The three-dimensional rectangular Multiple Bin Size Bin Packing Problem with transportation constraints
A case study in the field of air transportation

A dissertation submitted to the faculty of HEC Liège
in fulfilment of the requirement for the degree
of PhD in Economics and Management
by

Célia PAQUAY

J'ai toujours préféré la folie des passions à la sagesse de l'indifférence.

Anatole France

Contents

Acknowledgements

The first person I would like to thank is my advisor Professor Sabine Limbourg. This thesis would not have come to an end without her. Sabine, I thank you for your advice, your constant support, your availability at any time, your trust and your daily passion about what you undertake. You were a fantastic guide during these six years.

This thesis would not have started without my co-advisor, Professor Michael Schyns. I thank you for offering me the opportunity to experience a doctorate and teaching assistantship. I am also grateful for your trust, your support, your advice and all your suggestions.

I am sincerely grateful to Professor José Fernando Oliveira who took part in my thesis committee. Thank you for your ideas and suggestions, for your support and your availability and thank you for sharing your experience and your knowledge with me. You welcomed me in the ESICUP group which helped me and guided me during these six years.

I would like to acknowledge all jury members: Professors Ramón Alvares-Valdes, Yasemin Arda, An Caris and Greet Vanden Berghe who accepted to read this manuscript and to take part in this work.

I also thank Felix Brandt for providing appropriate data sets for the specific problem studied in this thesis.

A very special thanks to Virginie Lurkin and Julien Hambuckers for your support and your presence even from abroad. I really appreciated our discussions about sciences, life and anything else. Thanks for your passion and your refreshing spontaneity and authenticity.

To my fantastic officemates, Eli, Stéphanie, Anne-Sophie and Stefano, I would like to thank you for your constant good mood and your support. You brought a great atmosphere to this office with your jokes and your smiles. It has been a huge pleasure to share this office with you guys! I also thank every QuantOM member. I enjoyed working at the third floor of HEC during these six years also because of the people who are part of it. I will not forget our coffee breaks and lunch times, sometimes peaceful and quiet and sometimes really animated.

I also would like to thank Valérie, Colette and Eric, my teaching colleagues. You show how easy working in a team can be. Thank you for your discussions and your desire to improve the quality of your work again and again. Your passion and your good mood are contagious. It was an honour to share these six-year experience with you and to learn how teaching to students can be rewarding.

A big thanks to my friends. You shared the up and downs during these six years, thank you for your support and your advice. A special thanks to Audrey and Arnaud. You have been present for me at any time, thank you for always giving me good advice and being so supportive.

Je souhaite également remercier ma famille. Je remercie mon parrain, ma marraine et tonton M. pour toutes leurs petites attentions qui m'ont fait chaud au coeur et qui m'ont aidée à atteindre mes objectifs. Merci à Stéph, Sissi, Jade, Xavier et Chris pour vos encouragements et votre soutien. Merci à mes parents. Vous avez toujours été compréhensifs, patients et présents peu importe les projets un peu fous dans lesquels je me suis lancée, la thèse n'étant qu'un d'entre eux. Vous m'avez appris à m'investir dans chaque projet, mais aussi à garder les pieds sur terre et à cerner l'essentiel dans chaque chose.

Merci également à mes beaux-parents pour votre présence, votre soutien et vos encouragements, en particulier durant ces derniers mois.

Finally, I want to thank Lio. You have been fantastic during these six years. I am so grateful to share my life with you every single day. These last months have been tough for many different reasons but I could not expect a better coach and team-mate than you. Thank you for being so supportive and understanding, thank you for your outspokenness and your honesty. You make my life colourful. T'aime choubi

Summary

According to the International Air Transport Association and Air Transport Action Group, 51.3 million metric tons of goods were transported by airlines in 2014. To transport luggage, freight and mail, special containers, called Unit Load Devices (ULD), are used. The method of loading packages into ULDs represents a key element for cargo safety and aircraft weight and balance, as well as for the economy of airline companies.

This thesis aims to solve the problem of packing a set of boxes into containers of various shapes without wasting loading space. The goal is to select the best set of ULDs to pack all the boxes achieving a minimum unused volume. As for all the packing problems, geometric constraints have to be satisfied: items cannot overlap and have to lie entirely within the bins. The richness of this application is to manage additional and common constraints: the bin weight limit, rotations, stability and fragility of the boxes, and weight distribution within a ULD. In practice, this problem is manually solved with no strict guarantee that the constraints are met.

First, the problem is formulated as a mixed integer linear program. As this problem is NP-hard, it opens the way to heuristics. A second approach makes use of the formulation to apply three matheuristic methods, combining exact approaches and heuristics. Third, a tailored two-phase constructive heuristic is developed for this specific problem; it aims to find good initial solutions in short computational times. These approaches contain parameters that have been tuned using the irace parametrisation technique. For the experiments, several instances have been created on the basis of a box data set which stems from a real world case.

List of Abbreviations

ATAG	Air Transport Action Group
B&B	Branch-and-Bound
C&P	Cutting and Packing
CG	Centre of Gravity
EP	Extreme Point
FRF	Fractional Relax-and-Fix
I&F	Insert-and-Fix
J&S	Jump and Shift
IATA	International Air Transport Association
MILP	Mixed Integer Linear Program
MBSBPP	Multiple Bin Size Bin Packing Problem
R&F	Relax-and-Fix
RS	Residual Space
ULD	Unit Load Device

Chapter 1

Introduction

Air freight has been evolving for more than a hundred years. On November 7th 1910, the first cargo flight transported silk from Dayton to Columbus in Ohio, USA. However, freight flights remain rