 3-dimensional.

3DSLOPP 3-dimensional Single Large Object Placement Problem.

BPP Bin Packing Problem.

C&P Cutting and Packing.

CLP Container Loading Problem.

EPFULL MMEPMH Full.

EPH Extreme Points Heuristic.

EPMH Extreme Points Modified Heuristic.

FC2 Floating Cuts Model for k-staged Guillotine Cutting and Packing Problems.

FC4 Floating Cuts Model for Non-Staged Guillotine Cutting and Packing Problems.

FC5 Floating Cuts Model for First-Order Non-Guillotine Cutting and Packing Problems.

FIS First Improvement Strategy.

GRASP Greedy Randomized Adaptive Search Procedure.

GREP Reactive GRASP with Extreme Points Modified Heuristic.

GREPFULL MMGREP Full.

ILP Integer Linear Program.

MEPMH Mixed Extreme Points Modified Heuristic.

MILP Mixed-Integer Linear Programming.

MMEPMH EPMH with Mathematical Model.

MMGREP GREP with Mathematical Model.

NP Non-Deterministic Polynomial Time.

PBP Pallet Building Problem.

PLP Pallet Loading Problem.

RCP Rectangular Cutting Problem.

SLH Skylines Heuristic.

SPP Strip Packing Problem.

SSSCSP Single Stock-Size Cutting Stock Problem.

VRP Vehicle Routing Problem.

Acknowledgments

First and foremost, thank God for allowing me to experience this fantastic stage of my life.

I would like to thank each and every person who helped me during this work, in both professional and personal growth. This has been, quite literally, the most intense period of my life so far.

I would also like to thank Sabrina, my wife and my love, for all the support and understanding during this period, especially while I may have been right next to her physically, there were many times that my mind and thoughts were on my work. And, of course, thank to Bento, our "most original creation" and biggest blessing during this period.

Furthermore, thanks to my father and mother, Mauro and Cirene, for encouraging me to take on and supporting me in overcoming this challenge. My brothers Gustavo and Gabriel, I thank them for advising and helping me live in a faraway place with a completely different lifestyle.

I would like to thank the Catholic Church on behalf of Don Umberto Cocconi. This man who represents the image of God on Earth helped me not only in this work, but also in my own personal and spiritual growth, learning solidarity and generosity. Also, I thank my friends Abdullah, Márcio Rosário, and Érika dos Santos for helping me during the moments of difficulty.

My deepest gratitude also go to my dear professors. My advisor, Marco Locatelli (Don Marco), who worked tirelessly to help me complete this demanding task. My co-advisor, Manuel Iori, who not only provided opportunities and motivation for this research, but opened my mind while respecting my freedom of choice. Mayron Moreira, for his support prior to undertaking and during this project. Fabio Furini, for introducing me to the theory of the linear programming. Additionally, I would like to take the opportunity to express a special gratitude to those who taught me about so many cutting and packing problems and most importantly how to succeed when working in a team: José Fernando, Maria Antónia and Elsa Silva. Just as important, the technical and theoretical acknowledgements, here I would like to thank the reviewers of this thesis for their valuable corrections and advice, which helped me improve this research and contributed decisively to my professional growth as a researcher.

I also want to thank my friends involved in this journey. In Italy, they were of great support and helped make this period smoother: Gabriele Calzavara, Fabrizio, Serena Gambuto, Silvia Carusi, Enea, Louka Musa, Salvatore Diletto, Medhi, Bassem, Ihssan, Samba, Sabrina, Monia, and Murad, all of whom from Parma; and all DISMI research members from Reggio Emilia. A special thanks to those who were willing to interrupt their day do day lives to support my family during this time: Maria Teresa D'Amico, Enrico Giuranno, and Annagiulia Farella. For those in Brazil, who still provided strength regardless of the distance: Anderson Albuquerque and Humberto Brandão, thank you so much.

A special note of gratitude to Paul Campos, my friend and English teacher in this journey. I also thank my English teacher Marco Fernandez for his valuable contribution in writing this thesis.

Last but not least, I thank the University of Parma for this wonderful time, and the Federal University of Alfenas for their support and the opportunity.

“Christo nihil praeponere”. St. Cyprian of Carthage

Chapter 1

Introduction

"Verba volant, scripta manent"
"Paper is forbearing"

– Latin proverb

This work investigates two problems from the Cutting and Packing field named *Pallet Building Problem* (PBP) or *Distributor's Pallet Loading*, and *Rectangular Cutting Problem* (RCP).

Essentially, when addressing the PBP, we aim to load a predetermined set of small items (also called cargo) onto one or more large pallets (objects called containers) by meeting general and specific constraints to minimize the number of pallets used.

From a practical point of view, typical items include, e.g., food packaging, heavy boxes in rectangular containers for shipment of goods, soft drink bottles, and cans. Figure 1.1 shows a real-world solution for a regular PBP. Having said that, the loading of items and, therefore, the problem of the creation of pallets is fundamental to the logistics field, as it impacts not only the company's costs but also the customer's final price (e.g., with the impact of freight logistics).

The specific PBP that we address originates from a real-world automated application and is thus subject to some non-trivial operational constraints. Items should be packed into layers, that must then be piled one over the other while adhering to

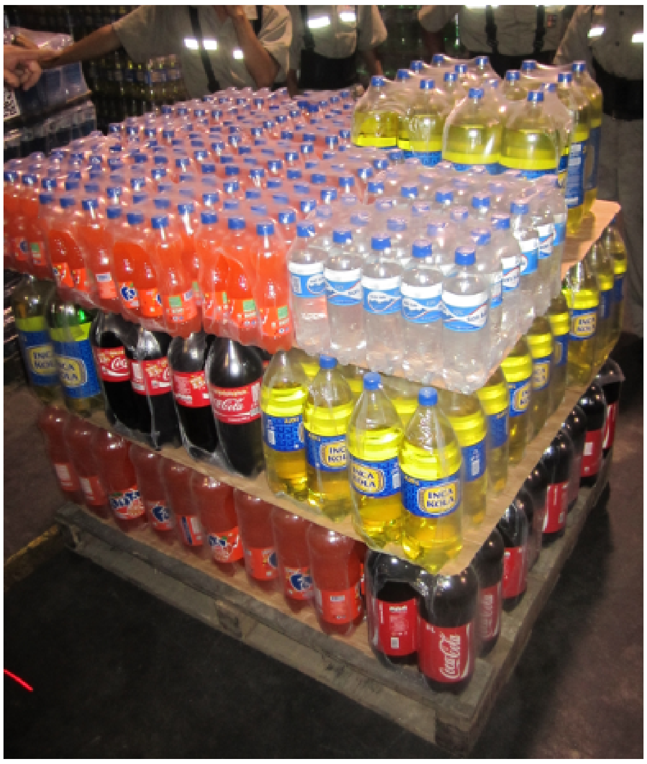


Figure 1.1: Real-world solution for the PBP.

stackability rules. Moreover, items are grouped into families, and, to facilitate loading/unloading operations, items from the same family packed into the same layer should be contiguous to one another.

While considering RCP, at first this may seem both simple and narrow problem, but in reality there are various challenging variants that arise in practice, driven by different operational, physical, and technological constraints. When it comes to practical applications, the RCP emerges in different industrial applications, such as wood, metal, glass, steel, fabric, and paper (e.g. Parreño and Alvarez-Valdes [81], Malaguti et al. [71], Polyakovskiy and M'Hallah [84], Libralesso and Fontan [66]). Figure 1.2 represents a real-world cutting process in a textile industry based on the solution (drawn lines) for the RCP.

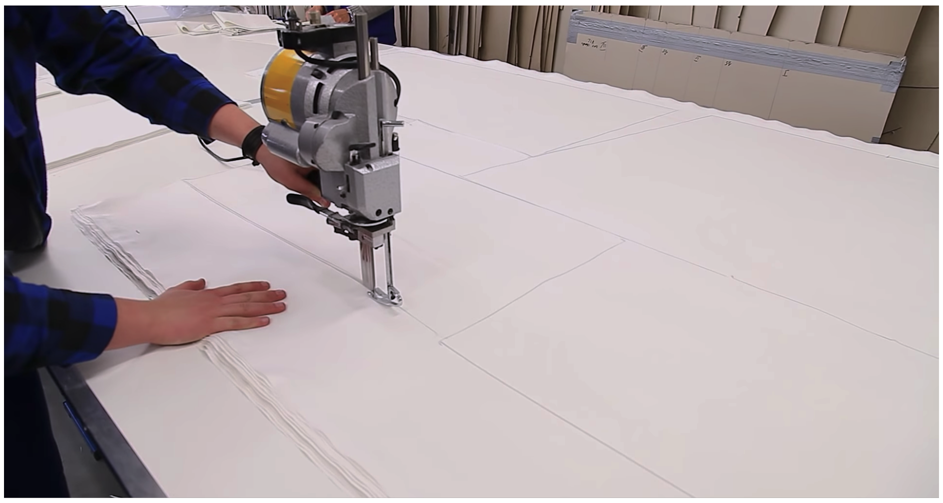


Figure 1.2: Cutting process of a solution for the RCP (image from Rexel [91]).

According to the constraints considered in the problem, the RCP is classified in a specific variant of the problem (see Section 2.3). All variants of the problem have been extensively studied, and different mixed-integer programming models have been proposed, taking advantage of the specificities that differentiate them. However, there is still need for a model that is both generally applicable and flexible, which would allow these variants to be easily addressed through small changes in the model. To this end, our approach covers these features while considering variants that include basic packing constraints, i.e., overlap constraints for items placed inside a container.

Theoretically speaking, both RCP and PBP are NP-hard because they are generalizations of the Bin Packing Problem (BPP), which is known to be NP-hard (Delorme et al. [30]). Considering algorithmic strategies to solve related problems, the literature presents efficient exact algorithms to very specific cases. When addressing generic cases, the exact approaches are less used, and heuristic solutions are widely available. Given this peculiarity, we propose some strategies to solve the discussed problems. We model and implement a new exact algorithm for the RCP and two exact techniques for the PBP. In addition, we propose heuristic solution algorithms for the PBP, since it is frequently encountered at the operational level during a robotized packing system's everyday working activity. Last, but not least, we create a new algorithm that combines the previous proposed strategies (exact and heuristic algorithms), which is also referred as matheuristic algorithm. The strategies and details of the algorithms are described later.

1.1 Cutting and Packing problems

The problems we face lie in the field of *Cutting and Packing* (C&P), one of the most studied fields of Operations Research and involve several interesting practical and theoretical problems (Bortfeldt and Wäscher [13], Crainic et al. [23], Iori et al. [57], Malaguti et al. [71], Parreño and Alvarez-Valdes [81], Polyakovskiy and M'Hallah [84], Scheithauer [96]). Due to their seemingly simple concept and vast applicability, C&P problems have attracted researchers' attention.

Although conceptually different, the C&P problems have a similar structure: to put it simply, in cutting problems, a set of stock units has to be cut to produce smaller items, while in packing problems, a set of items has to be packed into one or more containers. In general, the objective in these problems is maximizing the utilization of the container area in which items will be placed, thereby reducing the waste of raw materials. Some practical applications involve the production of materials that come in panels (such as wood or glass), the optimization of layouts (as in industry or newspaper paging), and the loading and subsequent transportation of items in containers, just to name a few.

C&P solution techniques are prevalent in companies that mass produce due to the downside of economies of scale, since they amplify possible "bad usage" in direct proportion to their production. Therefore, improvements in performance as well as the reduction of time/material and space waste are directly related to the use of efficient methods.

In regards to the creation of a solution to a C&P problem, in the past the most part of C&P tasks have been performed by skilled experts, but this has been changing over the years thanks to automated-packing systems. Currently, the literature presents two distinct scopes related to the algorithms used for this purpose: given an instance of the problem, companies may demand an immediate solution on the spot (online solution) or in an agreed predetermined period of time (offline solution). Although the reduction of the production costs stimulates the creation of algorithmic strategies for generating more efficient solutions, the development of new methods and algorithms to obtain online solutions is a complex task, since in many cases it is hard to improve the trade-off "cost-benefit" to obtain a better instant solution (i.e., solution created in a very short time limit) when considering a production chain. On the other hand, in those cases where considering an online solution is not necessary, it is easier to balance the trade-off "cost-benefit" to meet companies' business requirements, although the complexity of this type of problem remains the same. In this case, an algorithmic strategy could improve quality if the required constraints also keep the problem scalable, i.e., the time to solve the problem remains acceptable to the company after new constraints are added. With that in mind, we are herein proposing offline algorithms to the problems addressed, although some heuristics are so efficient that they could be used as an online version.

1.1.1 Structures

All C&P problems consider an item as a basic element. When we address a generic 2D problem, an item is represented by a simple polygon, i.e., a closed surface limited by straight segments connected to at most two other segments. Given such representation, it may be difficult to express restrictions on items, like, e.g., the non-overlap constraints. However, this representation can be simplified if all items are of a similar basic shape in nature (e.g. circles, triangles or rectangles).

When considering only rectangular shapes for both items and containers, as well as a packing in which the items are placed with the edges aligned parallel to the floor and the walls of the container, we have an *orthogonal* version of the problem. In this case, we could ignore the polygon structure of the edges to meet the overlap constraints, as it is enough to take into account the polygon vertices to check for overlaps. In fact, this representation can be simplified even further, with only one reference point and polygon dimensions being enough. For more details about structures, representation and algorithm complexities for polygons, we refer to Preparata and Shamos [86].

Another important characteristic when solving a C&P problem is the heterogeneity of the items. The literature classifies it into the following:

- *Homogeneous*: when the problem is formed by only one type of item;
- *Heterogeneous*: when the problem is formed by more than one type of item. In this case, we can classify it in two other ways:
  - Weakly heterogeneous: few types of items, but many items for each type;
  - Strongly heterogeneous: many types of items, but few items for each type.

When considering the PBP addressed in this work, we classify it as strongly heterogeneous. Regarding RCP, once a small quantity of items is used, we classify it as weakly heterogeneous.

1.1.2 Constraints

In terms of the constraints, we describe the most common ones found in the C&P literature, dividing them into groups as suggested by Bortfeldt and Wäscher [13].

- Container: constraints related to the container. Commonly referred to as *"Weight Limit"*, a maximum weight that can be loaded into the container, and *"Weight Distribution"*, the unyielding need to have the cargo spread as evenly as possible over the container floor for structural and safety reasons.
- Item: constraints related to an individual item. Commonly referred to as *"Loading Priorities"*, these are the scenarios where it is known before hand there is insufficient space for all items, therefore these must be loaded in accordance with their priority; *"Orientation"*, cases in which items can be rotated according to a set of allowed rotations (generic case), or vertically/horizontally (orthogonal case); *"Stacking"* any cases where it is clearly indicated that an item can be placed on top of others. When considering the demand d for an item i, the problem is either *"Unbounded"* ($d_i = \infty$) or *"Bounded"* ($d_i \neq \infty$).

As for the profit value v of each item i, the problem is called *"Unweighted"* when v_i is equal or proportional to the area of the item, or *"Weighted"* when v_i is a predefined value.

- Cargo: constraints related to the entire set or to a subset of items. In this case, we have two of them: *"Complete Shipment"*, which states that if at least one item of a type is loaded, all other items of the same type must be included in this shipment (Hit-or-Miss), additionally indicating whether or not the items are placed in the same container; and *"Allocation"*, which requires that a set or class of items must be placed together (connected/separated).
- Positioning: constraints related to the position of items inside the container. It is classified in *"Absolute"*, which forces specific items to (or not to) be placed in a certain position/area of the container, *"Relative"*, which forces certain items to be placed together in the container, and *"Multi-drop"*, which is a combination of Absolute and Relative positioning, i.e., the set of items should be placed together and the arrangement of the items should be the order of unloading.
- Load: constraints related to the result of the packing process. We relate *"Stability"* to the displacement of the items, being *"Vertical"* used to prevent the items from falling off the original stack (related to the force of gravity), and *"Horizontal"* used to prevent the items from shifting while the container is being transported (related to the force of inertia); and *"Cutting Pattern"* to the arrangement/layout intricacy, allowing complex placements to increase the container occupation, subdividing it into *"Guillotineable Pattern"*, in which the loading pattern can be obtained by parallel orthogonal "cuts", i.e., the container must be divided by continuous perpendicular cuts between their parallel edges, and *"Non Guillotineable Pattern"* or *"Robot-packable Pattern"*, where items can be loaded anywhere in a container by a "robot", i.e., items can be placed in designated positions that do not necessarily follow a guillotineable pattern. Regarding Guillotineable Pattern, it is possible to consider the number of stages of cuts. In a *"k-staged"* problem, there is a limit to cut in at most k stage sequences so that each stage represents a set of parallel guillotine cuts

performed on the "product" obtained in the previous stage (similarly, the cuts in each stage should be orthogonal to the cuts in the previous stage); on the other hand, if there is no limit to the number of stages, the problem is called *"Non-staged"*.

The problems we face include many of the aforementioned constraints. Specifically, when we have to take into account PBP, we have vertical and horizontal orientation, stacking by fragile items, complete shipment in different containers, allocation by maintaining contact of a specific set, relative positioning, and vertical stability of the load. When it comes to RCP, this set of constraints is simplified by making use of orientation and overlap constraints, although the difficulty of the problem remains.

In spite of the previous constraints being the most common in literature, we introduce new constraints not yet formally included in practical or theoretical work. In particular, we highlight the *visibility* and *contiguity* among items that belong to the same family. The concept of contiguity has already been addressed in the C&P literature (see, e.g., Scheithauer and Sommerwei*ß* [97] and Terno et al. [103]), although we could not find a univocal and formal description, more than likely because it is included in heuristic algorithms. Both constraints are the most important ones addressed in this work, and will be detailed in the following sections.

1.2 Contributions

PBP and RCP are hard practical problems related to the operational level in the day to day activities of automated-packing systems. For this reason, in order to effectively deal with these problems swiftly and efficiently, we found it necessary to adopt heuristic and exact algorithms, in addition to combining them to increase the quality of the solution. We derived part of the proposed algorithms from the most recent successful C&P studies, by embedding in them a tailored approach to the operational constraints of the problem at hand.

In summary, the main contributions of this work can be sketched as follows:

(i) the PBP that derived from a real-world industrial application is presented, in addition to the RCP in its theoretical version;

(ii) the concept of contiguity of items, that is so useful in practice during loading/unloading operations and yet it has never been formally addressed in literature, is discussed in detail;

(iii) a formal mathematical formulation of the PBP with contiguity and visibility constraints is presented, to the best of our knowledge, for the first time in the literature;

(iv) a formal mathematical formulation for the RCP for guillotine and 1st-order non-guillotine cuts, which does not include decision variables for neither the items position coordinates nor for cut coordinates, based on the original and innovative idea of floating cuts.

(v) new algorithms, made up by combining heuristics, metaheuristic, and mathematical models, are proposed to solve the problems discussed herein;

(vi) extensive computational tests on instances derived from a real-world case study and literature are given.

1.3 Outline of the work

This chapter introduces the main concepts of the problems addressed in this work, in addition to the contributions for the literature. Chapter 2 provides a formal description of the PBP and the RCP. We are focusing on describing the problems formally and, when it comes to the PBP, the main constraints named “visibility” and “contiguity”. Chapter 3 presents the main contributions we found in literature to solve the problems faced. We describe the papers related to the problems, their variations, similarities, constraints addressed, and algorithmic strategies in a concise way. Chapter 4 presents our first paper in which we describe the PBP with visibility and contiguity constraints and present the first heuristic algorithm to solve it. Chapter 5 presents a new algorithmic solution for the PBP. In this paper we solve the problem by means of a GRASP metaheuristic and a reactive method for finding more optimized solutions. Chapter 6 presents an extension of the work in Chapter 4. We mix the initial heuristic with a mathematical method to improve the packing process. We are not proposing a complete exact algorithm, but rather analyzing the impact of tailoring an exact part to the heuristic algorithm. Chapter 7 presents the main idea proposed in this thesis in a full formal description as well as heuristic and exact algorithms. Specifically, this chapter provides mathematical models for the layer creation, pallet creation, and the different solution algorithms that have been developed. Chapter 8 presents the new mixed-integer linear programming proposed to solve variants of the RCP based on floating cuts to generate the guillotine and 1st-order non-guillotine cuts. Conclusions and various future research directions are given in Chapter 9.

Chapter 2

Description of the Cutting and Packing Problems

2.1 Introduction

The following chapter presents the basic definitions for the problems addressed in this work. We formalize the Pallet Building Problem and the Rectangular Cutting Problem, highlighting the main constraints included therein. In addition to that, several basic structures are also detailed.

2.2 Pallet Building Problem with practical constraints

We are given a set R of identical pallets. Each pallet has a two-dimensional loading surface of width $W \in \mathbb{Z}_+^*$ and length $L \in \mathbb{Z}_+^*$, which can be used to load items up to a maximum height $H \in \mathbb{Z}_+^*$. We are also given a set $I = \{1, 2, \ldots, n\}$ of 3D rectangular item types, where each item type $i \in I$ contains b^i identical items, each having width $w_i \in \mathbb{Z}_+^*$, length $l_i \in \mathbb{Z}_+^*$, and height $h_i \in \mathbb{Z}_+^*$, such that $w_i \leq W$, $l_i \leq L$, and $h_i \leq H$. When loading the items, 90-degree rotations are allowed.

2.2.1 Item families

In addition to item type, the concept of family is also essential to describe the PBP. Type only refers to a specific characteristic, while family is a more general concept, covering at least one characteristic, e.g., geometric dimensions, material type, purpose of use, and so on. More in detail, item types are partitioned into a set F of families as follows. Each item type i belongs to a given family $f \in F$, which, in the real instances we have addressed, is defined as a set of item types having similar height and weight. Note, however, that in other applications, families could also be defined differently, e.g., each family could be made up of products of the same customer.

2.2.2 Layers classification

Items can be used to form layers. Each layer is a 2D packing of items whose total width does not exceed W, and whose total length does not exceed L. When building a 2D packing, we suppose a layer occupies the positive quadrant of the Cartesian system, with width parallel to the x-axis, length parallel to the y-axis, and bottom-left corner located in the origin of the axes, i.e., we refer to the point (0,0) as the reference point of a layer. Figure 2.1 describes this notation. When packing an item j in a layer, we refer to the reference point of j as its bottom-left vertex (x_{1j}, y_{1j}). The three other item vertices can be directly determined as (x_{2j}, y_{1j}), (x_{1j}, y_{2j}), (x_{2j}, y_{2j}), where $x_{2j} = x_{1j} + w_j$ and $y_{2j} = y_{1j} + l_j$. If item j is rotated, then width and length of the item must be switched. Regarding the composition of the layers, we consider three types of layers:

- *single-item layers* are formed by a unique type of items;
- *single-family layers* are formed by different item types, but all belonging to the same family;
- *residual layers* are formed by a combination of items of different families or by items belonging to the same family in case these do not cover a minimum predefined area of the loading surface (see the fill factor constraint described

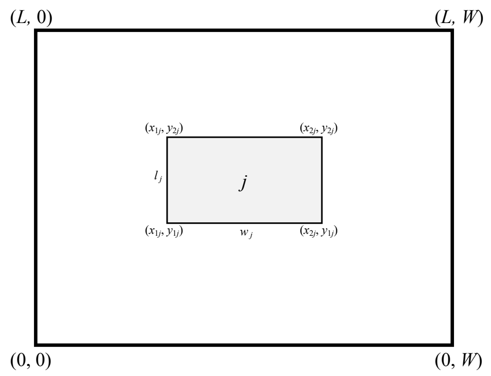


Figure 2.1: Graphic representation of a layer with width W and length L, and the representation of the vertices related to item j.

below). A pallet can contain at most one residual layer, which has to be placed on top of all other layers.

2.2.3 Constraints

Similar to all packing problems, items cannot overlap. To this aim, we apply the *Separating Axis theorem* (see, e.g., Gottschalk [47]) to find a feasible position for each item inside a layer. This theorem is applied to general shapes and states that two convex polytopes are disjoint if and only if there is an orthogonal axis, the separating axis, which separates either faces or edges of the two polytopes. Applying this theorem to our purpose, two rectangles do not overlap if and only if we can draw a straight horizontal line or straight vertical line or both between them. Figure 2.2 shows the cases addressed by this theorem.

2.2.3.1 Contiguity and Visibility constraints

Let us call a *group* a set of items having the same item type and being loaded in the same layer. Besides the overlap constraint, packings of items in a layer should fulfill two operational constraints that concern groups and are aimed at easing loading/unloading operations when the pallet is composed/delivered. These constraints

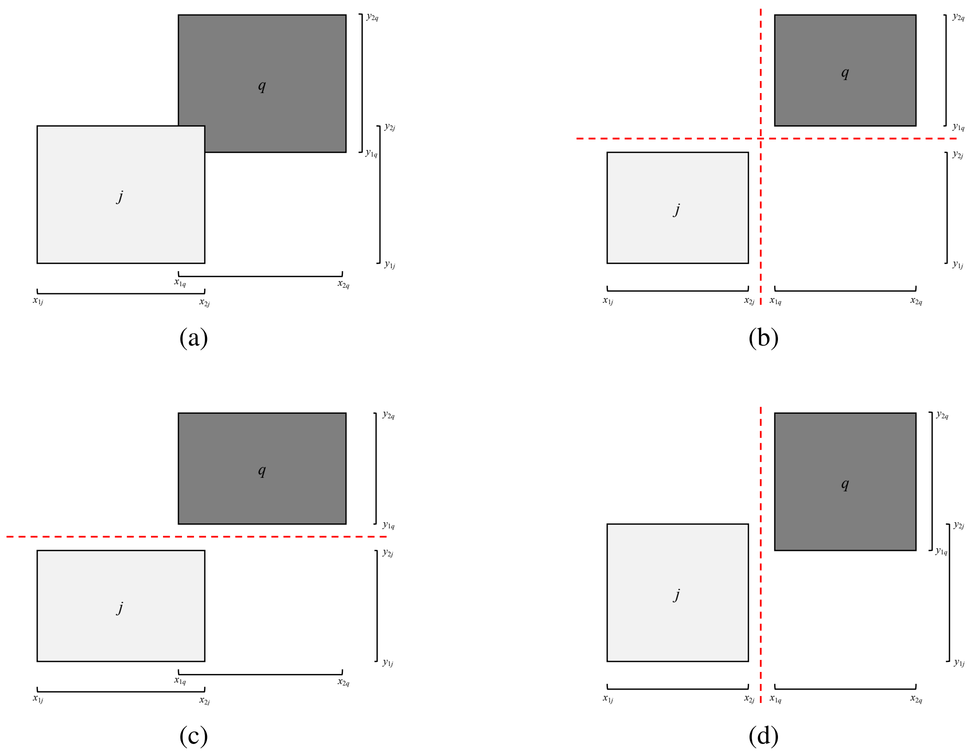


Figure 2.2: Item positions and separating axes: (2.2a) items overlap and no separating axis exists; no overlap between items includes the subcases: (2.2b), two separating axes; (2.2c) a horizontal separating axis; and (2.2d) a vertical separating axis.

are named *contiguity* and *visibility* constraints, and are defined as follows:

- *contiguity*: given two items of the same group, we introduce a maximum distance that can separate the two items in order to consider them as contiguous. Let $G \subseteq I$ be a generic subset of item types packed into a layer, $|G| \geq 3$ with at least two items of the same group, we define parameter ξ as

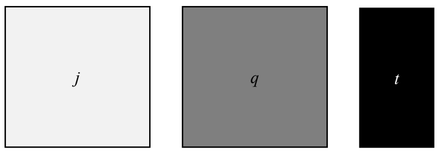
  $$0 \leq \xi \leq \min_{i \in G}(w_i, l_i),$$

  i.e., ξ lies in the interval between 0 and the smallest edge length among all items in G. This way, two items j, q selected from this layer are considered as contiguous if the smallest distance between the edges of j and the edges of q is strictly lower than ξ. In other words, we guarantee that no other item can fit between j and q. The contiguity relation can be modeled by the overlapping relation. Indeed, we can view contiguity between two items as an overlap between the two items if their sizes are augmented by $\frac{\xi}{2}$ in each direction. Figure 2.3 illustrates this fact, considering the items j and q when the parameter ξ is fixed to the value $\min_{i \in G}(w_i, l_i)$.

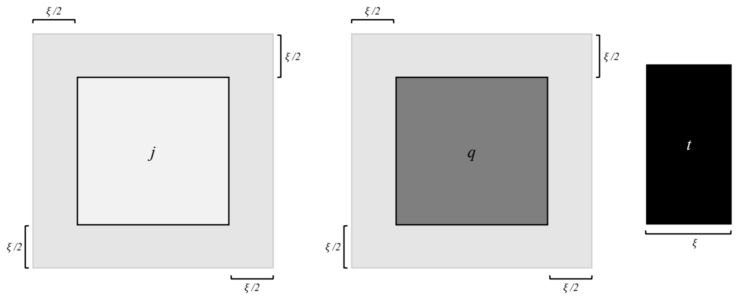
The notion of contiguity between items is employed to define the contiguity constraint for items of a given type i over a generic layer. Such constraint is met if, for each type $i \in G$, any item of type i is contiguous to at least another item of the same type (of course, provided that the number of items of that type is larger than one) and, moreover, there are no separated sub-groups of items of type i (i.e., subgroups whose distance is ξ or more). Stated in another way, the graph whose nodes are the items of type i and whose edges are defined by the contiguity relation, must be connected. Addressing a practical scenario, this constraint is very useful when a high number of different products (items) have to be placed in a container (layer). Indeed, contiguous item groups require less effort and time for load/unload operations.

- *visibility*: similarly to what we stated for the contiguity, we say that a group is visible from outside if, for at least one item in that group, the distance between its edges and one of the borders of the layer is strictly lower than ξ. In the same line, this constraint is very useful in a practical scenario to reduce the unload time because it allows to see from the outside which items compose the layer in a pallet. In general, this constraint is more commonly applied in small and medium companies due to their greater level of non robotized work in packaging logistics.

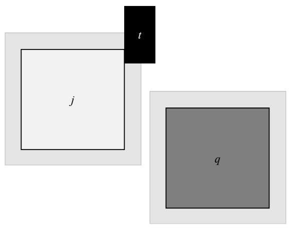
In Figure 2.4a, we show an example where the contiguity constraint is not met. The fact that items of the same type are spread around within the layer slows down both the loading and the unloading operations. In Figure 2.4b, we show an example where



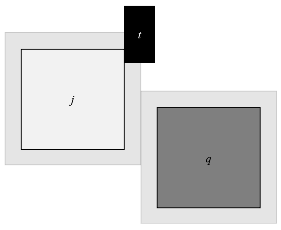
(a) Items set $G = \{j, q, t\}$



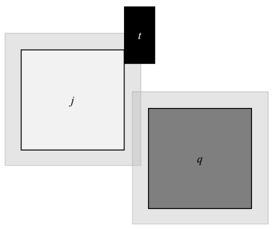
(b) Items j and q enlarged by $\xi/2$, considering the edge length ξ of item t



(c) j and q are not contiguous



(d) j and q are not contiguous



(e) j and q are contiguous

Figure 2.3: Contiguity relation based on the overlapping relation: given the items set G (2.3a), both similar items j and q are enlarged by $\xi/2$ in each direction (2.3b) according to the smallest edge length ξ: j and q will not be contiguous if their enlarged regions are either separated (2.3c) or, at most, share a portion of an edge (2.3d). Instead, they are contiguous if the enlarged regions overlap (2.3e), as it is impossible to insert another item between j and q.

the visibility constraint is not met. In this case, we notice that the set of items lying

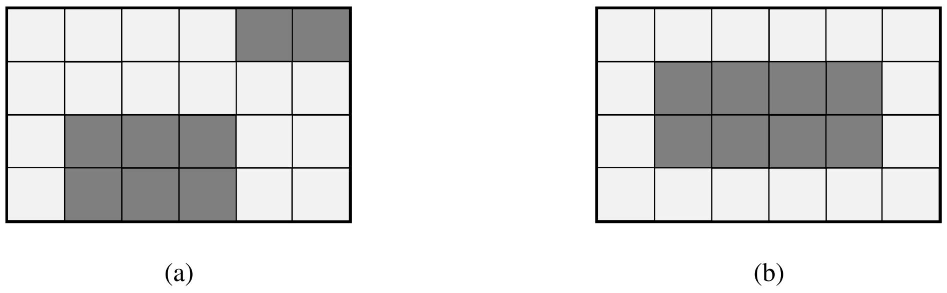


Figure 2.4: Examples of layouts that breach the constraints of either the contiguity or visibility when considering two types of items (light gray and dark gray) of similar dimensions: (2.4a) two groups of dark gray items separated by a distance equal to ξ; (2.4b) a single group of dark gray items placed at ξ distance from the layer border.

completely in the interior of the layer can only be loaded before and unloaded after the other items, thus forcing a precedence in the order with which the items are loaded and unloaded.

2.2.3.2 Stackability and Fill Factor constraints

Considering layers in a pallet, the height of each layer corresponds to the height of the highest item in the layer (recall that in single-family layers the heights of the items are quite similar). The height cannot exceed H. Moreover, we impose two additional conditions:

- a *stackability constraint* imposes that resistant items cannot be on top of fragile ones. For a layer f with a set of items φ, its stackability ρ_f is set equal to the largest stackability value among all items in φ. Layer f cannot be put below layer g if $\rho_f < \rho_g$ (see Figure 2.5).
- a layer can be used to support other layers only if its total area loaded with items reaches a minimum fraction α of the total loading surface WL. Parameter $0 < \alpha \leq 1$ is called *fill-factor*. A layer with a loaded area lower than αWL can still be used to build a pallet but can only be the topmost layer. We call this

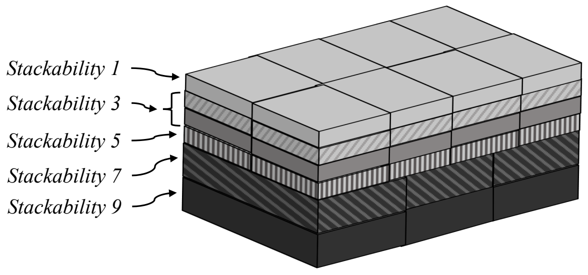


Figure 2.5: Graphical illustration of the stackability constraint.

the *fill factor constraint*. The choice of the parameter α will be commented in Chapter 7.

We note that the stackability constraint above introduces a simplification of the real weight that an item has to bear in a load. The simplification is widely adopted in the literature as it works well in practice, see, e.g., Bortfeldt and Wäscher [13]. For a more elaborate formulation of load-bearing constraints, we refer, e.g., to de Queiroz and Miyazawa [28, 29].

About layers classification, we remark that both single-item and single-family layers are required to meet a so called fill factor constraint, i.e., items on these layers should cover a minimum occupation area (see below for a formal description). Moreover, as previously mentioned, items belonging to the same family are required to have similar heights. Both requirements aim at guaranteeing that single-item and single-family layers can be used in a 3D packing to support other layers that are packed on top of them. Note that these requirements are not imposed for residual layers.

2.2.4 Problem outline

The PBP that we face aims to load all items into the minimum number of pallets by ensuring that the following constraints are satisfied:

- all items are packed in layers by satisfying contiguity and visibility constraints;

- at most one residual layer can be used per pallet, and, in such a case, it is placed at the top of the pallet;
- single-item layers can be used to support any type of layers on top of them, as well as single-family layers can be used to support single-family and residual layers, as long as stackability and fill-factor constraints are satisfied;
- the total height of the layers in any pallet does not exceed H.

Figure 2.6 shows a toy instance of the described PBP with a single pallet. Figure 2.6a represents family 1 with six item types. Figure 2.6b represents family 2 with one item type. Figure 2.6c represents family 3 with two item types. Figure 2.6d represents the pallet. First, layers are created as shown in Figure 2.6e with, from top to bottom, a residual, four single-family and two single-item layers, respectively; next, the layers are packed inside the pallet for building the final solution as displayed in Figure 2.6f.

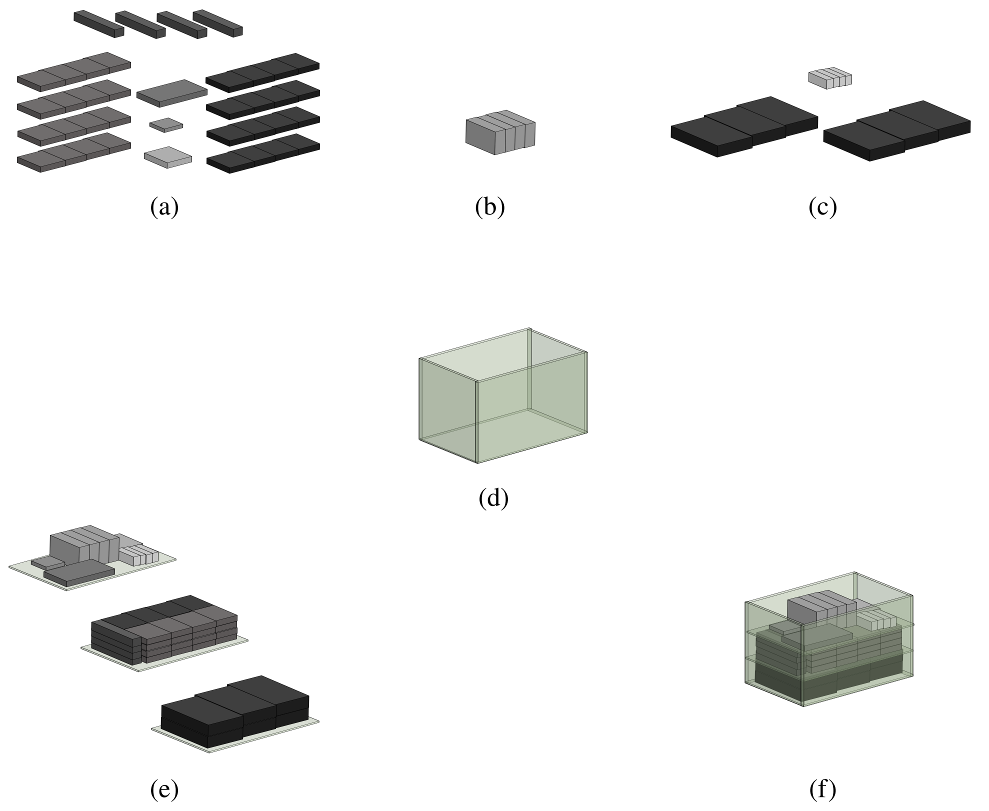


Figure 2.6: PBP toy instance.

2.3 Rectangular Cutting Problem

For this problem, there is one 2D rectangular plate with length $L \in \mathbb{Z}_+^*$ and width $W \in \mathbb{Z}_+^*$, and the goal is to cut from it a set of n 2D items indexed by $i, i = 1, ..., n$. Each item i has a length $l_i \in \mathbb{Z}_+^*$, a width $w_i \in \mathbb{Z}_+^*$, a profit value $v_i \in \mathbb{Z}_+^*$, and a demand $d_i \in \mathbb{Z}_+^*$, such that $l_i \leq L$, and $w_i \leq W$. The objective is to maximize the total profit from items assigned to the rectangular plate while fulfilling the constraints of the problem.

2.3.1 Classes addressed

When considering demand of the items, we deal with two RCP classes, according to the typology of Wäscher et al. [109]: dealing with $d_i = 1$ (the unitary demand) for each item type i, this problem is classified as the 2-dimensional Single Knapsack Problem (2DSKP); when it comes to a variable demand d_i for a type i, the problem is called the 2-dimensional Single Large Object Placement Problem (2DSLOPP). In both cases, the goal remains to cut a set of small rectangular items from a single large rectangular object (e.g., a bin, panel, or stock sheet) while maximizing the total profit associated with the items.

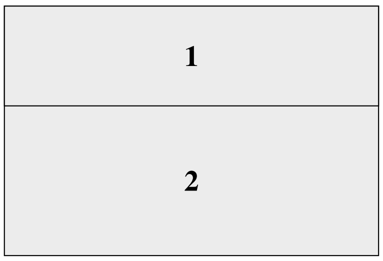
2.3.1.1 Variants

Depending on the industrial application, technological constraints must be considered within the cutting process. In this regard, different variants of the RCP arise. Therefore, from this perspective, we take into account the following constraints.

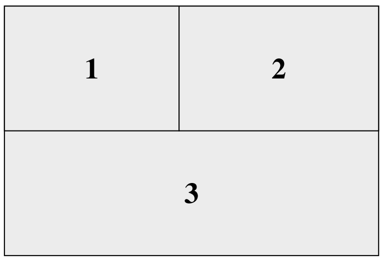
Firstly, regarding the items profit, we address the *Unweighted* and *Weighted* cases for both variants.

Secondly, an additional constraint regarding guillotine cuts is the number of stages. Thus, *k-staged* and *non-staged* versions of these problems are also considered.

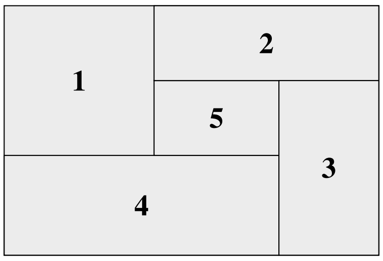
Finally, the main constraints we account for in this problem are those related to the cutting patterns. We are taking into consideration both "Guillotine" and "Non-Guillotine" cuts (the main concept of cutting patterns is explained in Chapter 1). Since a non-guillotine cut includes more cutting options, it tends to be more densely packed. Furthermore, when dealing with non-guillotine cuts, the literature presents some distinct types of cuts (see, e.g., the L-shaped cut from Lins et al. [67], and the higher order non-guillotine cut defined by Martins and Dell [76]), however we will only be looking at the *1st-order non-guillotine cut* in our work. In simple terms, this cut type produces five new rectangles arranged in such a way that it does not form a guillotine cutting pattern (Arenales and Morabito [8]). This pattern corresponds to a G5 recursive structure. Figure 2.7 shows a graphical example for the cutting patterns described. Note that the guillotine example is represented by images (2.7a) and (2.7b)



(a) 1-staged guillotine cut type



(b) 2-staged guillotine cut type



(c) First-order non-guillotine cut type

Figure 2.7: Types of orthogonal cuts.

in Figure 2.7, differing only by the number of stages, being 1-staged and 2-staged, respectively.

2.3.2 Problem outline

The RCP that we address aims at loading as many items as possible into a single rectangular plate resulting in a maximum profit by ensuring that the following constraints are satisfied:

- items in the plate cannot overlap with each other, and the cut pattern applied in the plate includes both guillotine and non-guillotine types;

- the problem should include the following problem classes: 2DSKP and 2DSLOPP;
- the problem classes handled should fulfill a combination of three specific constraints: number of stages, item rotation and type of profit. Specifically, we are interested in solving the following variants:

  (i) non-staged, no rotation, and unweighted;

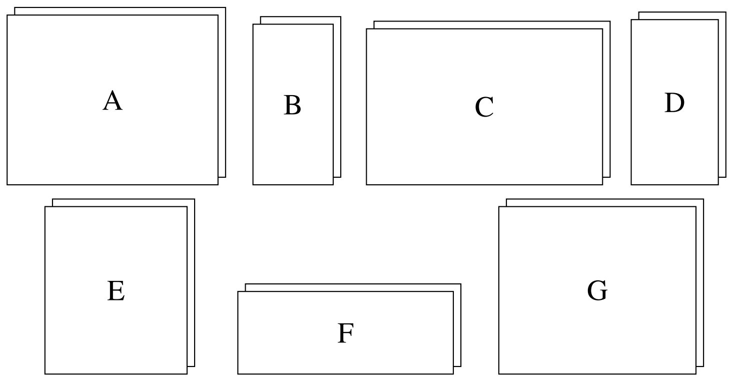
  (ii) non-staged, 90 degrees rotation allowed, and unweighted;

  (iii) non-staged, no rotation, and weighted;

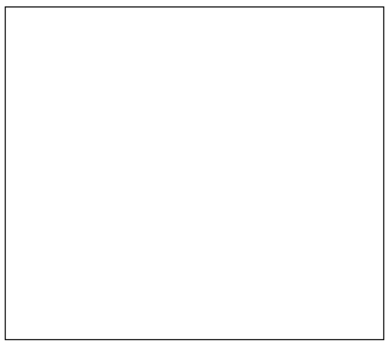
  (iv) non-staged, 90 degrees rotation allowed, and weighted;

  (v) 2-stages, no rotation, and unweighted;

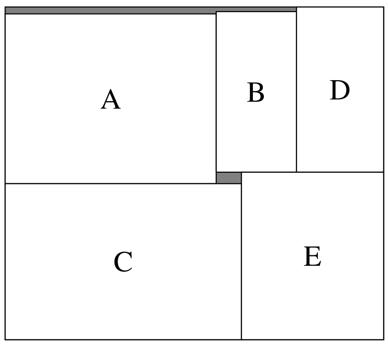
Figure 2.8 shows a toy instance for the RCP addressed. Figure (2.8a) represents a 2D item set. Figure (2.8b) represents the 2D plate. Figure (2.8c) represents a solution for the non-staged non-guillotine 2DSKP, and Figure (2.8d) represents a solution for the non-staged guillotine 2DSLOPP.



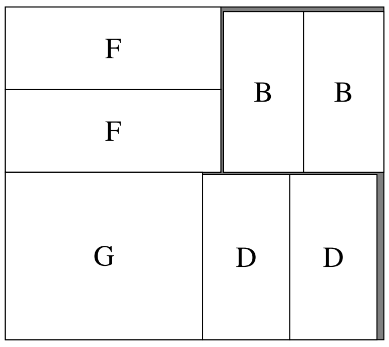
(a) 2D items



(b) 2D plate



(c) 2DSKP solution



(d) 2DSLOPP solution

Figure 2.8: RCP toy instance.

Chapter 3

Literature Review

3.1 Introduction

The field that involves C&P problems arouses interest in many researchers, which is mainly due to its challenging nature and the sheer number of real problems that can be modeled based on it. Every year, new techniques to develop solutions to these problems are addressed in literature, in addition to improvements to existing ones, both in terms of solution quality, performance and ways to manage the structures of the models developed.

Approaches to address C&P problems began to surface around 1960, with the adoption of simple mathematical models for 2D versions of the problem (see Gilmore and Gomory [45]). In contrast, approaches to handling practical cases of packing problems addressing 3D models were only developed years later thanks in large part to Tsai [104], who proposed a robotic 3D pallet loading with boxes of various sizes. Nowadays, we have a large assortment of strategies to solve these problems such as heuristics, exact and approximation methods.

Several interesting surveys and books have been published in recent years to try to review the fast-growing C&P literature, see, e.g., Crainic et al. [23], Iori et al. [57], Lodi et al. [69], Scheithauer [96] and Silva et al. [102]. For a recent review of the latest publications in the field of knapsack problems, we refer the reader to Cacchiani et al. [15, 16].

A typology of C&P problems has been proposed initially by Dyckhoff [34]. In that work, Dyckhoff was able to describe the basic structure concerning C&P, modifying the classification of the algorithmic strategies to obtain solutions, classifying them in specific groups. However, the proposed classification was not generic enough, and some problems remained yet unclassified. To cover these cases, a new typology was proposed by Wäscher et al. [109]. Later, Bortfeldt and Wäscher [13] extended the study in Wäscher [109] by considering the area of container loading and its main constraints. Specifically, they formalized the main concepts of loading containers with items and the related constraints. They noted that "the space available for packing above a pallet might be interpreted as a container, too", so they discussed constraints that can arise either when loading a container or a pallet. Indeed, apart from container-related constraints (such as weight limit and distribution), they also discussed item-related constraints (as loading priorities, orientation, dimension and stacking) that can be useful for pallets. The authors also considered cargo-related constraints, positioning constraints, and other load-related constraints that may appear during transportation (see also Iori and Martello [58, 59]) such as vertical and horizontal stability.

In the following, we highlight the main work that we have found in literature addressing C&P problems related to the PBP and RCP. We address the techniques to create partial and complete solutions, with exact and heuristic algorithms to solve the practical and theoretical versions of these problems.

3.2 PBP related works

The PBP emerges as a variant of the *Container Loading Problem* (CLP), which has received large attention in the last years. As previously mentioned, in Bortfeldt and Wäscher [13], a comprehensive survey of the main constraints used in the literature is presented. The authors verified that heuristic approaches are more frequently used than exact and approximation-guarantee algorithms. In Silva et al. [102], the *Pallet Loading Problem* (PLP) is considered. In this problem, a set of two-dimensional rectangular items needs to be packed without overlapping and by allowing a 90° rotation into a two-dimensional bin. The authors proposed a broad analysis of the solution methods and some aspects concerning computational complexity, upper bounds, and data sets most used in numerical experiments. In Crainic et al. [23], a survey is presented about 2D and 3D Orthogonal Packing Problems, focusing on data structures for the packing representation and the item-positioning rules. Concerning criteria to place items, they highlighted: (*i*) *interval graphs* (Fekete and Schepers [39]), to represent the overlap between items; (*ii*) *corner points* (Martello et al.[72]), that describe feasible places to pack an item in a partial solution; and (*iii*) *extreme points* (Crainic et al. [22]), that increase the amount of feasible regions on the partial packing of the corner points. Recently, Iori et al. [57] proposed a survey on variants of 2D packing problems, considering techniques to represent and handle items, relaxation methods, as well as exact and heuristic approaches. We also refer to Wäscher et al. [109] for a categorized typology of 2D packing problems, besides Delorme et al. [31] for a state-of-the-art computational analysis.

The PBP can be separated into two subsequent decisions: the first one consists of creating 2D layers, while the second one involves stacking layers to form pallets and thus considers the 3D characteristics. In the following, we discuss some relevant approaches for 2D and 3D, respectively.

For what concerns heuristics for 2D problems, Chazelle [17] described an efficient way to implement the famous bottom-left heuristic, which packs the items, one at a time in a given order, in the lowest and left-most position. In Burke et al. [14], a new placement heuristic, called best-fit, is presented for a 2D cutting problem, allowing non-guillotine packings and rotations of 90 degrees. This technique uses a dynamic search based on the "niches", which are the available bottom-most gaps for an item in the partial packing. In terms of metaheuristics, in Alvarez-Valdes et al. [7], a GRASP approach is developed for the 2D Strip Packing Problem (SPP), which is the problem of packing items into a strip of a given width by minimizing the used height. In the constructive phase, the items are placed into rectangles following specific criteria. A new rule attempts to foresee the future effect of the tallest object in the final solution to avoid spikes. The local search iteratively destroys and rebuilds portions of the current solution. Extensive computational experiments attested to the effectiveness of the proposed strategy. Imahori and Yagiura [56] improved the technique proposed by Burke et al. [14] by presenting a quicker implementation based on a balanced binary search tree and a priority heap with a doubly-linked list. In Leung et al. [65], a complete set of techniques to deal with the 2D SPP is presented. The authors use the so-called “skyline” approach in conjunction with greedy local search, a simulated annealing metaheuristic, and a multi-start diversification strategy. We also mention the so-called G4-Heuristic, developed by Scheithauer and Terno [99] for the PLP. In terms of exact algorithms, the use of Combinatorial Benders decompositions has recently lead to good computational results for a number of 2D packing problems, as in, e.g., Côté et al. [25, 26] and Delorme et al. [31].

Regarding 3D problems, Haessler and Talbot [51] addressed a real CLP involving some practical constraints. They proposed an integer programming formulation and a heuristic algorithm. Bischoff and Ratcliff [11] presented two heuristics for the CLP: the first one produces loading patterns with a high degree of stability; the second one considers a multi-drop situation in which a Last-In-First-Out constraint is imposed on the cargo. Terno et al. [103] proposed the parallel generalized layer-wise loading approach (PGL-approach) for the CLP. The constraints they consider are: (i) weight capacity; (ii) placement (some items cannot be inserted over the others); (iii) splitting (items of the same type should be loaded in a minimum number of containers); (iv) connectivity; and (v) stability. Using a complex branch-and-bound algorithm, the authors show a certain degree of competitiveness compared with classical solutions reported in the literature. Bortfeldt and Gehring [12] proposed a hybrid genetic algorithm to solve the CLP with a single container and a strongly heterogeneous set of boxes, considering orientation, stability, stacking, weight, and balance. The results showed a better efficiency with more significant heterogeneity of the box sets. Egeblad et al. [35] addressed the CLP for a single container (in the knapsack version), using irregular shaped items, and taking a stability constraint into account. They performed tests on randomly generated and real-world instances derived from a prominent European furniture producer. Józefowska et al. [63] studied the CLP considering rotation, stackability, and stability constraints. They considered a case study arising from a household equipment factory, proposing a best-fit heuristic based on the idea of wall-building over available space. Kurpel et al. [64] presented techniques to obtain bounds and exact approaches to solving input minimization and output maximization versions of the multiple CLP with rectangular boxes. They adapted and evaluated four discretization techniques from the literature and considered mathematical formulations to practical constraints such as rotation, vertical stability, and separation of the boxes. We also refer to a GRASP-based algorithm for the 3D BPP proposed by Parreño et al. [82]. The related algorithm was developed by using a maximal-space heuristic for the CLP during the constructive phase, and several moves were combined in a Variable Neighborhood Descent approach in the improvement phase. Da Silva et al. [27] proposed a matheuristic framework to solve two CLP named 3-dimensional Single Large Object Placement Problem (3DSLOPP) and Single Stock-Size Cutting Stock Problem (SSSCSP) (according to the typology proposed by Wäscher et al. [109]), considering seven practical constraints (orientation, load balance, loading priorities, positioning, stacking, stability and weight limit). The method consists of a wall-building approach by means of box arrangements: first, the solution is created by dividing the original problem into two smaller problems (Knapsack Problem and one-dimensional Packing Problem), which are solved through two mathematical formulations; then, a greedy improvement is carried out, using a heuristic that removes the walls with the lowest quality, recreates them, and re-applies the previous mathematical formulations to try to improve the solution. In Gzara et al. [48], an extension of the paper by Elhedhli et al. [37], is proposed to solve the 3D BPP. The authors proposed a layer-based column generation approach through a column generation framework, which provides a pool of layers to construct bins. The bins are created one at a time by means of a heuristic that selects layers ordered in descending order of density, meeting several constraints such as vertical support, load-bearing, planogram sequencing, and bin weight limits. About load-bearing, the algorithm ensures that the items of the top layer have sufficient support from below of, at least, 70%, and, in addition, it uses an efficient graph representation to update the total weight to which an item is subjected. Other approaches based on mathematical models can be found in Alvarez-Valdes et al. [6], Martins and Dell [76], Neliβen [79], Ribeiro and Lorena [92], and Wu and Ting [110]. For what concerns the use of 3D packing problems arising in freight transportation, we refer to the surveys by Iori and Martello [58], Schmid et al. [100] and Vidal et al. [106].

The works that, in our opinion, most resemble ours are Alonso et al. [4] and Ranck Júnior et al. [88], which, however, do not consider families nor visibility and contiguity constraints.

Ranck Júnior et al. [88] addressed a real problem of a beverage company for packing boxes into a multi-compartment container and delivering over a predefined route, meeting the practical constraints of orientation, stability, load-bearing strength, and load balancing. The proposed heuristic is a hybrid method to create a solution based on layers. These are created via the First Fit and Best Fit heuristics; next, to generate a final solution, a mathematical model is solved, meeting the delivery constraints over the boxes. An iterated local search is carried out over individual layers as an improvement phase, in an attempt to generate new box dispositions.

Alonso et al. [4] considered practical constraints through a real example originating from a logistics company required to load products into pallets (pallet building) and then load the created pallets into trucks (truck loading), by considering several practical constraints. For the pallet building, they incorporated orientation, support, priority, and stackability constraints. Regarding the truck loading, they adopted restrictions due to priority among pallets, stability, and stackability. They proposed a GRASP algorithm using a constructive phase, a randomized strategy to diversify the solutions, and an improvement phase. The efficiency of their GRASP was analyzed by comparing it with lower bounding procedures, showing good results. The study was later extended by Alonso et al. [1, 2], who developed mathematical models for the case of multiple container loading, addressing several additional practical constraints such as vertical and horizontal stability, multi-drop, and load balance. A further extension was proposed by Alonso et al. [3], which studied the advantages of using a metaheuristic algorithm when comparing to an integer programming formulation.

Finally, it is worth mentioning in relation to both similar works that, although they deal with several practical constraints and present similarities to our problem, their main idea is optimizing the occupation in bins, i.e., maximizing the fill factor. In this sense, our proposal goes further by additionally requiring other practical constraints (contiguity for items, visibility for families, layer disposition) to solve the problem addressed.

3.3 RCP related works

When it comes to RCP, this literature review will focus on exact methods to solve the 2DSKP and 2DSLOPP in both the non-guillotine and guillotine cuts. Mainly, it will be divided into two parts: papers dealing with non-guillotine cutting patterns and papers considering only guillotine cutting patterns. The literature considering guillotine cuts will be divided into: the papers considering a limited number of stages (k-staged); and the papers considering an unlimited number of stages (non-staged).

3.3.1 Non-guillotine cutting patterns

The first mathematical model dealing with general non-guillotine cutting patterns was proposed by Beasley [9]. The 2DSLOPP was modelled as a zero-one Integer Linear Program (ILP) model with $O(m|L||W|)$ decision variables and $O(|L||W|)$ constraints, since it is based on the discretization of the set of possible positions to place an item. The decision variables are related to whether a piece of a particular type is cut with its bottom-left corner at a certain position or not. The size of the model could be significantly reduced by considering normal patterns. However, the model was not solved monolithically, instead, an upper bound on the optimal solution was derived by using a tree search approach based upon Lagrangean relaxation of the ILP model, and a subgradient optimisation was used to minimise the upper bound obtained from the Lagrangean relaxation.

A zero-one ILP model for the general non-guillotine 2DSKP was proposed by Scheithauer and Terno [98]. The decision variables represent whether an item is cut to the right/left or above/below another item. In this problem formulation, all items are cut from an enlarged large rectangular object. However, a binary decision variable is used to count, in terms of contribution to the objective function, the items that are positioned in the initial large rectangular object. No computational experiments using this model was reported.

In Arenales and Morabito [8], the unbounded 2DSLOPP was solved using an AND/OR graph and a tree search procedure. A cutting pattern is built up by making successive cuts, either a single guillotine cut or a first-order non-guillotine cut. The tree search procedure includes heuristics that can be used to prune the tree; however, it may be the case, depending on the parameters, that the optimal solution is lost.

Hadjiconstantinou and Christofides [49] developed a tree-search exact algorithm for the 2DSLOPP that places the next item considering the left-most downward strategy. The search was limited by an upper bound based on a Lagrangian relaxation procedure and improved using subgradient optimization. Later, Hadjiconstantinou and Iori [50] solved the 2DSLOPP by means of genetic algorithm.

Fekete and Schepers [39, 40] and Fekete et al. [41] proposed an optimal tree search based algorithm based upon a graph-theoretic representation of the relative position of items in a feasible cutting pattern. The projections of cut items are made onto both the horizontal and vertical edges of the large rectangular object. Each projection is then translated into a graph, where the nodes in the graph are the cut items, and an edge joins two nodes if the projections of the corresponding cut items overlap. The authors proved that a cutting pattern is feasible if the projection graphs have certain properties. In the tree search procedure, branching to enumerate cutting patterns is related to whether an edge is included in or not from a projection graph. Upper bounds derived from the solution to knapsack problems and problem reduction tests are used to shorten the search.

3.3.2 Guillotine cutting patterns

3.3.2.1 k-staged guillotine

The 2DSKP has been widely studied in the literature. The seminal work considering two-staged guillotine cutting patterns goes back to Gilmore and Gomory [45], where a dynamic programming procedure was proposed and embedded in a column generation procedure to solve the 2-dimensional Cutting Stock Problem (2DCSP).

ILP models have been considered for problems limited to two or three stages. In [70] two models with a polynomial number of variables and constraints was proposed for the 2DSKP. The models were based on the observation that two-staged patterns are composed by levels, and it assumed that the total number of items is the maximum number of potential levels that can be initialized. The decision variables corresponded to the assignment of items to levels and the assignment of levels to the large object. The models were extended to different variants of the problem: the unbounded case, the possibility of a 90 degrees rotation of the small items, and the double constrained case, with a lower and an upper bound on the number of times that an item can be placed on the large object. In Puchinger and Raidl [87], one of the ILP models of Lodi and Monaci [70] was extended to three-staged patterns and used in a column generation procedure to solve the 2DBPP.

Hifi [53] studied the two- and three-staged problems. The two-staged problem considered that items could be rotated and was solved by an exact algorithm that resulted from the extension of Gilmore and Gomory's procedure. The three-staged problem considered both the fixed and rotated positioning of the item and was solved by an algorithm based on a graph search combined with dynamic programming procedures. Hifi and M'Hallah [54] proposed an exact algorithm based on a branch-and-bound procedure using a bottom-up strategy for the two-staged problem, introducing new upper bounds and new pruning strategies for the unbounded problem variant.

3.3.2.2 Non-staged guillotine

Christofides and Whitlock [19] presented the first exact tree search method for the non-staged 2DSLOPP. The size of the tree was limited by a bound derived from a state space relaxation of a dynamic programming formulation. Christofides and Hadjiconstantinou [18] improved the tree search algorithm by limiting the size of the tree search using a bound based on the Lagrangean relaxation of an integer programming model and defined dominance rules to remove repeated patterns through symmetry breaking and cut ordering. Similarly, Cung et al. [24] and Velasco and Uchoa [105] considered exact tree search algorithms with improved bounds using dynamic programming procedures.

Morabito et al. [78] proposed the AND/OR graph model for the unconstrained and non-staged problem. In this approach, the cutting patterns are represented as complete paths in the AND/OR graph, where the AND vertex corresponds to a cutting decision resulting in several sub-plates and the OR vertex represents a sub-plate. The authors considered a hybrid strategy combining a depth search heuristic and hill climbing to search the graph.

Cintra et al. [20] and Russo et al. [94, 95] considered a dynamic programming procedure for the unbounded case. Cintra et al. [20] used algorithms based on dynamic programming to solve the 2DSLOPP with different variants allowing the rotation of orthogonal items, k-staged patterns, and non-staged patterns. All cuts were performed on discretization points obtained by integer conic combinations of the width or the height. Russo et al. [94] corrected and improved one of the two dynamic programming procedures proposed by Gilmore and Gomory [46] and considered in their algorithm the reduction of the discretization points proposed by Cintra et al. [20]. Russo et al. [95] modified and improved the dynamic programming procedure proposed by Russo et al. [94], reducing the search space and avoiding redundant solutions. Combined with computational refinements, this allowed the authors to obtain the optimal solution for very large instances that had not yet been solved.

Dolatabadi et al. [32] also used a dynamic programming algorithm for the non-staged guillotine 2DSLOPP. The authors used a recursive exact procedure combined with an implicit enumeration of all feasible patterns by recursively dividing the bin into two parts by guillotine cuts (horizontal or vertical). The procedure was then embedded into two exact algorithms and computationally tested on a set of instances from the literature. In Russo et al. [93], a survey on relaxations for two-dimensional guillotine cutting problems and a categorization of the resulting bounds was presented. In this work, the authors pointed out that the procedure from Dolatabadi et al. [32] is not exact since the extension proposed for a dynamic programming from the literature considering maximal profit solutions was not accurate. More recently, Clautiaux et al. [21] proposed a hypergraph model for the non-staged guillotine 2DSKP, where a cutting pattern was represented by a flow in the directed acyclic hypergraph. The hypergraph model is derived from a forward labeling dynamic programming recursion that enumerated all non-dominated feasible cutting patterns. To reduce the hypergraph size, the authors used dominance rules and a filtering procedure based on Lagrangian reduced costs fixing of hyperarcs. The model was extended in order to consider other variants of the problem, as a k-staged version and the 2DSLOPP. Extensive computational experiments were performed and compared with the solutions obtained with the dynamic programming algorithm proposed by Dolatabadi et al. [32] and the solutions obtained by the ILP model proposed by Lodi and Monaci [70].

Pisinger and Sigurd [83] solved the 2DSKP by constraint programming, considering non-guillotine and guillotine cutting patterns with a limited and unlimited number of stages. The solution was then embedded in a column generation procedure for the 2DBPP.

Messaoud et al. [77] presented a polynomial algorithm to determine whether a given pattern is guillotinable. The authors defined the necessary and sufficient conditions for a cutting pattern to be guillotinable and formulated guillotine constraints into linear inequalities. Based on the mathematical constraints, a guillotine SPP capable of being extended to the 2DSKP was formulated. However, the linear relaxation of the model was weak and in the computational results, only instances with five items were solved with a small computing time.

Furini et al. [44] proposed a Mixed-Integer Linear Programming (MILP) model for the non-staged guillotine 2DSKP with a pseudo-polynomial number of variables and constraints. The model was an extension of the pseudo-polynomial model proposed by Silva et al. [101] for the two-staged and three-staged 2DCSP. The computational performance was also improved using an exact procedure that selected the sub-set of variables that contained an optimal solution. The modeling of non-staged guillotine cuts was extended to the 2DCSP and the SPP.

Recently, Martin et al. [73] proposed new ILP models for the 2DSLOPP which were based on the grid model proposed by Beasley [9] for the non-guillotine problem. The new models were developed for the non-staged problems, two-staged problems, and one-group cutting patterns. Based on the computational experiments, the authors concluded that the ILP models performed well in problem instances with items that were large with respect to the object, but performed disadvantageously in problem instances with many small item types.

Martin et al. [74] proposed new pseudo-polynomial MILP models for the non-staged and k-staged 2DSLOPP which were based on the classic bottom-up packing approach of Wang [107]. The computational experiments showed that the models are well suited to problem instances where the maximum number of small items that can be packed in the large object is small.

A top-down cutting approach for the two and three-dimensional SKP and SLOPP is presented by Martin et al. [75]. The pseudo-polynomial MILP model explicitly uses a binary tree structure as input for variables and constraints indexing. The nodes of the binary tree are rectangles, and the branches are guillotine cuts. The algorithm for the binary tree requires an upper bound on the number of items in the optimal solution. In this MILP model, the cut positioned is not previously defined as in the MILP models proposed in this work. The computational experiments with benchmark instances shows that the model is competitive with the literature when the number of items in the optimal solution is small or moderate.

Finally, we also refer to the empirical analysis of exact algorithms provided by Becker and Buriol [10] for the one-dimensional unbounded Knapsack Problem.

In Table 3.1, the papers considered in the literature review for the guillotine problem were classified according to the type of problem. The abbreviations used are 2DSLOPP for the 2-dimensional Single Large Object Placement Problem, and 2DSKP for the 2-dimensional Single Knapsack Problem. Other features taken into consideration are: the number of stages considered (k) – ∞ is used if the problem is non-staged –; if a 90 degrees rotation of the items is allowed; and if the problem is weighted or unweighted. Moreover, the instances considered in the computational experiments in the respective paper were identified.

Table 3.1: Classification of the problem variants found in the literature review.

Publication	**2DSLOPP**	**2DSKP**	**# stages**		**Rotation**	**Weighted**	**Instances**
			k	∞			
Christofides and Whitlock [19]	x			x	No	x	cgcut1-3
Christofides and Hadjiconstantinou [18]	x			x	No	x	gcut1-3 wang3 OF1-2
Lodi and Monaci [70]	x		2		Yes	x	HH 2 3 A1-2 STS2 STS4 CHL1-2 CW1-3 Hchl2 Hchl9 2s 3s A1s-2s STS2s STS4s OF1-2 W CHL1s-2s A3-5 CHL5-7 CU1-2 Hchl3s-8s BW HZ2 MW1 UW1 W1 B H HZ1 U1-2 UU1
Hifi and M'Hallah [54]	x		2		No	x	2 3 A1-5 STS2 STS4 CHL1-2 CW1-3 Hchl2 Hchl9 2s 3s A1s-2s STS2s STS4s OF1-2 W CHL1s-2s CHL5-7 CU1-2 Hchl3s-4s Hchl6s-Hchl8s HH
Puchinger and Raidl [87]		x	3		No	x	
Pisinger and Sigurd [83]		x	x	x	No	x	
Cung et al. [24]	x			x	Yes	x	2 3 A1-5 STS2 STS4 CHL1-4 2s 3s A1s-2s STS2s STS4s CHL1s-4s HH CHL5-7 Hchl1-2 Hchl9 Hchl3s-8s
Dolatabadi et al. [32]	x			x	No	x	gcut1-gcut-13, cgcut1-cgcut3, wang20, okp1-okp5, APT30-APT39, APT40-APT49
Furini et al. [44]	x			x	No	x	gcut1–12 wang20 cgcut1–3 okp1–5 2s 3s A1s-2s STS2s STS4s HH 2 3 A1-2 STS2 STS4 CHL1 OF1-2 W CHL1s-2s A3-5 CHL2 CW1-3 Hchl2 Hchl9 CHL5-7 CU1-2 Hchl3s-4s Hchl6s-8s
Clautiaux et al. [21]	x		2, 4	x	Yes	x	2s 3s A1s-2s STS2s STS4s OF1-2 W CHL1s-4s CHL5-7 ATP30-39 CU1-11 Hchl3s-8s cgcut1-3 A1-2 STS2 STS4 CHL1-4 CW1-11 ATP40-49 Hchl1-2 Hchl9 okp1-5
Velasco and Uchoa [105]	x			x	Yes	x	Wang1-3 OF1-2 CU1-11 ChW1-3 CW1-11 ATP30-49
Martin et al. [73]	x	x	2	x	No	x	cgcut1-3 OF1-2 Wang20 gcut1-12
Martin et al. [74]	x	x	2	x	Yes	x	cgcut1-3 OF1-2 Wang20 okp1-5 gcut1-13 ATP30-39 ATP40-49
Martin et al. [75]	x	x		x	No	x	cgcut1-3 OF1-2 Wang20 okp1-5 gcut1-13 CU1-11 CW1-11
'Floating cuts' model	x	x	x	x	Yes	x	cgcut1-3 CW1-3 okp1-okp5 gcut1-12 CU1-11 CW1-11 OF1-2 2s 3s A1s-2s STS2s STS4s W CHL1s-2s CHL5-7 Hchl3s-8s cgcut1-3 CHL5-6 A3-5

Chapter 4

Solution of a Practical Pallet Building Problem with Visibility and Contiguity Constraints

4.1 Abstract

We study a Pallet Building Problem (PBP) that originates from a case study at a company that produces pallet building automated systems. The problem takes into account well known constraints, such as rotation and stackability, and we introduce two practical constraints named visibility and contiguity between items of the same type. We formalize the problem and propose heuristic algorithms to solve it, using a strategy that first creates 2D layers and, then, creates the final 3D pallets. The proposed heuristic is based mainly on the Extreme Points Heuristic, that is tailored to choose feasible positions to pack items during the construction of the solution. Besides that, we adapt our proposed heuristic using other basic heuristics from the literature, considering different constraints. The performance of the algorithms is assessed through extensive computational tests on real-world instances, and the obtained results show the proposed heuristics are able to create compact packing in a very short time.

4.2 Contributions

Our main contributions in this work can be sketched as follows: an interesting problem that derives from a real-world industrial application is presented (Chapter 1); the concept of contiguity of items is used, that is very useful in practice during loading/unloading operations but has never been formally treated in the literature (Chapter 2); a 2-step heuristic algorithm to solve the problem addressed; and extensive computational tests on instances derived from the real-world case study are given.

4.3 Solution algorithm

This section presents the 2-step heuristic that we developed to solve the PBP with visibility and contiguity constraints, which we call *Extreme Points Modified Heuristic* (EPMH). EPMH produces feasible solutions by dividing the problem into two parts. First, we create layers to deal with individual items separately. This phase forces the presence of packing, visibility, and contiguity constraints through a process guided by an evaluation function. Second, the pallet generation step tries to minimize the quantity of pallets using a greedy strategy. Next, we particularly focus in the description of the first step (layers creation), as the second step (pallets creation) is based on a quite standard algorithm.

4.3.1 Creating layers

4.3.1.1 Item positioning

To pack an item in a layer, we propose an adaptation of the Extreme Points Heuristic (EPH) presented by Crainic et al. [22] . Originally, EPH was built for 3D packings, but we restrict here the discussion to the 2D case. EPH works with the concept of extreme points. An extreme point e is a point in the 2D space where an item can be packed by taking into account the partial solution built so far. For the sake of clarity, packing an item in an extreme point e means packing its bottom-left corner (reference point) in e.

In EPH, the items are packed one at a time in the layer, by considering a set E of available extreme points initialized with the origin point $(0,0)$. Then, each time a new item is packed, E is updated by removing the point used for the packing and possibly inserting new extreme points. These new extreme points are obtained by computing the projection of the last item packed over the partial packing solution under construction, considering the axes x and y. For the x-axis, EPH horizontally projects the top edge of the item to its left, until the projection touches a previously packed item or the left border of the layer (i.e., the y-axis). This is the first extreme point that is possibly created. For the y-axis, instead, EPH considers the right edge of the item and vertically projects it towards the bottom until the projection reaches a previously packed item or the bottom border of the layer (i.e., the x-axis). This is the second extreme point possibly created. These two points are added to E if they were not already included in it. Figure 4.1a depicts an example with a set E formed by white and black points. The white point is selected for packing an item of type k and is thus removed from E. The dark gray points are the new extreme points added to E after the packing.

Note that, besides the projections from the last item packed, it is mandatory to verify all projections from previously packed items on the last item. This step, that we name past projections, is needed to find new extreme points not yet available. We execute it right after having computed the projections from the last item packed.

The original EPH solves a pure 2D packing problem, so we need to include a set of changes to be able to solve the more complex PBP with visibility and contiguity constraints. The first modification we apply consists in increasing the search space inside a layer. The EPH creates only two new extreme points at a time, whereas visibility and contiguity constraints narrow the search space. As a consequence, a layer could be closed even when its occupation is low because E does not contain any feasible extreme point. Figure 4.1b shows an example where this situation occurs because none of the current feasible extreme points fulfills the contiguity constraints.

To overcome this limitation, we included some new extreme points, considering two cases for contiguity and visibility. In the first case, the extreme points are included around the last item packed to allow the contiguity with the next items of the same

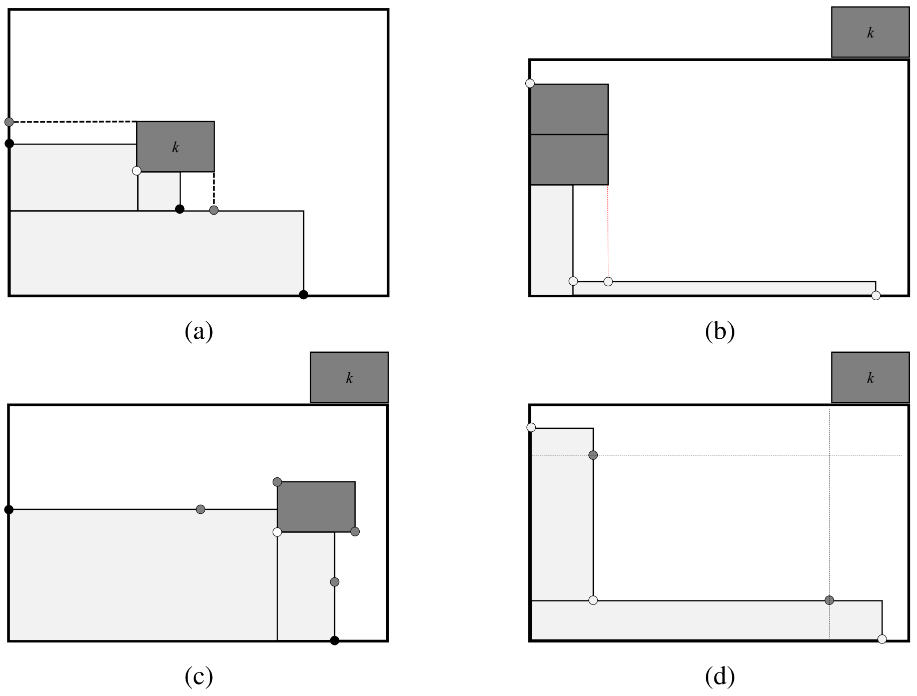


Figure 4.1: Extreme points: (4.1a) Black and white points form the current set E; the white point is chosen to pack k and is then removed from E; dark gray points are added to E after the packing; (4.1b) Case where EPH generates only extreme points that are infeasible for the PBP with visibility and contiguity constraints; (4.1c) New extreme points created for contiguity: after packing k in the white point, four extreme points (represented by dark gray points) are included in E; (4.1d) New extreme points created for visibility: new dark gray points that meet visibility constraints are added to E.

type. Basically, four new points are added: the item top-left and bottom-right corners allow the top and right connections, respectively, and the points on the left and below the item enable the left and bottom connections, respectively. Figure 4.1c shows the new extreme points.

Extreme points regarding the visibility case are included close to the top and right layer borders, when the current item has a different type from the previously packed item and there is no point in E that allows the current item to meet all constraints. In

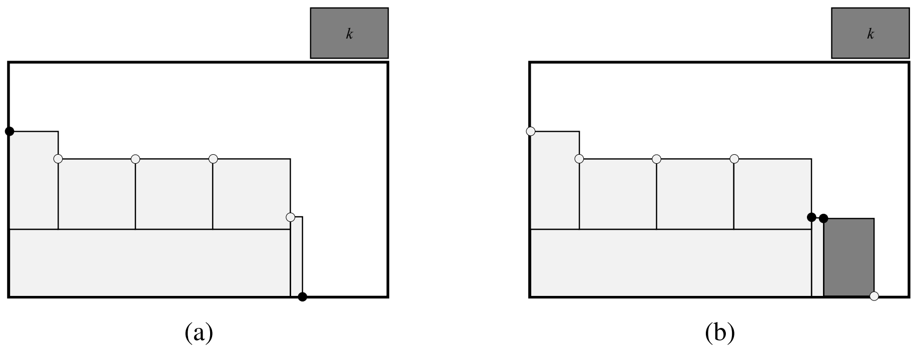


Figure 4.2: Partial solutions and their feasible (black) and infeasible (light gray) extreme points for the next item of type k: representation of the extreme points that meet visibility (4.2a) and contiguity (4.2b) constraints.

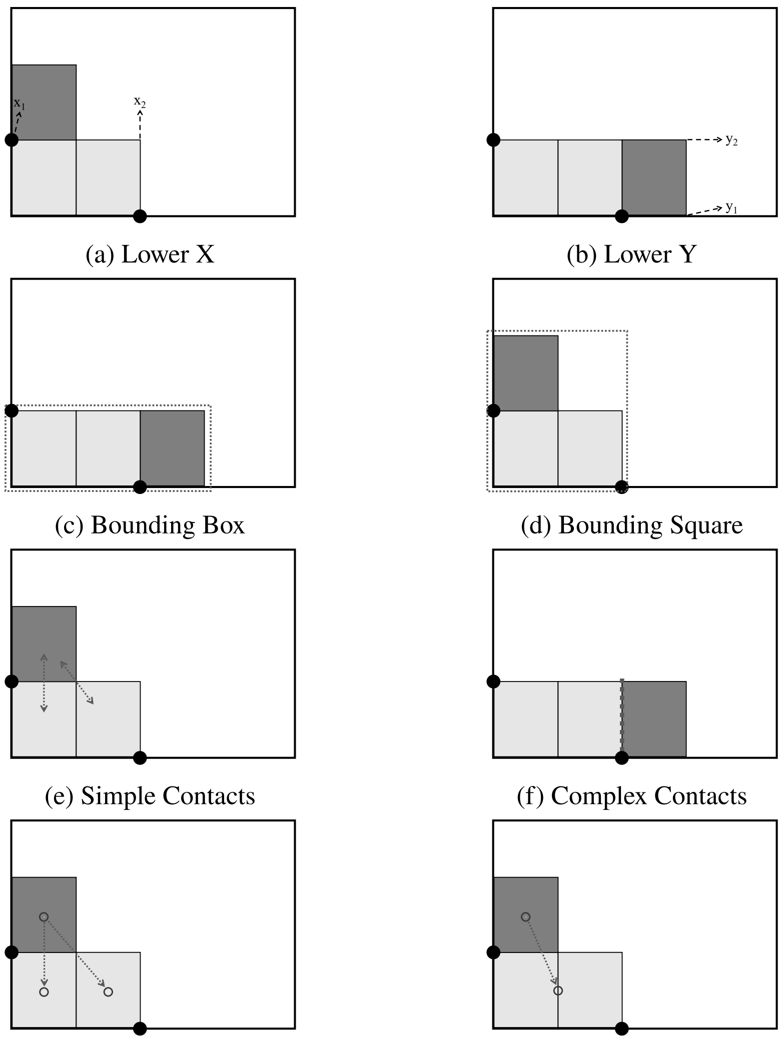
this case, two new points are added: the first feasible point that meets the contiguity to the top layer border, proceeding from right to the left; and the first feasible point that meets the contiguity to the right layer border, proceeding from top to bottom. Figure 4.1d shows the new extreme points for this case.

The second modification is related to performance. Current extreme points are directly linked to visibility and contiguity constraints. These points can be either feasible or infeasible for the current item according to its type and to the partial solution (because of contiguity and visibility). Thus, we work with two extreme points sets: *feasible extreme points*, E_f, and *infeasible extreme points*, E_i, representing, respectively, points that meet or do not meet the visibility and contiguity constraints for the current item. Therefore, before adding an item to the partial solution, we first update these two sets, and only after this is done we check if the item overlaps with previously packed items. Note that these two sets are related to the current configuration, and after the addition of a new item, infeasible points may become feasible and vice-versa. The benefit of using these sets is that we avoid checking the overlapping constraint for all extreme points. An example of sets E_f and E_i is provided in Figure 4.2.

4.3.1.2 Evaluation functions

Given the next item k, the point $e \in E_f$ that results in the best packing has to be chosen. For that purpose, the algorithm uses an evaluation function d to calculate the fitness for each e to aggregate k into a group. That said, we propose the following fitness evaluation functions:

- LOWEST X: finds the point that has the minimum x value among all feasible points;
- LOWEST Y: finds the point that has the minimum y value among all feasible points;
- BOUNDING BOX: calculates the bounding box area – minimum rectangle area that covers a set of items – with k, and finds the point which minimizes the area among all bounding boxes;
- BOUDING SQUARE: same as above but with squares instead of boxes;
- SIMPLE CONTACTS: calculates the number of contacts – a contact between two items happens when they meet the contiguity previously described in this Section, i.e., when their distance is lower than ξ – with k, and finds the point which maximizes this number;
- COMPLEX CONTACTS: similar to the previous one, but the number of contacts is switched by the contact length – the euclidian distance of edge overlap between different items – among the items that meet the contiguity with k. It finds the point which maximizes this value;
- DISTANCE SUM: calculates the distance sum among the center of gravity of each packed item with the center of gravity of k, and finds the point which minimizes this sum;
- CENTER OF GRAVITY: evaluates the distance between the center of gravity of the partial packing and the center of gravity of k, and finds the point that minimizes this distance.



(a) Lower X (b) Lower Y

(c) Bounding Box (d) Bounding Square

(e) Simple Contacts (f) Complex Contacts

(g) Distance Sum (h) Center of Gravity

Figure 4.3: Fitness evaluation functions of feasible extreme points, considering a new dark gray item.

Figure 4.3 shows a graphical example for each fitness evaluation function proposed.

4.3.1.3 Item grouping

The fitness of a new item k is calculated from the partial packing in the layer using a specific group of items, formed by some or the whole set of their items. To form these groups, we proposed the following strategy: when k is the first item of a specific type in the partial solution, its fitness is calculated considering all items of the partial packing in the layer. When there is at least one item of the same type of k, the fitness is calculated by considering the group formed by the items of this specific type.

4.3.1.4 Layer creation and classification

The heuristic tries in a first phase to create as many single-item and single-family layers as possible. Then, in a second phase, it takes care of creating the residual layers. This is because at most one residual layer can be inserted in a pallet, and so their number should be as small as possible.

In the first phase, the algorithm establishes a specific order for families and their item types, assigning them to the data structure S. Using this order, the heuristic tries to generate single-item layers by packing items one at a time according to the concept of modified extreme points that we describe below. If the packing obtained with a single item type meets the fill-factor constraint, but also has free space to potentially accommodate more items, then the heuristic tries to add more items of the same family to generate a single-family layer. Once no more item fits the residual space, the layer is closed. When, instead, the current packing of a layer does not fulfill the minimum fill-factor, then the layer is destroyed and all items are inserted in a residual item list. As speed-up technique, when the sum of the areas of the items of a specific family is not enough to meet the fill-factor constraint, then the items are directly inserted in the residual item list.

At the end of the first phase, no more single-item and single-family layers can be built, so the heuristic focuses on the items in the residual list. The same criterion used in the first phase is also used in this second phase to select items one at a time and fill a layer as much as possible. The process is repeated, layer after layer, until all items have been packed in residual layers.

The classification of a layer is made after it is closed, analyzing its occupation, the families and the item types that are included in it.

4.3.2 Building pallets

Once all layers have been generated, we need to stack them in the minimum number of pallets. To this aim, we developed a simple greedy constructive heuristic that works as follows. Recall the height of a layer is calculated as the height of the highest item in the layer, and, similarly, the stackability of a layer is computed as the maximum stackability among the items in the layer. Single-item, single-family and residual groups of layers are sorted according to non increasing stackability, breaking ties by non increasing height, and this order is maintained all throughout the algorithm.

As residual layers need to be packed in separated pallets, the heuristic starts by choosing the first residual layer, if any, in the order, and uses it to initialize a pallet. Then, it fills the current pallet with single-item layers, one at a time in the order above, by fulfilling stackability and maximum height constraints. If the current pallet has still unused height, but no more single-item layer can enter it, the heuristic attempts filling the pallet with single-family layers. In this case too, the layers are scanned according to the above order. The process is repeated until no more layer can enter the pallet. In such case the pallet is closed and a new pallet is considered. The process continues until all layers are packed into pallets, thus creating a feasible solution.

4.4 Computational results

In this section, we present the details of the instances that we adopted for our tests and then show the computational results that we achieved. All experiments have been conducted on a PC Intel Core i5 Dual-Core 1.8GHz CPU, 8GB RAM, macOS Catalina Operating System. The algorithms have been implemented in Java (Oracle® JDK 8). Due to its deterministic nature, each heuristic was run only once. Because of the high number of instances addressed, in the following we mostly report average results for groups of instances.

4.4.1 Instances

The results were obtained from a real-world database provided by an Italian company. We randomly selected 24 strongly heterogeneous instances, that are separated into 4 groups, each containing 6 instances, being characterized by different intervals in the number of distinct items: from 17 to 32; from 33 to 48; from 49 to 64; and from 65 to 80. The average number of items for each instance is approximately 690. Table 4.1 summarizes the details of the instances. Its columns are: ID number of the instance, number of item types (n), number of families, and total number of items ($\sum b^i$). The instances are of different difficulty. The larger the numbers of item types, families and items are, the more complex becomes the instance to be solved. In particular, the most challenging instances are those with more than 50 item types, 15 families, and 700 items. We report the database with the whole set of instances in `https://github.com/silveira-tt/PBP_instances`.

Table 4.1: Instances settings.

ID	n	N. of families	$\sum b^i$
1	20	4	877
2		12	269
3	23	4	388
4		13	115
5	29	6	438
6		15	2103
7	34	7	698
8		17	1322
9	37	6	300
10		17	449
11	48	9	395
12		18	748
13	59	10	455
14		19	633
15	64	11	658
16		19	683
17	61	10	300
18		19	797
19	75	16	790
20		20	829
21	79	17	855
22		21	944
23	66	13	738
24		19	767

4.4.2 Parametric configurations

For all instances, as a company requirement, the fill factor was set to 75%, and the container dimensions were set to 1500, 1250, and 1050 for height, width, and length, respectively. The algorithm uses an initial structure *S* sorted by non-increasing value of stackability for families, and a non-increasing value of height for item types. We tested it allowing rotation of 90 degrees of the items.

4.4.3 Evaluation

We solved the proposed instances with three algorithms. The first one is the basic EPH described in Section 4.3, and the second is the constructive heuristic based on skylines and proposed by Leung et al. [65], named Skylines Heuristic (SLH) in the following. These two methods are taken from the literature and do not take into account contiguity and visibility of the items. The third method is the newly-proposed EPMH method, attempted with the different evaluation functions discussed in the previous section. All such methods are adopted to generate 2D layers, which are then put together into pallets by the greedy heuristic of Section 4.3.2. In this way, it is possible to have a real estimate of how the composition of the layers interferes in the pallet building, even though the final pallets are not necessarily the best ones in all cases, due to its construction based on the greedy heuristic.

The computational results that we obtained are summarized in Table 4.2. Each row in the table provides average results for a given algorithm on the 24 attempted instances. We tested the two heuristics from the literature, namely EPH and SLH, with (R) and without (-) rotation. We then tested EPMH with 8 different evaluation functions, attempting 4 different configurations: R/B stands for 90 degrees rotation allowed and visibility constraint imposed; R/- stands for rotation allowed and no visibility constraint imposed; -/B stands for no rotation allowed and visibility constraint imposed; and -/- stands for no rotation allowed but also no visibility constraint imposed. In all cases, the EPMH meets the contiguity constraint among items in the same group. In this way, we provide information on 240 solutions obtained, 10 for each of the 24 instances.

Table 4.2: Computational results (average results on 24 instances on each line).

Algorithm	Constraint	N. of pallets	N. of layers	2D fill factor	Single-item layers	Single-family layers	Residual layers	Max. percent. of residual layers	N. of non-contiguous items	N. of non-contiguous layers	N. of non-visible items	N. of non-visible layers	Time(secs)
EPH	R	24.75	43.46	0.82	14.46	9.13	19.88	76.36%	12.17	10.13	4.17	3.25	0.023
	-	**25.58**	43.04	**0.82**	12.83	8.50	21.71	74.07%	4.75	3.88	5.42	4.33	0.026
SLH	R	22.83	42.17	0.84	16.75	7.17	18.25	73.08%	4.38	3.58	4.75	3.58	**0.009**
	-	27.58	45.25	0.78	**14.79**	6.00	24.46	77.19%	5.67	4.54	7.00	5.17	**0.011**
EPMH – LOWER X	R/B	22.83	42.00	0.84	14.42	9.38	18.21	73.08%	**0**	**0**	**0**	**0**	0.161
	R/-	22.79	42.04	0.84	14.38	9.46	18.21	71.70%	**0**	**0**	**2.92**	**2.42**	0.068
	-/B	**25.88**	43.29	0.81	12.83	**8.46**	**22.00**	**74.07%**	**0**	**0**	**0**	**0**	0.102
	-/-	**25.58**	43.17	**0.82**	12.83	**8.67**	**21.67**	**72.22%**	**0**	**0**	**3.71**	**2.67**	0.068
EPMH – LOWER Y	R/B	23.00	41.88	0.85	14.33	8.88	18.67	73.08%	**0**	**0**	**0**	**0**	0.137
	R/-	22.83	41.75	0.85	14.33	9.00	18.42	73.08%	**0**	**0**	4.42	3.33	0.079
	-/B	26.17	43.25	0.81	12.83	7.96	22.46	75.93%	**0**	**0**	**0**	**0**	0.108
	-/-	25.96	43.13	**0.82**	12.83	8.21	22.08	74.07%	**0**	**0**	4.67	3.46	0.077
EPMH – BOUND-ING BOX	R/B	**14.46**	**41.00**	**0.86**	**21.79**	**11.58**	**7.63**	66.67%	**0**	**0**	**0**	**0**	**0.104**
	R/-	**14.00**	**40.83**	**0.86**	**21.79**	**12.00**	**7.04**	66.67%	**0**	**0**	6.08	3.96	0.070
	-/B	26.17	43.25	0.81	12.83	7.79	22.63	75.47%	**0**	**0**	**0**	**0**	0.112
	-/-	25.67	**42.92**	**0.82**	12.83	8.04	22.04	75.93%	**0**	**0**	5.38	3.79	0.068
EPMH – BOUND-ING SQUARE	R/B	23.00	42.38	0.83	14.00	10.29	18.08	70.00%	**0**	**0**	**0**	**0**	0.110
	R/-	23.50	42.42	0.83	14.04	9.63	18.75	72.55%	**0**	**0**	5.13	3.83	0.081
	-/B	**25.88**	**43.17**	0.81	12.83	8.17	22.17	77.78%	**0**	**0**	**0**	**0**	0.106
	-/-	25.96	43.08	**0.82**	12.83	7.92	22.33	74.07%	**0**	**0**	5.08	3.54	0.067
EPMH – SIMPLE CONTACTS	R/B	23.13	42.21	0.84	14.21	9.46	18.54	73.08%	**0**	**0**	**0**	**0**	0.110
	R/-	23.33	42.29	0.84	14.21	9.29	18.79	74.51%	**0**	**0**	5.54	4.04	0.077
	-/B	26.13	43.38	0.81	12.83	8.13	22.42	75.93%	**0**	**0**	**0**	**0**	0.119
	-/-	26.04	43.38	0.81	12.83	8.38	22.17	75.93%	**0**	**0**	4.67	3.38	0.126
EPMH – COMPLEX CONTACTS	R/B	19.75	42.08	0.84	16.58	11.13	14.38	**61.46%**	**0**	**0**	**0**	**0**	0.110
	R/-	20.50	41.88	0.84	16.58	9.96	15.33	**62.50%**	**0**	**0**	6.58	4.54	0.086
	-/B	26.33	43.29	**0.82**	12.83	7.83	22.63	78.18%	**0**	**0**	**0**	**0**	0.116
	-/-	26.21	43.29	**0.82**	12.83	8.08	22.38	75.93%	**0**	**0**	6.04	4.21	0.076
EPMH – CENTER OF GRAVITY	R/B	23.08	42.46	0.83	14.00	10.21	18.25	64.71%	**0**	**0**	**0**	**0**	0.129
	R/-	23.88	42.29	0.83	14.00	8.88	19.42	71.15%	**0**	**0**	6.75	4.67	0.080
	-/B	26.38	43.33	0.81	**12.88**	7.63	22.83	75.93%	**0**	**0**	**0**	**0**	**0.100**
	-/-	26.67	43.17	**0.82**	12.88	7.33	22.96	75.93%	**0**	**0**	6.63	4.67	0.068
EPMH – DISTANCE SUM	R/B	23.17	42.38	0.83	14.04	10.04	18.29	66.67%	**0**	**0**	**0**	**0**	0.122
	R/-	23.71	42.25	0.84	14.04	9.29	18.92	71.15%	**0**	**0**	6.88	4.67	0.080
	-/B	26.42	43.42	0.81	**12.88**	7.79	22.75	76.36%	**0**	**0**	**0**	**0**	0.125
	-/-	26.50	43.25	0.81	12.88	7.25	23.13	75.93%	**0**	**0**	6.79	4.46	0.070

The information on algorithm and constraint configuration is contained in the first two columns. In the six successive columns, we report, respectively, the average of the total number of pallets and layers created in the solution, the average fill factor considering the 2D space of all layers, as well as the average number of single-item, single-family and residual layers. Then we report, in five additional columns, the maximum percentage of residual layers found among the instances (worst case result), the average number of items not satisfying the contiguity or the visibility constraint, the average number of layers for which at least an item does not meet the contiguity or the visibility constraint. The last column reports the computational time in seconds. To facilitate the analysis of the results, we highlight in bold the best average values for each column and constraint configuration.

Let us focus first on the performance of EPH and SLH. From Table 4.2, we can notice that disregarding rotation of the items has a considerable effect in the solution cost, with an important increase in the number of single-family and single-item layers, as well as a decrease in the number of residual layers. Between these two algorithms, SLH has a slightly better performance. This happens because the skyline structure is an efficient structure that allows one to analyze only a small part of the search space. However, this heuristic has also the disadvantage of forming empty holes depending on the sequence of the input items. We can also notice that the average 2D fill factor achieved is quite stable around 80%.

Now, let us consider the results obtained by the 8 attempted EPMH configurations. Also in this case, rotation has a relevant impact on the solution cost. As regards to the visibility constraint, we can notice a slight difference in the number of layers, considering both meeting rotation constraint and not. Concerning the runtime, the visibility constraint has a more significant influence than the rotation constraint. Comparing the results obtained by the proposed fitness evaluation functions, the Bounding Box function found the best average results, at least for the configurations R/B and R/-. The best average results for the configurations -/B and -/- were instead achieved by the Lower X function. For what concerns the number of residual layers, the values found by the EPMH with Bounding Box were considerably smaller than those obtained with other functions. All computational efforts were very low, always

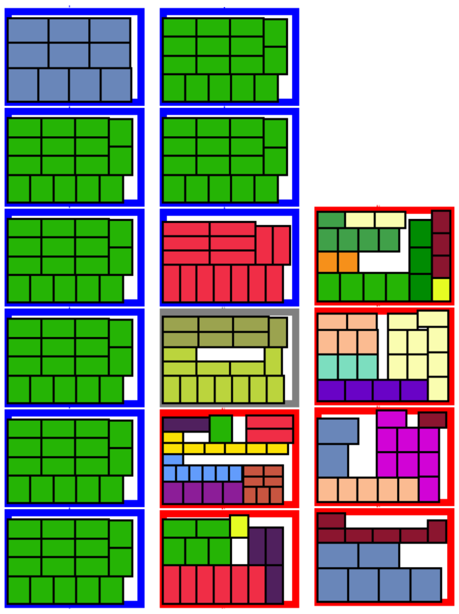


Figure 4.4: Layers for the PBP instance 2, formed by 9 single-item layers (layer border in blue), 1 single-family layer (layer border in gray), and 6 residual layers (layer border in red). Solution obtained by using the Bounding Box fitness evaluation function.

requiring less than 0.2 second on average. This is a very important detail for future research as we can expect to create more complex algorithms, maybe based on iterated executions of the proposed heuristics, without incurring in large computational efforts.

An example of a solution obtained for instance 2 is provided in Figure 4.4. We can notice 9 single-item layers, 1 single-family layer and 6 residual layers. The layers sum up to a total of 9 pallets.

4.5 Conclusions

In this chapter, we studied a pallet building problem with item rotations and practical constraints involving visibility and contiguity. We proposed a 2-step constructive heuristic that enlarges the well-known extreme points heuristic from the literature. In a first step, we generate 2D layers by considering a larger set of extreme points (candidate locations for packing) that are useful to model contiguity and visibility requirements. In the second step, we used the created 2D layers to build 3D pallets by using a simple greedy strategy. Extensive computational experiments on real-world instances proved the effectiveness of the proposed heuristic.

To evaluate the extreme points, we proposed several fitness evaluation functions, and found that the one based on the concept of Bounding Box gave better results than the other ones on average. We also analyzed the influence of some constraints (e.g., rotation and visibility) when tailoring our heuristic to basic 2D packing heuristics from the literature, gaining interesting insights in the difficulty of the real-world instances that we tested.

As future work, we intend to develop metaheuristic algorithms to try to enhance the quality of the solutions that we generated so far. We are also interested in extending the concept of family to address characteristics beyond the geometric ones. In that way, it will be possible to address more complex problems when using the contiguity and visibility constraints, as for example, the Vehicle Routing Problem (VRP). Bringing both problems together (PBP and VRP), the shared knowledge about well-defined families is useful for the whole process (packing and delivery), helping to solve both problems in a more efficient way.

Chapter 5

Reactive GRASP-based Algorithm for Pallet Building Problem with Visibility and Contiguity Constraints

5.1 Abstract

In this chapter, we study again the Pallet Building Problem (PBP) addressed in the previous one and we propose to solve it by means of a GRASP metaheuristic. The algorithm is based on the already discussed Extreme Points Modified Heuristic (EPMH) coupled with a reactive mechanism. It uses again a two-step strategy, in which items are first grouped into horizontal layers, and then layers are stacked one over the other to form pallets. The performance of the algorithm is assessed through extensive computational tests on real-world instances. The results show that the GRASP is able to create very compact packings for most of the instances with a limited computational effort.

5.2 Solution algorithm

This section presents the full technique to solve the problem addressed, which we call *Reactive GRASP with Extreme Points Modified Heuristic* (GREP). GREP produces feasible solutions by using a two-step heuristic (Section 5.2.1) inside the GRASP framework tailored to a reactive method (Section 5.2.2).

5.2.1 Two-step heuristic

GREP is based on the two-step heuristic named Extreme Points Modified Heuristic (EPMH) of Section 4.3. We have specifically adapted its deterministic parts in order to be able to include them within the GRASP framework. We detail the adaptations to the EPMH methods in the following section.

5.2.2 Reactive GRASP metaheuristic

The Greedy Randomized Adaptive Search Procedure (GRASP) is a metaheuristic originally developed by Feo and Resende [42, 43]. Roughly speaking, it is an iterative technique that mixes greedy and randomized processes with local search through a two-phase approach: first, a random greedy procedure is used to explore the solution space; then, local search procedures are invoked to improve solutions. Due to its simplicity and efficiency, the GRASP metaheuristic has been adopted to solve a large variety of problems, see, e.g., Resende and Ribeiro [90].

An original optimization heuristic can be changed to improve its efficiency without removing its original concept. To this regard, an interesting modification on the basic GRASP is to include the so-called *reactive method*, proposed firstly in Prais and Ribeiro [85]. Basically, this is a probabilistic method that selects an element among many options, where each option receives an independent probability that is adaptively modified according to the quality of the solutions previously found. This approach increases the robustness of the original GRASP metaheuristic, with the advantage of not requiring calibration efforts.

The *Reactive GRASP with Extreme Points Modified Heuristic* for the PBP proposed is described in Algorithm 1 (available in Appendix). Let us consider the input variables: I: item type set; L: pallet set; D: criteria set for selecting extreme points; $order\_family, order\_type$: criteria to sort families and item types; ϵ: randomness percentage; ϱ: reactive parameter; β: upper bound decrease percentage; ψ: number of iterations to update probabilities; and $\max_{\kappa}$: maximum number of rejected solutions. First, the required variables are initialized (line 2) and the item type set is sorted (line 3). Then, as long as the stopping condition is not met (i.e., a maximum execution time is not reached, line 4), the algorithm selects a fitness evaluation function (line 5) to construct a feasible solution to the problem (line 8). The solution is evaluated (line 9) according to a fitness function, to be maximized. We will use the fitness function (5.1), discussed later on in Section 5.2.2.3. The fitness of the solution is compared to the current reference value U (line 10): if the solution fitness is higher than U, then the algorithm tries to improve the solution by local search (line 12); otherwise, the local search is not performed and the reference value is decreased after a prefixed number of iterations without an improvement (lines 17-19). Lines 20-29 refer to the reactive method: the best (line 20), worst (line 23) and mean (lines 25-26) fitness values are possibly updated; next, after a prefixed number of iterations (line 27), all fitness values are re-evaluated (line 28) and the probability functions of the reactive method (which are used in the constructive phase, as discussed next) are updated (line 29).

In the remaining part of this section, we give full details on the procedures invoked by GREP and on the way parameters are used and updated.

5.2.2.1 Constructive phase

In this phase, we select one item at a time and pack it in an extreme point (if any), until a layer has been created. We reiterate until all items have been packed into layers, and then stack the layers into pallets.

Creating layers

Choosing an item

The choice of a new element is flexible in GRASP algorithms, since not necessarily the current best element, with respect to a given criterion, is selected, but rather a restricted candidate list of items is considered and the next element is randomly selected from such list. This concept is easily adapted to the deterministic method of selecting the next item described for the layer creation heuristic (Section 4.3.1.4) and leads to a greedy randomized heuristic that works as follows.

First, the algorithm establishes a specific order for the families and item types, assigning them to the data structure S. Then, it defines the selectable range based on a parameter ϵ. Let F_S be the sorted set of families representing the order of the families in S, and $G_f \subseteq S$ the sorted set of item types $s \in S$ representing the order of the item types of a family $f \in F_S$. A family f is selected randomly among the first $\epsilon|F_S|$ families, and then an item type $s \in G_f$ is selected among the first $\epsilon|G_f|$ item types.

Choosing an extreme point

Once an item of type $s \in S$ has been selected for packing, the heuristic determines in which extreme point it should be placed. This is again obtained by performing an adaptation in the greedy heuristic of Section 4.3.1.1 that takes into consideration ϵ. Let T be the set of all extreme points of E_f for the item type s. Each extreme point in T is evaluated according to a fitness function d. Then, the heuristic sorts all points in T by decreasing value of fitness, and randomly selects the extreme point where to pack the current item among the first $\epsilon|T|$ extreme points of T, i.e., the parameter ϵ stands for the percentage of the best extreme points of E_f that can be chosen randomly.

Reactive method

In this process, a key role is played by the fitness functions used to evaluate the quality of the assignment of an item to an extreme point. The reactive method that we adopted serves indeed to this aim. We use an overall set D, which is composed by different fitness evaluation functions, namely: Lower x; Lower y; Bounding Box; Bounding Square; Simple Contacts; Complex Contacts; Distance Sum; Center of Gravity (Section 4.3.1.2 for details).

The reactive method works as follows: given the fitness evaluation function set

D and the probability set P_d to choose in the current iteration the function $d \in D$ associated with the item type $s \in S$, the method initializes P_d to $1/|D|$ and updates these values after a given number ψ of iterations has been elapsed, considering the quality of the generated solutions. In particular, probabilities P_d are updated at line 29 of Algorithm 1. Note that the better the partial solution found by d is, the greater becomes the probability P_d to select d in the next iterations.

Layer creation and classification

The rest of the heuristic works exactly as defined previously (Section 4.3.1.4), and the algorithm continues until all items have been packed into layers.

Building pallets

After the layers have been created, GREP stacks them in the minimum number of pallets possible by using the constructive heuristic for pallet building presented in Section 4.3.2, concluding this constructive phase. It is important to highlight that the exploration of the search space provided by GRASP metaheuristic is adapted explicitly to the first step of the two-step heuristic (layer creation) being applied to the problem, since the remaining step (pallet building) is relatively simpler to solve.

5.2.2.2 Improvement phase

The improvement phase is based on a local search that tries to improve the quality of the solution generated in the constructive phase by applying small changes to it. This is not always an easy task, since in view of the hard PBP constraints, any change in the solution may create infeasibility. For example, in the PBP it is difficult to change the position of individual items without violating contiguity and visibility constraints. Therefore, we opted to generate new solutions from scratch, by applying the small local search changes directly to the initial order of S provided to the constructive heuristic.

In detail, we first randomly select two families f_a and f_b (if any) and switch the order of their items in S. Then, for each family $f \in F$, we randomly select two item types i_c and i_d (if any) in f and switch the order of their items in S. These changes are applied according to a *First Improvement Strategy* (FIS): considering the original data structure S, a change is applied as soon as it improves the quality of the solution, otherwise it is disregarded and a new one is generated, until a maximum number of iterations Ω is reached. It is worth highlighting that, if a change does not improve the solution, then the next change is applied to the original structure S, and not to the modified one.

Despite the improvement phase constitutes an essential part of the GREP, it is not always carried out, since low-quality starting solutions tend to produce low-quality final solutions. Thus, as shown at line 10 of Algorithm 1, we consider a threshold U to decide if it is worth looking for an improvement. The value of U is initialized with the value of the first solution created. Whenever a new solution has a fitness better than U, the improvement phase is performed and U is updated.

To avoid stagnation during the process, GREP controls U in the following way (lines 16-19 of Algorithm 1): if the fitness of a new solution does not reach U, then GREP increases an iteration counter κ that enumerates the number of consecutive solution rejections. Whenever κ gets larger than a maximum number of rejected solutions ($\max_\kappa$), U is decreased to βU ($\beta \in \mathbb{R}, 0 < \beta < 1$) and κ is reset to 0. Thus, the algorithm forces the improvement phase for solutions which are not necessarily the best ones but are of "good enough" quality. We also highlight that, as soon as a new solution with fitness higher than U is reached, U is set to this new fitness value and κ is reset to 0 (lines 11 and 14 of Algorithm 1).

5.2.2.3 Evaluating a solution

A basic strategy to compare solutions is to evaluate them on the basis of the number of pallets they produce. However, due to the complexity of the PBP with contiguity and visibility constraints, taking only this value into account may not be appropriate. Analyzing the structure of a solution, it is possible to extract more interesting information, and these peculiarities can help explore the search space. To this regard, let n_r be the number of residual layers, n_l the total number of layers, n_p the total number of pallets, and f_l the average fraction of the 2D occupation of the layers in a solution.

Obviously, a good solution is a solution with small n_r and n_p values and with large f_l value. Thus, GREP calculates the fitness function of a solution as:

$$V = \frac{v_{residual} + v_{pallets} + f_l}{3}, \tag{5.1}$$

where $v_{residual}$ and $v_{pallets}$ are logarithmic functions that consider the proportion of, respectively, n_r and n_p over n_l, being defined as

$$\begin{aligned} v_{residual} &= -\log_{n_l+1} \frac{n_r + 1}{n_l + 1}, \\ v_{pallets} &= -\log_{n_l+1} \frac{n_p + 1}{n_l + 1}. \end{aligned}$$

All components of the summation in (5.1) are in the range between 0 and 1, and V is their average value, which we would like to maximize. The idea behind the use of logarithmic functions is to perform evaluations that are more sensitive to variations in the numbers n_r of residual layers and n_p of pallets when these numbers are close to 0. To clarify this point, we can consider Figure 5.1. Such figure shows the behavior of $v_{residual}$ as a function of n_r, when considering $n_l = 70$ layers in total. We notice that the logarithmic curve decreases as n_r increases and has larger derivative at small values of n_r (value $v_{pallets}$ as a function of the number of pallets n_p behaves in a completely similar way).

5.3 Computational results

All experiments have been conducted on a Virtual Machine VMware®, Intel® Xeon® CPU E5-2640 v2 2.00GHz, 16GB RAM, Ubuntu Server 18.04 OS. The algorithms have been implemented in Java and ran using Oracle® JDK 11.

We used GREP (Section 5.2) to solve the same 24 instances presented in Section 4.4.1.

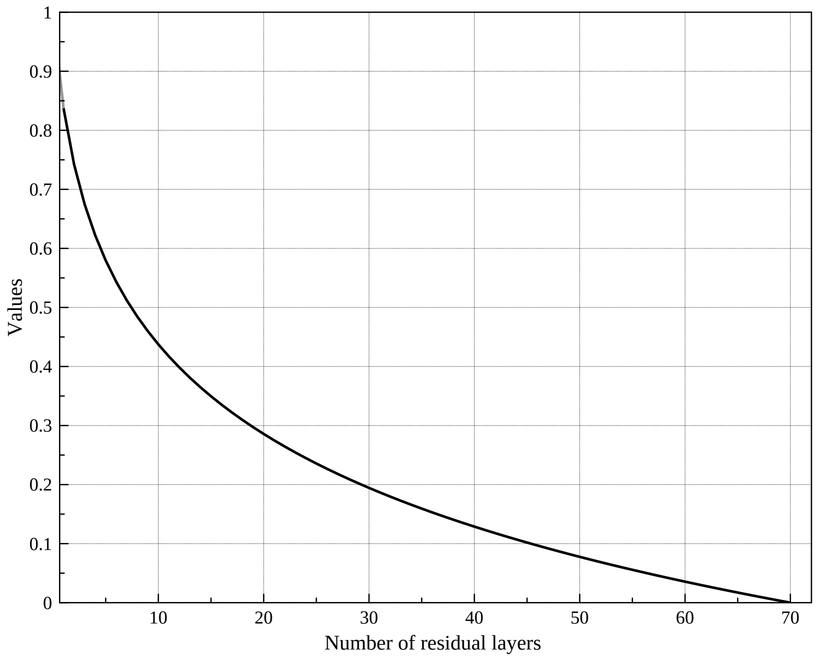


Figure 5.1: Behavior of the values for the logarithmic function which represents the variable $v_{residual}$ when considering the increase/decrease in the number of residual layers for a total of 70 layers.

5.3.1 Parametric configurations

Regarding GREP, we set its time limit to 5 minutes for each instance, running the algorithm 10 times. It is worth noting that the time limit was defined based on a simple parametric configuration test by considering a longer execution time. In the end, this value was considered more suitable because most of the improvements to the solution happened within this time range.

For all instances, the fill factor and the values for container dimensions are similar to those defined in Chapter 4. We tested a number of configurations, including allowing or not allowing rotation of 90 degrees of the items and the criteria to sort the items in the initial structure S. In this case, S is sorted by random order for families, and a non-increasing order of width for item types, since these parameters showed a better performance when comparing to the results of other sorting criteria.

Additionally, we carried out a simple test for parametric configuration of ϵ, Ω, and β using a restrict set of values, and we have chosen $\epsilon = 0.15$, $\Omega = 25$, and $\beta = 0.98$. The remaining parameters are $\psi = 500$, $\varrho = 10$, and $\max_{\kappa} = 5$, as suggested in Alonso et al. [5].

5.3.2 Evaluation

We report the individual and average results for GREP to analyze its performance. Besides that, we show the best average results obtained in Section 4.4.3 over the same instances, in order to compare the performance between both strategies in a consistent way. These results are summarized in Table 5.1.

Table 5.1 reports: algorithm; instance (average or individual ones), total number of pallets in the solution; total number of layers created; in the six successive columns it reports the minimum, maximum and average pallets utilization ('*Pallet filling (%)*') and layers utilization ('*2D fill factor (%)*'), respectively; average number of single-item, single-family and residual layers; maximum percentage of residual layers found among the instances (worst case result); minimum, maximum and average values of the objective function; total number of iterations; total number of local searches; ratio between the total number of local searches and the total number of iterations. Considering GREP, each row provides the average individual results after 10 iterations, except the last line which reports the average among all individual results, excluding the value for the column '*Max percent. RL*', which stands for the largest value among all. Considering EPMH, the respective row reports the best average among all individual results when using the Bounding Box function, meeting all constraints used in this work, i.e., those ones meeting contiguity, rotation (R), and visibility (B) constraints. We highlight in bold the best average values for each column.

From Table 5.1, considering the average results, we notice that the number of layers in EPMH is slightly lower than in GREP, but the number of pallets is considerably larger. Thus, we highlight that the minimization of the number of layers is less relevant than the way in which these are created, since the overall number of pallets depends on the disposition of items within the layers. Considering layer type fields, we observe

Table 5.1: Computational results (individual and average).

Alg.	Inst.	N. of pallets	N. of layers	Pallet filling (%)			2D fill factor (%)			Single-item layers	Single-family layers	Resid. layers (RL)	Max. perc. RL(%)	Objective funciton			Total iterat. (TI)	Tot. l. search. (LS)	Ratio LS/TI (%)
				Min	Max	Avg	Min	Max	Avg					Min	Max	Avg			
GREP	01	11.0	46.0	44	84	73	55	94	83	32.1	12.9	1.0	2.2	0.668	0.668	0.668	6984	487	6.97
	02	6.4	16.0	14	76	40	76	89	82	9.8	2.1	4.1	25.6	0.500	0.522	0.513	29773	1637	5.50
	03	6.0	27.0	65	89	78	66	95	84	17.0	9.0	1.0	3.7	0.682	0.682	0.682	19551	1047	5.36
	04	3.0	6.0	12	59	28	64	92	79	1.0	2.0	3.0	50.0	0.456	0.456	0.456	45276	2561	5.66
	05	6.0	29.3	35	85	70	78	94	87	20.3	7.0	2.0	6.8	0.654	0.661	0.659	18060	1382	7.65
	06	29.0	108.0	13	89	77	69	98	87	100.0	4.0	4.0	3.7	0.600	0.600	0.600	3448	307	8.89
	07	12.2	52.0	43	88	75	71	93	86	36.3	13.7	2.0	3.9	0.639	0.645	0.644	10153	827	8.15
	08	22.4	91.1	20	85	66	68	95	86	61.7	15.5	13.9	15.3	0.521	0.526	0.523	4749	577	12.14
	09	4.9	16.0	23	86	63	74	96	86	8.6	5.4	2.0	12.5	0.613	0.634	0.615	13611	1316	9.67
	10	9.0	24.0	13	77	42	71	94	82	9.4	8.6	6.0	25.0	0.499	0.499	0.499	11022	779	7.07
	11	7.0	19.0	11	85	46	76	96	85	7.6	8.4	3.0	15.8	0.566	0.566	0.566	10075	795	7.89
	12	14.8	42.0	11	84	47	62	95	84	16.3	17.6	8.1	19.3	0.502	0.512	0.506	6745	653	9.68
	13	8.7	33.0	24	88	66	69	95	86	13.7	14.7	4.6	13.9	0.565	0.593	0.575	12603	820	6.50
	14	10.0	32.2	11	87	51	69	98	83	13.6	13.6	5.0	15.5	0.542	0.547	0.546	6946	526	7.57
	15	11.1	48.0	47	85	74	68	95	85	20.7	23.3	4.0	8.3	0.593	0.599	0.599	9304	709	7.62
	16	12.1	50.0	54	85	72	72	97	87	22.9	19.4	7.7	15.4	0.552	0.562	0.554	8637	711	8.24
	17	6.0	16.0	12	82	42	70	96	83	2.8	10.2	3.0	18.8	0.552	0.552	0.552	15303	1217	7.95
	18	12.6	38.1	11	89	50	71	98	85	13.4	16.8	7.9	20.7	0.511	0.518	0.515	5333	490	9.18
	19	12.0	50.1	32	86	69	75	96	86	19.7	24.6	5.8	11.6	0.568	0.591	0.574	6465	530	8.21
	20	13.6	55.2	16	87	69	58	95	86	25.6	23.3	6.3	11.4	0.564	0.575	0.568	5897	518	8.79
	21	12.1	53.0	44	85	70	68	95	85	20.6	26.5	5.9	11.1	0.572	0.579	0.573	5693	418	7.35
	22	15.2	59.6	15	84	63	63	95	84	28.8	22.8	8.0	13.4	0.537	0.552	0.543	4468	387	8.66
	23	10.7	44.9	27	86	69	67	95	85	13.2	26.4	5.3	11.8	0.564	0.585	0.576	6204	434	6.99
	24	12.5	46.3	12	84	61	58	95	84	16.6	22.8	6.9	14.9	0.541	0.559	0.544	6125	465	7.60
	AVG	**11.2**	41.8	25	84	61	68	95	85	**22.2**	**14.6**	**5.0**	**50.0**	0.565	0.574	0.569	11351	816	7.89
EPMH	AVG	14.5	**41.0**	-	-	-	-	-	**86**	21.8	11.6	7.6	66.7	-	-	-	1	-	-

an increase in the number of single-item and single-family layers, while the number of residual layers is reduced, when using GREP. About the quality metric that analyze the worst case result (‘*Max percent. RL*’), the reported value is significantly better when using GREP (at most 50%). Regarding to execution time, EPMH carried out in 104 milliseconds (not reported in that table), in contrast to 5 minutes for GREP.

Considering the individual results of GREP from Table 5.1, let us focus first on the utilization of layers and pallets. Regarding the 2D fill factor, this value is quite stable (85% in average). In contrast, this behaviour is quite different when we analyze the pallet filling, in which the values fluctuate between 25% and 84% in average.

For what concerns the GRASP metaheuristic, we highlight the influence of the strategy to readapt the UB value. As the UB value is related to the local search process, it is interesting to adjust it according to the quality and time spent to find a new solution. In particular, the rate of change of the UB value is quite important: large and frequent variations of UB tend to favor a larger number of local searches, while small and not frequent variations tend to reduce the number of local searches. Thus, we balance this trade-off using a small (2%) but fast (after 5 iterations without improvement) decrement in UB, which contributes to both exploitation and exploration of the search space. In the end, GREP carried out on average a percentage of local searches equal to 8%.

5.4 Conclusions

In this chapter, we proposed an extension of the work proposed in Chapter 4 based on the GRASP metaheuristic with reactive method when applying the Extreme Points Modified Heuristic (EPMH) for creating 2D layers and the Greedy heuristic for creating 3D pallets. Regarding the reactive method, it allows a more flexible search on the solution space when using a set of fitness evaluation functions in the EPMH to find a new place to pack items. Additionally, the GRASP upper bound updating strategy is essential to control the exploration and exploitation of the search space, and, consequently, to improve the solution quality. Extensive computational experiments on real-world instances reported good final results, thus showing that the algorithm proposed is an efficient strategy for this type of problem.

As future work, we intend to develop other types of local searches to try to enhance the quality of the solutions generated so far. Additionally, we intend to study extensions of the GRASP metaheuristic (Resende and Ribeiro [90]), as well as to consider the recently developed fixed set search metaheuristic (Jovanovic et al. [60], Jovanovic and Voß [61, 62]), and the most recent improvement strategies from the literature that address learning mechanism, such as Support Vector Machine.

Chapter 6

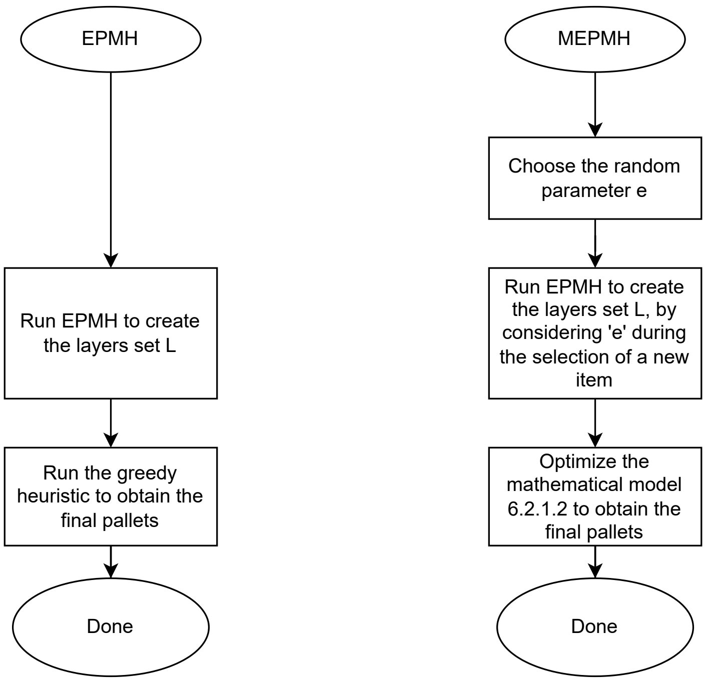
A Mixed Approach for Pallet Building Problem with Practical Constraints

6.1 Abstract

We study the Pallet Building Problem (PBP) with contiguity and visibility constraints addressed in the previous chapters and we propose to solve it by extending the Extreme Points Modified Heuristic (EPMH) (Chapter 4.3) by invoking an exact method to stack layers one over the other to form pallets. The performance of the algorithm is assessed through extensive computational tests on real-world instances. The results show that the exact model considerably increases the solution quality, creating very compact packings with a limited computational effort.

6.2 Solution algorithm

This section presents the full technique that we developed to solve the PBP with visibility and contiguity constraints, which we call *Mixed Extreme Points Modified Heuristic* (MEPMH). MEPMH produces feasible solutions by using a two-step heuris-


EPMH
Run EPMH to create the layers set L
Run the greedy heuristic to obtain the final pallets
Done
MEPMH
Choose the random parameter e
Run EPMH to create the layers set L, by considering 'e' during the selection of a new item
Optimize the mathematical model 6.2.1.2 to obtain the final pallets
Done

(a) EPMH framework (b) MEPMH framework

Figure 6.1: Frameworks of the constructive approaches for the PBP: (6.1a) EPMH is an algorithm based on deterministic methods to create layers and pallets; (6.1b) MEPMH is an algorithm based on methods to: (*i*) create layers by using random choices, and (*ii*) build pallets by solving a mathematical model.

tic: first, it creates layers to deal with individual items separately using a heuristic method (Section 6.2.1.1), forcing the presence of packing, fill factor, visibility, and contiguity constraints; second, the pallet generation step tries to minimize the number of pallets using an exact method (Section 6.2.1.2), forcing the presence of stackability and layers sequence constraints. Figure 6.1 shows the frameworks for both the EPMH and MEPMH in order to outline the modifications introduced in the algorithm proposed.

6.2.1 Two-step heuristic

MEPMH is based on the Extreme Points Modified Heuristic (EPMH) described in Section 4.3. We have replaced the layer stacking heuristic method with an exact algorithm, as well as modified the layer creation method in order to test the effectiveness of random choices in the deterministic process while making it more flexible.

6.2.1.1 Creating layers

This section corresponds to the EPMH methods to create layers (Section 4.3.1). We include slight modifications to a few of them, that are detailed below.

The first modification is to the evaluation function used. In this case, function d, the *Bounding Box function*, as suggested in Chapter 4 thanks to its superior performance over the results obtained with other fitness evaluation functions. In summary, this function calculates the bounding box area – minimum rectangle area covering a set of items – with k and finds the extreme point e that minimizes the area among all bounding boxes generated when positioning the reference point of k over e.

The next modifications increases the search space by adding unpredictability to the process. The first modification addresses the selection of the next item. This process has already been described and applied in Section 5.2.2.1, in which it uses the structure S composed of sorted items, and the available range over S is based on the parameter ϵ.

The final modification includes the possibility of selecting extreme points that are not the most suitable when basing on the greedy rank. As before, the selection of an extreme point also uses the parameter ϵ to include unpredictability to the process. This modification has been described in Section 5.2.2.1.

6.2.1.2 Creating pallets

The proposed mathematical model deals with the insertion of the generated layers into pallets in such a way that the number of pallets is minimized. In simple words, this mathematical model “sorts” the layers in pallets as described in Section 2: a pallet is made up by single-item layers in its base (if any), single-family layers in the middle (if any), and, possibly, by a residual layer in its top part. This practical constraint is defined to simplify the problem avoiding the addition of the stability constraint, since this structure generates more stable pallets when compared to the use of shuffled layer types.

For the sake of clarity, we present the following notation for data of the mathematical model: P stands for the set of pallets; R stands for the set of residual layers; I stands for the set of single-item layers; F stands for the set of single-family layers; $M = I \cup F$; $L = R \cup M$; H stands for the height of the pallet; h_t stands for the height of layer t; and S_t stands for the stackability of layer t. We make use of two families of binary variables:

x_{tp} : it is equal to 1 if layer $t \in L$ is packed in pallet $p \in P$, 0 otherwise;

y_p : it is equal to 1 if pallet $p \in P$ is used, 0 otherwise.

Then, we obtain the following mathematical model:

$$\begin{aligned}
&\text{Minimize} \sum_{p \in P} y_p && && (6.1)\\
&\text{subject to} \sum_{t \in L} h_t x_{tp} \le H y_p && p \in P && (6.2)\\
&\sum_{p \in P} x_{tp} = 1 && t \in L && (6.3)\\
&\sum_{t \in R} x_{tp} \le y_p && p \in P && (6.4)\\
&\sum_{t' \in M : S_{t'} < S_t} x_{t'p} \le |M|(1 - x_{tp}) && t \in R,\ p \in P && (6.5)\\
&\sum_{t' \in I : S_{t'} < S_t} x_{t'p} \le |I|(1 - x_{tp}) && t \in F,\ p \in P && (6.6)\\
&x_{tp} \in \{0, 1\} && t \in L,\ p \in P && (6.7)\\
&y_p \in \{0, 1\} && p \in P. && (6.8)
\end{aligned}$$

The objective function (6.1) aims at minimizing the number of used pallets. Constraints (6.2) ensure that the sum of the heights of all layers in p is not higher than H if pallet p is used. Constraints (6.3) ensure that each layer is placed in exactly one pallet. Constraints (6.4) ensure that at most a residual layer $t \in R$ can be packed in a pallet $p \in P$ (namely, at the top of the pallet). Constraints (6.5) ensure that, for each layer $t \in R$, the layers $t' \in M$ with stackability lower than t cannot be placed on the same pallet with t (recall that t is placed on top of all other layers and, thus, its stackability cannot be larger than that of the other layers in the pallet). Constraints (6.6) are similar to the previous ones, but we have the correspondence between single-item and single-family layers in the pallet, by requiring that all stackability values of the single-item layers are greater than the stackability values of the single-family layers (since the former are always placed below the latter). Note that stackability constraints should also be imposed between each pair of single-item or single-family layers. However, once we have established which single-item (single-family) layers are placed on the pallet, we only need to order them according to their stackability value. Constraints (6.7) and (6.8) are binary conditions for the variables.

As described in Chapter 2, we highlight that the height of each layer is the height of the highest item in the layer. Similarly, the stackability of a layer is computed as the maximum stackability value among the items in the layer. This exact algorithm concludes the 2-step heuristic and allows us to have a feasible solution.

6.3 Computational results

We used MEPMH (Section 6.2) to solve the same 24 instances presented in Section 4.4.1.

All experiments have been conducted on a Virtual Machine VMware®, Intel Xeon® CPU E5-2640 v2 2.00GHz, 16GB RAM, Linux Ubuntu Server 18.04 OS. The algorithms have been implemented in Java and ran using Oracle® JDK 11. Additionally, we used the solver IBM ILOG® CPLEX 12.10 to solve the mathematical model.

6.3.1 Parametric configurations

The algorithm uses an initial structure S sorted by a random order for families, and a non-increasing order of width for item types. For each instance, we ran the algorithm 15 times. The time limit for the solver was set to 30 seconds. Additionally, the container dimensions were set equal to 1500, 1250, and 1050 for height, width, and length, respectively. We tested them, by allowing rotation of 90 degrees of the items.

We carried out a detailed test for selecting the value of both fill factor and parameter ϵ. About the fill factor, we have tested values $\{0.55, 0.65, 0.75\}$; these values define a threshold that represents the minimum value of occupation for defining a residual layer. In this way, as long as the final layers present a fill factor that is able to create stable pallets, a proper adjustment of this parameter might allow us to generate more single-item and single-family layers, increasing the quality of the final solution as a result. About parameter ϵ, we have chosen the value set $\{0.00, 0.05, 0.15, 0.25\}$. In this case, the larger the value is, the more random the process is.

6.3.2 Experimental evaluation

We report the average results for MEPMH to analyze its performance, considering all the previous configurations. Besides that, we show the average results obtained by EPMH, which uses the greedy algorithm to create pallets, as described in Chapter 4. We have conducted these experiments over the same instances to compare the performance of the two strategies consistently. It is worth saying that the layer solution is always constructed from scratch however, since the random number generator was reset for each new test, the resulting number of layers may be slightly different.

The average results for both MEPMH and EPMH are summarized in Table 6.1. In this table, each row provides average results for a given algorithm on the 360 solutions obtained, 15 for each of the 24 attempted instances. The information about the algorithm is contained in the first column, followed by the minimum fill factor to create a layer, the choice factor (value of ϵ), the total number of pallets in the solution and the total number of layers created. In the six successive columns, we

Table 6.1: Computational results (individual average).

Algorithm	Min. fill factor	Choice factor (ϵ)	N. of pallets	N. of layers	Pallet filling (%)			2D fill factor (%)			Single-item layers	Single-family layers	Resid. layers (RL)	Time (sec.mil)
					Min	Max	Avg	Min	Max	Avg				
EPMH	0.75	0.00	12.84	40.93	16	85	53	41	96	86	21.79	11.53	7.61	0.016
		0.05	12.79	40.92	17	85	53	42	96	86	21.79	11.53	7.59	0.016
		0.15	13.81	42.29	13	82	49	41	95	83	21.36	12.20	8.73	0.020
		0.25	18.49	42.93	9	**80**	37	39	94	82	18.13	10.46	14.35	0.023
	0.65	0.00	11.41	41.55	22	84	59	36	96	84	22.79	14.00	4.76	0.015
		0.05	11.40	41.57	22	84	59	37	96	84	22.79	14.03	4.74	**0.015**
		0.15	11.91	42.96	16	**83**	57	40	95	82	23.05	15.11	4.80	0.017
		0.25	13.06	44.46	12	**80**	52	39	94	79	23.72	14.25	6.49	0.018
	0.55	0.00	11.26	41.81	18	85	59	41	96	84	23.13	15.00	3.68	**0.014**
		0.05	11.30	41.79	18	85	59	41	96	84	23.13	14.99	3.68	**0.015**
		0.15	11.48	43.25	20	**83**	59	41	95	81	23.42	16.20	3.64	**0.016**
		0.25	11.94	45.43	20	**80**	56	35	94	78	26.15	15.44	3.84	**0.017**
MEPMH	0.75	0.00	12.36	40.94	16	85	55	41	96	86	21.79	11.52	7.63	12.114
		0.05	12.29	40.91	17	84	56	43	96	86	21.79	11.51	7.61	11.133
		0.15	13.36	42.28	15	81	51	41	95	83	21.33	12.11	8.84	13.731
		0.25	17.96	42.98	10	79	39	38	95	82	18.26	10.44	14.28	13.796
	0.65	0.00	10.75	41.56	27	**86**	63	36	96	84	22.79	14.00	4.76	9.157
		0.05	10.74	41.57	28	**86**	63	38	96	84	22.79	14.03	4.74	9.562
		0.15	11.16	42.98	23	83	61	42	95	82	23.17	15.00	4.81	10.672
		0.25	12.39	44.45	17	**80**	55	40	94	79	23.44	14.53	6.48	12.723
	0.55	0.00	**10.36**	41.80	**30**	**86**	**66**	41	96	84	23.13	15.00	3.68	8.188
		0.05	**10.38**	41.77	**30**	85	**65**	43	96	84	23.13	14.99	3.65	8.719
		0.15	**10.81**	43.35	**27**	**83**	**63**	37	94	81	23.44	16.19	3.72	9.426
		0.25	**11.24**	45.38	**28**	79	**60**	37	94	78	26.08	15.45	3.85	9.417

report, respectively, the minimum, maximum, and average pallet utilization and the minimum, maximum, and average fill factor, considering the 2D area of all layers. Then, the average number of single-item, single-family and residual layers, and the computational time (represented by seconds.milliseconds). About the EPMH, we reported the best results that met rotation, contiguity and visibility constraints (all constraints used in this work), using the Bounding Box function.

The first comment on Table 6.1 is about the number of created layers. As EPMH and MEPMH use the same algorithm to create layers, the 2D fill factor and the number of layers are very similar, showing only slight differences due to the randomness of the process provided by the choice factor. Besides that, the distribution among layers (single-item, single-family and residual) presents the same similarities.

Although the creation of layers is similar for both algorithms, the analysis of the parameters of minimum fill factor and choice factor is critical. Let us focus first on the analysis of the minimum fill factor. Here, the lower is this value, the higher the improvement in the quality of the solution is. Therefore, we notice an interesting situation: even though the algorithm uses the smallest value for the minimum fill factor (0.55), the final average and minimum 2D fill factor remain quite similar to the values obtained when we use the highest value for this parameter (0.75). This is an important consideration, because, notwithstanding a company needs a minimum occupation of a layer and even when we allow the algorithm to use a value for this parameter that is lower than this threshold, the algorithm still creates better solutions maintaining the stability of the pallets. Of course, a layer with a small fill factor may cause instability if another layer is posed on the top of it. In this case the occupation of the layer can be increased by so called filler boxes, which allow to increase the occupation area and, thus, the stability.

Considering the choice factor, this parameter presents a general behavior: a more significant improvement in the quality of the solution made by the configurations 0.05 and 0.15. It is important to highlight that the higher the values for this parameter, the larger the algorithm’s randomness. Therefore, analyzing the results, we notice that high randomness brings a broad exploration, but the algorithm is not able to find good-quality solutions.

Last but not least, let us consider the part of the algorithm to create pallets. In this scenario, we can notice substantial differences. The first one is about the execution time. In this case, the execution time of MEPMH is greater than the execution time of EPMH, since the mathematical model in MEPMH is more complex to solve than the respective heuristic algorithm in EPMH. However, the final execution time of

MEPMH leveled out between 8.188 and 13.796 seconds on average, thus showing a high performance, that can be easily accepted for an industrial software application.

The most important analysis when considering the final solution is about the comparison of the final number of pallets between the two algorithms. In this case, we can notice the improvement that MEPMH adds to the algorithm. To understand this comparison, we highlight the respective values in Table 6.2.

Table 6.2: Improvement in solution when using MEPMH.

Min. fill factor	Choice factor (ϵ)	Improvement (%)
0.75	0.00	3.74
	0.05	3.91
	0.15	3.26
	0.25	2.87
0.65	0.00	5.78
	0.05	5.79
	0.15	6.30
	0.25	5.13
0.55	0.00	7.99
	0.05	8.14
	0.15	5.84
	0.25	5.86

More in detail, Table 6.2 shows the minimum fill factor to create a layer, the choice factor (value of the parameter ϵ), and the percentage of improvement that MEPMH presented over EPMH, i.e., the percentage reduction of the number of pallets when comparing both algorithms. In this table, we can notice the improvement leveling out between 2.87% and 8.14%. Therefore, in the same line as in the previous analysis, the greatest improvements occur when the algorithm uses the lowest value for the minimum fill factor (0.55) and low values for the choice factor (0.05 and 0.15).

6.4 Conclusions

This chapter addressed a Pallet Building Problem from a case study in a company by considering practical constraints, such as rotation of the items, fill factor, stackability, visibility, and contiguity constraints. We proposed an extension of the work in Chapter

4, using a modification of a 2-step heuristic that includes a mathematical model to create pallets. We use the efficient Extreme Points Modified Heuristic (EPMH) for creating 2D layers, including in the algorithm the exact method to increase the quality of the solution to create 3D pallets. To analyze the effectiveness of the proposed heuristic, we carried out extensive computational experiments on real-world instances, testing a set of parameter values for the minimum fill factor of a layer, as well as for the random choice factor used to choose the next item to be packed. Despite the complexity of the real-world instances, good final results are reported in a short execution time, thus showing that the inclusion of the mathematical model in the algorithm is an efficient strategy to create compact solutions for this type of problems.

As future work, we intend to create matheuristics by means the EPMH and an exact mathematical model to solve the layer creation to try to enhance the quality of the solutions generated. Additionally, we are interested in proposing formal mathematical models to express the concept of contiguity and visibility of items, thus filling a gap in the existing literature.

Chapter 7

Mathematical Models and Heuristic Algorithms for Pallet Building Problems with Practical Constraints

7.1 Abstract

In this paper, we conclude our study in the Pallet Building Problem (PBP) with contiguity and visibility constraints addressed in the previous chapters. We present a formal mathematical description for layer and pallet creation subproblems and then we propose new algorithms, made up by combining heuristics, metaheuristic, and mathematical model to solve the overall problem. The performance of the algorithms is assessed through extensive computational tests on real-world instances.

7.2 A MILP formulation

While a mathematical programming model for the whole problem would be possible, its dimension would be quite large and impossible to solve within reasonable computing times. For this reason, we define two separate Mixed-Integer Linear Programming (MILP) formulations, one for the creation of single layers from a selected set of items, and one for the creation of pallets once the layers are given.

7.2.1 MILP formulation to create layers

Let $\mathcal{G}$ be the set of items that can be inserted into a given layer. Let m be the number of different item types in $\mathcal{G}$ and, for each item type i, let b^i be the number of items of type i in $\mathcal{G}$. The proposed mathematical model finds a feasible disposition for items in $\mathcal{G}$ by meeting the following constraints.

Non overlap

Let us define $\Phi_{\mathcal{G}} = \{(i, p, j, q) : i, p = 1, \dots, m, j = 1, \dots, b^i, q = 1, \dots, b^p, i > p \vee (i = p \wedge j > q)\}$. The problem of determining if the items of the set $\mathcal{G}$ can fit into a layer without overlapping can be modeled as follows:

$$\sum_{u=1}^{4} o^{ip}_{jqu} \geq 1 \qquad (i,p,j,q) \in \Phi_{\mathcal{G}} \tag{7.1}$$

$$x^p_{2q} - x^i_{1j} \leq W(1 - o^{ip}_{jq1}) \qquad (i,p,j,q) \in \Phi_{\mathcal{G}} \tag{7.2}$$

$$x^i_{2j} - x^p_{1q} \leq W(1 - o^{ip}_{jq2}) \qquad (i,p,j,q) \in \Phi_{\mathcal{G}} \tag{7.3}$$

$$y^p_{2q} - y^i_{1j} \leq L(1 - o^{ip}_{jq3}) \qquad (i,p,j,q) \in \Phi_{\mathcal{G}} \tag{7.4}$$

$$y^i_{2j} - y^p_{1q} \leq L(1 - o^{ip}_{jq4}) \qquad (i,p,j,q) \in \Phi_{\mathcal{G}} \tag{7.5}$$

$$x^i_{2j} = x^i_{1j} + (1 - r^i_j)w^i + r^i_j l^i \qquad i = 1, \dots, m, j = 1, \dots, b^i \tag{7.6}$$

$$y^i_{2j} = y^i_{1j} + r^i_j w^i + (1 - r^i_j) l^i \qquad i = 1, \dots, m, j = 1, \dots, b^i \tag{7.7}$$

$$0 \leq x^i_{1j} \leq x^i_{2j} \leq W \qquad i = 1, \dots, m, j = 1, \dots, b^i \tag{7.8}$$

$$0 \leq y^i_{1j} \leq y^i_{2j} \leq L \qquad i = 1, \dots, m, j = 1, \dots, b^i \tag{7.9}$$

$$r^i_j \in \{0, 1\} \qquad i = 1, \dots, m, j = 1, \dots, b^i \tag{7.10}$$

$$o^{ip}_{jqu} \in \{0, 1\} \qquad (i,p,j,q) \in \Phi_{\mathcal{G}}, u = 1, \dots, 4 \tag{7.11}$$

For item j of type i, variables x^i_{1j} and y^i_{1j} represent the minimum x and y coordinates, respectively, while x^i_{2j} and y^i_{2j} represent the maximum x and y coordinates. The binary variable r^i_j defines the orientation for item j of type i. It is equal to 1 when a 90-degree rotation is applied to the item, 0 otherwise. The binary variable o^{ip}_{jqu} is equal to 1 when there is a separation axis between item j of type i and item q of type p, more precisely, a vertical separation axis for $u \in \{1, 2\}$, and a horizontal separation axis for $u \in \{3, 4\}$. Constraints (7.1) ensure that between items j and q there is at least one separating axis. Constraints (7.2) ensure that there is a vertical separating axis between item j and item q if the binary variable o^{ip}_{jq1} is equal to 1 (in this case j lies at the right of q). Similarly for constraints (7.3) (in this case j lies at the left of q). Constraints (7.4) ensure that there is a horizontal separating axis between item j and item q if the binary variable o^{ip}_{jq3} is equal to 1 (in this case j lies above q). Similarly for constraints (7.5) (in this case j lies below q). Constraints (7.6) and (7.7) define the x_2 and y_2 coordinates of the items, taking into account their orientation. Constraints (7.8) and (7.9) ensure that the items lie inside the layer.

Grouping items

In order to meet the contiguity constraints, we first need to model the contiguity relation between items of the same type i. As previously seen in Section 2.2.3.1, contiguity is modeled as an overlap between enlarged items (enlarged by $\xi/2$ in each direction). Thus, we can model contiguity by turning the non overlapping constraints (7.2)-(7.5) into overlapping constraints between the enlarged items. More precisely, for two items j and q of the same type, contiguity is modeled as follows:

$$\left(x_{2q} + \frac{\xi}{2}\right) - \left(x_{1j} - \frac{\xi}{2}\right) \geq -W(1 - \bar{d}_{jq})$$

$$\left(x_{2j} + \frac{\xi}{2}\right) - \left(x_{1q} - \frac{\xi}{2}\right) \geq -W(1 - \bar{d}_{jq})$$

$$\left(y_{2q} + \frac{\xi}{2}\right) - \left(y_{1j} - \frac{\xi}{2}\right) \geq -L(1 - \bar{d}_{jq})$$

$$\left(y_{2j} + \frac{\xi}{2}\right) - \left(y_{1q} - \frac{\xi}{2}\right) \geq -L(1 - \bar{d}_{jq})$$
$$\bar{d}_{jq} \in \{0, 1\}.$$

The binary variable $\bar{d}_{jq}$ models contiguity since it can be equal to 1 only when there is an overlap between the enlarged items j and q. Now, let $\tau_{\mathcal{G}} = \{(i, j, q) : i = 1, \dots, m, 1 \leq q < j \leq b^i\}$. The problem of determining if item j is contiguous to item q of the same type i can be modeled as follows:

$$x^i_{2q} - x^i_{1j} + \frac{\xi}{2} \geq -W\,(1 - \bar{d}^i_{jq}) \quad (i, j, q) \in \tau_{\mathcal{G}} \tag{7.12}$$
$$x^i_{2j} - x^i_{1q} + \frac{\xi}{2} \geq -W\,(1 - \bar{d}^i_{jq}) \quad (i, j, q) \in \tau_{\mathcal{G}} \tag{7.13}$$
$$y^i_{2q} - y^i_{1j} + \frac{\xi}{2} \geq -L\,(1 - \bar{d}^i_{jq}) \quad (i, j, q) \in \tau_{\mathcal{G}} \tag{7.14}$$
$$y^i_{2j} - y^i_{1q} + \frac{\xi}{2} \geq -L\,(1 - \bar{d}^i_{jq}) \quad (i, j, q) \in \tau_{\mathcal{G}} \tag{7.15}$$
$$\bar{d}^i_{jq} \in \{0, 1\} \quad (i, j, q) \in \tau_{\mathcal{G}}. \tag{7.16}$$

Remark 1 *The notion of contiguity that we have introduced is a mild one. It may happen that items are contiguous according to this definition but there is no visible space between them. A simple example is detailed in Figure 7.1.*

The adoption of a suitable objective function which tends to group items of the same type makes unlikely the situation displayed in Figure 7.1. However, we can also deal with this by strengthening the conditions under which contiguity holds. To this end, in addition to the previous constraints (7.12)-(7.16), we may also add the following ones, where $\Upsilon_i = \min\{w_i, h_i\}$, i.e., Υ_i is the length of the shortest edge for items of type i:

$$\bar{d}^i_{jq} \leq d^i_{jq1} + d^i_{jq2} + d^i_{jq3} + d^i_{jq4} - 2 \qquad (i, j, q) \in \tau_{\mathcal{G}},$$
$$x^i_{2q} - x^i_{1j} \geq \frac{\Upsilon_i}{2} - (W + \frac{\Upsilon_i}{2})\,(1 - d^i_{jq1}) \quad (i, j, q) \in \tau_{\mathcal{G}}$$

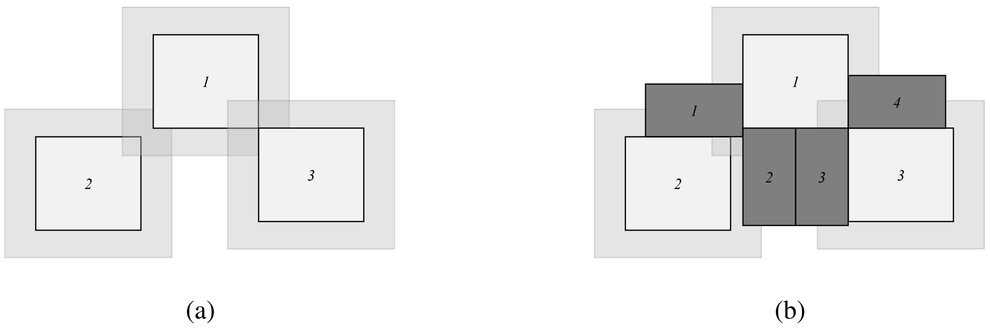


Figure 7.1: Special case for the contiguity constraint: the contiguity constraint is always met when considering the group of light and dark items (Figures 7.1a and 7.1b). For what concerns the light items, items 1 and 2 are contiguous and so are items 2 and 3, so that the contiguity relation defines a connected graph. The same holds for the dark items. However, in practice, when considering Figure 7.1b, we notice that the group of dark items creates a physical barrier between the light ones.

$$
\begin{aligned}
& x^i_{2j} - x^i_{1q} \geq \frac{\Upsilon_i}{2} - (W + \frac{\Upsilon_i}{2})\,(1 - d^i_{jq2}) && (i,j,q) \in \tau_{\mathcal{G}} \\
& y^i_{2q} - y^i_{1j} \geq \frac{\Upsilon_i}{2} - (L + \frac{\Upsilon_i}{2})\,(1 - d^i_{jq3}) && (i,j,q) \in \tau_{\mathcal{G}} \\
& y^i_{2j} - y^i_{1q} \geq \frac{\Upsilon_i}{2} - (L + \frac{\Upsilon_i}{2})\,(1 - d^i_{jq4}) && (i,j,q) \in \tau_{\mathcal{G}} \\
& d^i_{jqu} \in \{0,1\} && (i,j,q) \in \tau_{\mathcal{G}}, u = 1,\ldots,4.
\end{aligned}
$$

According to these constraints, contiguity between items j and q occurs or, equivalently, $\bar{d}^i_{jq}$ *can be equal to 1, if and only if three of the binary variables* d^i_{jqu}, $u = 1,\ldots,4$, *are equal to 1. In this case the projections along the x-axis (or along the y-axis) of the horizontal sides (if* $d^i_{jq1} = d^i_{jq2} = 1$*) or vertical sides (if* $d^i_{jq3} = d^i_{jq4} = 1$*) of items j and q share a segment of length at least equal to* $\frac{\Upsilon_i}{2}$*. For instance, if* $d^i_{jqu} = 1$ *for* $u = 1,2,3$*, then the upper horizontal side of an item and the lower horizontal side of the other item are at distance lower than* ξ *in view of constraints (7.12)-(7.16), and, moreover, their projections along the x-axis share a segment of length at least* $\frac{\Upsilon_i}{2}$*. Note that two out of four of these binary variables, namely, one for the horizontal sides (*$u \in \{1,2\}$*) and one for the vertical sides (*$u \in \{3,4\}$*), can always be set equal* *to 1. We point out that we do not include the last set of constraints in our computational experiments. Their addition is advised if the notion of contiguity needs to be strengthened.*

Unique connected component

We can associate an oriented graph $F_i = (V_i, A_i)$ to each item type i as follows, where J_i denotes the set of all items of type i (note that $|J_i| = b^i$):

- a node j is associated to each item in J_i; one of these nodes (it does not matter which one) is selected as a representative for all items of type i, and we denote it with index 1;
- two additional nodes are introduced, a source S and a destination D;
- an arc from S to each node in J_i is introduced and its capacity is fixed to 1;
- a single oriented arc from node 1 to node D with infinite capacity is introduced;
- given $(i, j, q) \in \tau_{\mathcal{G}}, q \neq 1$, an arc between node $j \in J_i$ and node $q \in J_i$ and an arc between node $q \in J_i$ and node $j \in J_i$ are introduced both with capacity $b^i \cdot \bar{d}^i_{jq}$. Note, in particular, that if items j and q are not contiguous, i.e., if $\bar{d}^i_{jq} = 0$, then the capacity of the arcs is 0. In fact, arcs exiting from node 1 can be omitted.

Figures 7.2-7.4 illustrate the notions introduced above. In all cases the dotted and continuous arcs represent arcs with null capacity and with strictly positive capacity, respectively. For two distinct item types, Figure 7.2 illustrates the two graphs induced by the contiguity relations between the items, while the complete graph is represented in Figure 7.3. Given graph F_i, whose arcs have capacity depending on the contiguity variables $\bar{d}^i_{jq}$, the contiguity constraint for items of type i is met if the restriction of the graph to the nodes in J_i, i.e., nodes corresponding to items of type i, is connected. This is equivalent to require that the maximum flow between node S and node D, which cannot be larger than b^i, is exactly equal to b^i.

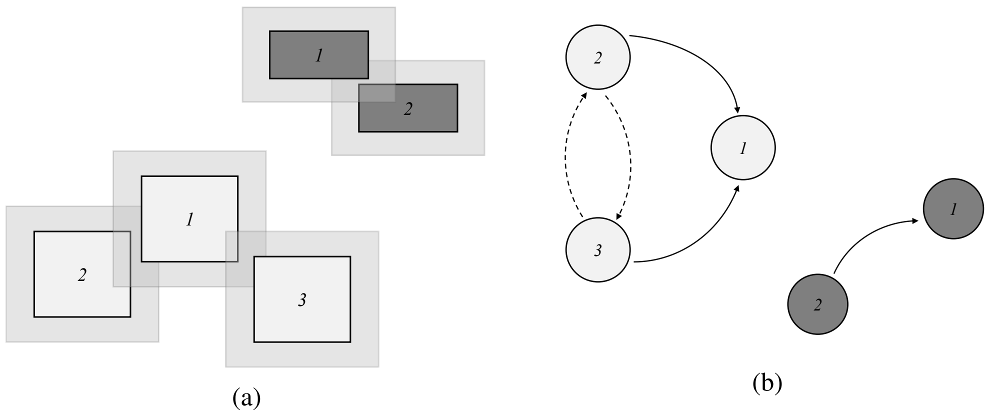


Figure 7.2: Contiguous items by item type: (7.2a) graphic and (7.2b) graph representation.

Figure 7.3 shows the max-flow problem for the item disposition presented in Figure 7.2a. In this case the contiguity constraint is met. On the other hand, Figure 7.4 shows a case where the contiguity constraint is not met. Thus, we propose the following constraints to guarantee contiguity between items of the same type, based on the above max-flow formulation:

$$f^i_{jq} \geq 0 \qquad (i, j, q) \in \tau_{\mathcal{G}} \tag{7.17}$$

$$f^i_{qj} \geq 0 \qquad (i, j, q) \in \tau_{\mathcal{G}}, q \neq 1 \tag{7.18}$$

$$f^i_{jq} \leq \bar{d}^i_{jq} \, . \, b^i \qquad (i, j, q) \in \tau_{\mathcal{G}} \tag{7.19}$$

$$f^i_{qj} \leq \bar{d}^i_{jq} \, . \, b^i \qquad (i, j, q) \in \tau_{\mathcal{G}}, q \neq 1 \tag{7.20}$$

$$f^i_{Sj} \leq 1 \qquad i = 1, \ldots, m, j = 1, \ldots, b^i \tag{7.21}$$

$$f^i_{Sj} \geq 0 \qquad i = 1, \ldots, m, j = 1, \ldots, b^i \tag{7.22}$$

$$\sum_{q=1}^{b^i} f^i_{jq} = \sum_{q=2}^{b^i} f^i_{qj} + f^i_{Sj} \qquad i = 1, \ldots, m, j = 2, \ldots, b^i, q \neq j \tag{7.23}$$

$$f^i_{1D} = \sum_{j=2}^{b^i} f^i_{j1} + f^i_{S1} \qquad i = 1, \ldots, m \tag{7.24}$$

$$f^i_{1D} = b^i \qquad i = 1, \ldots, m \tag{7.25}$$

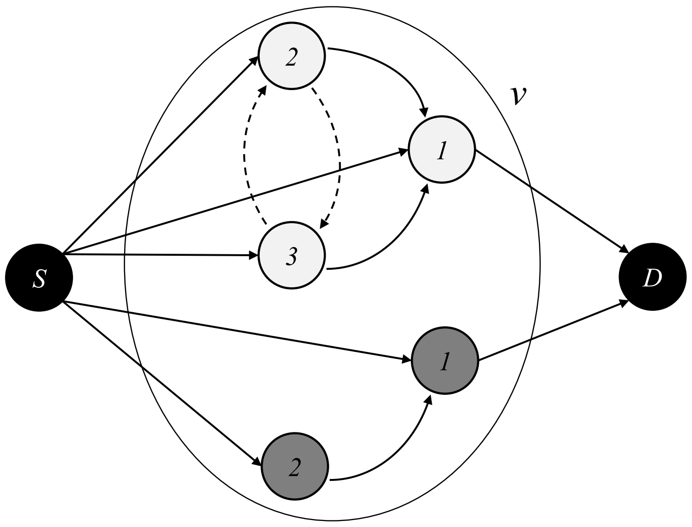


Figure 7.3: Max-flow representation: light items 2 and 3 are not contiguous, but both are contiguous to light item 1, maintaining the overall connection of this group. The same holds for the dark items.

Variable f^i_{jq} can be strictly positive only when j and q of the same type i are contiguous, otherwise its value is equal to 0. Constraints (7.17) and (7.18) ensure nonnegativity of the flow along arcs between nodes in J_i; (7.19) and (7.20) are the capacity constraints for the same arcs (note again that for arcs between nodes such that $\bar{d}^i_{jq} = 0$ the capacity is equal to 0); constraints (7.21) and (7.22) impose nonnegativity of the flow and the capacity, respectively, for arcs between node S and nodes in J_i; constraints (7.23) ensure flow conservation for all nodes in J_i except node 1; constraints (7.24) ensure flow conservation for node 1, and, thus, also defines the overall flow received by node D; finally, constraints (7.25) impose that the maximum flow between S and D is b^i, which ensures that items in J_i form a connected component and, thus, that the contiguity constraint for items of type i is met.

Item visibility

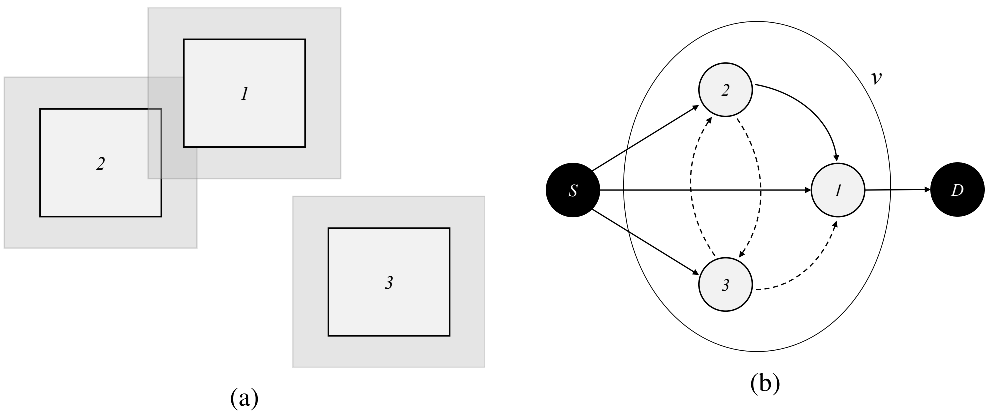


Figure 7.4: Item group not fulfilling the contiguity constraint: for the item disposition in 7.4a, the max-flow in 7.4b shows that only items 1 and 2 are contiguous, resulting in a flow to D equal to 2, that is different from the number of items (3 in this case).

To meet the visibility constraint, each group of items of type $i \in G$ must have at least one item visible from the border. In simple terms, we apply the idea of grouping items considering the borders as generic items (top, bottom, right, left) that must be contiguous to a specific item of the group i (without loss of generality, we set this item equal to 1). More precisely, visibility is modeled by imposing the contiguity between item 1 of the given type, enlarged by $\frac{\xi}{2}$ in all directions, and the borders of the layer, all moved by $\frac{\xi}{2}$ towards the interior of the layer. That leads to the following constraints to model the group visibility:

$$\sum_{u=1}^{4} v_u^i \geq 1 \qquad i = 1, \ldots, m \tag{7.26}$$

$$x_{11}^i - \frac{\xi}{2} \leq W\,(1 - v_1^i) \qquad i = 1, \ldots, m \tag{7.27}$$

$$x_{21}^i + \frac{\xi}{2} \geq W\,v_2^i \qquad i = 1, \ldots, m \tag{7.28}$$

$$y_{11}^i - \frac{\xi}{2} \leq L\,(1 - v_3^i) \qquad i = 1, \ldots, m \tag{7.29}$$

$$y_{21}^i + \frac{\xi}{2} \geq L\,v_4^i \qquad i = 1, \ldots, m \tag{7.30}$$

$$v_u^i \in \{0, 1\} \qquad i = 1, \ldots, m, u = 1, \ldots, 4 \tag{7.31}$$

The binary variable v_u^i is equal to 1 when item 1 of type i is contiguous (i..e, within distance ξ) to one of the borders of the layer. In particular, taking into account constraints (7.27)-(7.30), $v_1^i = 1$ ensures contiguity to the leftmost vertical side of the layer, $v_2^i = 1$ ensures contiguity to the rightmost vertical side, $v_3^i = 1$ ensures contiguity to the lowermost horizontal side, $v_4^i = 1$ ensures contiguity to the uppermost horizontal side. Constraint (7.26) ensure that item 1 of type i is within distance ξ from at least one of the sides of the layer.

Bounding box

In the proposed approach, for a given set of items, we would like to establish whether there exists a way to place them over the layer so that all constraints are met. Under this respect the problem is a feasibility one. However, it is convenient to introduce an objective function which should favor compact arrangements for items of the same type. Here we use the concept of Bounding Box (rectangle convex hull) to group items of the same type. Therefore, we propose a mathematical model to minimize the sum of the (semi)perimeters of the bounding boxes that cover all items of each item type $i \in G$ inside the layer. The overall model to place items over the layer is the following:

$$\begin{aligned}
\text{Minimize} \quad & \sum\nolimits_{i=1}^{m} [(X_2^i - X_1^i) + (Y_2^i - Y_1^i)] & & (7.32)\\
\text{subject to} \quad & X_1^i \le x_{1j}^i & i = 1,\dots,m, j = 1,\dots,b^i \quad & (7.33)\\
& Y_1^i \le y_{1j}^i & i = 1,\dots,m, j = 1,\dots,b^i \quad & (7.34)\\
& X_2^i \ge x_{2j}^i & i = 1,\dots,m, j = 1,\dots,b^i \quad & (7.35)\\
& Y_2^i \ge y_{2j}^i & i = 1,\dots,m, j = 1,\dots,b^i \quad & (7.36)\\
& (7.1) - (7.31). & &
\end{aligned}$$

Variables X_1^i and Y_1^i represent, respectively, the minimum x and y bounding box coordinates for items of type i, while X_2^i and Y_2^i represent the maximum x and y bounding box coordinates when considering item group of type i. Constraints (7.33)-(7.36) define the extreme points of a bounding box enclosing all items of type i, while the objective function is the sum of the (semi)perimeters of the bounding boxes for all the item types inside the layer, to be minimized.

As a final remark, we point out once again that the model is applied to a fixed set of items. Once the model has been solved, if a solution has been detected, then a new item is added to the set and the new model is solved. Instead, if no feasible solution is detected, the layer is declared complete and a new one is started.

7.2.2 Mathematical formulation for building pallets

After all items have been packed into layers, we need to put layers together into the minimum number of pallets. This mathematical formulation has already been described previously, and we refer to the Section 6.2.1.2 for details.

7.3 Solution algorithms

In this section, we present the heuristic algorithms that we developed to solve the PBP addressed. We first adopted a constructive heuristic (Section 7.3.1), then extended it to obtain a reactive GRASP metaheuristic (Section 7.3.2), and finally embedded the mathematical models of Section 7.2, obtaining a so-called matheuristic (Section 7.3.3). All these methods are based on the decomposition of the PBP into its two main components, first layer creation and then pallet building, already adopted in Section 7.2.

7.3.1 Constructive heuristic

The constructive heuristic is based on the flexible Extreme Points Modified Heuristic (EPMH) of Section 6.2. We refer to Chapter 6 for details.

7.3.2 Reactive GRASP metaheuristic

The reactive GRASP metaheuristic is based on the Reactive GRASP with Extreme Points Modified Heuristic (GREP) (Section 5.2). We refer to Chapter 5 for details.

7.3.3 Embedding the mathematical models in the heuristics

We created some variants of the aforementioned heuristics by making use of the mathematical models of Section 7.2.

For what concerns the creation of the pallets once the layers have been built, we created a variant of the EPMH, called *EPMH with Mathematical Model* (MMEPMH), in which we first attempt to create the pallets through the quick greedy heuristic of Section 4.3.2. Then, we perform a second attempt in which we create a (hopefully better) solution by using the model of Section 6.2.1.2 with a short time limit. Note that this model is relatively simple and is usually solved within a very short time. However, in some cases it takes more time and for this reason we imposed a time limit to its solution. To increase the execution speed of the solver, in the second attempt we set the initial model solution to the solution found by the heuristic during the first attempt. The main idea of this combination is to take advantage of the EPMH in order to create a solution in a very short time and then, check the possibility of reducing the final number of pallets by using a mathematical model, which, although more time consuming, usually leads to better solutions. The same idea has been used in the GREP algorithm of Section 5.2.2, leading to an alternative algorithm that we call *GREP with Mathematical Model* (MMGREP). In this case, the GRASP metaheuristic is exploited to find solutions better than those returned by the EPMH.

For what concerns the creation of the layers, we did not find convenient using the model of Section 7.2.1 from scratch, as too time consuming, but we opted, instead, to use it to create denser layers than the ones produced by the heuristic of Section 7.3.1. In detail, after the heuristic has created a layer, we attempt to include other items in the layer, one at a time, while preserving feasibility. In simple terms, we insert in the set of items to be packed by the model a new item of an item type already contained in the layer: if the model returns a feasible packing, then we attempt inserting a new item of the same type (if any); if the model does not find a feasible solution, then we attempt inserting an item of a different type (if any). In a family layer, the process is executed considering the only family in the layer. For residual layers, instead, all families and item types are attempted. At each iteration, the model is allowed to run for a short time limit. The process is iterated until no more item can fit into the current layer or a maximum overall time limit is reached. We obtained in this way a modified version of MMEPMH, that we call *MMEPMH Full* (EPFULL), and a modified version of MMGREP, that we call *MMGREP Full* (GREPFULL). These strategies allow to create more compact layers and to detect better solutions. However, that comes at the cost of larger execution times, since the mathematical model is complex and its solution time consuming.

7.4 Computational results

All experiments have been conducted on a Virtual Machine VMware, Intel Xeon CPU E5-2640 v2 2.00GHz, 16GB RAM, Linux Ubuntu Server 18.04 Operating System. The algorithms have been implemented in Java and executed using Oracle JDK 11. We used the solver Gurobi Optimizer 9.1.0 to solve the mathematical models.

We used the algorithms proposed (Section 7.3.3) to solve the same 24 instances presented in Section 4.4.1.

7.4.1 Parametric configurations

For all instances, the minimum required fill factor was set to 55%, and the container dimensions were set to 1500, 1250, and 1050 for height, width, and length, respectively. We allowed rotation of 90 degrees of the items. Note that the minimum fill factor value has been set equal to a relatively small value (55%) since we experimentally observed that larger values impose stronger constraints on the creation of layers and lead to poorer solutions (for more details, see Chapter 6). We also remark that this is just a minimum fill factor value, but, as we will see in the reported results, the average fill factor of the layers is considerably larger than such minimum. As already pointed out in the previous chapter, the occupation of the layer with a small fill factor can be increased by so called filler boxes, which allow to increase the occupation area and, thus, the stability.

We solved the instances by using all the algorithmic variants described in Section 7.3. For each variant, we carried out a simple test for parametric configuration. When GRASP is applied, we set $\epsilon = 0.15$, $\varrho = 10$, $\max_\kappa = 5$, $\beta = 0.98$, $\Omega = 25$, and $\psi = 500$, as described in Chapter 5. For the variant GREPFULL, instead, we set $\Omega = 1$ and $\psi = 10$, due to the reduced number of iterations when using the mathematical models.

All variants involving GRASP have been run 15 times each, so as to obtain a robust evaluation of the performance. These algorithms use an initial structure S sorted by *random* order for families, and a non-increasing order of *width* for item types. The deterministic algorithms (EPMH, MMEPMH and EPFULL) use instead the Bounding Box function to evaluate the creation of item groups in a layer, as suggested in Chapter 4, and are run only once. After preliminary experiments, when considering the variants that use only the model for pallet creation (MMEPMH and MMGREP), we set the time limit for the solver to 30 seconds. When considering the variants that use both models (EPFULL and GREPFULL), we set the time limit to 1 second for the layer creation model and 3 seconds for the pallet creation model. Actually, we also tried larger time limits, but the increased computing times were not compensated by higher quality solutions. When considering GRASP variants, the whole time for the algorithms was set to 5 minutes, except for GREPFULL, where it was set to 15 minutes.

Due to the large number of tests that we performed, in the following we report only aggregate results. For the sake of clarity, we separated the results in different tables in which each row provides average results for a given algorithm on the 360 solutions obtained (15 for each of the 24 attempted instances) in case of randomized algorithms and 24 solutions (one per instance) in case of deterministic algorithms. For a more detailed comparison among the algorithms, we refer to `https://github.com/silveira-tt/ANOR_PBP-Appendix`, where we report full details of all the computational tests that we performed over each instance.

7.4.2 Evaluation

The average results are summarized in Table 7.1. In the first three columns the table reports, respectively, the name of the algorithm, the total number of pallets in the solution obtained and the total number of layers created. In the six successive columns, it reports the minimum, maximum and average pallet utilization percentage and fill factor percentage (2D space of all layers). Note that the minimum fill factor is below the imposed lower bound (55%), since layers on top of all the others (in particular, residual layers) are allowed to violate such bound. The last column reports the computational time (in seconds). To facilitate the analysis of the results, we highlight in bold the best results of pallets and layers obtained.

From Table 7.1, we can notice that the number of layers in EPMH is slightly lower than in GREP, but the number of pallets is larger. Thus, we can highlight that the minimization of the number of layers is less relevant than the way in which these are created, since the overall number of pallets depends on the disposition of items within the layers. This situation also occurs when comparing MMEPMH and MMGREP, but not when considering the algorithms with mathematical models EPFULL and GREPFULL. For what concerns 3D pallet filling and 2D fill factor, the best performance is, in general, obtained by model-based algorithms, but with the disadvantage of a larger computational time. The choice of the algorithm obviously depends on how much time we can dedicate to the whole operation. MMEPMH

Table 7.1: Computational results for the proposed algorithms (the best results are highlighted in bold).

Algorithm	N. of pallets	N. of layers	3D pallet filling (%)			2D fill factor (%)			Seconds
			Min	Max	Avg	Min	Max	Avg	
EPMH	11.29	41.88	22	84	59	36	96	83	0.03
GREP	10.80	42.02	26	85	62	49	95	84	300.00
MMEPMH	10.33	41.88	32	86	66	36	96	83	3.69
MMGREP	10.33	42.71	34	85	66	50	95	83	300.00
EPFULL	10.13	41.13	31	86	67	45	95	86	184.91
GREPFULL	**9.97**	**40.78**	33	86	68	54	95	86	900.00

Table 7.2: Computational results per layer type.

Algorithm	Single-item layers	Single-family layers	Residual layers
EPMH	23.08	15.08	3.71
GREP	23.79	15.00	3.24
MMEPMH	23.08	15.08	3.71
MMGREP	24.59	14.89	3.24
EPFULL	16.96	20.88	3.29
GREPFULL	16.59	21.09	3.09

represents a good compromise between quality of the solutions and computing times but, of course, if minutes rather than seconds are allowed to perform the computations, then EPFULL and GREPFULL appears to be the best choices.

Secondly, we present an analysis of layers type in the solution, summarized in Table 7.2. This table presents, for each algorithm, the average number of single-item, single-family and residual layers.

Considering layer type, we observe a tendency to create more homogeneous layers when using the GRASP metaheuristic, by increasing the number of single-item or single-family layers, and reducing the number of residual ones. About the quality metric that analyzes the worst case result, the reported value is significantly better when using GRASP-based algorithms in general. Also note that in the algorithms where layer creation is based on the solution of mathematical models, the number of single-family layers increases with respect to the number of single-item layers, in view of the higher ability of the mathematical model solvers in further adding items of the same type in single-item layers, thus reducing the number of such layers.

For a deeper analysis of the GRASP-based algorithms, we present Table 7.3. In this table, for each algorithm, we report, respectively, the minimum, maximum and average values of the fitness function (5.1), adopted to guide the search, the average number of iterations, and the average number of local searches performed.

For what concerns the fitness function values, GREP presents the most stable results, whereas MMGREP presents a large variation among average and extreme values. The best results are obtained by GREPFULL, which also delivers the lowest

Table 7.3: Computational results for GRASP-based algorithms.

Algorithm	Objective function			Iterations	Local searches
	Min	Max	Avg		
GREP	0.5936	0.5952	0.5946	7888	1097
MMGREP	0.5794	0.6045	0.5962	5574	807
GREPFULL	0.6031	0.6125	0.6087	8.62	1.82

number of pallets. This confirms that the fitness function provides a good representation of the search space.

Considering the obtained results, we have a large reduction of the number of local searches in GREPFULL with respect to the other ones, but this is obviously due to the reduced number of iterations when using that algorithm.

Considering the previous results, we may highlight the most important parts of each algorithm. Algorithm EPMH obtains the lowest computing time, but it does not always find a solution of good quality. Hence, this strategy is suitable to be applied within metaheuristic or matheuristic approaches. For what concerns algorithm MMEPMH, it provides a better compromise between computational effort and quality, since it brings an improvement in the solution quality without excessively increasing the execution time. When taking metaheuristic into account, we can notice a considerable increment in the execution time. However, in general, this extra time is used to perform a better exploration of the search space, which is obtained by the randomized greedy construction and local search improvement phases of Algorithm 1. For example, algorithm GREP brings that benefit, improving the solution quality while maintaining a good compromise between time and quality. However, for algorithm MMGREP that does not happen, since the average quality of the solution is similar to algorithm MMEPMH, besides increasing the execution time. Algorithms EPFULL and GREPFULL make full use of the mathematical models introduced in Sections 7.2.1-6.2.1.2. Through a good mix of quick greedy construction of layers and pallets, better exploration of the solution space by means of randomization and local search, and search for improved solution by means of mathematical models invoked for short time limits, these approaches are able to return better quality solutions. They require computing times larger than the other approaches but still compatible with the needs of a company if the pallet creation is performed offline.

7.5 Conclusions

In this chapter, we concluded our studies regarding the Pallet Building Problem with item rotations and practical constraints involving visibility and contiguity. We proposed matheuristic algorithms and a mathematical formulation for the addressed problem, filling a gap in the existing literature. In general, our complete algorithm is based on a two-step heuristic approach, first creating layers and then pallets. We tailored each proposed strategy to GRASP metaheuristic with reactive method. Regarding to heuristic algorithms, these are an Extreme Points Modified Heuristic for creating 2D layers and a greedy heuristic for creating 3D pallets. Regarding to mathematical models, we embedded them in the previous heuristics, creating in this way variants to the two-step heuristic both to compact items into layers and to stack layers into pallets. Extensive computational experiments on real-world instances proved the effectiveness of the proposed algorithms.

When it comes to the matheuristics proposed, the embedded mathematical models have shown a capacity of improvement of the quality of the solutions found, without increasing too much the computing time. In spite of the complexity of the real-world instances and the short execution times, interesting final results are reported, thus showing that the combination heuristics/mathematical models/metaheuristic is an efficient strategy for this type of problem.

As future work, as an alternative to the approaches proposed in this chapter, and, thus, a possible further topic for future research, we mention heuristics based on building blocks (see, e.g., Liu et al. [68], Ren et al. [89], Wang et al. [108], Zhang et al. [111]). Each block is made up by items of the same type, so that a block automatically satisfies contiguity between these items. However, basic building block heuristics should be adapted to meet also the visibility constraint.

Chapter 8

The "Floating Cuts" Model: A General and Flexible Mixed-Integer Programming Model for Rectangular Cutting Problems

8.1 Abstract

Cutting and Packing problems are challenging combinatorial optimization problems that have many relevant industrial applications and arise whenever a raw material has to be cut into smaller parts while minimizing waste, or products have to be packed, minimizing the empty space. Thus, the optimal solution to these problems has a positive economic and environmental impact.

In many practical applications, both the raw material and the cut parts have a rectangular shape, and cutting plans are generated for one raw material rectangle (also known as plate) at a time. This is what is known in the literature as the (2-dimensional)

Rectangular Cutting Problem. Current cutting technologies impose several geometric and quantitative constraints to this problem, the most relevant of which are the guillotine cuts.

Based on the idea of the floating cuts, a general and flexible mixed-integer programming model for the non-guillotine and guillotine problem is proposed. To the best of our knowledge, it is the first Mixed-Integer Linear Programming model in the literature for both non-guillotine and guillotine problems, and a few of their variations. The basic idea of this model is a tree search where branching occurs by successive first-order non-guillotine-type cuts. The exact position of the cuts is not fixed, but instead remains floating until a concrete small rectangle (also known as item) is assigned to a child node. This model does not include decision variables either for the position coordinates of the items or for the coordinates of the cuts. This highlights the main similarities and differences when comparing the proposed model to the 'one-cuts' model from Dyckhoff [33] (also presented by Furini et al. [44]): both models contain divisions of the rectangles into smaller ones, however, this model includes the decision of the sizes of the subrectangles in a decision variable, unlike the 'one-cuts' model. Not only that but, the 'one-cuts' model requires the calculation of all container positions prior to optimizing the model, in contrast to the 'floating cuts' model.

Extensive computational experiments were run to evaluate the model's performance considering the many different problem variants and to compare it with the state-of-the-art formulations of each variant. The results confirm the power of this flexible model, as it outperforms the state-of-the-art approaches for some variants while remaining very competitive for the others. But, even more importantly, this is a new way of looking at these problems which may trigger even better approaches, with the consequent economic and environmental benefits.

8.2 Contributions

In this work, a new Mixed-Integer Linear Programming (MILP) model for first-order non-guillotine cutting patterns is proposed. The MILP model can be adapted for dealing with guillotine constraints, allowing guillotine cutting patterns.

The main contribution of this new MILP model is its flexibility, since: it can be used for non-guillotine and guillotine cutting patterns, without affecting the structure of the model; it can be used for the 2DSKP and for the 2DSLOPP (introduced in Section 2.3) and to different variants of the problem; the weighted and unweighted versions can be considered; 90 degrees rotation of the items is allowed; both the bounded and unbounded variants of the problem are considered.

Based on the proposed model, a specific k-staged MILP for the guillotine version is also developed, taking advantage of the structure related to the guillotine constraint. Any k can be defined and in extreme cases, ∞ can be considered, allowing non-staged cutting patterns. Moreover, these new MILP models do not require decision variables either for the coordinates where items are placed or for the position of the cuts, substantially reducing the size of the model.

Contributing to the wide use of this model is the fact that the 2DSKP and 2DSLOPP also arise as sub-problems (pricing problems) embedded in column generation procedures to solve the 2-dimensional Bin Packing Problem (2DBPP) and the 2-dimensional Cutting Stock Problem (2DCSP), meaning that this new MILP can also be used in the resolution of these problems.

8.3 Work organization

The work is organized as follows.

In Section 8.4, the new MILP formulation is introduced and the ‘floating cuts’ model and the supporting algorithms for the formulation are described. Valid inequalities are proposed and added to the model to reduce symmetry and to strengthen the formulation.

In Section 8.5, the effectiveness of the valid inequalities is assessed through computational experiments.

In Section 8.6, an extension of the ‘floating cuts’ model through new valid inequalities to the guillotine problem is proposed and assessed through extensive computational experiments. Additionally, a new model to deal with k-staged problems is developed.

Section 8.7 is dedicated to the computational experiments for guillotine cutting patterns. The first floating cuts model extended for the guillotine constraint is compared against the model specifically designed for the guillotine problem. Both models are also compared with state-of-the-art formulations.

Finally, in Section 8.8, conclusions and future work are discussed.

8.4 The 'floating cuts' MILP model for the two-dimensional SLOPP

For the sake of simplicity, the model will be presented and explained taking into account the 2DSLOPP in terms of formulation. All other formulations will be developed from this one, by adding different objectives or new constraints.

As already stated and explained, the cutting problem considered in this work allows cuts of type first-order non-guillotine. The model is based on the assumption that from a given rectangle, a maximum of five new rectangles can be obtained.

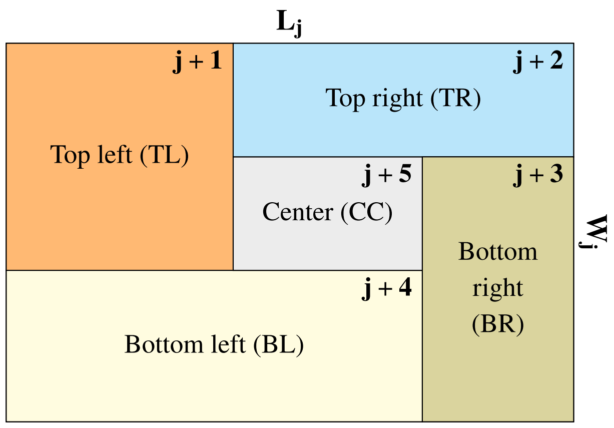
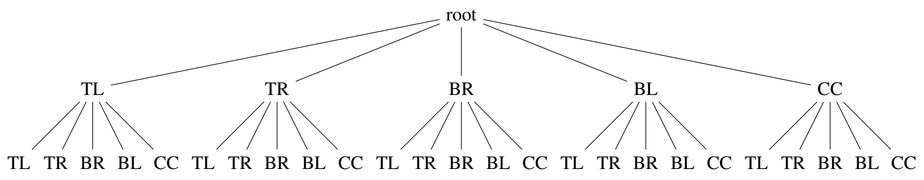


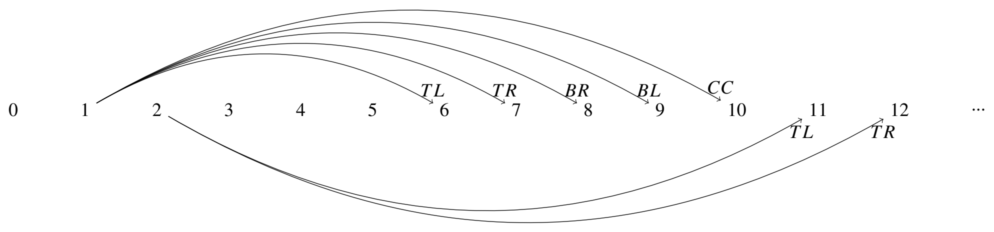
Figure 8.1: 1st order cut.

In Figure 8.1, rectangles and sub-rectangles are numbered on the top left corner. From a cut in a sub-rectangle j may result a maximum of five new sub-rectangles:

top left ($j + 1$), top right ($j + 2$), bottom right ($j + 3$), bottom left ($j + 4$) and center ($j + 5$). The new sub-rectangles can be further cut in a recursive way. Note that if the center sub-rectangle ($j + 5$) does not exist, the cut is of type guillotine.

In Figure 8.2 (a), an example of a tree of sub-rectangles is presented. If index 0 is assigned to the root node (initial plate), indexes 1, 2, 3, 4, and 5 are assigned to the top left, top right, bottom right, bottom left and center sub-rectangles (respectively). In the same way, if sub-rectangle 1 is cut, it can result in sub-rectangles 6, 7, 8, 9 and 10 (top left, top right, bottom right, bottom left, and center), and so forth for the other sub-rectangles. An illustration of the sub-rectangles indexing is presented in Figure 8.2 (b).


root
TL
TR
BR
BL
CC
TL TR BR BL CC TL TR BR BL CC TL TR BR BL CC TL TR BR BL CC TL TR BR BL CC
(a) Tree example



(b) Sub-rectangle index

Figure 8.2: Example.

Sub-rectangles index enumeration

For a sub-rectangle tree with h levels, it is possible to enumerate all indexes of the sub-rectangles and store them in a vector M^h. Since M^h depends on the number of levels of the tree and from a given cut, five new sub-rectangles are obtained, the number of elements of M^h is $|M^h| = \sum_{j=0}^{h} 5^j$.

The model proposed is parametric since their dimension (number of variables and constraints) is dependent on the number of levels of the tree (h) and the corresponding number of sub-rectangles ($|M^h|$). Thus, the model becomes more complex as the value of h increases.

Two special sets of indexes are considered: 1) element 0 is the root node and does not have a father sub-rectangle; and 2) the elements of the last level of the tree, known as leaves, do not have children. Every element j (not a leaf) has 5 children in the M^h vector (top left (TL), top right (TR), bottom right (BR), bottom left (BL) and center (CC)), which can be traced: $TL = 5 \cdot j + 1$; $TR = 5 \cdot j + 2$; $BR = 5 \cdot j + 3$; $BL = 5 \cdot j + 4$ and $CC = 5 \cdot j + 5$. Therefore, given an index j ($\forall j \neq 0$), it is possible to obtain the sub-rectangle's relative position (top left, top right, bottom right, bottom left and center) using Algorithm 2 ("mod" stands for the remainder of the integer division). Having the relative position of a given sub-rectangle j, it is possible to identify the index of the father of j using Algorithm 3. It is also possible to identify the level of the tree of a given sub-rectangle j using Algorithm 4. The algorithms mentioned in this section can be found in Appendix.

For this problem, there is one rectangular plate with length L and width W, and the goal is to cut from it a set of n items indexed by $i, i = 1, ..., n$. Each item i has a length l_i, a width w_i, a maximum demand d_i and a value v_i. The main objective is to maximize the value of the items assigned to the rectangular plate.

The main and new idea of this MILP model is that it is possible to enumerate the set m ($m = |M^h|$) of all possible sub-rectangles j, $j = 1, ..., m$ from any initial plate, without explicitly identifying their dimensions. Therefore, a decision variable is associated with a cut, and not to the position of the cut. The decision variables are formally defined below.

Decision Variables:

$$x_j = \begin{cases} 1, \text{if sub-rectangle } j \text{ is cut }, & j = 0, ..., m \\ 0, \text{ otherwise} \end{cases}$$

$$\delta_{ij} = \begin{cases} 1, \text{if item } i \text{ is assigned to sub-rectangle } j, & j = 0, ..., m \\ 0, \text{ otherwise} \end{cases}$$

s_{ij} - number of items of type i placed side by side (horizontally) in sub-rectangle j, $j = 0, ..., m$.

t_{ij} - number of items of type i placed on the top of each other (vertically) in sub-rectangle j, $j = 0, ..., m$.

$$z_j = \begin{cases} 1, \text{if items are placed side by side in sub-rectangle } j, j = 0, ..., m \\ 0, \text{ otherwise} \end{cases}$$

L_j - length of sub-rectangle j, $j = 0, ..., m$;

W_j - width of sub-rectangle j, $j = 0, ..., m$;

Data:

$TL(j)$ - Top left sub-rectangle index resulting from a cut in sub-rectangle j, $j = 0, ..., m$;

$TR(j)$ - Top right sub-rectangle index resulting from a cut in sub-rectangle j, $j = 0, ..., m$;

$CC(j)$ - Center sub-rectangle index resulting from a cut in sub-rectangle j, $j = 0, ..., m$;

$BL(j)$ - Bottom Left sub-rectangle index resulting from a cut in sub-rectangle j, $j = 0, ..., m$;

$BR(j)$ - Bottom right sub-rectangle index resulting from a cut in sub-rectangle j, $j = 0, ..., m$;

For the sake of clarity, the mathematical formulation (8.1)-(8.33) is divided and explained in four sub-sections:

8.4.1 Objective function and general constraints

$$\text{Maximize} \quad \sum_{i=1}^{n}\sum_{j=1}^{m} v_i \cdot (s_{ij} + t_{ij}) \tag{8.1}$$

$$\text{Subject to:} \quad x_j + \sum_{i=1}^{n} \delta_{ij} \leq 1 \qquad \forall j = 0, ..., m; \tag{8.2}$$

$$\sum_{j=0}^{m}(s_{ij} + t_{ij}) \leq d_i, \qquad \forall i = 1, ..., n; \tag{8.3}$$

$$s_{ij} \leq d_i \cdot \delta_{ij}, \qquad \forall i = 1, ..., n; j = 0, ..., m; \tag{8.4}$$

$$t_{ij} \leq d_i \cdot \delta_{ij}, \qquad \forall i = 1, ..., n; j = 0, ..., m; \tag{8.5}$$

$$\sum_{i=1}^{n} l_i \cdot s_{ij} \leq L_j, \qquad \forall j = 0, ..., m; \tag{8.6}$$

$$\sum_{i=1}^{n} w_i \cdot t_{ij} \leq W_j, \qquad \forall j = 0, ..., m; \tag{8.7}$$

$$l_i \cdot \delta_{ij} \leq L_j + L(1 - \delta_{ij}), \qquad \forall i = 1, ..., n; j = 0, ..., m; \tag{8.8}$$

$$w_i \cdot \delta_{ij} \leq W_j + W(1 - \delta_{ij}), \qquad \forall i = 1, ..., n; j = 0, ..., m; \tag{8.9}$$

$$s_{ij} \leq d_i \cdot z_j, \qquad \forall i = 1, ..., n; j = 0, ..., m; \tag{8.10}$$

$$t_{ij} \leq d_i \cdot (1 - z_j), \qquad \forall i = 1, ..., n; j = 0, ..., m; \tag{8.11}$$

$$L_0 = L; \tag{8.12}$$

$$W_0 = W; \tag{8.13}$$

$$s_{ij} + t_{ij} \geq \delta_{ij}, \qquad \forall i = 1, ..., n; j = 0, ..., m; \tag{8.14}$$

The objective function (8.1) maximizes the values of the items assigned to the rectangular plate. Constraints (8.2) ensure that for each sub-rectangle j, only three scenarios can exist: the sub-rectangle is cut or items of type i are assigned to the sub-rectangle; or the sub-rectangle stays without any modification. Constraints (8.3) ensure that the demand of each item type i is not exceeded. Constraints (8.4) e (8.5) are used to connect decision variables s_{ij}, t_{ij} and δ_{ij}. It is guaranteed that the items geometrically fit in the sub-rectangle in constraints (8.6) - (8.7). Constraints (8.8) - (8.9) ensure that at least one item fits in sub-rectangle j. Constraints (8.10) and (8.11) ensure that for a given sub-rectangle j it is only allowed the assignment of items grouped horizontally or vertically. Constraints (8.12) and (8.13) initialize the root sub-rectangle with the dimensions of the plate. Constraints (8.14) are used to connect s_{ij}, t_{ij} and δ_{ij}, regarding the quantity of items and the assignment to a sub-rectangle.

8.4.2 Length of children sub-rectangles constraints

$$L_{TL(j)} + L_{TR(j)} \leq L(1 - x_j) + L_j, \qquad \forall j = 0, ..., m; \quad (8.15)$$

$$L_{TL(j)} + L_{TR(j)} \geq L_j - L(1 - x_j), \qquad \forall j = 0, ..., m; \quad (8.16)$$

$$L_{TL(j)} + L_{CC(j)} + L_{BR(j)} \leq L(1 - x_j) + L_j, \qquad \forall j = 0, ..., m; \quad (8.17)$$

$$L_{TL(j)} + L_{CC(j)} + L_{BR(j)} \geq L_j - L(1 - x_j), \qquad \forall j = 0, ..., m; \quad (8.18)$$

$$L_{BL(j)} + L_{BR(j)} \leq L(1 - x_j) + L_j, \qquad \forall j = 0, ..., m; \quad (8.19)$$

$$L_{BL(j)} + L_{BR(j)} \geq L_j - L(1 - x_j), \qquad \forall j = 0, ..., m; \quad (8.20)$$

$$L_{TL(j)} + L_{TR(j)} + L_{CC(j)} + L_{BL(j)} + L_{BR(j)} \leq 3 \cdot L \cdot x_j, \quad \forall j = 0, ..., m; \quad (8.21)$$

Constraints (8.15) - (8.21) are associated with length of the sub-rectangles if a cut is performed in the j sub-rectangles. Constraints (8.15) - (8.16) ensure that when a cut is performed, the sum of the lengths of the resulting top left and top right sub-rectangles is equal to the length of the father sub-rectangle. Constraints (8.17) - (8.18) ensure that when a cut is performed, the sum of the lengths of the resulting top left, center and bottom right sub-rectangles is equal to the length of the father sub-rectangle. Constraints (8.19) - (8.20) ensure that when a cut is performed, the sum of the lengths of the resulting bottom left, and bottom right sub-rectangles is equal to the length of the father sub-rectangle. In the opposite way, constraints (8.21) ensure that if the father sub-rectangle is not cut the resulting sub-rectangles have length equal to zero.

8.4.3 Width of children sub-rectangles constraints

$$W_{TL(j)} + W_{BL(j)} \leq W(1 - x_j) + W_j, \qquad \forall j = 0, ..., m; \tag{8.22}$$

$$W_{TL(j)} + W_{BL(j)} \geq W(x_j - 1) + W_j, \qquad \forall j = 0, ..., m; \tag{8.23}$$

$$W_{TR(j)} + W_{CC(j)} + W_{BL(j)} \leq W(1 - x_j) + W_j, \qquad \forall j = 0, ..., m; \tag{8.24}$$

$$W_{TR(j)} + W_{CC(j)} + W_{BL(j)} \geq W(x_j - 1) + W_j, \qquad \forall j = 0, ..., m; \tag{8.25}$$

$$W_{TR(j)} + W_{BR(j)} \leq W(1 - x_j) + W_j, \qquad \forall j = 0, ..., m; \tag{8.26}$$

$$W_{TR(j)} + W_{BR(j)} \geq W(x_j - 1) + W_j, \qquad \forall j = 0, ..., m; \tag{8.27}$$

$$W_{TL(j)} + W_{TR(j)} + W_{CC(j)} + W_{BL(j)} + W_{BR(j)} \leq 3 \cdot W \cdot x_j, \qquad \forall j = 0, ..., m; \tag{8.28}$$

Constraints (8.22) - (8.28) are associated with the width of the resulting sub-rectangles if sub-rectangle j is cut. Constraints (8.22) - (8.23) ensure that if sub-rectangle j is cut, the sum of the width of the top left and bottom left sub-rectangle is equal to the width of the father sub-rectangle. Similarly, (8.24) - (8.25) ensure that if sub-rectangle j is cut, the sum of the width of the top right, center and bottom left sub-rectangles is equal to the width of the father sub-rectangle. Constraints (8.26) - (8.27) ensure that if sub-rectangle j is cut, the sum of the width of the top right and bottom right sub-rectangles is equal to the width of the father sub-rectangle. Constraints (8.28) guarantee that if sub-rectangle j is not cut, the resulting sub-rectangles have width equal to zero.

8.4.4 Decision variables domain

$$x_j \in \{0, 1\}, \qquad \forall j = 0, ..., m; \tag{8.29}$$

$$\delta_{ij} \in \{0,1\}, \qquad \forall i = 1,...,n; j = 0,...,m; \tag{8.30}$$

$$s_{ij}, t_{ij} \in \mathbb{N}_0, \qquad \forall i = 1,...,n; j = 0,...,m; \tag{8.31}$$

$$z_j \in \{0,1\}, \qquad j = 0,...,m; \tag{8.32}$$

$$L_j, W_j \geq 0, \qquad \forall j = 0,...,m. \tag{8.33}$$

Decision variables related to the cutting of a sub-rectangle are binary and presented in constraints (8.29). The decision variables related to the assignment of items to sub-rectangles are also of the binary type and are defined by constraints (8.30). The decision variables related to the quantity of items of type i that are horizontally and vertically assigned to sub-rectangle j are integer and greater or equal to zero (8.31). Decision variables (8.32) are related to the type of assignment of items to sub-rectangles, it is equal to one if the items are assigned side by side (horizontally) and zero otherwise. Lastly, constraints (8.33) are related to the size of the sub-rectangles and can take any value greater or equal to zero.

8.4.5 Model strengthening

To strengthen the MILP formulation (8.1)-(8.33), a set of valid inequality constraints were added to the model and are described next.

Symmetry

The four non-guillotine cutting patterns in Figure 8.3 are equivalent. Constraints (8.34) - (8.35) are added to the model to reduce symmetry by ensuring that the model can generate only the first-order non-guillotine cutting pattern of type (a) of Figure 8.3.

$$L_{BL(j)} \geq L_{TL(j)}, \qquad \forall j = 0,...,m; \tag{8.34}$$

$$L_{TL(j)} \geq L_{BR(j)}, \qquad \forall j = 0,...,m; \tag{8.35}$$

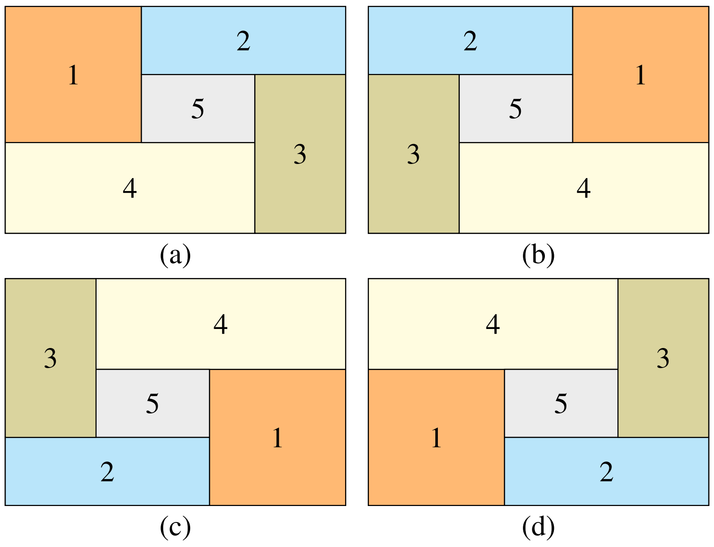


Figure 8.3: Symmetric patterns.

Area bound

Constraints (8.36) state that the total area of the rectangles cut does not exceed the initial rectangle area.

$$\sum_{i=1}^{n} l_i \cdot w_i \cdot \sum_{j=0}^{m} (s_{ij} + t_{ij}) \leq L \cdot W \tag{8.36}$$

Leaves don’t branch

The set $|M^{h-1}|, ..., |M^h|$ represents the sub-rectangles’ indexes belonging to the last level of the sub-rectangles tree. Constraints (8.37) ensure that in the last level of the tree, the sub-rectangles can not be further cut.

$$x_j = 0 \qquad \forall j = |M^{h-1}|, ..., |M^h| \tag{8.37}$$

If a sub-rectangle is not cut, it has no descendants

If a sub-rectangle is not cut ($x_j = 0$), then it has no descendants.

$$\sum_{k=1}^{5}(x_{5\cdot j+k} + \sum_{i=1}^{n} \delta_{i(5\cdot j+k)}) \leq 5 \cdot x_j, \qquad \forall j = 0, ..., m; \tag{8.38}$$

8.5 Computational experiments for the non-guillotine problem

Computational experiments were performed to evaluate the performance of the original 'floating cuts' model with the model strengthened with a set of valid inequality constraints. Since five new sub-rectangles can be generated from a cut, the 'floating cuts' model will be represented by the acronym "FC5", more precisely the model (8.1)-(8.33) is named "FC5_Base", and the model (8.1) - (8.38) is named "FC5_Strengthened". Graphical examples of non-guillotine solutions obtained are provides in Appendix.

The proposed models can be used to solve the two-dimensional SLOPP and SKP. The computational experiments include four different variants for each type of problem:

- **F/U** -The orientation of the items is fixed (no 90 degrees rotation is allowed), and the problem is unweighted, i.e., the value of each item is equal to its area;
- **R/U** - The items can be rotated 90 degrees, and the problem is unweighted;
- **F/W** - The orientation of the items is fixed, and the problem is weighted, i.e., the value of each item is not equal to its area;
- **R/W** - The items can be rotated 90 degrees, and the problem is weighted.

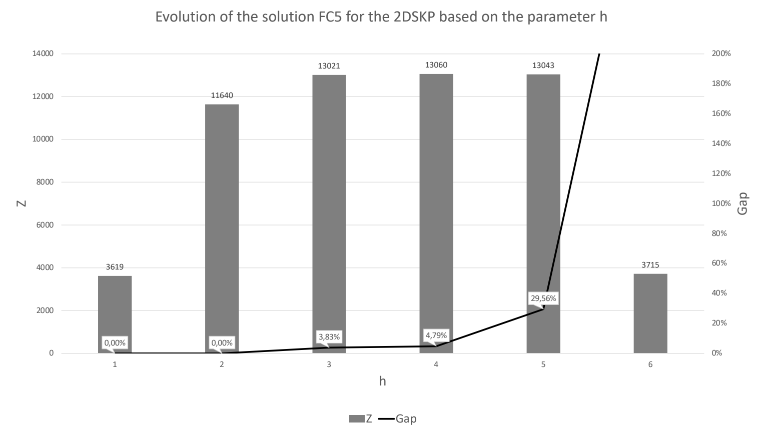
The computational experiments were run on Gurobi 9.1.2 using an Intel Xeon Gold 6148 CPU 2.40GHz, with 96.0 GB of RAM, under Windows 10 operating system, and the model was implemented with Python. The optimality gap was set to 0.01% (Gurobi default value) and the time limit was set to 900 seconds.

Model size increases with larger values of h, and as it is known, the performance of the MILP solvers decreases as the size of models to be solved increases. The first preliminary experiments were performed to determine the value of h to consider. Parameter h was initially set to the lowest value ($h = 1$) and iteratively incremented while there was an average improvement in the quality of the solution. Figure 8.4 shows the average results for the 4 variations when considering both the 2DSKP and 2DSLOPP, according to the h value. Therefore, we concluded that h equal to 4 is the most suitable value. In Appendix, Tables 1 and 2 present the preliminary experiments. We highlight that the maximum number of levels of the sub-rectangles tree was set to 6, since, for some instances, the solver was not able to find a feasible solution within the time limit.

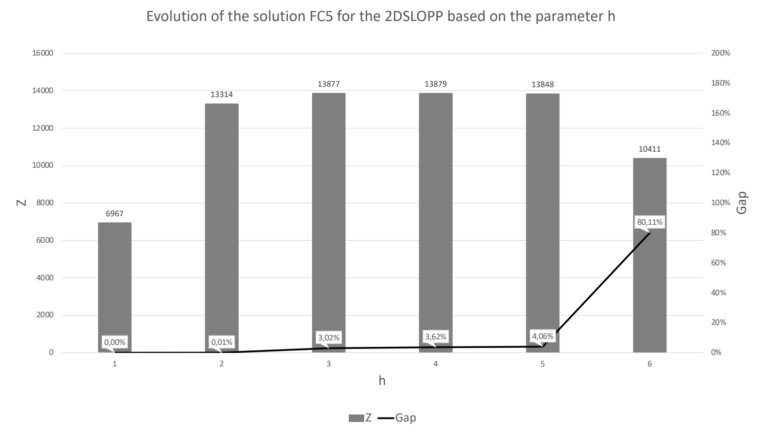
The computational experiments were run over a selection of problem instances taken from the literature. The selection was made considering problem instances for the weighted 2DSLOPP (as this is the most complete type/variant of the problem, in terms of piece data), which were used in at least three papers published since 2016. Three data sets, in a total of 11 instances, were identified with these characteristics: the cgcut1-3, originally proposed by Christofides and Whitlock [19], the CW1-3, originally proposed by Hifi [53], and the okp1-5, proposed by Fekete and Schepers [40]. Besides, the set of instances ngcut1-12 originally proposed by Beasley [9] for the first mathematical model dealing with non-guillotine cutting patterns were also considered.

A description of the problem instances is presented in Table 8.1. For each instance, the following is identified: the number of different item types (n); the total number of items ($N = \sum_{i=1}^{n} d_i$); the length (L) and the width (W) of the large object; the smallest edge length (l_{min}) and width (w_{min}) among all the item types.

In a problem instance for the weighted 2DSLOPP, the item types are characterized by a length l_i, a width w_i, a value v_i, and a maximum demand d_i. When addressing the unweighted variant, the value of the items is set to $v_i = l_i \cdot w_i$. If the problem type is the 2DSKP, the maximum demand is set to one ($d_i = 1$). The only exceptions go to



(a) 2DSKP



(b) 2DSLOPP

Figure 8.4: Summary of the preliminary experiments for (8.4a) 2DSKP and (8.4b) 2DSLOPP, when considering the average for the 4 variants addressed. In both cases, $h = 4$ presents the best trade-off between the Z value of the objective function and the solver Gap.

instances cgcut1-3 and ngcut1-12, since in these instances, the number of item types (n) is small, the total number of items (N) is used by replicating the items according to its demand. Therefore, depending on the problem types and variants, the demand (d_i) and the value (v_i) of the items were updated as described.

Table 8.1: Characteristics of the instances.

Instance	n	N	L	W	l_{min}	w_{min}
cgcut1	7	16	15	10	2	1
cgcut2	10	23	40	70	9	7
cgcut3	19	62	40	70	9	11
CW1	25	67	105	125	21	25
CW2	35	63	165	145	34	34
CW3	40	96	207	267	45	59
okp1	15	50	100	100	1	1
okp2	30	30	100	100	1	3
okp3	30	30	100	100	3	3
okp4	33	57	100	100	1	2
okp5	29	97	100	100	1	3
ngcut1	5	10	10	10	2	2
ngcut2	7	17	10	10	1	1
ngcut3	10	21	10	10	1	1
ngcut4	5	7	15	10	7	1
ngcut5	7	14	15	10	2	1
ngcut6	10	15	15	10	2	1
ngcut7	5	8	20	20	1	1
ngcut8	7	13	20	20	6	1
ngcut9	10	18	20	20	1	2
ngcut10	5	13	30	30	1	2
ngcut11	7	15	30	30	3	2
ngcut12	10	22	30	30	2	1

In Table 8.2 we present the results for the four variants of the 2DSKP. The table is divided into four parts regarding the problem variants (F/U, R/U, F/W and F/W). The first column identifies the problem instance and the following column refers to the number of levels of the sub-rectangle tree considered ($h = 4$). The following three columns are dedicated to report the results obtained with FC5_Base model. Column (Z) presents the value of the objective function of the solution obtained by Gurobi, column (Gap^s(%)) presents the gap value reported by Gurobi and the total computational time in seconds is reported in column (T(s)), "TL" means the time limit of 900 seconds was reached. The following columns present the results for the FC5_Strengthened model, and the last column is used to present the value of the optimal solution when it is known. This structure is repeated for each problem variant. It is highlighted in bold in column Z whenever the optimal solution is obtained, compared to the values in column OPT from the literature. In column (Gap^s(%)), the cases where the solver proves optimality are highlighted in bold.

Analysing the results, it is clear that constraints (8.34) - (8.38) strengthened the

Table 8.2: Computational results for first-order non-guillotine SKP.

		F/U						OPT		R/U						OPT
Instance	h	FC5_Base			FC5_Strengthened				h	FC5_Base			FC5_Strengthened			
		Z	Gap^s(%)	T(s)	Z	Gap^s(%)	T(s)			Z	Gap^s(%)	T(s)	Z	Gap^s(%)	T(s)	
cgcut1		144	56.25	TL	145	3.45	TL	-		**150**	50.00	TL	**150**	**0.00**	9.69	150
cgcut2		2778	56.37	TL	2740	2.19	TL	-		2775	56.54	TL	2738	2.26	TL	-
cgcut3		2721	509.00	TL	2721	2.90	TL	-		2723	1221.04	TL	2754	1.67	TL	-
CW1		13057	288.93	TL	12797	2.56	TL	-		12797	296.84	TL	12797	2.56	TL	-
CW2		23240	360.20	TL	23524	1.71	TL	-		23599	554.55	TL	23299	2.69	TL	-
CW3		54343	351.14	TL	54343	1.70	TL	-		54721	582.96	TL	54721	1.00	TL	-
okp1		9284	38.35	TL	9284	7.71	TL	-		9688	32.58	TL	9688	3.22	TL	-
okp2		9749	222.06	TL	9681	3.30	TL	-		9876	217.92	TL	9921	0.80	TL	-
okp3		9734	270.60	TL	9838	1.65	TL	-		9893	264.64	TL	9799	2.05	TL	-
okp4		9824	231.60	TL	9818	1.85	TL	-		9841	270.56	TL	9854	1.48	TL	-
okp5		9749	197.69	TL	9876	1.26	TL	-		9921	192.53	TL	9833	1.70	TL	-
ngcut1	4	95	100.00	TL	95	5.26	TL	-	4	97	95.88	TL	97	3.09	TL	-
ngcut2		97	185.57	TL	97	3.09	TL	-		**100**	177.00	TL	**100**	**0.00**	160.72	100
ngcut3		**100**	177.00	TL	**100**	**0.00**	75.19	100		**100**	177.00	TL	**100**	**0.00**	530.19	100
ngcut4		138	17.39	TL	138	2.17	TL	-		138	17.39	TL	138	2.17	TL	-
ngcut5		140	152.14	TL	140	7.14	TL	-		**150**	135.33	TL	**150**	**0.00**	4.05	150
ngcut6		**150**	93.33	TL	**150**	**0.00**	29.13	150		**150**	93.33	TL	**150**	**0.00**	18.60	150
ngcut7		175	**0.00**	3.16	175	**0.00**	1.00	-		175	**0.00**	3.91	175	**0.00**	1.84	-
ngcut8		380	66.58	TL	380	5.26	TL	-		387	63.57	TL	386	3.63	TL	-
ngcut9		390	149.74	TL	390	2.56	TL	-		**400**	143.50	TL	**400**	**0.00**	198.47	400
ngcut10		879	95.68	TL	879	1.93	TL	-		879	95.68	TL	879	2.16	TL	-
ngcut11		842	76.13	TL	842	6.89	TL	-		873	69.87	TL	873	3.09	TL	-
ngcut12		894	156.82	TL	898	0.22	TL	-		**900**	155.11	TL	899	0.11	TL	900
Average			167.50	861.96		2.82	788.06				215.82	862.00		1.46	666.94	
		F/W						OPT		R/W						OPT
cgcut1		**244**	49.18	TL	**244**	6.56	TL	244		260	40.0	TL	260	**0.00**	175.52	-
cgcut2		2856	55.78	TL	2856	2.24	TL	2892		2866	55.2	TL	2860	2.10	TL	-
cgcut3		**1860**	450.00	TL	**1860**	8.60	TL	1860		1780	900.6	TL	1940	4.12	TL	-
CW1		5010	116.91	TL	4860	9.82	TL	-		4944	119.8	TL	4944	10.05	TL	-
CW2		4767	161.88	TL	4767	13.36	TL	-		5083	206.4	TL	5062	6.64	TL	-
CW3		4593	319.70	TL	4719	12.82	TL	-		5008	284.9	TL	4938	7.82	TL	-
okp1		23771	31.77	TL	23771	12.40	TL	-		26081	20.1	TL	25721	4.19	TL	-
okp2		21976	170.54	TL	21947	11.44	TL	-		23288	155.3	TL	24263	2.54	TL	-
okp3		24019	220.74	TL	23740	11.85	TL	-		25216	205.5	TL	25278	6.16	TL	-
okp4		26470	180.96	TL	26470	9.25	TL	-		27568	198.2	TL	27568	5.15	TL	-
okp5		21976	157.20	TL	21976	11.29	TL	-		23645	139.0	TL	24263	2.62	TL	-
ngcut1	4	**164**	102.44	TL	**164**	22.56	TL	164	4	193	72.0	TL	193	13.47	TL	-
ngcut2		**230**	153.48	TL	**230**	11.30	TL	230		250	133.2	TL	250	2.00	TL	-
ngcut3		**247**	128.34	TL	**247**	7.69	TL	247		259	117.8	TL	259	2.70	TL	-
ngcut4		**268**	12.69	TL	**268**	2.61	TL	268		268	12.7	TL	268	2.61	TL	-
ngcut5		**358**	71.79	TL	**358**	4.19	TL	358		370	66.2	TL	370	0.81	TL	-
ngcut6		**289**	76.82	TL	**289**	9.69	TL	289		298	71.5	TL	298	6.38	TL	-
ngcut7		**430**	**0.00**	2.85	**430**	**0.00**	1.54	430		430	**0.00**	3.07	430	**0.00**	2.04	-
ngcut8		**834**	69.19	TL	**834**	12.47	TL	834		886	59.3	TL	886	5.87	TL	-
ngcut9		**924**	107.90	TL	**924**	4.11	TL	924		908	111.6	TL	918	4.79	TL	-
ngcut10		**1452**	81.20	TL	**1452**	**0.00**	682.57	1452		1452	81.2	TL	1452	4.48	TL	-
ngcut11		**1688**	72.45	TL	**1688**	12.20	TL	1688		1786	63.0	TL	1786	4.93	TL	-
ngcut12		**1865**	133.62	TL	**1865**	9.44	TL	1865		1932	125.5	TL	1932	5.64	TL	-
Average			127.15	861.95		8.95	852.40				140.83	861.96		4.57	830.37	

base ‘floating cuts’ mathematical model with a positive impact on the gaps reported by Gurobi. The average Gurobi gap for the FC5_Base model for the four problem variants (F/U, R/U, F/W, R/W) was 162.83%, while for the FC5_Strengthened model was only 4.45%.

Although the addition of valid inequality constraints improved the linear programming relaxation bounds (LP-bounds), the impact on the value of the solutions was not so impressive. In some instances, lower values of the objective function were reported for the FC5_Strengthened model. The solver proved the optimality in 14 cases with the FC5_Strengthened model and only 4 cases with the FC5_Base model from 92 cases tested for the 2DSKP.

For problem variant F/W, the optimal solutions is known from the literature in 15 instances. In this case, the optimal value was obtained by the FC5_Base and FC5_Strengthened model in 14 instances. However, the solver could only certificate optimality in three of them.

In Table 8.3 we present the results for 2DSLOPP. The structure of this table is similar to the previous one. The behaviour observed for 2DSKP remains for 2DSLOPP. For the FC5_Strengthened model, the solver obtained lower gaps in comparison to the FC5_Base model, and this difference is more impressive for the unweighted versions (F/U and R/U). The best performance was obtained for problem variant R/U with an average gap of 0.97% from the solver. For problem variant F/W, the optimal solution is known from the literature for 20 instances. The solver was able to reach the optimal solution for 17 instances for both models. However, it could only prove optimality for one instance (ngcut7) with FC5_Base model and two instances (ngcut7 and ngcut10) for the FC5_Strengthened model.

Column "Best" in problem variant R/W presents the best solutions found in Egeblad and Pisinger [36] and He et al. [52], all the values are marked with "*". The solver was able to find the same values for the FC5_Base and FC5_Strengthened models in 15 instances out of 20.

In Figure 8.5 we present the box plots for the gaps given by Gurobi on the different experiments. The box plot with no fill color were obtained for the gaps of the FC5_Base model. It is clear that the valid inequalities improved the gaps. For the base model, worse gaps were obtained for the 2DSLOPP in comparison with 2DSKP.

8.6 Extension of the model to the guillotine problem

An important constraint from practice regarding the cutting problems is related to the guillotine cuts. The mathematical formulation (8.1) - (8.38) can be extended to ensure

Table 8.3: Computational results for first-order non-guillotine SLOPP.

		F/U						OPT		R/U						OPT
Instance	h	FC5_Base			FC5_Strengthened				h	FC5_Base			FC5_Strengthened			
		Z	Gap^s(%)	T(s)	Z	Gap^s(%)	T(s)			Z	Gap^s(%)	T(s)	Z	Gap^s(%)	T(s)	
cgcut1		144	56.25	TL	144	4.17	TL	-		**150**	50.00	TL	**150**	**0.00**	4.34	150
cgcut2		2740	58.54	TL	2778	0.79	TL	-		2775	56.54	TL	2775	0.90	TL	-
cgcut3		2721	936.24	TL	2721	2.90	TL	-		2776	1146.36	TL	2771	1.05	TL	-
CW1		13057	746.05	TL	12777	2.72	TL	-		12972	751.60	TL	12972	1.18	TL	-
CW2		23240	1003.64	TL	23418	2.17	TL	-		23670	983.59	TL	23643	1.19	TL	-
CW3		53997	1356.25	TL	54708	1.03	TL	-		54608	1339.95	TL	54477	1.45	TL	-
okp1		9938	251.32	TL	9974	0.26	TL	-		9960	250.54	TL	9968	0.32	TL	-
okp2		9749	222.06	TL	9681	3.30	TL	-		9876	217.92	TL	9921	0.80	TL	-
okp3		9734	270.60	TL	9734	2.73	TL	-		9893	264.64	TL	9893	1.08	TL	-
okp4		9958	505.74	TL	9976	0.24	TL	-		9977	504.59	TL	9979	0.21	TL	-
okp5		9982	668.36	TL	9982	0.18	TL	-		**10000**	666.98	TL	**10000**	**0.00**	30.97	10000
ngcut1	4	95	100.00	TL	95	5.26	TL	-	4	97	95.88	TL	97	3.09	TL	-
ngcut2		97	185.57	TL	97	3.09	TL	-		**100**	177.00	TL	**100**	**0.00**	4.27	100
ngcut3		**100**	177.00	TL	**100**	**0.00**	2.78	100		**100**	177.00	TL	**100**	**0.00**	5.44	100
ngcut4		138	17.39	TL	138	2.17	TL	-		138	17.39	TL	138	2.17	TL	-
ngcut5		140	152.14	TL	140	7.14	TL	-		**150**	135.33	TL	**150**	**0.00**	1.46	150
ngcut6		**150**	93.33	TL	**150**	**0.00**	19.63	150		**150**	93.33	TL	**150**	**0.00**	4.49	150
ngcut7		175	**0.00**	2.20	175	**0.00**	1.14	-		175	**0.00**	1.62	175	**0.00**	0.75	-
ngcut8		380	66.58	TL	380	5.26	TL	-		387	63.57	TL	387	3.36	TL	-
ngcut9		390	149.74	TL	390	2.56	TL	-		**400**	143.50	TL	**400**	**0.00**	7.02	400
ngcut10		879	95.68	TL	879	2.39	TL	-		879	95.68	TL	879	2.39	TL	-
ngcut11		842	76.13	TL	842	6.89	TL	-		873	69.87	TL	873	3.09	TL	-
ngcut12		898	155.68	TL	898	0.22	TL	-		**900**	155.11	TL	**900**	**0.00**	16.89	900
Average			319.32	861.92		2.41	784.50				324.19	861.90		0.97	551.72	
		F/W						OPT		R/W						Best
cgcut1		**244**	49.18	TL	**244**	6.56	TL	244		260	40.00	TL	260	**0.00**	98.23	260*
cgcut2		2856	55.78	TL	2856	2.24	TL	2892		2880	54.48	TL	2883	1.28	TL	2909*
cgcut3		1840	681.52	TL	**1860**	9.68	TL	1860		1900	1006.32	TL	1920	6.25	TL	1940*
CW1		6402	355.51	TL	6402	12.09	TL	-		6936	320.44	TL	6808	5.41	TL	-
CW2		5354	426.90	TL	5543	8.70	TL	-		5548	408.47	TL	5604	7.51	TL	-
CW3		5689	684.67	TL	5689	7.00	TL	-		5689	684.67	TL	5736	6.12	TL	-
okp1		**27718**	225.74	TL	**27718**	5.48	TL	27718		28423	217.66	TL	28423	2.86	TL	28423*
okp2		21976	170.54	TL	22119	10.19	TL	22502		23288	155.30	TL	23645	5.34	TL	24263*
okp3		**24019**	220.74	TL	23740	11.85	TL	24019		25216	205.52	TL	25081	6.82	TL	25216*
okp4		**32893**	343.99	TL	**32893**	3.59	TL	32893		32784	345.46	TL	32893	3.00	TL	32893*
okp5		**27923**	492.10	TL	**27923**	4.38	TL	27923		27983	490.83	TL	27983	4.15	TL	27983*
ngcut1	4	**164**	102.44	TL	**164**	22.56	TL	164	4	193	72.02	TL	193	4.15	TL	193*
ngcut2		**230**	153.48	TL	**230**	10.00	TL	230		250	133.20	TL	250	2.80	TL	250*
ngcut3		**247**	128.34	TL	**247**	7.69	TL	247		259	117.76	TL	259	2.70	TL	259*
ngcut4		**268**	12.69	TL	**268**	2.61	TL	268		268	12.69	TL	268	2.61	TL	268*
ngcut5		**358**	71.79	TL	**358**	4.19	TL	358		370	66.22	TL	370	0.81	TL	370*
ngcut6		**289**	76.82	TL	**289**	9.69	TL	289		298	71.48	TL	298	6.38	TL	300*
ngcut7		**430**	**0.00**	1.72	**430**	**0.00**	1.44	430		430	**0.00**	1.22	430	**0.00**	1.21	430*
ngcut8		**834**	69.19	TL	**834**	12.47	TL	834		886	59.26	TL	886	5.87	TL	886*
ngcut9		**924**	107.90	TL	**924**	4.11	TL	924		924	107.90	TL	924	4.11	TL	924*
ngcut10		**1452**	81.20	TL	**1452**	**0.00**	286.01	1452		1452	81.20	TL	1452	4.48	TL	1452*
ngcut11		**1688**	72.45	TL	**1688**	11.85	TL	1688		1786	62.99	TL	1786	6.27	TL	1786*
ngcut12		**1865**	133.62	TL	**1865**	10.78	TL	1865		1932	125.52	TL	1932	5.33	TL	1932*
Average			205.07	861.90		7.73	835.15				210.41	861.88		4.10	826.98	

that only cutting patterns of type guillotine are obtained.

In a first-order non-guillotine cutting pattern (Figure 8.1), if the sub-rectangle on the center does not exist, i.e., $L_{CC} = 0$ and/or $W_{CC} = 0$, the cutting pattern would be of type guillotine. An example of guillotine cutting patterns obtained when $L_{CC} = 0$ and/or $W_{CC} = 0$ is presented in Figure 8.6.

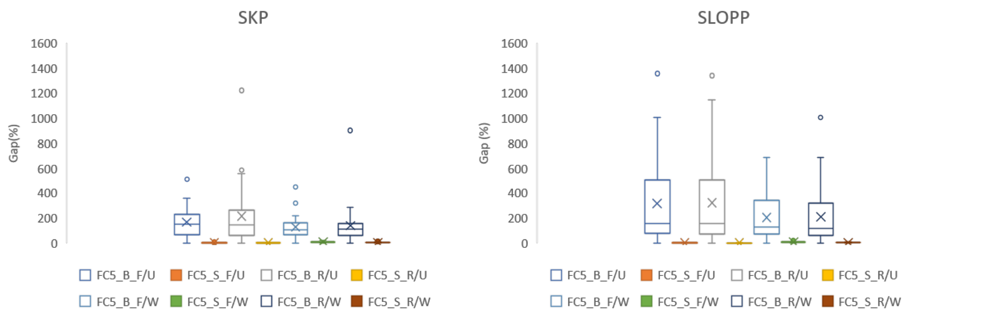


Figure 8.5: Analysis of the gap reported by Gurobi for FC5_Base and FC5_Strengthened models.

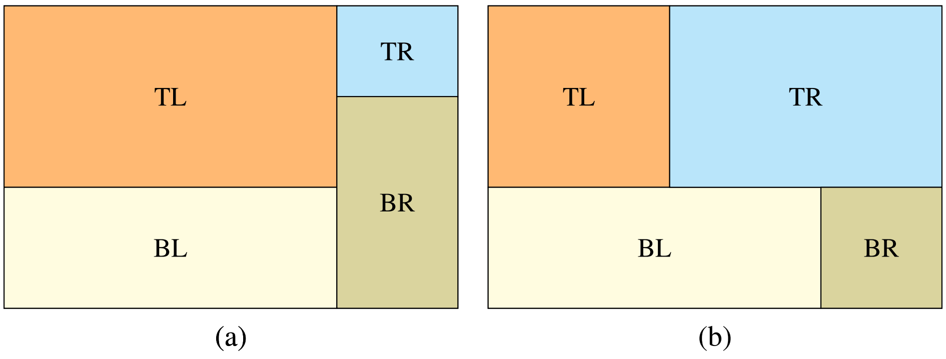


Figure 8.6: Example of a guillotine cutting pattern if $L_{CC} = 0$ and/or $W_{CC} = 0$.

Based on this insight, decision variable $\beta_{CC(j)}$ was defined:

$$\beta_{CC(j)} = \begin{cases} 1, \text{ ensures } W_{CC(j)} = 0, & j = 0, \dots, m \\ 0, \text{ ensures } L_{CC(j)} = 0, & j = 0, \dots, m \end{cases}$$

$$L_{CC(j)} \leq \beta_{CC(j)} \cdot L, \qquad \forall j = 0, \dots, m; \tag{8.39}$$

$$W_{CC(j)} \leq (1 - \beta_{CC(j)}) \cdot W, \qquad \forall j = 0, \dots, m; \tag{8.40}$$

Constraints (8.39) ensure that if decision variable $\beta_{CC(j)}$ is equal to one, the length of the sub-rectangle on the center will be zero. Constraints (8.40) ensure that if decision variable $\beta_{CC(j)}$ is equal to zero, the width of the sub-rectangle on the center, will be zero. These constraints ensure that the sub-rectangle on the center of a first-order non-guillotine pattern does not exist, and the cutting pattern becomes a guillotine one.

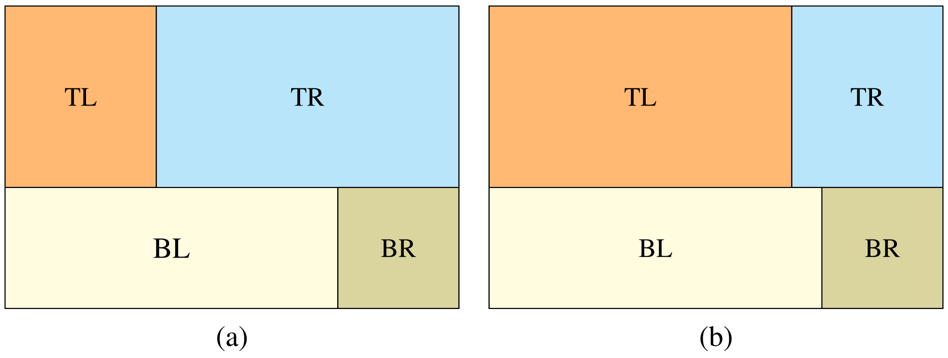


Figure 8.7: Example of equivalent guillotine cutting patterns when $W_{CC} = 0$.

8.6.1 Symmetry reduction

The addition of constraints (8.39) and (8.40) to the MILP model ensures guillotine cutting patterns. However, the relative positions of the five sub-rectangles of the first order cutting patterns remain the same. The main characteristic is that at least one of the dimensions of the center sub-rectangle is equal to zero. Constraints (8.34) and (8.35), to reduce symmetry, are also maintained for the guillotine version. However, when $W_{CC} = 0$ equivalent patterns, as presented in Figure 8.7, can be obtained. Constraints (8.41) and (8.42) are added to the mathematical formulation to reduce this symmetry.

$$L_{BR(j)} \leq L_{BL(j)} + L \cdot (1 - \beta_{CC(j)}), \qquad \forall j = 0, ..., m; \qquad (8.41)$$

$$L_{TR(j)} \leq L_{TL(j)} + L \cdot (1 - \beta_{CC(j)}), \qquad \forall j = 0, ..., m; \qquad (8.42)$$

Similarly, when $L_{CC} = 0$ the four equivalent cutting patterns of Figure 8.8 can be produced. To address this symmetry, constraints (8.43) and (8.44) are added to the model.

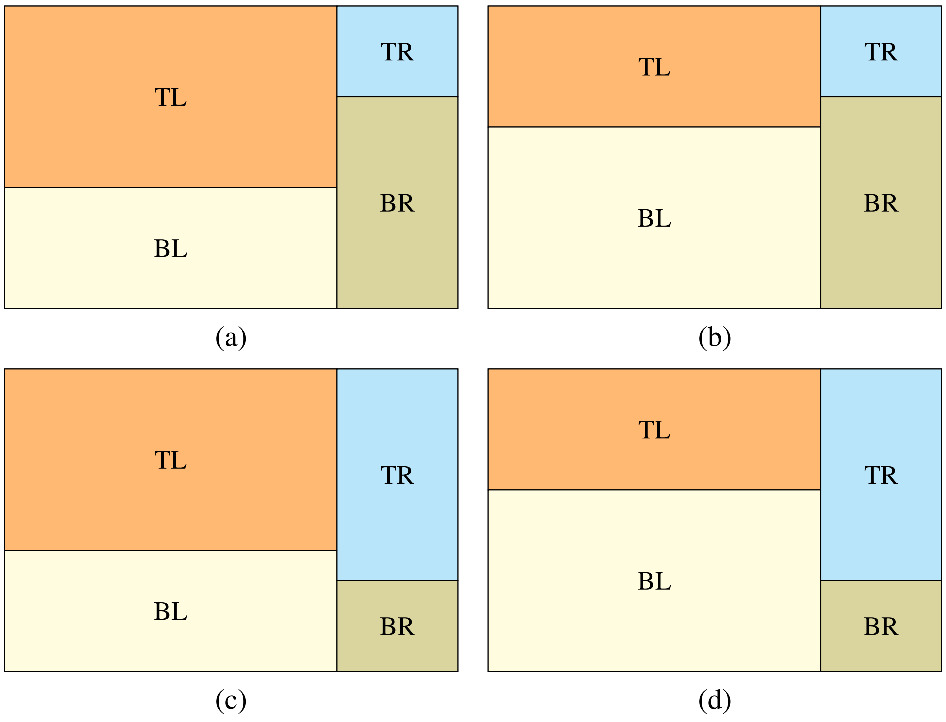


Figure 8.8: Example of equivalent guillotine cutting patterns when $L_{CC} = 0$.

$$W_{BL(j)} \leq W_{TL(j)} + W \cdot \beta_{CC(j)}, \qquad \forall j = 0, ..., m; \tag{8.43}$$

$$W_{TR(j)} \leq W_{BR(j)} + W \cdot \beta_{CC(j)}, \qquad \forall j = 0, ..., m; \tag{8.44}$$

8.6.2 Computational experiments - guillotine cutting patterns

The computational experiments presented in this subsection aim to evaluate the impact of valid inequality constraints (8.41) - (8.44) on the MILP model for the guillotine problem. The guillotine cutting patterns obtained are of type non-staged, which means there is no limitation on the number of stages.

The ‘floating cuts’ model with guillotine constraints (8.1)-(8.40) is named “FC4_-Base” since from the guillotine cut, a maximum of four sub-rectangles are generated. The ‘floating cuts’ model for the guillotine problems strengthened with the set of valid inequality constraints (8.41) - (8.44) is named “FC4_Strengthened”.

The computational results for 2DSKP and 2DSLOPP with FC4_Base and FC4_-Strengthened are presented in tables 8.4 and 8.5. The structure of the tables remains the same as the previous tables.

The addition of reducing symmetry constraints (8.41) - (8.44) impacted the values of the gap reported by Gurobi for both 2DSKP and 2DSLOPP. In Figure 8.9 we present the box-plots for the values of the Gurobi’s gap for the different variants for the 2DSKP and 2DSLOPP with FC4_Base and FC4_Strengthened. The box plots for the FC4_Base model are represented without fill color. Constraints (8.41) - (8.44) had a higher impact on variants with the orientation of the items fixed (F/U and F/W) for 2DSKP and 2DSLOPP. Generally, the gaps are smaller for both the FC4_Strengthened and FC4_Base model for the 2DSLOPP compared to 2DSKP.

The full occupation of the plate was obtained for the FC4_Strengthened model for 2DSKP variant R/U in 6 instances (cgcut1, ngcut2-3, ngcut 5-6 and ngcut9), and Gurobi proved optimality. For these 6 instances the solver’s average time was 12.66

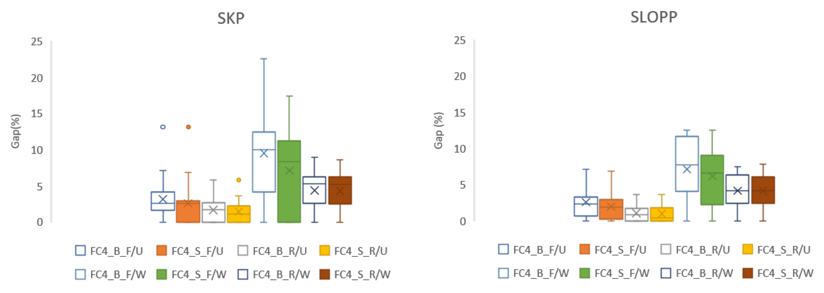


Figure 8.9: Analysis of the gap reported by Gurobi for the models FC4_Base and FC4_Strengthened.

Table 8.4: Computational results for guillotine SKP.

		F/U						OPT		R/U						OPT
Instance	h	FC4_Base			FC4_Strengthened				h	FC4_Base			FC4_Strengthened			
		Z	Gap^s(%)	T(s)	Z	Gap^s(%)	T(s)			Z	Gap^s(%)	T(s)	Z	Gap^s(%)	T(s)	
cgcut1		144	4.17	TL	144	4.17	TL	-		**150**	**0.00**	18.22	**150**	**0.00**	12.41	150
cgcut2		2740	2.19	TL	2740	2.19	TL	-		2759	1.49	TL	2769	1.12	TL	-
cgcut3		2721	2.90	TL	2721	2.90	TL	-		2754	1.67	TL	2754	1.67	TL	-
CW1		12797	2.56	TL	12797	2.56	TL	-		12777	2.72	TL	12797	2.56	TL	-
CW2		23240	2.95	TL	23240	2.95	TL	-		23299	2.69	TL	23643	1.19	TL	-
CW3		53755	2.82	TL	53755	2.82	TL	-		54041	2.27	TL	54070	2.22	TL	-
okp1		8836	13.17	TL	8836	13.17	TL	-		9448	5.84	TL	9448	5.84	TL	-
okp2		9773	2.32	TL	9749	2.58	TL	-		9828	1.75	TL	9828	1.75	TL	-
okp3		9838	1.65	TL	9838	1.65	TL	-		9843	1.60	TL	9893	1.08	TL	-
okp4		9831	1.72	TL	9818	1.85	TL	-		9816	1.87	TL	9810	1.94	TL	-
okp5		9773	2.32	TL	9749	2.58	TL	-		9921	0.80	TL	9780	2.25	TL	-
ngcut1	4	95	5.26	TL	95	**0.00**	326.07	-	4	97	3.09	TL	97	3.09	TL	-
ngcut2		97	3.09	TL	97	3.09	TL	-		**100**	**0.00**	23.25	**100**	**0.00**	11.38	100
ngcut3		**100**	**0.00**	7.06	**100**	**0.00**	4.83	100		**100**	**0.00**	52.91	**100**	**0.00**	19.53	100
ngcut4		138	**0.00**	646.89	138	**0.00**	25.64	-		138	2.17	TL	138	**0.00**	18.13	-
ngcut5		140	7.14	TL	140	**0.00**	10.28	-		**150**	**0.00**	4.40	**150**	**0.00**	2.38	150
ngcut6		148	1.35	TL	148	1.35	TL	-		**150**	**0.00**	10.82	**150**	**0.00**	13.48	150
ngcut7		175	**0.00**	1.12	175	**0.00**	0.74	-		175	**0.00**	1.96	175	**0.00**	2.16	-
ngcut8		380	5.26	TL	380	5.26	TL	-		386	3.63	TL	386	3.63	TL	-
ngcut9		390	2.56	TL	390	2.56	TL	-		**400**	**0.00**	38.30	**400**	**0.00**	21.81	400
ngcut10		879	1.93	TL	879	**0.00**	18.97	-		879	2.16	TL	879	0.91	TL	-
ngcut11		842	6.89	TL	842	6.89	TL	-		867	3.81	TL	873	3.09	TL	-
ngcut12		894	0.67	TL	894	0.67	TL	-		891	1.01	TL	895	0.56	TL	-
Average			3.17	811.96		2.58	682.76				1.68	633.30		1.43	592.01	
		F/W						OPT		R/W						OPT
cgcut1		**244**	6.56	TL	**244**	6.15	TL	244		260	**0.00**	114.08	260	**0.00**	97.64	-
cgcut2		2856	2.24	TL	2856	2.24	TL	2892		2892	0.97	TL	2848	2.53	TL	-
cgcut3		**1860**	8.60	TL	**1860**	8.60	TL	1860		1900	6.32	TL	1900	6.32	TL	-
CW1		4841	10.25	TL	4837	10.17	TL	-		5012	7.00	TL	4930	8.62	TL	-
CW2		4673	15.56	TL	4673	15.37	TL	-		5062	6.68	TL	5017	7.57	TL	-
CW3		4593	15.92	TL	4593	16.07	TL	-		4887	8.94	TL	4936	7.68	TL	-
okp1		22752	17.70	TL	22752	17.38	TL	-		25721	4.22	TL	25721	4.14	TL	-
okp2		21814	11.81	TL	22502	8.33	TL	-		23513	5.84	TL	23513	5.86	TL	-
okp3		23743	11.94	TL	24019	10.84	TL	-		25216	6.22	TL	25216	6.22	TL	-
okp4		25939	11.45	TL	25939	11.23	TL	-		27568	5.62	TL	27568	5.25	TL	-
okp5		21814	13.36	TL	21814	12.08	TL	-		23523	5.87	TL	24263	2.58	TL	-
ngcut1	4	164	22.56	TL	164	**0.00**	243.10	-	4	193	4.15	TL	193	4.15	TL	-
ngcut2		230	10.00	TL	230	2.61	TL	-		250	1.20	TL	250	1.20	TL	-
ngcut3		247	7.69	TL	247	7.29	TL	-		259	2.70	TL	259	2.70	TL	-
ngcut4		268	2.61	TL	268	**0.00**	34.42	-		268	2.61	TL	268	**0.00**	36.21	-
ngcut5		358	4.19	TL	358	**0.00**	8.93	-		370	0.81	TL	370	**0.00**	10.36	-
ngcut6		289	9.69	TL	289	4.15	TL	-		298	6.38	TL	298	6.38	TL	-
ngcut7		430	**0.00**	1.48	430	**0.00**	0.73	-		430	**0.00**	1.94	430	**0.00**	2.14	-
ngcut8		834	12.47	TL	834	12.47	TL	-		886	5.87	TL	886	5.87	TL	-
ngcut9		924	4.11	TL	924	**0.00**	23.93	-		924	4.11	TL	908	5.95	TL	-
ngcut10		1452	**0.00**	52.32	1452	**0.00**	16.41	-		1452	4.48	TL	1452	4.48	TL	-
ngcut11		1688	10.72	TL	1688	9.66	TL	-		1786	5.32	TL	1780	5.45	TL	-
ngcut12		1865	9.38	TL	1865	9.38	TL	-		1932	5.64	TL	1932	5.59	TL	-
Average			9.51	824.99		7.13	680.20				4.39	827.70		4.28	750.67	

seconds. For the guillotine 2DSKP the solver proved optimality in 14 cases for the FC4_Base model and 23 cases for the FC4_Strengthened model out of 92 cases tested.

For the 2DSLOPP, a similar behaviour of 2DSKP was observed. The FC4_Base and FC4_Strengthened models performed better for the unweighted version (solver proved optimality in 25 cases out of 92) of the problem than the weighted (solver proved optimality in 13 cases out of 92). Besides, for the 2DSLOPP R/U version, the

Table 8.5: Computational results for guillotine SLOPP.

Instance	h	F/U						OPT	h	R/U						OPT
		FC4_Base			FC4_Strengthened					FC4_Base			FC4_Strengthened			
		Z	Gap^s(%)	T(s)	Z	Gap^s(%)	T(s)			Z	Gap^s(%)	T(s)	Z	Gap^s(%)	T(s)	
cgcut1		144	4.17	TL	144	4.17	TL	-		**150**	**0.00**	22.24	**150**	**0.00**	4.46	150
cgcut2		2740	2.19	TL	2778	0.79	TL	-		2776	0.87	TL	2751	1.78	TL	-
cgcut3		2721	2.90	TL	2721	2.90	TL	-		2754	1.67	TL	2754	1.67	TL	-
CW1		12825	2.34	TL	12777	2.72	TL	-		12906	1.70	TL	13053	0.55	TL	-
CW2		23240	2.95	TL	23418	2.17	TL	-		23607	1.35	TL	23670	1.08	TL	-
CW3		54063	2.23	TL	54063	2.23	TL	-		54608	1.21	TL	54290	1.80	TL	-
okp1		9938	0.62	TL	9938	0.62	TL	-		9968	0.32	TL	9960	0.40	TL	-
okp2		9681	3.30	TL	9743	2.64	TL	-		9842	1.61	TL	9842	1.61	TL	-
okp3		9838	1.65	TL	9812	1.92	TL	-		9893	1.08	TL	9893	1.08	TL	-
okp4		9976	0.24	TL	9976	0.24	TL	-		9977	0.23	TL	9979	0.21	TL	-
okp5		9982	0.18	TL	9952	0.48	TL	-		**10000**	**0.00**	20.64	**10000**	**0.00**	31.60	10000
ngcut1	4	95	5.26	TL	95	5.26	TL	-	4	97	3.09	TL	97	3.09	TL	-
ngcut2		97	3.09	TL	97	3.09	TL	-		**100**	**0.00**	6.24	**100**	**0.00**	2.67	100
ngcut3		**100**	**0.00**	8.97	**100**	**0.00**	1.84	100		**100**	**0.00**	6.00	**100**	**0.00**	5.51	100
ngcut4		138	2.17	TL	138	**0.00**	5.47	-		138	2.17	TL	138	**0.00**	99.24	-
ngcut5		140	7.14	TL	140	**0.00**	8.00	-		**150**	**0.00**	2.48	**150**	**0.00**	1.52	150
ngcut6		148	1.35	TL	148	1.35	TL	-		**150**	**0.00**	8.75	**150**	**0.00**	3.54	150
ngcut7		175	**0.00**	0.68	175	**0.00**	0.42	-		175	**0.00**	1.90	175	**0.00**	1.48	-
ngcut8		380	5.26	TL	380	5.26	TL	-		386	3.63	TL	386	3.63	TL	-
ngcut9		390	2.56	TL	390	2.56	TL	-		**400**	**0.00**	54.89	**400**	**0.00**	21.20	400
ngcut10		879	2.39	TL	879	**0.00**	32.55	-		879	2.39	TL	879	2.39	TL	-
ngcut11		842	6.89	TL	842	6.89	TL	-		873	3.09	TL	873	3.09	TL	-
ngcut12		898	0.22	TL	898	0.22	TL	-		**900**	**0.00**	36.44	**900**	**0.00**	11.12	900
Average			2.57	823.07		1.98	707.23				1.06	555.37		0.97	517.19	
		F/W						OPT		R/W						OPT
cgcut1		**244**	6.56	TL	**244**	6.56	TL	244		260	**0.00**	25.99	260	**0.00**	31.03	-
cgcut2		2856	2.24	TL	2856	2.24	TL	2892		2861	2.06	TL	2865	1.92	TL	-
cgcut3		**1860**	8.60	TL	**1860**	8.60	TL	1860		1900	7.37	TL	1900	7.37	TL	-
CW1		**6402**	11.67	TL	**6402**	11.67	TL	6402		6766	6.06	TL	6766	6.06	TL	-
CW2		**5354**	12.53	TL	**5354**	12.53	TL	5354		5604	7.51	TL	5604	7.51	TL	-
CW3		**5689**	6.52	TL	**5689**	6.40	TL	5689		5689	6.86	TL	5689	6.52	TL	-
okp1		**27589**	5.97	TL	**27589**	5.88	TL	27589		28423	2.86	TL	28090	3.99	TL	-
okp2		21976	10.90	TL	21947	11.05	TL	22503		23523	5.95	TL	23513	5.83	TL	-
okp3		23743	11.83	TL	23743	11.81	TL	24019		25174	6.40	TL	24881	7.81	TL	-
okp4		**32893**	3.00	TL	**32893**	2.88	TL	32893		32893	3.34	TL	32893	3.63	TL	-
okp5		**27923**	4.33	TL	**27923**	4.20	TL	27923		27983	4.15	TL	27983	4.15	TL	-
ngcut1	4	164	9.15	TL	164	7.32	TL	-	4	193	4.15	TL	193	4.15	TL	-
ngcut2		230	11.74	TL	230	8.70	TL	-		250	2.40	TL	250	2.40	TL	-
ngcut3		247	7.69	TL	247	6.88	TL	-		259	2.70	TL	259	2.70	TL	-
ngcut4		268	**0.00**	369.86	268	**0.00**	7.73	-		268	**0.00**	345.72	268	**0.00**	72.77	-
ngcut5		358	4.19	TL	358	**0.00**	9.93	-		370	**0.00**	676.13	370	**0.00**	33.34	-
ngcut6		289	9.69	TL	289	4.50	TL	-		298	6.38	TL	298	6.38	TL	-
ngcut7		430	**0.00**	0.58	430	**0.00**	0.60	-		430	**0.00**	1.11	430	**0.00**	1.07	-
ngcut8		834	12.47	TL	834	12.47	TL	-		886	5.87	TL	886	5.87	TL	-
ngcut9		924	4.11	TL	924	**0.00**	306.37	-		924	4.11	TL	924	4.11	TL	-
ngcut10		1452	**0.00**	113.35	1452	**0.00**	313.26	-		1452	4.82	TL	1452	4.48	TL	-
ngcut11		1688	11.85	TL	1688	8.59	TL	-		1786	6.27	TL	1786	5.32	TL	-
ngcut12		1865	9.01	TL	1865	9.01	TL	-		1932	6.73	TL	1932	5.07	TL	-
Average			7.13	804.51		6.14	732.86				4.17	789.91		4.14	750.31	

average gap for the FC4_Strengthened model was less than 1%, and full occupancy was obtained for 8 instances, 73% of ngcut instances.

8.6.3 The 'floating cuts' MILP model for the guillotine k-staged two-dimensional SLOPP

In practice, it is common to consider an additional constraint related to the number of stages regarding guillotine cutting patterns. The MILP model (8.1) - (8.44) allows guillotine cutting patterns of type non-staged. However, in some cases, the number of stages should be limited due to machinery constraints. Introducing this constraint of the number of stages in MILP model (8.1) - (8.44) is not straightforward. This section proposes a new MILP for k-staged cutting patterns based on the one presented to deal with first-order non-guillotine cutting problems.

The MILP model for the k-staged problem is based on the assumption that in a given rectangle, only three situations may occur: the rectangle is cut horizontally; the rectangle is cut vertically; or an item is assigned to the rectangle. If a rectangle is cut vertically, two new rectangles are obtained, i.e., the left and the right sub-rectangles. The width of the new sub-rectangles is equal to the width of the rectangle/sub-rectangle of the previous level and the sum of the lengths of the left and right sub-rectangles is equal to the length of the rectangle/sub-rectangle of the previous level. Similarly, if a horizontal cut is performed, two new rectangles are obtained, i.e., the top and bottom sub-rectangles. Their dimensions can also be calculated considering the dimensions of the rectangle/sub-rectangle of the previous level. The MILP model for the k-staged problem hereafter will be named "FC2", since from a cut a maximum of two sub-rectangles are generated.

In Figure 8.10, rectangles and sub-rectangles are numbered on the top left corner. In rectangle j, two types of cuts can be performed, i.e., vertical and horizontal. If a vertical cut is performed, a maximum of two new sub-rectangles is obtained, i.e., sub-rectangle $j + 1$ (left) and sub-rectangle $j + 2$ (right), and their dimensions are obtained considering the dimension of rectangle j. If a horizontal cut is performed, a maximum of two new sub-rectangles is created, i.e., sub-rectangle $j + 3$ (top) and sub-rectangle $j + 4$ (bottom), and, similarly, their dimensions are obtained considering the dimensions of rectangle j. The new sub-rectangles can be further cut vertically or horizontally, in a recursive way.

In Figure 8.11 (a), an example of a tree of sub-rectangles is presented. If index 0 is

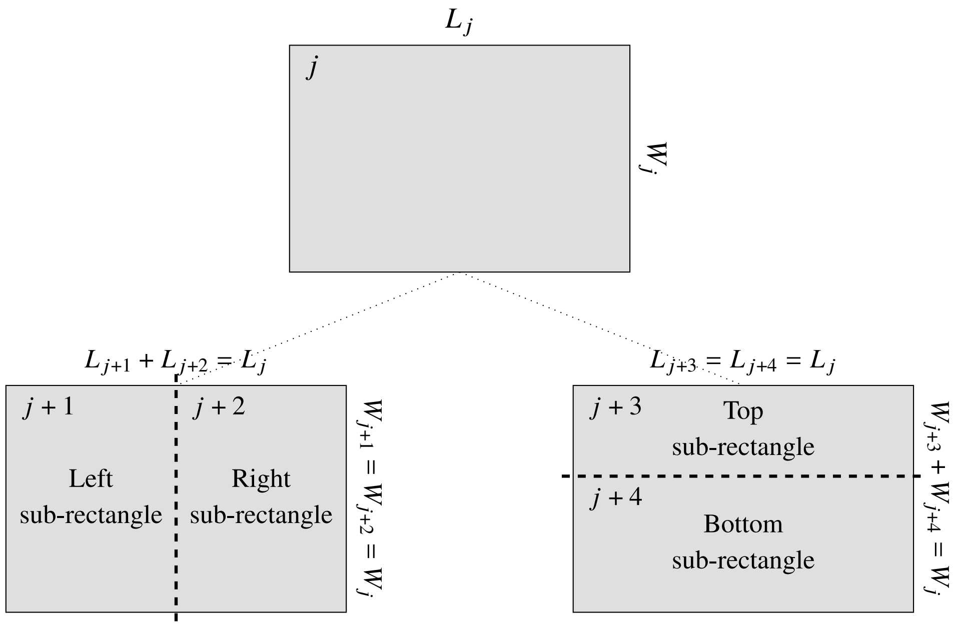
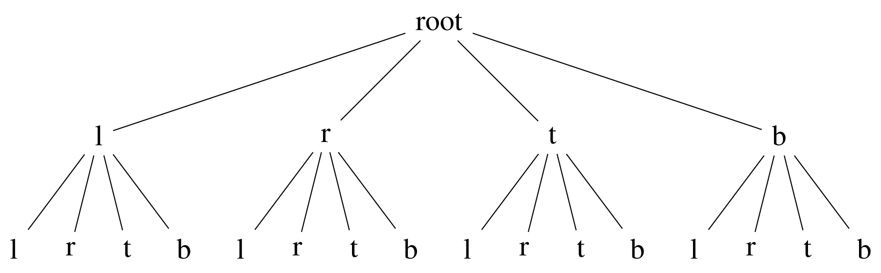
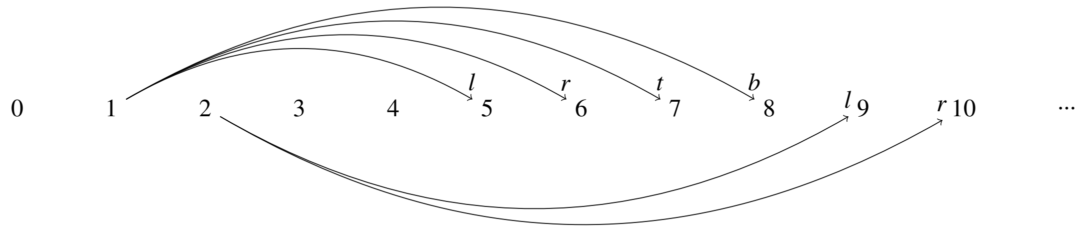


Figure 8.10: Types of cuts and sub-rectangles generated.

assigned to the root node (initial plate), indexes 1, 2, 3, and 4 are assigned to the left, right, top and bottom sub-rectangles (respectively). In the same way, if sub-rectangle 1 is cut, it can result in sub-rectangles 5, 6, 7 and 8 (left, right, top, and bottom), and so forth for the other sub-rectangles. An illustration of the sub-rectangles indexing is presented in Figure 8.11 (b).



(a) Tree example



(b) Sub-rectangle index

Figure 8.11: Example.

Decision variables δ_{ij}, s_{ij}, t_{ij}, z_j, L_j and W_j remain as in the formulation for the first-order non-guillotine cutting patterns. The new decision variables considered are the following:

$$x_j = \begin{cases} 1, \text{if sub-rectangle } j \text{ is cut vertically}, & j = 0, ..., m \\ 0, \text{ otherwise} \end{cases}$$

$$y_j = \begin{cases} 1, \text{if sub-rectangle } j \text{ is cut horizontally}, & j = 0, ..., m \\ 0, \text{ otherwise} \end{cases}$$

$$\begin{aligned}
&\text{Maximize} && (8.1) && \\
&\text{Subject to:} && x_j + y_j + \sum_{i=1}^{n} \delta_{ij} \leq 1, && \forall j = 0, ..., m; \quad (8.45) \\
& && (8.3)\ -\ (8.14), &&
\end{aligned}$$

The objective function is the same as (8.1) and constraints (8.3) - (8.14) remain from the model for the first-order non-guillotine cutting patterns. Constraints (8.2) are replaced by (8.45), ensuring that a sub-rectangle j can be cut vertically or horizontally or an item is assigned to it.

The constraints of the new MILP for the k-staged problem are grouped regarding the vertical and horizontal cuts constraints and the decision variables domain. The parameter D used in those constraints stands for the rectangular plate's largest dimension, i.e., $D = max\{L, W\}$.

8.6.3.1 Vertical cuts constraints

$$L_{l(j)} + L_{r(j)} \leq D(1 - x_j) + L_j, \qquad \forall j = 0, ..., m; \qquad (8.46)$$

$$L_{l(j)} + L_{r(j)} \geq L_j - D(1 - x_j), \qquad \forall j = 0, ..., m; \qquad (8.47)$$

$$L_{l(j)} + L_{r(j)} \leq D \cdot x_j, \qquad \forall j = 0, ..., m; \qquad (8.48)$$

$$W_{l(j)} \leq D(1 - x_j) + W_j, \qquad \forall j = 0, ..., m; \qquad (8.49)$$

$$W_{l(j)} \geq D(x_j - 1) + W_j, \qquad \forall j = 0, ..., m; \qquad (8.50)$$

$$W_{r(j)} \leq D(1 - x_j) + W_j,, \qquad \forall j = 0, ..., m; \qquad (8.51)$$

$$W_{r(j)} \geq D(x_j - 1) + W_j, \qquad \forall j = 0, ..., m; \qquad (8.52)$$

$$W_{l(j)} + W_{r(j)} \leq 2 \cdot D \cdot x_j, \qquad \forall j = 0, ..., m; \qquad (8.53)$$

Constraints (8.46) - (8.53) are associated with vertical cuts performed in the father sub-rectangles. Constraints (8.46) - (8.47) ensure that when a vertical cut is performed, the sum of the lengths of the resulting left and right sub-rectangles is equal to the length of the father sub-rectangle. Similarly, constraints (8.49) - (8.52) ensure that when a vertical cut is performed, the width of the resulting left and right sub-rectangles is equal to the width of the father sub-rectangle. Oppositely, if a vertical cut is not performed on the father sub-rectangle ($x_j = 0$), constraints (8.48) and (8.53) ensure that the length and the width of the left and right sub-rectangles are set to zero, respectively.

8.6.3.2 Horizontal cuts constraints

$$W_{t(j)} + W_{b(j)} \leq D(1 - y_j) + W_j, \qquad \forall j = 0, ..., m; \qquad (8.54)$$

$$W_{t(j)} + W_{b(j)} \geq W_j - D(1 - y_j), \qquad \forall j = 0, ..., m; \qquad (8.55)$$

$$W_{t(j)} + W_{b(j)} \leq D \cdot y_j, \qquad \forall j = 0, ..., m; \qquad (8.56)$$

$$L_{t(j)} \leq D(1 - y_j) + L_j, \qquad \forall j = 0, ..., m; \qquad (8.57)$$

$$L_{t(j)} \geq D(y_j - 1) + L_j, \qquad \forall j = 0, ..., m; \qquad (8.58)$$

$$L_{b(j)} \leq D(1 - y_j) + L_j, \qquad \forall j = 0, ..., m; \qquad (8.59)$$

$$L_{b(j)} \geq D(y_j - 1) + L_j, \qquad \forall j = 0, ..., m; \qquad (8.60)$$

$$L_{t(j)} + L_{b(j)} \leq 2 \cdot D \cdot y_j, \qquad \forall j = 0, ..., m; \qquad (8.61)$$

Constraints (8.54) - (8.61) are related to horizontal cuts performed in the father sub-rectangle. Constraints (8.54) and (8.55) ensure that if a horizontal cut is performed in the father sub-rectangle, the sum of the width of the resulting top and bottom sub-rectangles is equal to the width of the father sub-rectangle. Constraints (8.57) - (8.60) guarantee that the length of the resulting top and bottom sub-rectangles is equal to the length of the father sub-rectangle.

Constraints (8.56) and (8.61) ensure that if a horizontal cut is not performed ($y_j = 0$) in the father sub-rectangle, the length and the width of the top and bottom sub-rectangles are set to zero, respectively.

8.6.3.3 Decision variables domain

$$\begin{aligned} &(8.30) - (8.33), && \\ &x_j, y_j \in \{0,1\}, && \forall j = 0,...,m; \qquad (8.62) \end{aligned}$$

Decision variables related to the type of cut are binary, as can be observed in constraints (8.62).

8.6.3.4 Sub-rectangles index enumeration

Similarly to the MILP for the non-guillotine problem, it is possible to enumerate all possible indexes of the sub-rectangles of the tree and store them in vector M^h. In this case, a tree with h levels has $\sum_{j=0}^{h} 4^j$ elements.

Every element j (not a leaf) has 4 children in the M^h vector (left, right, top, and bottom), which can be traced: $l = 4 \cdot j + 1$; $r = 4 \cdot j + 2$; $t = 4 \cdot j + 3$; e $b = 4 \cdot j + 4$. Therefore, given an index j ($\forall j \neq 0$), it is possible to obtain the sub-rectangle's relative position (left, right, top, and bottom) adapting Algorithm 2 and use an adaptation of Algorithm 3 to identify the father of a sub-rectangle j.

8.6.3.5 Model strengthening

To strengthen the MILP formulation for the staged problem, a set of valid inequality constraints were added to the model. The area bound (8.36) is also considered.

Last level rectangles

Similarly to constraints (8.37), constraints (8.63) ensure that the sub-rectangles of the last level cannot be further cut, i.e., only items can be assigned to them.

$$x_j + y_j = 0, \qquad \forall j = |M^{h-1}|, ..., |M^h| \qquad (8.63)$$

Symmetry reduction

$$L_{4\cdot j+1} - L_{4\cdot j+2} \geq D(x_j - 1), \qquad \forall j = 0, ..., |M^{h-1}| \tag{8.64}$$

$$W_{4\cdot j+3} - W_{4\cdot j+4} \geq D(y_j - 1), \qquad \forall j = 0, ..., |M^{h-1}| \tag{8.65}$$

Constraints (8.64) and (8.65) are used to reduce symmetry in the generation of sub-rectangles by imposing that the left (top) sub-rectangle is larger than the right (bottom) sub-rectangle when a vertical (horizontal) cut is applied.

Relation between father and children sub-rectangles

$$\sum_{k=1}^{4}(x_{4\cdot j+k} + y_{4\cdot j+k} + \sum_{i=1}^{n} \delta_{i(4\cdot j+k)}) \leq 2 - 2 \cdot \sum_{i=1}^{n} \delta_{ij}, \qquad \forall j = 0, ..., |M^{h-2}| \tag{8.66}$$

$$\sum_{k=1}^{4}\sum_{i=1}^{n} \delta_{i(4\cdot j+k)} \leq 2 - 2 \cdot \sum_{i=1}^{n} \delta_{ij}, \quad \forall j = |M^{h-2}|, ..., |M^{h-1}| \tag{8.67}$$

Constraints (8.66) and (8.67) ensure that if an item is allocated to a sub-rectangle j, then j has no descendants.

$$\sum_{k=1}^{4}(x_{4\cdot j+k} + y_{4\cdot j+k} + \sum_{i=1}^{n} \delta_{i(4\cdot j+k)}) \leq 2 \cdot (x_j + y_j), \qquad \forall j = 0, ..., |M^{h-2}| \tag{8.68}$$

$$\sum_{k=1}^{4}\sum_{i=1}^{n} \delta_{i(4\cdot j+k)} \leq 2 \cdot (x_j + y_j), \quad \forall j = |M^{h-2}|, ..., |M^{h-1}| \tag{8.69}$$

Constraints (8.68) and (8.69) ensure that if a sub-rectangle j is not cut, then j has no descendants. This condition prevents a rectangle from remaining "undecided" (i.e., it is neither cut nor it has an item assigned to it) along several levels of the tree.

8.6.3.6 k-staged cutting patterns

The limitation on the number of stages can be considered by simply removing the decision variables associated with the sub-rectangles that require the generation of a number of stages larger than the k stage limit. To identify these decision variables, we need to calculate how many changes in the orientation of the cuts (horizontal or vertical) need to occur up to the j sub-rectangle. For example, if a sub-rectangle is generated after two vertical cuts, followed by a horizontal cut, this corresponds to two stages. Algorithm 5 (available in Appendix) determines the number of changes in the orientation of the guillotine cuts for a given sub-rectangle j.

8.6.3.7 Item rotation

To consider item rotation, as it is done in a generalized way, a replica of each item rotated by 90 degrees (the length of the replica is equal to the width of the original item, and vice-versa) is added to the data. Constraints (8.3) should be updated accordingly to guarantee that the original item and its replica do not exceed the original item's demand.

An interesting insight pops up when item rotation is allowed. Until now, we have considered that the first cut on the plate could be either vertical or horizontal. However, when items can rotate, we can fix the first cut (to make the explanation simpler, let us assume that we fix it vertically) without loss of generality/optimality, as any cutting pattern we would obtain with a horizontal first cut can be obtained with a vertical first cut using the rotated items **and a rotated plate**. Figure (8.12) visually depicts this insight.

This means that if we allow the rotation of the plate, we can fix the first cut and cut off half of the search tree, at the cost of inserting one additional binary variable, which stands for the decision of rotating the plate or not. Therefore, to the previously presented model, we insert a new binary variable r:

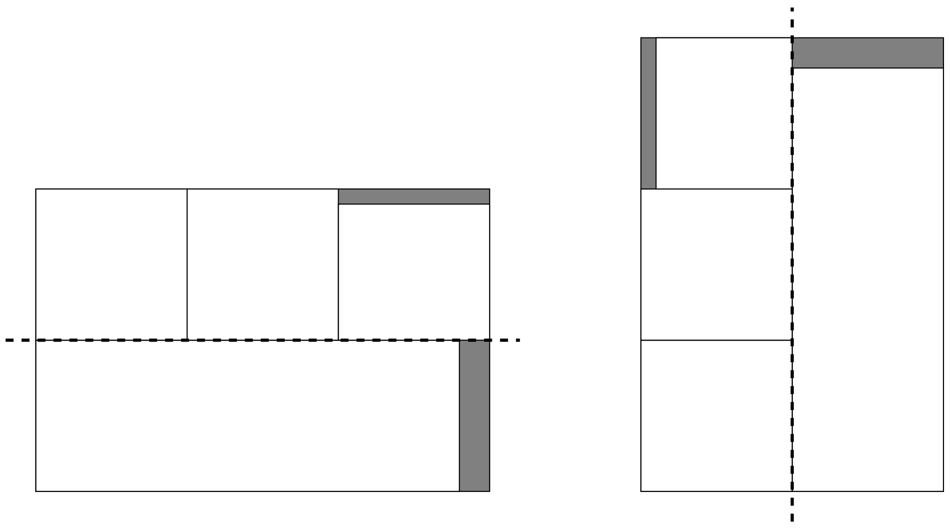


Figure 8.12: A cutting pattern with a horizontal first cut, and the correspondent pattern with the items and the plate rotated obtained with a vertical first cut.

$$r \in \{0, 1\}; \tag{8.70}$$

a new constraint forcing the first cut to be vertical:

$$x_0 = 1; \tag{8.71}$$

and replace constraints (8.12) and (8.13) with the following constraints:

$$L_0 = W \cdot r + L \cdot (1 - r); \tag{8.72}$$

$$W_0 = L \cdot r + W \cdot (1 - r); \tag{8.73}$$

heavily reducing symmetry in the model.

8.7 Computational experiments - guillotine problem

In this section, the computational experiments are divided into two parts. Firstly it is presented a comparison between the model for first-order non-guillotine patterns with guillotine constraints ((8.1) - (8.44)) the FC4_Strengthened, and the model designed explicitly for guillotine cutting patterns “FC2”. Secondly, it is presented a comparison of the FC4_Strengthened and FC2 models proposed for guillotine cutting patterns with the existing models from the literature. Graphical examples of guillotine solutions obtained are provides in Appendix.

8.7.1 Non-staged 2DSKP and 2DSLOPP - comparison between FC4_-Strengthened and FC models

Similarly to what was done for the model for first-order non-guillotine cutting patterns, preliminary experiments were conducted to determine the best h for the FC2 model. In Appendix, Tables 3 and 4 for 2DSKP and 2DSLOPP, respectively, present all the experiments. We performed an analysis for the parameter h similar to that in Section 8.5, although we do not present the charts in this section. In the computational experiments for the 2DSKP, it will be considered $h = 6$, and for the 2DSLOPP, it will be $h = 5$.

The comparison between both models is presented in Tables 8.6 and 8.7. For the 2DSKP, the FC4_Strengthened model performed better in problem variants F/U, F/W and R/U in terms of the number of optimal solutions proved by the solver and regarding the average gap. The opposite behaviour was found in problem variant R/W, in which the FC2 model had six optimal solutions against 5 from FC4_Strengthened model.

Regarding the 2DSLOPP, the FC2 model had the worst performance regarding the gap with an average value of 39.99%, while the gap for the FC4_Strengthened model was 5.42%. In Figure 8.13, it can be seen that the worst gaps were obtained for variants F/W and R/W.

For 2DSLOPP variant R/U the full occupation was found in seven instances with both models. The solver proved optimality in these seven instances with the

Table 8.6: Computational results for guillotine SKP comparison between FC4_-Strengthened and FC2.

	F/U								OPT	R/U								OPT
Instance	FC4_Strengthened				FC2					FC4_Strengthened				FC2				
	h	Z	Gap^s(%)	T(s)	h	Z	Gap^s(%)	T(s)		h	Z	Gap^s(%)	T(s)	h	Z	Gap^s(%)	T(s)	
cgcut1		144	4.17	TL		144	4.17	TL	-		**150**	**0.00**	12.41		**150**	**0.00**	28.34	150
cgcut2		2740	2.19	TL		2740	2.19	TL	-		2769	1.12	TL		2678	4.56	TL	-
cgcut3		2721	2.90	TL		2721	2.90	TL	-		2754	1.67	TL		2754	1.67	TL	-
CW1		12797	2.56	TL		12797	2.56	TL	-		12797	2.56	TL		12777	2.72	TL	-
CW2		23240	2.95	TL		23240	2.95	TL	-		23643	1.19	TL		23299	2.69	TL	-
CW3		53755	2.82	TL		53755	2.82	TL	-		54070	2.22	TL		54022	2.31	TL	-
okp1		8836	13.17	TL		8836	13.17	TL	-		9448	5.84	TL		9448	5.84	TL	-
okp2		9749	2.58	TL		9615	4.00	TL	-		9828	1.75	TL		9863	1.39	TL	-
okp3		9838	1.65	TL		9838	1.65	TL	-		9893	1.08	TL		9751	2.55	TL	-
okp4		9818	1.85	TL		9677	3.34	TL	-		9810	1.94	TL		9823	1.80	TL	-
okp5		9749	2.58	TL		9615	4.00	TL	-		9780	2.25	TL		9863	1.39	TL	-
ngcut1	4	95	**0.00**	326.07	6	95	5.26	TL	-	4	97	3.09	TL	6	77	23.38	TL	-
ngcut2		97	3.09	TL		97	3.09	TL	-		**100**	**0.00**	11.38		92	5.44	TL	100
ngcut3		**100**	**0.00**	4.83		**100**	**0.00**	115.16	100		**100**	**0.00**	19.53		**100**	**0.00**	76.55	100
ngcut4		138	**0.00**	25.64		138	2.17	TL	-		138	**0.00**	18.13		108	**0.00**	2.81	-
ngcut5		140	**0.00**	10.28		140	7.14	TL	-		**150**	**0.00**	2.38		143	4.90	TL	150
ngcut6		148	1.35	TL		148	1.35	TL	-		**150**	**0.00**	13.48		148	1.35	TL	150
ngcut7		175	**0.00**	0.74		175	**0.00**	5.41	-		175	**0.00**	2.16		154	**0.00**	1.55	-
ngcut8		380	5.26	TL		380	5.26	TL	-		386	3.63	TL		312	**0.00**	3.32	-
ngcut9		390	2.56	TL		387	3.36	TL	-		**400**	**0.00**	21.81		391	2.30	TL	400
ngcut10		879	**0.00**	18.97		660	**0.00**	9.58	-		879	0.91	TL		690	**0.00**	2.10	-
ngcut11		842	6.89	TL		651	15.05	TL	-		873	3.09	TL		749	**0.00**	2.93	-
ngcut12		894	0.67	TL		792	13.64	TL	-		895	0.56	TL		820	9.76	TL	-
Average			2.58	682.76			4.35	788.27				1.43	592.01			3.22	631.20	
	F/W								OPT	R/W								OPT
cgcut1		**244**	6.15	TL		**244**	7.79	TL	244		260	**0.00**	97.64		260	**0.00**	447.82	-
cgcut2		2856	2.24	TL		2856	2.24	TL	2892		2848	2.53	TL		2860	2.10	TL	-
cgcut3		**1860**	8.60	TL		**1860**	8.60	TL	1860		1900	6.32	TL		1900	6.32	TL	-
CW1		4837	10.17	TL		4837	11.83	TL	-		4930	8.62	TL		4874	10.69	TL	-
CW2		4673	15.37	TL		4673	15.77	TL	-		5017	7.57	TL		5062	6.82	TL	-
CW3		4593	16.07	TL		4593	15.76	TL	-		4936	7.68	TL		4874	9.23	TL	-
okp1		22752	17.38	TL		22752	17.88	TL	-		25721	4.14	TL		25721	4.18	TL	-
okp2		22502	8.33	TL		21416	16.59	TL	-		23513	5.86	TL		23513	5.80	TL	-
okp3		24019	10.84	TL		23740	13.09	TL	-		25216	6.22	TL		25216	6.18	TL	-
okp4		25939	11.23	TL		25939	12.50	TL	-		27568	5.25	TL		27568	5.33	TL	-
okp5		21814	12.08	TL		21127	15.77	TL	-		24263	2.58	TL		23513	6.00	TL	-
ngcut1	4	164	**0.00**	243.10	6	141	**0.00**	16.47	-	4	193	4.15	TL	6	145	15.86	TL	-
ngcut2		230	2.61	TL		198	**0.00**	37.81	-		250	1.20	TL		198	4.04	TL	-
ngcut3		247	7.29	TL		201	19.90	TL	-		259	2.70	TL		214	12.62	TL	-
ngcut4		268	**0.00**	34.42		207	**0.00**	3.84	-		268	**0.00**	36.21		207	**0.00**	2.13	-
ngcut5		358	**0.00**	8.93		262	12.98	TL	-		370	**0.00**	10.36		284	4.23	TL	-
ngcut6		289	4.15	TL		282	9.57	TL	-		298	6.38	TL		298	3.69	TL	-
ngcut7		430	**0.00**	0.73		374	**0.00**	4.67	-		430	**0.00**	2.14		374	**0.00**	3.55	-
ngcut8		834	12.47	TL		673	**0.00**	7.03	-		886	5.87	TL		673	**0.00**	3.77	-
ngcut9		924	**0.00**	23.93		839	3.34	TL	-		908	5.95	TL		839	3.46	TL	-
ngcut10		1452	**0.00**	16.41		998	**0.00**	7.87	-		1452	4.48	TL		1063	**0.00**	2.89	-
ngcut11		1688	9.66	TL		1195	20.42	TL	-		1780	5.45	TL		1439	**0.00**	4.33	-
ngcut12		1865	9.38	TL		1615	14.30	TL	-		1932	5.59	TL		1661	8.31	TL	-
Average			7.13	680.20			9.49	668.60				4.28	750.67			4.99	685.41	

FC4_Strengthened model, while with the FC2 model it only proved optimality in two instances.

The optimal solution in 2DSLOPP variant F/W was known from the literature in 11 instances (cgcut1-3, CW1-3, okp1-5). Both models reached the optimal solution in eight instances. However, the solver could not prove it and the gaps are much smaller in the FC4_Strengthened model compared to the FC2 model.

Table 8.7: Computational results for guillotine SLOPP comparison between FC4_Strengthened and FC2.

Instance	F/U								OPT	R/U								OPT
	FC4_Strengthened				FC2					FC4_Strengthened				FC2				
	h	Z	Gap^s(%)	T(s)	h	Z	Gap^s(%)	T(s)		h	Z	Gap^s(%)	T(s)	h	Z	Gap^s(%)	T(s)	
cgcut1		144	4.17	TL		144	56.25	TL	-		**150**	**0.00**	4.46		**150**	50.00	TL	150
cgcut2		2778	0.79	TL		2740	58.54	TL	-		2751	1.78	TL		2757	57.56	TL	-
cgcut3		2721	2.90	TL		2721	147.01	TL	-		2754	1.67	TL		2754	164.60	TL	-
CW1		12777	2.72	TL		12885	175.94	TL	-		13053	0.55	TL		12972	262.00	TL	-
CW2		23418	2.17	TL		23196	319.65	TL	-		23670	1.08	TL		23670	207.02	TL	-
CW3		54063	2.23	TL		53983	350.35	TL	-		54290	1.80	TL		54608	355.14	TL	-
okp1		9938	0.62	TL		9922	213.18	TL	-		9960	0.40	TL		9960	186.37	TL	-
okp2		9743	2.64	TL		9534	4.89	TL	-		9842	1.61	TL		9828	1.75	TL	-
okp3		9812	1.92	TL		9681	3.30	TL	-		9893	1.08	TL		9755	2.51	TL	-
okp4		9976	0.24	TL		9852	203.09	TL	-		9979	0.21	TL		9979	156.30	TL	-
okp5		9952	0.48	TL		9965	291.51	TL	-		**10000**	**0.00**	31.60		**10000**	243.80	TL	10000
ngcut1	4	95	5.26	TL	5	95	**0.00**	476.36	-	4	97	3.09	TL	5	97	22.68	TL	-
ngcut2		97	3.09	TL		97	22.68	TL	-		**100**	**0.00**	2.67		**100**	55.00	TL	100
ngcut3		**100**	**0.00**	1.84		**100**	105.00	TL	100		**100**	**0.00**	5.51		**100**	102.00	TL	100
ngcut4		138	**0.00**	5.47		138	**0.00**	392.35	-		138	**0.00**	99.24		138	**0.00**	304.16	-
ngcut5		140	**0.00**	8.00		140	**0.00**	288.20	-		**150**	**0.00**	1.52		**150**	18.00	TL	150
ngcut6		148	1.35	TL		148	63.51	TL	-		**150**	**0.00**	3.54		**150**	69.33	TL	150
ngcut7		175	**0.00**	0.42		175	**0.00**	3.40	-		175	**0.00**	1.48		175	**0.00**	1.97	-
ngcut8		380	5.26	TL		380	56.58	TL	-		386	3.63	TL		386	63.99	TL	-
ngcut9		390	2.56	TL		390	59.74	TL	-		**400**	**0.00**	21.20		**400**	65.50	TL	400
ngcut10		879	**0.00**	32.55		879	**0.00**	666.14	-		879	2.39	TL		879	19.57	TL	-
ngcut11		842	6.89	TL		842	14.61	TL	-		873	3.09	TL		873	49.26	TL	-
ngcut12		898	0.22	TL		894	68.12	TL	-		**900**	**0.00**	11.12		**900**	71.33	TL	900
Average			1.98	707.23			96.26	783.76				0.97	517.19			96.68	835.05	
	F/W								OPT	R/W								OPT
cgcut1		**244**	6.56	TL		**244**	49.18	TL	244		260	**0.00**	31.03		260	40.00	TL	-
cgcut2		2856	2.24	TL		2856	55.78	TL	2892		2865	1.92	TL		2863	55.40	TL	-
cgcut3		**1860**	8.60	TL		**1860**	112.37	TL	1860		1900	7.37	TL		1900	169.47	TL	-
CW1		**6402**	11.67	TL		**6402**	148.56	TL	6402		6766	6.06	TL		6766	165.36	TL	-
CW2		**5354**	12.53	TL		**5354**	177.16	TL	5354		5604	7.51	TL		5471	167.19	TL	-
CW3		**5689**	6.40	TL		**5689**	160.63	TL	5689		5689	6.52	TL		5744	196.38	TL	-
okp1		**27589**	5.88	TL		**27589**	219.06	TL	27589		28090	3.99	TL		28090	182.50	TL	-
okp2		21947	11.05	TL		21154	15.44	TL	22503		23513	5.83	TL		23513	5.99	TL	-
okp3		23743	11.81	TL		23743	12.48	TL	24019		24881	7.81	TL		24485	9.68	TL	-
okp4		**32893**	2.88	TL		**32893**	152.60	TL	32893		32893	3.63	TL		32893	140.94	TL	-
okp5		**27923**	4.20	TL		**27923**	308.26	TL	27923		27983	4.15	TL		27983	174.29	TL	-
ngcut1	4	164	7.32	TL	5	164	**0.00**	542.73	-	4	193	4.15	TL	5	193	**0.00**	645.25	-
ngcut2		230	8.70	TL		230	23.91	TL	-		250	2.40	TL		250	33.20	TL	-
ngcut3		247	6.88	TL		247	52.63	TL	-		259	2.70	TL		259	56.37	TL	-
ngcut4		268	**0.00**	7.73		268	**0.00**	336.92	-		268	**0.00**	72.77		268	**0.00**	353.22	-
ngcut5		358	**0.00**	9.93		358	**0.00**	295.82	-		370	**0.00**	33.34		370	**0.00**	417.62	-
ngcut6		289	4.50	TL		276	42.03	TL	-		298	6.38	TL		298	50.00	TL	-
ngcut7		430	**0.00**	0.60		430	**0.00**	2.88	-		430	**0.00**	1.07		430	**0.00**	2.54	-
ngcut8		834	12.47	TL		828	41.91	TL	-		886	5.87	TL		886	59.26	TL	-
ngcut9		924	**0.00**	306.37		924	35.82	TL	-		924	4.11	TL		924	40.58	TL	-
ngcut10		1452	**0.00**	313.26		1452	**0.00**	161.26	-		1452	4.48	TL		1452	10.40	TL	-
ngcut11		1688	8.59	TL		1688	17.36	TL	-		1786	5.32	TL		1786	43.84	TL	-
ngcut12		1865	9.01	TL		1865	60.75	TL	-		1932	5.07	TL		1932	59.52	TL	-
Average			6.14	732.86			73.30	762.59				4.14	750.31			72.19	805.16	

Summing up, the FC4_Strengthened model with guillotine constraint outperformed the FC2 model in both problem types. This supremacy is because when a cut is performed in this model, four new rectangles can be produced, while with the FC2 model, only two new rectangles are obtained. In this way, the FC2 model requires a deeper level in the tree search to find similar solutions, which results in a more complex tree search.

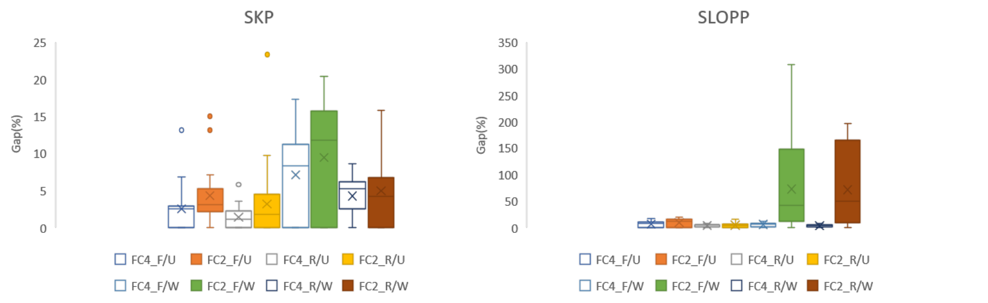


Figure 8.13: Analysis of the gap reported by Gurobi for the FC4_Strengthened and FC2 models.

8.7.2 Comparison with mathematical models from the literature

In this section, the performance of the 'floating cuts' FC4_Strengthened and FC2 models are compared with other exact approaches. For each instance we detail the number of item types n and the value of the optimal solution known from the literature (OPT), for each approach from the literature it is presented the gap to the optimal solution (Gap^{opt}(%)), and solution time (T(s)) reported by the authors. For the 'floating cuts' model, the following is presented: the h considered, the gap to the optimal solution (Gap^{opt}(%)), the gap reported by the solver (Gap^{s}(%)) and the total time (T(s)). The gap to the optimal solution (Gap^{opt}(%))) is computed as $100 * (opt - ofv)/opt$, where opt and ofv are, respectively, the optimal solution and objective function value of the model.

2DSKP

The computational experiments were performed on the gcut1-12 instances originally proposed by Beasley [9]. Recently, these instances have been used in Furini et al. [44], Martin et al. [73], and Martin et al. [75].

For the 2-stages - F/U problem variant, FC2 model was considered, since it allows the limitation on the number of stages. Preliminary experiments were also conducted to define the best h to use for the variant 2-stages F/U and the results can be assessed in Appendix, Tables 3 and 4 for 2DSKP and 2DSLOPP, respectively. It was concluded that $h = 6$ will be used. It is considered that the first cut is horizontal, as in Martin et al. [73], so the decision variable related to the horizontal cut in the initial rectangle is fixed to one ($y_0 = 1$). The computational results are compared in Table 8.8. In Martin et al. [73], a time limit of one hour was defined, and the symbol "*" is used when the solver ran out of memory during its execution and a gap of 100 means that no integer solution was found.

The solver proved optimality for the 'floating cuts' model in all but one instance. Although the value obtained was the optimal (gcut8). The average time was 251.87 seconds. For the model from Martin et al. [73], the solver ran out of memory in three instances, and the average time was 333.74 seconds.

For the 2DSKP with guillotine constraints and no limitation on the number of stages, the FC4_Strengthened model will be used with $h = 4$ as previously defined in the preliminary experiments. The results of this model with gcut1-12 problem instances were compared with the results of Furini et al. [44], and Martin et al. [75], and can be analysed in Table 8.9.

The 'floating cuts' model and the 2d-Top model from Martin et al. [75] outperform the model from Furini et al. [44] since the solver proved optimality in only two out

Table 8.8: Computational results for 2-stage - F/U SKP comparison between Martin et al. [73] and FC2.

			2-stage - F/U						
Instance	n	OPT	Martin et al. [73]		FC2				
			Gap^{opt} (%)	T(s)	h	Gap^{opt} (%)	Gap^{s} (%)	T(s)
gcut1	10	43024	**0.00**	1.72		**0.00**	**0.00**	3.56
gcut2	20	57996	**0.00**	39.02		**0.00**	**0.00**	85.55
gcut3	30	59895	**0.00**	571.51		**0.00**	**0.00**	108.98
gcut4	50	60504	100.00	*		**0.00**	**0.00**	772.78
gcut5	10	193379	**0.00**	1.96		**0.00**	**0.00**	34.46
gcut6	20	224399	**0.00**	32.44	6	**0.00**	**0.00**	50.24
gcut7	30	238974	**0.00**	96.12		**0.00**	**0.00**	310.33
gcut8	50	245758	100.00	*		**0.00**	1.73	TL
gcut9	10	919476	**0.00**	1.35		**0.00**	**0.00**	29.10
gcut10	20	856445	**0.00**	15.32		**0.00**	**0.00**	108.22
gcut11	30	942219	**0.00**	2244.23		**0.00**	**0.00**	389.58
gcut12	50	970744	100.00	*		**0.00**	**0.00**	229.61
Average			25.00	333.74		**0.00**	0.14	251.87

Table 8.9: Computational results non-staged - F/U SKP comparing Furini et al. [44], Martin et al. [75] and FC4_Strengthened.

Instance	n	OPT	Non-staged - F/U							
			Furini et al. [44]		Martin et al. [75]		FC4_Strengthened			
			Gap^{opt} (%)	T(s)	Gap^{opt} (%)	T(s)	h	Gap^{opt} (%)	Gap^{s}(%)	T(s)
gcut1	10	48368	**0.00**	8.80	**0.00**	0.04	4	**0.00**	**0.00**	642.25
gcut2	20	59307	100.00	TL	**0.00**	4.65		**0.00**	5.38	TL
gcut3	30	60241	100.00	TL	**0.00**	13.12		**0.00**	3.75	TL
gcut4	50	60942	100.00	TL	**0.00**	509.84		**0.00**	2.56	TL
gcut5	10	195582	1.40	TL	**0.00**	0.21		**0.00**	**0.00**	646.35
gcut6	20	236305	100.00	TL	**0.00**	0.70		**0.00**	5.80	TL
gcut7	30	238974	100.00	TL	**0.00**	2.74		**0.00**	4.61	TL
gcut8	50	245758	100.00	TL	**0.00**	62.44		**0.00**	1.73	TL
gcut9	10	919476	**0.00**	2847.90	**0.00**	0.28		**0.00**	8.76	TL
gcut10	20	903435	100.00	TL	**0.00**	0.78		**0.00**	10.69	TL
gcut11	30	955389	100.00	TL	**0.00**	14.48		**0.00**	4.67	TL
gcut12	50	970744	100.00	TL	**0.00**	14.74		**0.00**	3.01	TL
Average			75.12	2152.23	**0.00**	52.00		**0.00**	4.25	857.38

of 12 instances. The solver performed very well with the 2d-Top model from Martin et al. [75], proving optimality in all instances with an average of 52 seconds. In comparison, the solver for the ‘floating cuts’ model could only prove one optimal solution. However, the value of all solutions is optimal since Gap^{opt}(%) was always zero.

2DSLOPP

For the 2DSLOPP in variant F/U with 2-stages as guillotine constraint, 24 problem instances proposed in Hifi and Roucairol [55] were considered in the computational experiments and compared with the results obtained with the exact model ‘M1’ proposed in Lodi and Monaci [70] with the first cut horizontal. The computational results can be analysed in Table 8.10. The FC2 model was used in these experiments due to the limitation on the number of stages.

For the integer programming model proposed in Lodi and Monaci [70], the solver was able to prove optimality in all problem instances with an average computational time of 43.11 seconds.

The solver proved optimality in 7 out of 24 instances for the ‘floating cuts model’. However, the optimal value of the solution is obtained for 15 out of the 24 instances.

Table 8.10: Computational results for 2-stage -F/U SLOPP comparison between Lodi and Monaci [70] and FC2.

			2-stage - F/U					
Instance	n	OPT	Lodi and Monaci [70]		FC2			
			Gap^{opt} (%)	T(s)	h	Gap^{opt} (%)	Gap^{s}(%)	T(s)
2s	10	2430	**0.00**	0.48		**0.00**	18.60	TL
3s	20	2599	**0.00**	0.33		**0.00**	**0.00**	383.60
A1s	20	2950	**0.00**	0.27		**0.00**	**0.00**	692.58
A2s	20	3423	**0.00**	2.57		**0.00**	**0.00**	819.96
STS2s	30	4569	**0.00**	10.12		0.04	40.53	TL
STS4s	20	9481	**0.00**	13.10		0.35	33.52	TL
OF1	10	2713	**0.00**	0.07		**0.00**	**0.00**	77.56
OF2	10	2515	**0.00**	0.28		**0.00**	**0.00**	260.17
W	20	2623	**0.00**	0.75		**0.00**	**0.00**	470.23
CHL1s	30	13036	**0.00**	4.30		0.07	46.50	TL
CHL2s	10	3162	**0.00**	0.18		**0.00**	14.80	TL
A3	20	5380	**0.00**	1.78	6	**0.00**	28.18	TL
A4	20	5885	**0.00**	1.58		**0.00**	22.18	TL
A5	20	12553	**0.00**	3.97		**0.00**	26.06	TL
CHL5	10	363	**0.00**	0.03		**0.00**	**0.00**	123.00
CHL6	30	16572	**0.00**	21.50		0.57	50.15	TL
CHL7	35	16728	**0.00**	54.23		1.14	48.25	TL
CU1	25	12312	**0.00**	11.78		**0.00**	43.79	TL
CU2	35	26100	**0.00**	3.67		**0.00**	28.71	TL
Hchl3s	10	11961	**0.00**	312.93		0.60	36.40	TL
Hchl4s	10	11408	**0.00**	402.13		0.47	20.50	TL
Hchl6s	22	60170	**0.00**	19.60		1.36	50.24	TL
Hchl7s	40	62459	**0.00**	168.20		0.56	87.98	TL
Hchl8s	10	729	**0.00**	0.72		**0.00**	6.72	TL
Average			**0.00**	43.11		0.21	25.13	756.00

For the 2DSLOPP with variant F/U with guillotine constrained with unlimited number of stages the following instances were used: OF1-2 originally proposed by Oliveira and Ferreira [80], CU1-11 proposed by Fayard et al. [38] and Wang20 proposed by Wang [107].

The results of the FC4_Strengthened for all problem instances can be compared with the results from Martin et al. [75]. In Martin et al. [75] a time limit of 900 seconds was considered. The gap presented in Martin et al. [75] is for the optimal solution, and no information is given about the gap of the solver. Only for instances of2 and wang20, the optimality was proven by the solver for the 2d-Top model from Martin et al. [75] since the time limit was not reached. For this model, the solver reached the optimal value in 7 instances out of 14, while the FC4_Strengthened reached the optimal solution in 4 instances, although the solver was not able to prove it. Regarding the value of the solution, the model from Martin et al. [75] obtained lower gaps than the proposed model in 8 instances. The proposed model reached lower gaps in three instances on the other side. In this way, both models seem to share the same weakness, the certificate of quality due to weak LP-bounds.

The computational results for OF1-2, CU1-2 and Wang20 problem instances, obtained by the ‘floating cuts’ model, were compared with the results of the mathematical model presented in Furini et al. [44]. The model from Furini et al. [44] was able to prove optimality in 4 out of 5 instances, while the proposed model reached the optimal value in 3 out of 5. However, for problem instance cu2, the solver was not able to find a feasible solution in 1 hour for the model from Furini et al. [44], while the solver stopped in 900 seconds for the proposed model with a gap of 0.56% for the optimal solution.

The three data sets used in the first set of computational experiments presented in Section 8.5 were also used for the 2DSLOPP with F/W variant (cgcut1-3, CW1-3, and okp1-5). Besides, problem instances CW4-11 from Fayard et al. [38] are also included.

As in the unweighted version of the problem, the results of the proposed model

Table 8.11: Computational results for non-staged - F/U SLOPP comparing Furini et al. [44], Martin et al. [75] and FC4_Strengthened.

			Non-staged - F/U							
Instance	n	OPT	Furini et al. [44]		Martin et al. [75]		FC4_Strengthened			
			Gap^{opt} (%)	T(s)	Gap^{opt} (%)	T(s)	h	Gap^{opt} (%)	Gap^{s}(%)	T(s)
of1	10	2737	**0.00**	53.90	**0.00**	TL	4	0.88	3.21	TL
of2	10	2690	**0.00**	66.00	**0.00**	5.94		**0.00**	4.09	TL
cu1	25	12330	**0.00**	1460.80	0.15	TL		**0.00**	1.38	TL
cu2	35	26100	100.00	TL	**0.00**	TL		0.56	1.14	TL
cu3	45	16723	-	-	0.69	TL		1.00	1.18	TL
cu4	45	99495	-	-	0.20	TL		0.23	1.64	TL
cu5	50	173364	-	-	0.53	TL		0.20	1.47	TL
cu6	45	158572	-	-	**0.00**	TL		0.51	0.87	TL
cu7	25	247150	-	-	**0.00**	TL		0.41	4.76	TL
cu8	35	433331	-	-	0.14	TL		0.26	2.68	TL
cu9	25	657055	-	-	**0.00**	TL		**0.00**	2.27	TL
cu10	40	773772	-	-	0.85	TL		0.51	1.59	TL
cu11	50	924696	-	-	2.02	TL		2.95	3.75	TL
wang20	19	2721	**0.00**	60.70	**0.00**	435.10		**0.00**	2.90	TL
Average			20.00	1048.28	0.33	803.79		0.54	2.35	TL

were compared to the results from Martin et al. [75], and Furini et al. [44].

The average gap obtained by the solver for the proposed model is larger for the weighted instances (10.45%) than the unweighted instances (2.35%). The proposed model reached the optimal solution in 14 out of 19 instances. Although, the solver was not able to prove optimality in none of them.

For the 14 instances solved with the model from Martin et al. [75], the solver reached the optimality in all but one instance. However, the solver proved the optimality in only 5 instances.

The solver was able to obtain the optimal solution for the model from Furini et al. [44] in 5 out of the 11 instances tested. The solver could not find a feasible solution in two instances (CW2 and CW3). For the instances in which the optimality was not reached for the model of Furini et al. [44], the proposed model could obtain lower gaps.

Table 8.12: Computational results for non-staged - F/W SLOPP comparing Furini et al. [44], Martin et al. [75] and FC4_Strengthened.

			Non-staged - F/W							
Instance	n	OPT	Furini et al. [44]		Martin et al. [75]		FC4_Strengthened			
			Gap^{opt} (%)	T(s)	Gap^{opt} (%)	T(s)	h	Gap^{opt} (%)	Gap^{s}(%)	T(s)
cgcut1	7	244	**0.00**	0.10	**0.00**	4.92		**0.00**	6.56	TL
cgcut2	10	2892	**0.00**	58.40	1.24	TL		1.24	2.24	TL
cgcut3	19	1860	**0.00**	58.60	**0.00**	8.28		**0.00**	8.60	TL
okp1	15	27589	**0.00**	490.50	-	-		**0.00**	5.88	TL
okp2	30	22503	30.00	TL	-	-		2.47	11.05	TL
okp3	30	24019	8.80	TL	-	-		1.15	11.81	TL
okp4	33	32893	**0.00**	684.40	-	-		**0.00**	2.88	TL
okp5	29	27923	**0.00**	2118.60	-	-		**0.00**	4.20	TL
CW1	25	6402	70.70	TL	**0.00**	TL		**0.00**	11.67	TL
CW2	35	5354	100.00	TL	**0.00**	409.66	4	**0.00**	12.53	TL
CW3	40	5689	100.00	TL	**0.00**	TL		**0.00**	6.40	TL
CW4	39	6175	-	-	**0.00**	TL		**0.00**	33.04	TL
CW5	35	11659	-	-	**0.00**	TL		**0.00**	7.69	TL
CW6	55	12923	-	-	**0.00**	TL		2.10	9.45	TL
CW7	45	9898	-	-	**0.00**	TL		**0.00**	15.39	TL
CW8	60	4605	-	-	**0.00**	TL		2.19	13.28	TL
CW9	50	10748	-	-	**0.00**	TL		**0.00**	14.77	TL
CW10	60	6515	-	-	**0.00**	583.77		**0.00**	10.02	TL
CW11	60	6321	-	-	**0.00**	214.53		**0.00**	11.11	TL
Average			28.14	1946.42	0.09	666.44		0.48	10.45	TL

8.8 Conclusions and future work

In this Chapter, first-order non-guillotine and guillotine cutting and packing problems were solved by a new general and flexible Mixed-Integer Linear Programming (MILP) model: the ‘floating cuts’ model. In this tree search-based model the position of the cut is not fixed when branching. The ‘floating cuts’ model depends on the definition of the number of levels of the sub-rectangles tree, which turns it into a parametric model. As the model is based on a tree search formulation, we propose algorithms to identify the relative position of a sub-rectangle, the father of a sub-rectangle, and the level of the tree where a sub-rectangle is located.

Three different ‘floating cuts’ models were proposed: the FC5 for first-order non-guillotine cutting and packing problems, the FC4 for the non-staged guillotine cutting and packing problems and the FC2 for k-staged guillotine cutting and packing problems.

The computational experiments were run with two goals: (1) to evaluate the performance of the MILP models for the several problem types and variants, always considering the same set of problem instances and only adapting the data parameters according to the problem variant; and (2) to compare the ‘floating cuts’ model with the state-of-the-art exact methods.

For the test cases for the non-guillotine problem for which the optimal solutions were known, the solver for the FC5 reached the optimal solution in 91% of the cases. Besides, 75% of the gaps reported by the solver for the FC5_Strengthened were less than or equal to 6,15% with an average value of 4,07%.

Regarding the extension of the FC5 to the guillotine version of the problem, the FC4 (FC4_Base and FC4_Strengthened), the solver obtained the optimal value for 8 out of 11 instances for which the optimal solution was known from the literature.

This supports the claim that the ‘floating cuts’ model is a general and flexible model, but, above all, paves the way for further research on this idea, which may lead to even better results in the future.

Regarding the comparison with other exact methods, the FC4_Strengthened model outperforms the exact method proposed by Furini et al. [44] for the 2-staged and non-staged F/U variants of the 2DSKP, and also outperforms the results of Martin et al. [73] for the 2-staged F/U problem variant.

For the non-staged - F/U 2DSKP, the value of the solutions obtained for Martin et al. [75] and the FC4_Strengthened is the same. However, Martin et al. [75] outperformed the proposed floating cuts model since the solver was able to prove optimality for all tested instances, while only two were proven for the FC4_Strengthened. For the 2DSLOPP similar results were obtained for both models.

For the non-staged - F/W 2DSLOPP, the model from Martin et al. [75] and the FC4_Strengthened model outperformed the model from Furini et al. [44], and the results from Martin et al. [75] are slightly better than the ones from the FC4_Strengthened model.

The main drawback of the 'floating cuts' models is that it cannot prove the overall optimality of the solution in a single run, i.e., the solutions are optimal for the tree height (h) considered in that run. Overall optimality requires an iterative framework where the model is set up and run with an increasing value of h. However, when solving instances for which the optimal solution is known, it was clear that most of the times the model could find the optimal solutions. In the other cases, the gaps were fairly low. The fast convergence to very good solutions and the low solution times open future research opportunities regarding the use of the 'floating cuts' model in the context of matheuristics.

Chapter 9

Conclusions

We presented a broad yet concise study of the C&P problems named *Pallet Building Problem* (PBP) and *Rectangular Cutting Problem* (RCP). When considering PBP, we solve a variation of the root problem, including in it standard constraints (rotation, stackability etc.) as well as new practical constraints (visibility and contiguity). On the other hand, when considering RCP, we solve the classic variations of the problem by using the innovative idea of floating-cuts to place items inside a container. We proposed techniques to solve both problems based on heuristic, metaheuristic, matheuristic and exact methods. The instances used were taken from an Italian company (PBP) in addition to instances taken from the related literature (RCP). Our work yielded effectively compact and competitive results.

In Chapter 4 a new heuristic technique was introduced to solve the PBP. The advantage of using this technique is its low execution time, allowing it to create a solution in milliseconds. Additionally, we analyzed the grouping of items by using several evaluation functions. It includes generality to the proposal, when adding the possibility to choose different functions tailored to the instance that will be solved.

In Chapter 5 we have included improvements on our previous work now allowing automated choice of the evaluation functions during the optimization process. This is possible by using GRASP metaheuristic with the choice of evaluation function based on the reactive method. In this case, in addition to the extensive exploration of the search space provided by the GRASP metaheuristic, the optimization process has improved because the reactive method interacts dynamically during the solution construction, resulting in a broader process which is less related to parametric settings.

In Chapter 6 we extended the basic technique for solving the PBP by replacing the heuristic method for piling layers with an exact method. The key point here is the fact that most of the time it is more efficient to devote additional time on layer optimization rather than starting a new solution at the individual items level. Nevertheless, given the inherent challenges of optimizing anything, a more precise analysis for different scenarios is needed.

In Chapter 7 we conclude our proposal for the PBP by formally describing the problem in detail, which to the best of our knowledge has never been documented in literature, and by proposing exact methods to help the heuristic algorithm resolve the problem of packing items in layers. We recognize the optimization potential of this method, as well as its weakness when addressing complex scenarios. However, the combination of exact and heuristic methods guided by a metaheuristic one can improve the results and make the algorithm competitive both theoretically and practically.

In Chapter 8 we propose new Mixed-Integer Linear Programming (MILP) models to solve guillotine and non-guillotine cuts, based on the recent and innovative idea of floating cuts on a tree search. The model proposed is parametric since its dimension (number of variables and constraints) is dependent on the number h of levels of the tree, i.e., the model becomes more complex as the value of h increases. Simply put, the technique consists of enumerating all possible solutions from a single plate, which are strongly reduced by including valid inequality constraints for eliminating symmetric solutions. The proposed models are flexible enough to solve two classes of the RCP as well as many of their variants. Despite the drawback of not proving optimality in a single run, the models are very competitive both in terms of quality of the solution and in terms of solution gap when the optimal solution value is known.

9.1 What are the main questions answered?

Via our work we were able to answer many questions about PBP and RCP, although there are many more unanswered questions to be addressed by future research.

In the category of questions answered, we highlight the combination of heuristic, exact and metaheuristic techniques, and the impact each one has on the whole process. Naturally, not all the cases were included, given the large number of possible scenarios. However, in these works we were able to analyze various scenarios in order to standardize the execution of the algorithm developed.

Additionally, this work also highlights the necessity for parametric tuning of the solution techniques when used independently or combined with others to create an efficient algorithm. As stated, it was not a trivial task to define a configuration that would yield a reasonably efficient solution, since the set of parameters used is directly linked to the quality of the solutions.

An important achievement of this work is directly related to the RCP technique, which in turn is based on floating cuts. As previously mentioned, the floating cuts strategy makes the mathematical models for C&P more efficient and flexible, since there is no need for the variables that decide the position of the cuts in a container. Therefore, we were able to include a series of simplifications to the model as well as substantially reducing the number of variables and constraints when developing the MILP model based on this strategy.

9.2 What could have been done better?

Taking into consideration the steps taken in the development of this research, I would like to highlight some points that could help, were it to start today.

The first point is related to the database and the partnership with the company to solve the PBP with practical constraints. Our partnership was explicitly restricted to the use of the real database, leading to no direct contact with the real life scenario in day to day operations. Research that is limited in such practical applications so that it can only be developed in the theoretical field has its effectiveness drastically reduced, given that theory and practice are intrinsically linked. Therefore, keeping theory and practice together would be ideal not only for this research, but also for any successive research that may be required.

When it comes to the algorithms, the wide application of the technique for the PBP is another point that I wish had been done. The reasoning for this is the wide application achieved with the implementation of said model for several variants for the RCP. In that regard, we were able to approach a considerably wider range of problems, which in practical situations is much more advantageous. A simple example of situations not covered adequately was dealing with instances consisting of only similar items, this requires a more robust algorithm than the one used herein in order to efficiently create compact solutions.

Regarding the practical problem, we gave too much attention to the new practical constraints, inadvertently overlooking other constraints that should have had similar consideration, for example, the 'stackability' constraint, which is of utmost importance in the PBP. Even though the problem can still be solved efficiently in several cases, however the disregard of a more refined formulation for this constraint leads to a solution with less applicability in practical real-world situations.

Regarding the planning of experiments, a few points could have been taken into consideration to improve our research. For example, a study of the evolution of the solution quality for the algorithms proposed would have increased the scientific contributions reached in this work. In addition to that, I would like to point out the lack of a more accurate study of GRASP-based algorithms. We took for granted that they would be more efficient algorithms, however we overlooked the analysis which could have brought new scientific knowledge.

Finally, a critical thinking deduction that fits the philosophical field of research. The process to develop the best algorithm for a problem requires a long journey, with small contributions each day. I think the main point is not to try to make the perfect algorithm in one try, but to start from a basic initial proposal, and the improvements will come through dedication and hard work. Additionally, suggesting new solutions is commendable even if they may seem silly, because the most awesome solutions are often born from simple concepts, and the attitude of advancing into the unknown is what differentiates the research activity from other professions.

9.3 What are the possible directions for the future?

To finish, possible areas of interest in the near future are included below:

- How does one improve the trade-off of the algorithm while meeting all constraints included here in a real world situation? Are there other constraints that must be fulfilled or other situations that have yet to be addressed while considering the current set of constraints?
- Would including other techniques during the optimization process in the PBP help the optimization? And more specifically, would it be useful to use machine learning to choose the next item to be placed in a container?
- Is it possible to develop an efficient pseudo-polynomial algorithm to solve the PBP? If so, what would be its best complexity threshold when considering a generic case?
- Is there a simple way to choose the level h of the tree search when solving the RCP? Would it be possible to apply other cuts to further strengthen the model?
- Taking into account the floating cuts technique, would it be useful to expand this idea to other cutting patterns? And how useful would it be to include other packing constraints in this model (e.g., visibility and contiguity constraints)?

In view of all that has been presented, it is our honest hope that this work can and will be a useful contribution for future researchers.

Bibliography

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

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

[3] Alonso, M., Alvarez-Valdes, R., Parreño, F.: A GRASP algorithm for multi container loading problems with practical constraints. 4OR-Q J Oper Res **18**, 49–72 (2020)

[4] Alonso, M., Alvarez-Valdes, R., Parreño, F., Tamarit, J.: Algorithms for pallet building and truck loading in an interdepot transportation problem. Mathematical Problems in Engineering **2016**, 1–11 (2016)

[5] Alonso, M., Alvarez-Valdes, R., Tamarit, J., Parreño, F.: A reactive GRASP algorithm for the container loading problem with load-bearing constraints. European Journal of Industrial Engineering **8**, 669–694 (2014)

[6] Alvarez-Valdes, R., Parreño, F., Tamarit, J.: A branch-and-cut algorithm for the pallet loading problem. Computers & Operations Research **32**, 3007–3029 (2005)

[7] Alvarez-Valdes, R., Parreño, F., Tamarit, J.: Reactive GRASP for the strip-packing problem. Computers & Operations Research **35**, 1065–1083 (2008)

[8] Arenales, M., Morabito, R.: An and/or-graph approach to the solution of two-dimensional non-guillotine cutting problems. European Journal of Operational Research **84**(3), 599–617 (1995)

[9] Beasley, J.: An exact two-dimensional non-guillotine cutting tree search procedure. Operations Research **33**(1), 49–64 (1985)

[10] Becker, H., Buriol, L.: An empirical analysis of exact algorithms for the unbounded knapsack problem. European Journal of Operational Research **277**(1), 84–99 (2019)

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

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

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

[14] Burke, E., Kendall, G., Whitwell, G.: A new placement heuristic for the orthogonal stock-cutting problem. Operations Research **52**, 655–671 (2004)

[15] Cacchiani, V., Iori, M., Locatelli, A., Martello, S.: Knapsack problems - an overview of recent advances. Part I: Single Knapsack Problems. Computers & Operations Research (2022, forthcoming)

[16] Cacchiani, V., Iori, M., Locatelli, A., Martello, S.: Knapsack problems - an overview of recent advances. Part II: Multiple, Multidimensional, and Quadratic Knapsack Problems. Computers & Operations Research (2022, forthcoming)

[17] Chazelle, B.: The bottomn-left bin-packing heuristic: An efficient implementation. IEEE Transactions on Computers **C-32**, 697–707 (1983)

[18] Christofides, N., Hadjiconstantinou, E.: An exact algorithm for orthogonal 2-d cutting problems using guillotine cuts. European Journal of Operational Research **83**(1), 21–38 (1995)

[19] Christofides, N., Whitlock, C.: An algorithm for two-dimensional cutting problems. Operations Research **25**(1), 30–44 (1977)

[20] Cintra, G., Miyazawa, F., Wakabayashi, Y., Xavier, E.: Algorithms for two-dimensional cutting stock and strip packing problems using dynamic programming and column generation. European Journal of Operational Research **191**(1), 61–85 (2008)

[21] Clautiaux, F., Sadykov, R., Vanderbeck, F., Viaud, Q.: Combining dynamic programming with filtering to solve a four-stage two-dimensional guillotine-cut bounded knapsack problem. Discrete Optimization (2018)

[22] Crainic, T., Perboli, G., Tadei, R.: Extreme point-based heuristics for three-dimensional bin packing. INFORMS Journal on Computing **20**, 368–384 (2008)

[23] Crainic, T., Perboli, G., Tadei, R.: Recent advances in multi-dimensional packing problems. In: New Technologies, chap. 5. IntechOpen (2012)

[24] Cung, V.D., Hifi, M., Cun, B.L.: Constrained two-dimensional cutting stock problems a best-first branch-and-bound algorithm. International Transactions in Operational Research **7**(3), 185–210 (2000)

[25] Côté, J.F., Dell'Amico, M., Iori, M.: Combinatorial benders' cuts for the strip packing problem. Operations Research **62**, 643–661 (2014)

[26] Côté, J.F., Haouari, M., Iori, M.: Combinatorial benders decomposition for the two-dimensional bin packing problem. INFORMS Journal on Computing pp. 1–16 (2021)

[27] da Silva, E.F., Leão, A.A.S., Toledo, F.M.B., Wauters, T.: A matheuristic framework for the three-dimensional single large object placement problem with practical constraints. Computers & Operations Research **124**, 105058 (2020)

[28] de Queiroz, T., Miyazawa, F.: Two-dimensional strip packing problem with load balancing, load bearing and multi-drop constraints. International Journal of Production Economics **145**, 511–530 (2013)

[29] de Queiroz, T., Miyazawa, F.: Order and static stability into the strip packing problem. Annals of Operations Research **223**, 137–154 (2014)

[30] Delorme, M., Iori, M., Martello, S.: Bin packing and cutting stock problems: Mathematical models and exact algorithms. European Journal of Operational Research **255**, 1–20 (2016)

[31] Delorme, M., Iori, M., Martello, S.: Logic based benders decomposition for orthogonal stock cutting problems. Computers & Operations Research **78**, 290–298 (2017)

[32] Dolatabadi, M., Lodi, A., Monaci, M.: Exact algorithms for the two-dimensional guillotine knapsack. Computers and Operations Research **39**(1), 48–53 (2012)

[33] Dyckhoff, H.: A new linear programming approach to the cutting stock problem. Operations Research **29**(6), 1092–1104 (1981)

[34] Dyckhoff, H.: A typology of cutting and packing problems. European J. Operational Research **44**, 145–159 (1990)

[35] Egeblad, J., Garavelli, C., Lisi, S., Pisinger, D.: Heuristics for container loading of furniture. European Journal of Operational Research **200**, 881–892 (2010)

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

[37] Elhedhli, S., Gzara, F., Yildiz, B.: Three-dimensional bin packing and mixed-case palletization. INFORMS Journal on Optimization **1**(4), 323–352 (2019)

[38] Fayard, D., Hifi, M., Zissimopoulos, V.: An efficient approach for large-scale two-dimensional guillotine cutting stock problems. Journal of the Operational Research Society **49**(12), 1270–1277 (1998)

[39] Fekete, S., Schepers, J.: A new exact algorithm for general orthogonal d-dimensional knapsack problems. In: Algorithms — ESA '97, pp. 144–156 (1997)

[40] Fekete, S., Schepers, J.: On more-dimensional packing III: Exact algorithms (2000). http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.40.9488

[41] Fekete, S., Schepers, J., van der Veen, J.: An exact algorithm for higher-dimensional orthogonal packing. Operations Research **55**(3), 569–587 (2007)

[42] Feo, T., Resende, M.: A probabilistic heuristic for a computationally difficult set covering problem. Operations Research Letters **8**, 67–71 (1989)

[43] Feo, T., Resende, M.: Greedy randomized adaptive search procedures. Journal of Global Optimization **6**, 109–133 (1995)

[44] Furini, F., Malaguti, E., Thomopulos, D.: Modeling two-dimensional guillotine cutting problems via integer programming. INFORMS Journal on Computing **28**(4), 736–751 (2016)

[45] Gilmore, P., Gomory, R.: Multistage cutting stock problems of two or more dimensions. Operations Research **13**, 94–120 (1965)

[46] Gilmore, P., Gomory, R.: The theory and computation of knapsack functions. Operations Research **14**(6), 1045–1074 (1966)

[47] Gottschalk, S.: Separating axis theorem. Tech. rep., Technical Report TR96-024, Department of Computer Science, UNC Chapel Hill (1996)

[48] Gzara, F., Elhedhli, S., Yildiz, B.C.: The pallet loading problem: Three-dimensional bin packing with practical constraints. European Journal of Operational Research **287**(3), 1062–1074 (2020)

[49] Hadjiconstantinou, E., Christofides, N.: An exact algorithm for general, orthogonal, two-dimensional knapsack problems. European Journal of Operational Research **83**(1), 39–56 (1995)

[50] Hadjiconstantinou, E., Iori, M.: A hybrid genetic algorithm for the two-dimensional single large object placement problem. European Journal of Operational Research **183**(3), 1150–1166 (2007)

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

[52] He, K., Huang, W., Jin, Y.: An efficient deterministic heuristic for two-dimensional rectangular packing. Computers & Operations Research **39**(7), 1355–1363 (2012)

[53] Hifi, M.: Exact algorithms for large-scale unconstrained two and three staged cutting problems. Computational Optimization and Applications **18**(1), 63–88 (2001)

[54] Hifi, M., M'Hallah, R.: An exact algorithm for constrained two-dimensional two-staged cutting problems. Operations Research **53**(1), 140–150 (2005)

[55] Hifi, M., Roucairol, C.: Approximate and exact algorithms for constrained (un) weighted two-dimensional two-staged cutting stock problems. Journal of Combinatorial Optimization **5**(4), 465–494 (2001)

[56] Imahori, S., Yagiura, M.: The best-fit heuristic for the rectangular strip packing problem: An efficient implementation and the worst-case approximation ratio. Computers & Operations Research **37**, 325–333 (2010)

[57] Iori, M., de Lima, V., Martello, S., Miyazawa, F., Monaci, M.: Exact solution techniques for two-dimensional cutting and packing. European Journal of Operational Research **289**(2), 399–415 (2021)

[58] Iori, M., Martello, S.: Routing problems with loading constraints. TOP **18**, 4–27 (2010)

[59] Iori, M., Martello, S.: An annotated bibliography of combined routing and loading problems. Yugoslav Journal of Operations Research **23**, 311–326 (2013)

[60] Jovanovic, R., Tuba, M., Voß, S.: Fixed set search applied to the traveling salesman problem. In: Hybrid Metaheuristics, International Workshop on Hybrid Metaheuristics, pp. 63–77. Springer International Publishing (2019)

[61] Jovanovic, R., Voß, S.: Fixed set search applied to the minimum weighted vertex cover problem. In: Analysis of Experimental Algorithms, *SEA 2019*, vol. 11544, pp. 490–504. Springer (2019)

[62] Jovanovic, R., Voß, S.: The fixed set search applied to the power dominating set problem. Expert Systems p. e12559 (2020)

[63] Józefowska, J., Pawlak, G., Pesch, E., Morze, M., Kowalski, D.: Fast truck-packing of 3D boxes. Engineering Management in Production and Services **10**, 29–40 (2018)

[64] Kurpel, D., Scarpin, C., Pécora Junior, J., Schenekemberg, C., Coelho, L.: The exact solutions of several types of container loading problems. European Journal of Operational Research **284**, 87–107 (2020)

[65] Leung, S., Zhang, D., Sim, K.: A two-stage intelligent search algorithm for the two-dimensional strip packing problem. European Journal of Operational Research **215**, 57–69 (2011)

[66] Libralesso, L., Fontan, F.: An anytime tree search algorithm for the 2018 ROADEF/EURO challenge glass cutting problem. CoRR **abs/2004.00963** (2020). URL `https://arxiv.org/abs/2004.00963`

[67] Lins, L., Lins, S., Morabito, R.: An l-approach for packing (l, w)-rectangles into rectangular and l-shaped pieces. The Journal of the Operational Research Society **54**(7), 777–789 (2003)

[68] Liu, J., Yue, Y., Dong, Z., Maple, C., Keech, M.: A novel hybrid tabu search approach to container loading. Computers and Operations Research **38**, 797–807 (2011)

[69] Lodi, A., Martello, S., Monaci, M., Vigo, D.: Two-Dimensional Bin Packing Problems, pp. 107–129. John Wiley & Sons, Ltd (2014)

[70] Lodi, A., Monaci, M.: Integer linear programming models for 2-staged two-dimensional knapsack problems. Mathematical Programming **94**(2–3), 257–278 (2003)

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

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

[73] Martin, M., Birgin, E., Lobato, R., Morabito, R., Munari, P.: Models for the two-dimensional rectangular single large placement problem with guillotine cuts and constrained pattern. International Transactions in Operational Research **27**(2), 767–793 (2020)

[74] Martin, M., Morabito, R., Munari, P.: A bottom-up packing approach for modeling the constrained two-dimensional guillotine placement problem. Computers & Operations Research **115**, 104851 (2020)

[75] Martin, M., Morabito, R., Munari, P.: A top-down cutting approach for modeling the constrained two- and three-dimensional guillotine cutting problems. Journal of the Operational Research Society **72**(12), 2755–2769 (2021)

[76] Martins, G., Dell, R.: Solving the pallet loading problem. European Journal of Operational Research **184**, 429–440 (2008)

[77] Messaoud, S., Chu, C., Espinouse, M.L.: Characterization and modelling of guillotine constraints. European Journal of Operational Research **191**(1), 112–126 (2008)

[78] Morabito, R., Arenales, M., Arcaro, V.: An and—or-graph approach for two-dimensional cutting problems. European Journal of Operational Research **58**(2), 263–271 (1992)

[79] Neliβen, J.: How to use structural constraints to compute an upper bound for the pallet loading problem. European Journal of Operational Research **84**, 662–680 (1995)

[80] Oliveira, J., Ferreira, J.: An improved version of Wang's algorithm for two-dimensional cutting problems. European Journal of Operational Research **44**(2), 256–266 (1990)

[81] Parreño, F., Alvarez-Valdes, R.: Mathematical models for a cutting problem in the glass manufacturing industry. Omega p. 102432 (2021)

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

[83] Pisinger, D., Sigurd, M.: Using decomposition techniques and constraint programming for solving the two-dimensional bin-packing problem. INFORMS Journal on Computing **19**(1), 36–51 (2007)

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

[85] Prais, M., Ribeiro, C.: Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment. INFORMS Journal on Computing **12**(3), 164–176 (2000)

[86] Preparata, F.P., Shamos, M.I.: Computational Geometry: An Introduction. Springer-Verlag, Berlin, Heidelberg (1985)

[87] Puchinger, J., Raidl, G.: Models and algorithms for three-stage two-dimensional bin packing. European Journal of Operational Research **183**(3), 1304–1327 (2007)

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

[89] Ren, J., Tian, Y., Sawaragi, T.: A tree search method for the container loading problem with shipment priority. European Journal of Operational Research **214**, 526–535 (2011)

[90] Resende, M., Ribeiro, C.: Greedy Randomized Adaptive Search Procedures: Advances and Extensions, pp. 169–220. Springer International Publishing, Cham (2019)

[91] Rexel: REXEL straight knife machine (2022). URL `http://www.rexel.com.pl`. [Online; accessed 19-March-2022]

[92] Ribeiro, G., Lorena, L.: Lagrangean relaxation with clusters and column generation for the manufacturers pallet loading problem. Computers & Operations Research **34**, 2695–2708 (2007)

[93] Russo, M., Boccia, M., Sforza, A., Sterle, C.: Constrained two-dimensional guillotine cutting problem: upper-bound review and categorization. International Transactions in Operational Research **27**(2), 794–834 (2020)

[94] Russo, M., Sforza, A., Sterle, C.: An improvement of the knapsack function based algorithm of Gilmore and Gomory for the unconstrained two-dimensional guillotine cutting problem. International Journal of Production Economics **145**(2), 451–462 (2013)

[95] Russo, M., Sforza, A., Sterle, C.: An exact dynamic programming algorithm for large-scale unconstrained two-dimensional guillotine cutting problems. Computers & Operations Research **50**, 97–114 (2014)

[96] Scheithauer, G.: Introduction to Cutting and Packing Optimization. Springer International Publishing (2018)

[97] Scheithauer, G., Sommerweiβ, U.: 4-block heuristic for the rectangle packing problem. European Journal of Operational Research **108**, 509–526 (1998)

[98] Scheithauer, G., Terno, J.: Modeling of packing problems. Optimization **28**(1), 63–84 (1993)

[99] Scheithauer, G., Terno, J.: The G4-heuristic for the pallet loading problem. The Journal of the Operational Research Society **47**, 511–522 (1996)

[100] Schmid, V., Doerner, K., Laporte, G.: Rich routing problems arising in supply chain management. European Journal of Operational Research **224**, 435–448 (2013)

[101] Silva, E., Alvelos, F., de Carvalho, J.: An integer programming model for two- and three-stage two-dimensional cutting stock problems. European Journal of Operational Research **205**(3), 699–708 (2010)

[102] Silva, E., Oliveira, J., Wäscher, G.: The pallet loading problem: a review of solution methods and computational experiments. International Transactions in Operational Research **23**, 147–172 (2016)

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

[104] Tsai, D.: Modeling and analysis of three-dimensional robotic palletizing systems for mixed carton sizes. Ph.D. thesis, Iowa State University (1987)

[105] Velasco, A., Uchoa, E.: Improved state space relaxation for constrained two-dimensional guillotine cutting problems. European Journal of Operational Research **272**(1), 106–120 (2019)

[106] Vidal, T., Crainic, T., Gendreau, M., Prins, C.: Heuristics for multi-attribute vehicle routing problems: A survey and synthesis. European Journal of Operational Research **231**, 1–21 (2013)

[107] Wang, P.: Two algorithms for constrained two-dimensional cutting stock problems. Operations Research **31**(3), 573–586 (1983)

[108] Wang, Z., Li, K., Levy, J.: A heuristic for the container loading problem: A tertiary-tree-based dynamic space decomposition approach. European Journal of Operational Research **191**, 86–99 (2008)

[109] Wäscher, G., Hauβner, H., Schumann, H.: An improved typology of cutting and packing problems. European Journal of Operational Research **183**, 1109–1130 (2007)

[110] Wu, K., Ting, C.: A two-phase algorithm for the manufacturer's pallet loading problem. In: IEEE International Conference on Industrial Engineering and Engineering Management, pp. 1574–1578 (2007)

[111] Zhang, D., Peng, Y., Leung, S.: A heuristic block-loading algorithm based on multi-layer search for the container loading problem. Computers and Operations Research **39**, 2267–2276 (2012)

Appendix

GREP algorithm for the PBP with visibility and contiguity constraints

Algorithm 1: GREP(I, D, $order\_family$, $order\_type$, ϵ, ϱ, β, ψ, $\max_\kappa$)

```
1  begin
2      INITIALIZATION($numIter$, $V_{best}$, $V_{worst}$, $U$, $sum_d$, $mean_d$, $\kappa$, $n_d$, $P_d$);
3      $S \leftarrow$ SORTITEMS($I$, $order\_family$, $order\_type$);
4      while NOTSTOPCONDITION(time) do
5          $d \leftarrow$ CHOOSECRITERION($D$, $P_d$);    // select a function $d$
6          $n_d = n_d + 1$;
7          $numIter = numIter + 1$;
8          $Sol \leftarrow$ CONSTRUCTIVE($S$, $L$, $d$, $\epsilon$);          // EPMH solution
9          $V \leftarrow$ EVALUATESOLUTION($Sol$);
10         if $V > U$ then               // upper bound is improved
11             $\kappa = 0$;
12             $Sol \leftarrow$ IMPROVEMENT($S$, $L$, $d$, $\epsilon$);        // local search
13             $V \leftarrow$ EVALUATESOLUTION($Sol$);
14             $U \leftarrow V$;                         // new upper bound
15         else                     // upper bound is not improved
16             $\kappa \leftarrow \kappa + 1$;
17             if $\kappa > \max_\kappa$ then
18                 $U \leftarrow \beta U$;               // reduce the upper bound
19                 $\kappa = 0$;
           /* Basic update (fitness and solution)       */
20         if $V > V_{best}$ then
21             $V_{best} \leftarrow V$;
22             $Sol_{best} \leftarrow Sol$;
23         if $V < V_{worst}$ then
24             $V_{worst} \leftarrow V$;
           /* Update reactive variables (function $d$)    */
25         $sum_d = sum_d + V$;
26         $mean_d = \frac{sum_d}{n_d}$;
           /* Update reactive structures (set $D$)        */
27         if $mod(numIter, \psi) = 0$ then
28             $eval_d \leftarrow (\frac{mean_d - V_{worst}}{V_{best} - V_{worst}})^\varrho$ , $\forall\, d \in D$;
29             $P_d \leftarrow \frac{eval_d}{\sum_{d' \in D}(eval_{d'})}$ , $\forall\, d \in D$;
30     return $Sol_{best}$;
```

Floating cuts algorithms for non-guillotine cuts

Algorithm 2: Identify the relative position of sub-rectangle j

```
Function RelativePosition
    Input: $j$.
    Output: $TL||TR||BR||BL||CC$.
if $j$ mod $5 == 1$ then
    return $TL$;
end
else if $j$ mod $5 == 2$ then
    return $TR$;
end
else if $j$ mod $5 == 3$ then
    return $BR$;
end
else if $j$ mod $5 == 4$ then
    return $BL$;
end
else
    return $CC$;
end
```

Algorithm 3: Identify the father of sub-rectangle j

```
Function FATHER
Input: $j$.
Output: Index of the father of sub-rectangle $j$.
if RELATIVEPOSITION($j$) == $TL$ then
    return ($\frac{j-1)}{5}$);
end
else if RELATIVEPOSITION($j$) == $TR$ then
    return ($\frac{(j-2)}{5}$);
end
else if RELATIVEPOSITION($j$) == $BR$ then
    return ($\frac{(j-3)}{5}$);
end
else if RELATIVEPOSITION($j$) == $BL$ then
    return ($\frac{(j-4)}{5}$);
end
else
    return ($\frac{(j-5)}{5}$);
end
```

Algorithm 4: Identify the level of the tree where sub-rectangle j is located

```
Function LEVEL
Input: $j$.
Output: level $h$ of the tree.
if $j$ == 0 then
    return 0;
end
$h = 1$;
while $\frac{j}{5^{(h)}} \geq 1$ do
    $h$++;
end
return $h$;
```

Algorithm 5: Number of stages to reach sub-rectangle j

```
Function STAGES
Input: $j$.
Output: number of stages $nS$ to reach sub-rectangle $j$.
if $j == 0$ then
    return ($j$);
end
$nS = 1$;
while $j \neq 0$ do
    if (((RELATIVEPOSITION(FATHER($j$)) == $t$ or RELATIVEPOSITION(FATHER($j$)) == $b$) and
      (RELATIVEPOSITION($j$) == $l$ or RELATIVEPOSITION($j$) == $r$)) or
      ((RELATIVEPOSITION(FATHER($j$)) == $l$ or RELATIVEPOSITION(FATHER($j$)) == $r$) and
      (RELATIVEPOSITION($j$) == $t$ or RELATIVEPOSITION($j$) == $b$))) then
        $nS++$;
    end
    $j$ = FATHER($j$);
end
return ($nS$);
```

Floating cuts: experiments for parameter settings

Table 1: FC5_Strengthened preliminary computational experiments for the first-order non-guillotine 2DSKP.

Variant	Instance	h																	
		1			2			3			4			5			6		
		Z	Gap(%)	T(s)	Z	Gap(%)	T(s)	Z	Gap(%)	T(s)	Z	Gap(%)	T(s)	Z	Gap(%)	T(s)	Z	Gap(%)	T(s)
F/U	cgcut1	32	0.00	0.01	113	0.00	0.04	145	0.00	351.10	145	3.45	TL	144	41.67	TL	0	inf	TL
	cgcut2	462	0.00	0.05	1810	0.00	0.08	2737	2.30	TL	2740	2.19	TL	2778	0.79	TL	0	inf	TL
	cgcut3	1333	0.00	0.07	2470	0.00	2.40	2726	2.72	TL	2721	2.90	TL	2721	2.90	TL	2284	145.18	TL
	CW1	4752	0.00	0.01	12450	0.00	0.28	13057	0.52	TL	12797	2.56	TL	13057	0.52	TL	0	inf	TL
	CW2	9672	0.00	0.07	23250	0.00	0.89	23524	1.71	TL	23524	1.71	TL	23240	105.90	TL	23240	105.90	TL
	CW3	19250	0.00	0.02	54343	0.00	0.72	54343	1.70	TL	54343	1.70	TL	54343	1.70	TL	53705	105.82	TL
	okp1	2268	0.00	0.01	7406	0.00	0.16	8776	0.00	308.36	9284	7.71	TL	9284	7.71	TL	3766	241.05	TL
	okp2	2387	0.00	0.07	8976	0.00	0.42	9876	1.26	TL	9681	3.30	TL	9618	3.97	TL	9445	111.75	TL
	okp3	2350	0.00	0.07	8425	0.00	1.16	9838	1.65	TL	9838	1.65	TL	9838	103.29	TL	9681	106.59	TL
	okp4	2376	0.00	0.02	9160	0.00	0.59	9824	1.79	TL	9818	1.85	TL	9801	2.03	TL	8994	122.37	TL
	okp5	2387	0.00	0.06	8976	0.00	0.19	9876	1.26	TL	9876	1.26	TL	9876	1.26	TL	0	inf	TL
R/U	cgcut1	32	0.00	0.01	117	0.00	0.19	150	0.00	6.53	150	0.00	9.69	148	52.03	TL	0	inf	TL
	cgcut2	462	0.00	0.03	1810	0.00	0.19	2766	1.23	TL	2738	2.26	TL	2742	2.12	TL	0	inf	TL
	cgcut3	1333	0.00	0.09	2680	0.00	4.33	2756	1.60	TL	2754	1.67	TL	2723	2.83	TL	0	inf	TL
	CW1	4752	0.00	0.07	12568	0.00	1.67	13057	0.52	TL	12797	2.56	TL	13057	0.52	TL	11978	119.15	TL
	CW2	9672	0.00	0.07	23424	0.00	3.39	23535	1.66	TL	23299	2.69	TL	23370	104.75	TL	23421	104.30	TL
	CW3	19250	0.00	0.04	54343	0.00	3.47	54719	1.01	TL	54721	1.00	TL	54332	103.45	TL	0	inf	TL
	okp1	2268	0.00	0.03	8168	0.00	0.21	9688	3.22	TL	9688	3.22	TL	9776	2.29	TL	0	inf	TL
	okp2	2387	0.00	0.04	8976	0.00	2.24	9876	1.26	TL	9921	0.80	TL	9665	24.73	TL	0	inf	TL
	okp3	2350	0.00	0.07	9402	0.00	1.18	9917	0.84	TL	9799	2.05	TL	9893	1.08	TL	0	inf	TL
	okp4	2376	0.00	0.04	9392	0.00	1.58	9892	1.09	TL	9854	1.48	TL	9830	103.46	TL	0	inf	TL
	okp5	2387	0.00	0.04	8976	0.00	0.62	9875	1.27	TL	9833	1.70	TL	9776	2.29	TL	0	inf	TL
F/W	cgcut1	66	0.00	0.01	208	0.00	0.06	244	6.56	TL	244	6.56	TL	244	7.79	TL	240	51.67	TL
	cgcut2	582	0.00	0.05	1930	0.00	0.12	2851	2.42	TL	2856	2.24	TL	2856	2.24	TL	0	inf	TL
	cgcut3	500	0.00	0.08	1280	0.00	3.12	1860	8.60	TL	1860	8.60	TL	1780	14.61	TL	980	295.92	TL
	CW1	704	0.00	0.01	3350	0.00	0.10	5090	4.70	TL	4860	9.82	TL	5010	8.60	TL	0	inf	TL
	CW2	749	0.00	0.03	3398	0.00	0.42	4673	15.26	TL	4767	13.36	TL	4673	17.12	TL	4599	76.58	TL
	CW3	742	0.00	0.03	3591	0.00	0.14	4719	12.40	TL	4719	12.82	TL	4593	16.42	TL	4457	107.09	TL
	okp1	6668	0.00	0.01	19277	0.00	0.31	22623	0.00	471.95	23771	12.40	TL	23771	13.73	TL	6668	369.74	TL
	okp2	4850	0.00	0.03	17979	0.00	0.44	21976	10.90	TL	21947	11.44	TL	22502	10.96	TL	0	inf	TL
	okp3	6032	0.00	0.07	19957	0.00	0.67	23743	11.82	TL	23740	11.85	TL	24019	96.52	TL	0	inf	TL
	okp4	6601	0.00	0.07	23868	0.00	0.69	26470	3.73	TL	26470	9.25	TL	25939	12.68	TL	0	inf	TL
	okp5	4850	0.00	0.03	17979	0.00	0.46	21976	10.98	TL	21976	11.29	TL	21814	14.46	TL	0	inf	TL
R/W	cgcut1	66	0.00	0.01	215	0.00	0.15	260	0.00	33.77	260	0.00	175.52	260	40.00	TL	0	inf	TL
	cgcut2	582	0.00	0.06	1930	0.00	0.21	2892	0.97	TL	2860	2.10	TL	2756	5.95	TL	0	inf	TL
	cgcut3	500	0.00	0.10	1280	0.00	4.33	1900	6.32	TL	1940	4.12	TL	1900	104.21	TL	0	inf	TL
	CW1	704	0.00	0.07	3350	0.00	0.19	5090	4.81	TL	4944	10.05	TL	4860	11.96	TL	0	inf	TL
	CW2	749	0.00	0.08	3591	0.00	0.27	5062	6.40	TL	5062	6.64	TL	4923	74.22	TL	0	inf	TL
	CW3	742	0.00	0.08	3616	0.00	0.43	5013	6.04	TL	4938	7.82	TL	5013	68.22	TL	0	inf	TL
	okp1	6668	0.00	0.07	22065	0.00	0.32	25721	4.22	TL	25721	4.19	TL	25721	5.11	TL	0	inf	TL
	okp2	4850	0.00	0.07	19197	0.00	1.45	23645	5.19	TL	24263	2.54	TL	24263	2.91	TL	0	inf	TL
	okp3	6032	0.00	0.08	21890	0.00	2.06	24932	7.43	TL	25278	6.16	TL	25216	6.50	TL	0	inf	TL
	okp4	6601	0.00	0.04	25771	0.00	3.56	27568	6.01	TL	27568	5.15	TL	27510	92.03	TL	0	inf	TL
	okp5	4850	0.00	0.08	19197	0.00	0.74	23645	5.07	TL	24263	2.62	TL	24251	2.96	TL	0	inf	TL

Table 2: FC5_Strengthened preliminary computational experiments for the first-order non-guillotine 2DSLOPP.

Variant	Instance	h																	
		1			2			3			4			5			6		
		Z	Gap(%)	T(s)	Z	Gap(%)	T(s)	Z	Gap(%)	T(s)	Z	Gap(%)	T(s)	Z	Gap(%)	T(s)	Z	Gap(%)	T(s)
	cgcut1	64	0.00	0.02	141	0.00	0.53	145	3.45	TL	144	4.17	TL	144	4.17	TL	144	4.17	TL
	cgcut2	945	0.00	0.02	2458	0.00	6.15	2740	2.19	TL	2778	0.79	TL	2740	2.19	TL	2741	2.15	TL
	cgcut3	2277	0.00	0.04	2710	0.00	26.97	2721	2.90	TL	2721	2.90	TL	2721	2.90	TL	2721	2.90	TL
	CW1	7280	0.00	0.08	12885	0.00	79.45	12861	2.05	TL	12777	2.72	TL	13057	0.52	TL	13057	0.52	TL
	CW2	14144	0.00	0.16	23418	0.00	228.49	23301	2.68	TL	23418	2.17	TL	23240	2.95	TL	0	inf	TL
F/U	CW3	34680	0.00	0.16	54571	0.00	252.29	54708	1.03	TL	54708	1.03	TL	54343	1.70	TL	54343	1.70	TL
	okp1	5670	0.00	0.05	9594	0.00	14.62	9974	0.26	TL	9974	0.26	TL	9974	0.26	TL	9938	0.62	TL
	okp2	2387	0.00	0.02	8976	0.00	0.62	9876	1.26	TL	9681	3.30	TL	9681	3.30	TL	9671	3.40	TL
	okp3	2350	0.00	0.03	8425	0.00	5.26	9838	1.65	TL	9734	2.73	TL	9838	1.65	TL	9812	1.92	TL
	okp4	6960	0.00	0.06	9796	0.00	28.24	9976	0.24	TL	9976	0.24	TL	9958	0.42	TL	9976	0.24	TL
	okp5	6324	0.00	0.08	9982	0.00	93.44	9982	0.18	TL	9982	0.18	TL	9982	0.18	TL	9976	0.24	TL
	cgcut1	64	0.00	0.06	147	0.00	24.36	150	0.00	4.41	150	0.00	4.34	150	0.00	58.21	145	3.45	TL
	cgcut2	945	0.00	0.05	2565	0.00	18.83	2727	2.68	TL	2775	0.90	TL	2737	2.30	TL	2765	1.27	TL
	cgcut3	2277	0.00	0.06	2756	0.00	64.29	2754	1.67	TL	2771	1.05	TL	2756	1.60	TL	2723	2.83	TL
	CW1	7280	0.00	0.07	13053	0.00	520.50	13053	0.55	TL	12972	1.18	TL	13057	0.52	TL	0	inf	TL
	CW2	14144	0.00	0.11	23750	0.00	765.18	23580	1.46	TL	23643	1.19	TL	23715	0.89	TL	0	inf	TL
R/U	CW3	34680	0.00	0.12	55015	0.46	TL	54750	0.95	TL	54477	1.45	TL	54515	1.38	TL	0	inf	TL
	okp1	5670	0.00	0.09	9920	0.00	52.72	9974	0.26	TL	9968	0.32	TL	9974	0.26	TL	9960	0.40	TL
	okp2	2387	0.00	0.05	8976	0.00	1.20	9828	1.75	TL	9921	0.80	TL	9921	0.80	TL	9720	2.88	TL
	okp3	2350	0.00	0.06	9402	0.00	1.36	9893	1.08	TL	9893	1.08	TL	9893	1.08	TL	9730	2.78	TL
	okp4	6960	0.00	0.12	9913	0.00	168.93	9979	0.21	TL	9979	0.21	TL	9816	1.87	TL	9884	1.17	TL
	okp5	6324	0.00	0.10	10000	0.00	0.75	10000	0.00	4.33	10000	0.00	30.97	10000	0.00	108.06	10000	0.00	709.55
	cgcut1	132	0.00	0.02	230	0.00	1.53	244	6.15	TL	244	6.56	TL	244	7.38	TL	243	8.23	TL
	cgcut2	945	0.00	0.02	2578	0.00	6.86	2856	2.24	TL	2856	2.24	TL	2856	2.24	TL	2782	4.96	TL
	cgcut3	1080	0.00	0.03	1820	0.00	14.63	1860	8.60	TL	1860	9.68	TL	1860	9.68	TL	1820	12.09	TL
	CW1	2112	0.00	0.14	6230	0.00	9.90	6402	11.64	TL	6402	12.09	TL	6402	12.31	TL	6402	12.31	TL
	CW2	2088	0.00	0.06	5394	0.00	18.16	5543	8.08	TL	5543	8.70	TL	5354	12.53	TL	5450	10.55	TL
F/W	CW3	2226	0.00	0.09	5557	0.00	10.96	5689	6.40	TL	5689	7.00	TL	5689	7.00	TL	5689	7.00	TL
	okp1	16008	0.00	0.06	26892	0.00	10.46	27617	5.19	TL	27718	5.48	TL	27617	5.86	TL	26746	9.31	TL
	okp2	4850	0.00	0.07	17979	0.00	0.67	21976	0.00	804.12	22119	10.19	TL	22092	13.02	TL	21154	18.03	TL
	okp3	6032	0.00	0.03	19957	0.00	1.72	23743	0.00	857.18	23740	11.85	TL	23743	12.98	TL	23740	13.13	TL
	okp4	18684	0.00	0.11	31238	0.00	14.10	32893	2.75	TL	32893	3.59	TL	32893	3.65	TL	32893	3.65	TL
	okp5	15956	0.00	0.05	27923	0.00	11.55	27923	4.20	TL	27923	4.38	TL	27923	4.38	TL	27923	4.38	TL
	cgcut1	132	0.00	0.08	244	0.00	25.20	260	0.00	11.97	260	0.00	98.23	260	0.00	734.93	259	1.54	TL
	cgcut2	945	0.00	0.08	2608	0.00	49.19	2885	1.21	TL	2883	1.28	TL	2839	2.85	TL	2863	1.99	TL
	cgcut3	1080	0.00	0.11	1900	0.00	50.70	1900	7.37	TL	1920	6.25	TL	1900	7.37	TL	1900	7.37	TL
	CW1	2136	0.00	0.14	6574	0.00	39.65	6766	5.63	TL	6808	5.41	TL	6746	6.58	TL	0	inf	TL
	CW2	2223	0.00	0.17	5471	0.00	58.16	5604	7.03	TL	5604	7.51	TL	5604	7.51	TL	5150	16.99	TL
R/W	CW3	2226	0.00	0.14	5689	0.00	76.33	5736	5.53	TL	5736	6.12	TL	5689	7.00	TL	5398	12.76	TL
	okp1	16008	0.00	0.09	28090	0.00	64.42	28423	2.75	TL	28423	2.86	TL	28090	4.04	TL	28090	4.08	TL
	okp2	4850	0.00	0.08	19197	0.00	0.69	23645	5.10	TL	23645	5.34	TL	23513	6.19	TL	23013	8.50	TL
	okp3	6032	0.00	0.09	21890	0.00	1.60	24932	7.43	TL	25081	6.82	TL	25216	6.22	TL	0	inf	TL
	okp4	18684	0.00	0.18	31907	0.00	59.80	32893	3.00	TL	32893	3.00	TL	32577	4.66	TL	31334	8.81	TL
	okp5	15956	0.00	0.09	27983	0.00	29.49	27983	4.11	TL	27983	4.15	TL	27983	4.15	TL	27923	4.38	TL

Table 3: FC4_Strengthened preliminary computational experiments for the guillotine 2DSKP.

Variant	Instance	h																				
		1			2			3			4			5			6			7		
		Z	Gap(%)	T(s)	Z	Gap(%)	T(s)	Z	Gap(%)	T(s)	Z	Gap(%)	T(s)	Z	Gap(%)	T(s)	Z	Gap(%)	T(s)	Z	Gap(%)	T(s)
	cgcut1	32	0.00	0.01	64	0.00	0.04	97	0.00	0.15	134	0.00	3.88	140	7.14	TL	144	4.17	TL	144	4.17	TL
	cgcut2	462	0.00	0.01	865	0.00	0.02	1495	0.00	0.06	2242	0.00	4.02	2628	6.55	TL	2740	2.19	TL	2740	2.19	TL
	cgcut3	1333	0.00	0.01	2092	0.00	0.06	2470	0.00	0.98	2721	0.00	84.78	2721	2.90	TL	2721	2.90	TL	2623	6.75	TL
	CW1	4752	0.00	0.05	8143	0.00	0.05	12450	0.00	0.21	12797	0.00	35.36	12777	2.72	TL	12797	2.56	TL	12797	2.56	TL
	CW2	9672	0.00	0.01	14043	0.00	0.14	23240	0.00	0.38	23240	0.00	50.92	23240	2.95	TL	23240	2.95	TL	23240	2.95	TL
inf/F/U	CW3	19250	0.00	0.01	32288	0.00	0.05	53755	0.00	0.29	53768	0.00	124.37	53755	2.82	TL	53755	2.82	TL	53755	2.82	TL
	okp1	2268	0.00	0.01	4158	0.00	0.07	6282	0.00	0.24	8110	0.00	10.26	8754	0.00	541.73	8836	13.17	TL	8836	13.17	TL
	okp2	2387	0.00	0.05	4725	0.00	0.14	7684	0.00	0.67	9423	0.00	56.16	9767	2.39	TL	9615	4.00	TL	9615	4.00	TL
	okp3	2350	0.00	0.01	4650	0.00	0.03	8962	0.00	0.31	9681	0.00	82.73	9734	2.73	TL	9838	1.65	TL	9681	3.30	TL
	okp4	2376	0.00	0.05	4707	0.00	0.03	8151	0.00	0.31	9666	0.00	54.55	9666	3.46	TL	9677	3.34	TL	9387	6.53	TL
	okp5	2387	0.00	0.01	4355	0.00	0.07	7432	0.00	0.38	9390	0.00	50.68	9615	4.00	TL	9615	4.00	TL	9394	6.45	TL
	cgcut1	32	0.00	0.05	64	0.00	0.02	101	0.00	0.06	148	0.00	8.39	150	0.00	2.11	150	0.00	28.34	150	0.00	34.98
	cgcut2	462	0.00	0.01	865	0.00	0.06	1495	0.00	0.06	2452	0.00	1.58	2671	4.83	TL	2678	4.56	TL	2678	4.56	TL
	cgcut3	1333	0.00	0.01	2361	0.00	0.07	2723	0.00	3.19	2754	0.00	887.48	2754	1.67	TL	2754	1.67	TL	2723	2.83	TL
	CW1	4752	0.00	0.01	9276	0.00	0.01	12568	0.00	0.66	12797	0.00	87.68	12777	2.72	TL	12777	2.72	TL	12797	2.56	TL
	CW2	9672	0.00	0.01	16676	0.00	0.05	23278	0.00	1.24	23557	0.00	262.55	23517	1.74	TL	23299	2.69	TL	23286	2.74	TL
inf/R/U	CW3	19250	0.00	0.01	33110	0.00	0.05	54022	0.00	0.88	54648	0.00	386.67	54022	2.31	TL	54022	2.31	TL	54022	2.31	TL
	okp1	2268	0.00	0.01	4158	0.00	0.01	7142	0.00	0.04	8970	0.00	11.27	9426	6.09	TL	9448	5.84	TL	9426	6.09	TL
	okp2	2387	0.00	0.01	4763	0.00	0.06	8088	0.00	0.22	9656	0.00	61.16	9863	1.39	TL	9863	1.39	TL	9701	3.08	TL
	okp3	2350	0.00	0.01	4650	0.00	0.01	9131	0.00	0.16	9795	0.00	37.95	9879	1.23	TL	9751	2.55	TL	9799	2.05	TL
	okp4	2376	0.00	0.01	4707	0.00	0.01	8264	0.00	0.24	9816	0.00	37.90	9823	1.80	TL	9823	1.80	TL	9823	1.80	TL
	okp5	2387	0.00	0.01	4736	0.00	0.01	8088	0.00	0.20	9656	0.00	36.81	9776	2.29	TL	9863	1.39	TL	9683	3.27	TL
	cgcut1	66	0.00	0.04	132	0.00	0.04	184	0.00	0.14	236	0.00	8.01	244	6.56	TL	244	7.79	TL	244	7.79	TL
	cgcut2	582	0.00	0.04	985	0.00	0.02	1615	0.00	0.06	2362	0.00	5.01	2759	5.84	TL	2856	2.24	TL	2782	4.96	TL
	cgcut3	500	0.00	0.01	860	0.00	0.09	1280	0.00	0.95	1720	0.00	13.44	1860	8.60	TL	1860	8.60	TL	1860	9.68	TL
	CW1	704	0.00	0.01	1397	0.00	0.04	2710	0.00	0.08	4512	0.00	4.57	4837	10.79	TL	4837	11.83	TL	4580	18.80	TL
	CW2	749	0.00	0.01	1490	0.00	0.03	2935	0.00	0.30	4673	0.00	29.79	4673	15.62	TL	4673	15.77	TL	4673	17.12	TL
inf/F/W	CW3	742	0.00	0.01	1480	0.00	0.08	2879	0.00	0.24	4593	0.00	20.12	4593	16.07	TL	4593	15.76	TL	4529	19.32	TL
	okp1	6668	0.00	0.01	12004	0.00	0.07	18093	0.00	0.15	21224	0.00	5.92	22573	0.00	761.41	22752	17.88	TL	22752	18.83	TL
	okp2	4850	0.00	0.05	8881	0.00	0.05	16159	0.00	0.41	20824	0.00	35.44	21154	15.61	TL	21416	16.59	TL	21976	13.41	TL
	okp3	6032	0.00	0.01	11479	0.00	0.16	21012	0.00	0.33	23285	0.00	23.55	23740	11.86	TL	23740	13.09	TL	23743	13.11	TL
	okp4	6601	0.00	0.05	12829	0.00	0.06	21006	0.00	0.60	25146	0.00	28.02	25754	10.73	TL	25939	12.50	TL	25939	12.66	TL
	okp5	4850	0.00	0.01	8881	0.00	0.09	16159	0.00	0.32	20824	0.00	22.56	21254	14.81	TL	21127	15.77	TL	21254	17.48	TL
	cgcut1	66	0.00	0.06	132	0.00	0.05	191	0.00	0.05	258	0.00	11.54	260	0.00	32.56	260	0.00	447.82	260	1.15	TL
	cgcut2	582	0.00	0.05	985	0.00	0.06	1615	0.00	0.06	2572	0.00	1.86	2795	4.47	TL	2860	2.10	TL	2813	3.80	TL
	cgcut3	500	0.00	0.06	940	0.00	0.03	1280	0.00	1.36	1720	0.00	39.75	1900	6.32	TL	1900	6.32	TL	1900	7.37	TL
	CW1	704	0.00	0.01	1397	0.00	0.02	2710	0.00	0.07	4704	0.00	13.16	5012	6.56	TL	4874	10.69	TL	4930	10.37	TL
	CW2	749	0.00	0.01	1490	0.00	0.05	2935	0.00	0.13	5062	0.00	11.08	5062	6.40	TL	5062	6.82	TL	4981	9.88	TL
inf/R/W	CW3	742	0.00	0.05	1480	0.00	0.05	2930	0.00	0.10	4874	0.00	15.51	4936	7.72	TL	4874	9.23	TL	4747	13.84	TL
	okp1	6668	0.00	0.01	12004	0.00	0.01	20290	0.00	0.04	24553	0.00	11.52	25721	0.00	386.05	25721	4.18	TL	25721	4.81	TL
	okp2	4850	0.00	0.01	9409	0.00	0.01	16832	0.00	0.19	22573	0.00	15.79	23513	5.99	TL	23513	5.80	TL	23138	7.91	TL
	okp3	6032	0.00	0.01	11490	0.00	0.01	21012	0.01	0.25	24355	0.00	32.54	24840	7.79	TL	25216	6.18	TL	24881	7.81	TL
	okp4	6601	0.00	0.06	12829	0.00	0.05	21006	0.00	0.25	27568	0.00	31.65	27568	4.86	TL	27568	5.33	TL	27568	5.86	TL
	okp5	4850	0.00	0.04	9409	0.00	0.01	16832	0.00	0.21	22573	0.00	20.39	23513	6.00	TL	23513	6.00	TL	22969	8.71	TL
	cgcut1	0	inf	0.01	64	0.00	0.04	80	0.00	0.02	92	0.00	0.55	122	0.00	1.40	140	0.00	52.11	140	7.14	TL
	cgcut2	0	inf	0.01	865	0.00	0.02	1291	0.00	0.08	1712	0.00	0.99	2208	0.00	18.68	2375	17.90	TL	2430	15.23	TL
	cgcut3	0	inf	0.01	2092	0.00	0.02	2277	0.00	0.19	2470	0.00	8.44	2599	0.00	49.62	2599	7.73	TL	2599	7.73	TL
	CW1	0	inf	0.03	7337	0.00	0.01	7961	0.00	0.36	12450	0.00	0.57	12777	0.00	55.06	12777	0.00	433.25	12777	2.72	TL
	CW2	0	inf	0.03	13812	0.00	0.01	13924	0.00	0.31	23240	0.00	0.89	23240	0.00	25.35	23240	0.00	714.79	23240	2.95	TL
2/F/U	CW3	0	inf	0.00	32120	0.00	0.02	32120	0.00	0.42	53755	0.00	0.66	53755	0.00	34.89	53755	2.82	TL	53755	2.82	TL
	okp1	0	inf	0.00	4158	0.00	0.01	5158	0.00	0.16	6382	0.00	0.89	7166	0.00	5.20	7166	0.00	51.72	7166	7.17	TL
	okp2	0	inf	0.00	4355	0.00	0.02	6291	0.00	0.15	8192	0.00	2.28	9390	0.00	46.62	9396	6.43	TL	9396	6.43	TL
	okp3	0	inf	0.01	4650	0.00	0.01	5530	0.00	0.29	8076	0.00	0.88	8682	0.00	31.60	9399	1.12	TL	9399	6.39	TL
	okp4	0	inf	0.02	4707	0.00	0.01	6892	0.00	0.20	8206	0.00	7.79	9300	0.00	32.89	9593	0.00	619.93	9593	4.24	TL
	okp5	0	inf	0.01	4355	0.00	0.02	6291	0.00	0.09	8192	0.00	1.47	9390	0.00	46.27	9396	6.43	TL	9396	6.43	TL

Table 4: FC4_Strengthened preliminary computational experiments for the guillotine 2DSLOPP.

Variant	Instance	h																				
		1			2			3			4			5			6			7		
		Z	Gap(%)	T(s)	Z	Gap(%)	T(s)	Z	Gap(%)	T(s)	Z	Gap(%)	T(s)	Z	Gap(%)	T(s)	Z	Gap(%)	T(s)	Z	Gap(%)	T(s)
	cgcut1	64	0.000	0.01	108	0.00	0.09	129	0.00	1.71	144	0.00	66.87	144	56.25	TL	144	56.25	TL	118	90.68	TL
	cgcut2	945	0.000	0.06	1593	0.00	0.13	2310	0.00	2.16	2686	0.00	536.26	2740	58.54	TL	2740	58.54	TL	2740	58.54	TL
	cgcut3	2277	0.000	0.10	2470	0.00	0.44	2667	0.00	5.28	2721	0.00	566.18	2721	147.01	TL	2700	318.56	TL	2667	340.05	TL
	CW1	7280	0.000	0.02	11920	0.00	0.24	12890	0.00	24.06	12890	40.13	TL	12885	175.94	TL	12692	413.41	TL	0	inf	TL
	CW2	14144	0.000	0.03	21273	0.00	0.77	23418	0.00	45.53	23418	69.04	TL	23196	319.65	TL	23216	335.78	TL	0	inf	TL
inf/F/U	CW3	34680	0.000	0.08	51752	0.00	0.49	54450	0.00	53.14	53995	79.10	TL	53983	350.35	TL	53983	363.61	TL	0	inf	TL
	okp1	5670	0.000	0.11	8416	0.00	0.20	9580	0.00	4.62	9898	0.00	892.07	9922	213.18	TL	9898	243.99	TL	0	inf	TL
	okp2	2387	0.000	0.02	4725	0.00	0.13	7684	0.00	1.59	9423	0.00	247.16	9534	4.89	TL	9402	6.36	TL	0	inf	TL
	okp3	2350	0.000	0.06	4650	0.00	0.06	8962	0.00	0.90	9681	0.00	318.99	9681	3.30	TL	9681	3.30	TL	8568	133.43	TL
	okp4	6960	0.000	0.06	8501	0.00	0.36	9824	0.00	4.87	9958	13.18	TL	9852	203.09	TL	9836	263.52	TL	0	inf	TL
	okp5	6324	0.000	0.06	9832	0.00	0.21	9862	0.00	22.62	9952	55.52	TL	9965	291.51	TL	9832	453.48	TL	0	inf	TL
	cgcut1	64	0.000	0.02	112	0.00	0.07	140	0.00	0.78	150	0.00	454.23	150	50.00	TL	150	50.00	TL	150	50.00	TL
	cgcut2	945	0.000	0.10	1593	0.00	0.10	2442	0.00	1.73	2757	26.88	TL	2757	57.56	TL	2762	57.28	TL	2730	59.12	TL
	cgcut3	2277	0.000	0.18	2716	0.00	1.26	2756	0.00	29.21	2771	56.08	TL	2754	164.60	TL	2756	438.64	TL	2644	461.46	TL
	CW1	7280	0.000	0.10	12106	0.00	0.29	12972	0.00	59.09	12972	81.46	TL	12972	262.00	TL	12972	431.56	TL	0	inf	TL
	CW2	14144	0.000	0.13	21321	0.00	0.94	23670	0.00	60.18	23670	111.29	TL	23670	207.02	TL	23530	351.07	TL	2967	3477.22	TL
inf/R/U	CW3	34680	0.000	0.23	53517	0.00	1.26	54608	0.00	328.18	54164	125.21	TL	54608	355.14	TL	53820	470.67	TL	12210	2415.43	TL
	okp1	5670	0.000	0.14	9396	0.00	0.23	9960	0.00	8.62	9960	20.42	TL	9960	186.37	TL	9960	241.85	TL	960	17028189.17	TL
	okp2	2387	0.000	0.03	4763	0.00	0.09	8088	0.00	0.85	9656	0.00	314.04	9828	1.75	TL	9772	2.33	TL	648	2986.42	TL
	okp3	2350	0.000	0.07	4650	0.00	0.11	9131	0.00	0.65	9795	2.09	TL	9755	2.51	TL	9751	2.55	TL	5665	253.05	TL
	okp4	6960	0.000	0.12	8501	0.00	0.38	9854	0.00	10.79	9979	26.50	TL	9979	156.30	TL	9970	258.64	TL	2304	1451.91	TL
	okp5	6324	0.000	0.08	10000	0.00	0.21	10000	0.00	62.50	10000	71.76	TL	10000	243.80	TL	10000	444.18	TL	648	8297.84	TL
	cgcut1	132	0.000	0.02	167	0.00	0.15	225	0.00	3.15	244	0.00	108.30	244	49.18	TL	244	49.18	TL	243	49.79	TL
	cgcut2	945	0.000	0.02	1593	0.00	0.22	2388	0.00	2.22	2768	2.89	TL	2856	55.78	TL	2856	55.78	TL	0	inf	TL
	cgcut3	1080	0.000	0.05	1440	0.00	0.29	1800	0.00	7.13	1860	0.00	855.97	1860	112.37	TL	1860	268.82	TL	0	inf	TL
	CW1	2112	0.000	0.11	4107	0.00	0.22	5698	0.00	5.96	6402	22.20	TL	6402	148.56	TL	6402	245.92	TL	0	inf	TL
	CW2	2088	0.000	0.09	3570	0.00	0.65	5150	0.00	12.95	5354	46.06	TL	5354	177.16	TL	5033	233.68	TL	0	inf	TL
inf/F/W	CW3	2226	0.000	0.09	4428	0.00	0.38	5387	0.00	12.61	5689	34.79	TL	5689	160.63	TL	5493	299.73	TL	0	inf	TL
	okp1	16008	0.000	0.02	23674	0.00	0.16	27021	0.00	5.27	27589	0.00	253.81	27589	219.06	TL	27589	223.01	TL	0	inf	TL
	okp2	4850	0.000	0.02	8881	0.00	0.11	16159	0.00	0.54	20824	0.00	241.54	21154	15.44	TL	20054	24.51	TL	0	inf	TL
	okp3	6032	0.000	0.02	11479	0.00	0.10	21012	0.00	0.53	23285	0.00	144.93	23743	12.48	TL	23285	15.34	TL	0	inf	TL
	okp4	18684	0.000	0.06	24804	0.00	0.24	31238	0.00	8.60	32893	0.00	324.22	32893	152.60	TL	32893	202.29	TL	0	inf	TL
	okp5	15956	0.000	0.07	27923	0.00	0.21	27923	0.00	7.86	27923	26.67	TL	27923	308.26	TL	27923	360.22	TL	0	inf	TL
	cgcut1	132	0.000	0.07	204	0.00	0.14	242	0.00	2.29	260	0.00	254.48	260	40.00	TL	260	40.00	TL	257	41.63	TL
	cgcut2	945	0.000	0.10	1593	0.00	0.15	2442	0.00	3.49	2826	29.12	TL	2863	55.40	TL	2858	55.67	TL	2753	61.61	TL
	cgcut3	1080	0.000	0.22	1700	0.00	0.75	1900	0.00	21.44	1900	63.68	TL	1900	169.47	TL	1840	363.04	TL	1260	576.19	TL
	CW1	2136	0.000	0.15	4272	0.00	0.50	6746	0.00	9.79	6746	48.16	TL	6766	165.36	TL	6746	241.65	TL	0	inf	TL
	CW2	2223	0.000	0.20	4311	0.00	0.51	5458	0.00	20.98	5604	61.87	TL	5471	167.19	TL	5354	221.97	TL	315	5470.16	TL
inf/R/W	CW3	2226	0.000	0.21	4440	0.00	0.48	5687	0.00	48.03	5689	65.09	TL	5744	196.38	TL	5687	297.35	TL	436	67175500.00	TL
	okp1	16008	0.000	0.08	26894	0.00	0.21	28090	0.00	9.81	28090	7.85	TL	28090	182.50	TL	28090	217.25	TL	1685	23935463.86	TL
	okp2	4850	0.000	0.12	9409	0.00	0.11	16832	0.00	0.58	22573	0.00	107.10	23513	5.99	TL	23513	6.19	TL	953	4578.17	TL
	okp3	6032	0.000	0.03	11490	0.00	0.09	21012	0.00	0.73	24355	0.00	172.70	24485	9.68	TL	24881	7.94	TL	756	6524.74	TL
	okp4	18684	0.000	0.10	24804	0.00	0.28	31238	0.00	6.47	32893	7.12	TL	32893	140.94	TL	32893	189.09	TL	6120	11268014.15	TL
	okp5	15956	0.000	0.09	27923	0.00	0.18	27983	0.00	9.03	27983	41.91	TL	27983	174.29	TL	27983	359.23	TL	953	13384.47	TL
	cgcut1	0	inf	0.00	108	0.00	0.06	108	0.00	0.34	126	0.00	0.65	140	0.00	2.89	140	0.00	79.10	140	13.57	TL
	cgcut2	0	inf	0.00	1593	0.00	0.06	2082	0.00	0.76	2361	0.00	10.22	2430	0.00	76.49	2430	18.93	TL	2430	78.77	TL
	cgcut3	0	inf	0.05	2277	0.00	0.68	2405	0.00	1.95	2558	0.00	5.75	2599	0.00	21.79	2599	0.00	382.47	2599	37.32	TL
	CW1	0	inf	0.00	9744	0.00	0.34	12342	0.00	2.34	12890	0.00	11.76	12890	0.00	61.28	12890	18.86	TL	12890	88.19	TL
	CW2	0	inf	0.04	21273	0.00	0.17	21273	0.00	1.84	23240	0.00	9.93	23240	0.00	67.41	23240	38.55	TL	23240	85.96	TL
2/F/U	CW3	0	inf	0.01	51752	0.00	0.20	51752	0.00	2.95	54063	0.00	15.13	54063	0.00	145.23	54063	35.47	TL	54063	95.56	TL
	okp1	0	inf	0.03	8280	0.00	0.14	8380	0.00	4.96	9196	0.00	24.43	9366	0.00	54.14	9366	11.30	TL	9366	76.70	TL
	okp2	0	inf	0.01	4355	0.00	0.05	6291	0.00	0.13	8192	0.00	1.31	9390	0.00	29.61	9396	6.43	TL	9396	6.43	TL
	okp3	0	inf	0.04	4650	0.00	0.02	5530	0.00	0.80	8076	0.00	0.93	8682	0.00	25.54	9399	0.00	854.98	9399	6.39	TL
	okp4	0	inf	0.01	8360	0.00	0.13	8960	0.00	1.95	9824	0.00	6.31	9958	0.00	62.58	9958	17.38	TL	9958	94.69	TL
	okp5	0	inf	0.05	8804	0.00	0.11	9787	0.00	3.34	9846	0.00	39.76	9874	0.72	TL	9874	35.03	TL	9874	160.73	TL

Solutions from floating cuts models

2DSKP

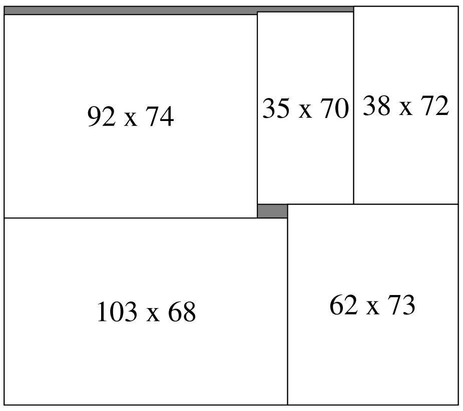


Figure 1: Solution of instance ‘CW2’ on the variant F/U non-staged as non-guillotine constraint, when considering FC5_Strengthened with h=4 (objective function value = 23524).

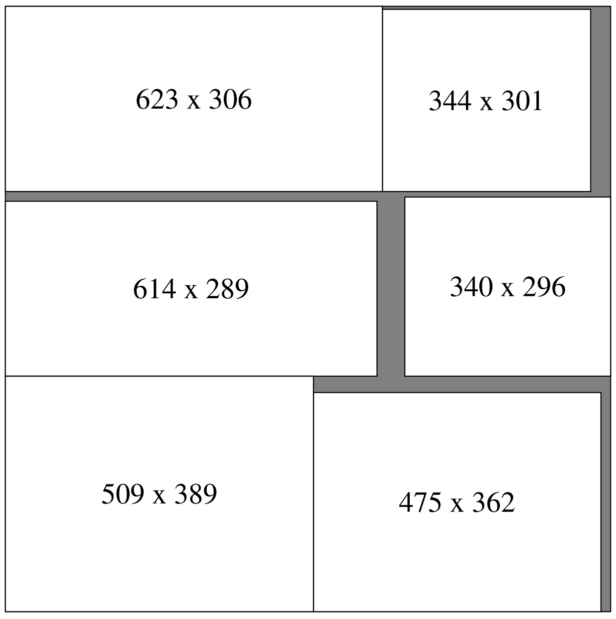


Figure 2: Solution of instance ‘gcut11’ on the variant F/U with 2-stages as guillotine constraint, when considering FC2 with h=6 (objective function value = 942219).

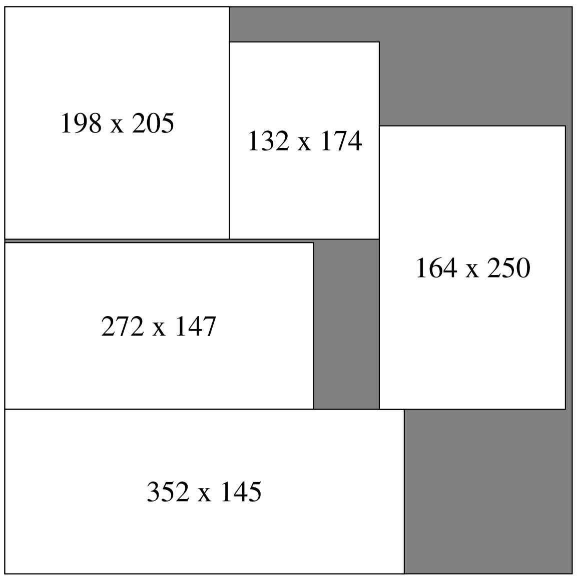


Figure 3: Solution of instance ‘gcut5’ on the variant F/U non-staged as guillotine constraint, when considering FC4_Strengthened with h=4 (objective function = 195582).

2DSLOPP

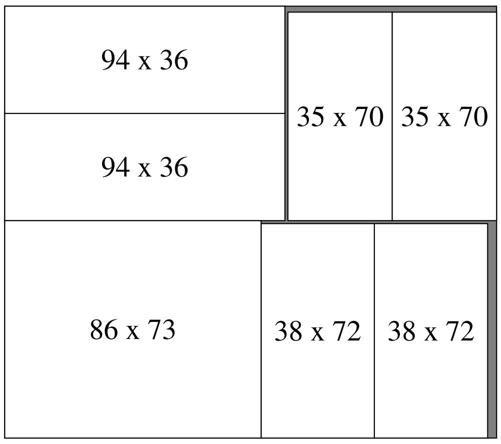


Figure 4: Solution of instance ‘CW2’ on the variant F/U non-staged as guillotine constraint, when considering FC4_Strengthened with h=4 (objective function = 23418).

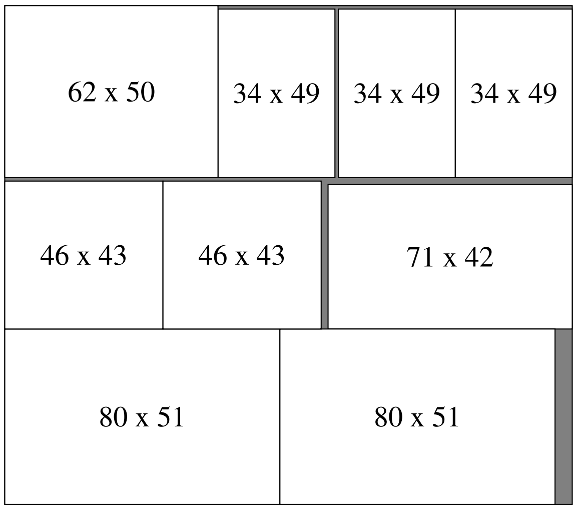


Figure 5: Solution of instance ‘CW2’ on the variant F/U non-staged as guillotine constraint, when considering FC2 with h=5 (objective function value = 23196).

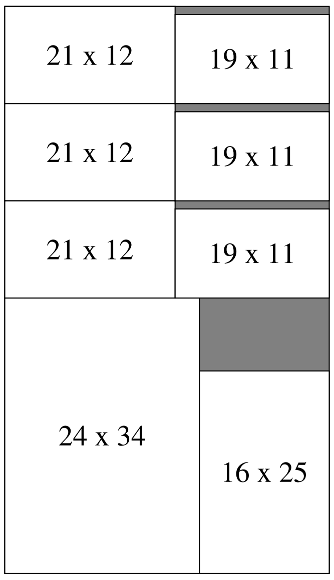


Figure 6: Solution of instance ‘3s’ on the variant F/U with 2-stages as guillotine constraint, when considering FC2 with h=6 (objective function value = 2599).

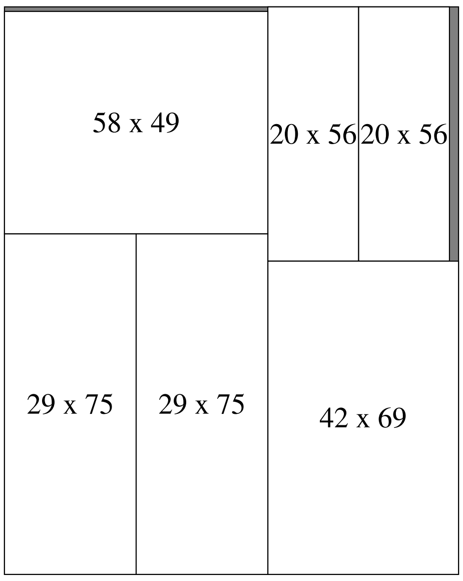


Figure 7: Solution of instance ‘cu1’ on the variant F/U non-staged as guillotine constraint, when considering FC4_Strengthened with h=4 (objective function = 12330).

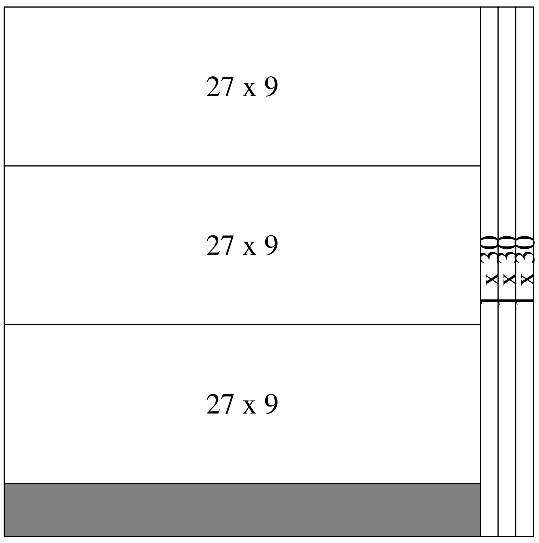


Figure 8: Solution of instance ‘ngcut10’ on the variant F/W non-staged as non-guillotine constraint, when considering FC5_Strengthened with h=4 (objective function = 1452).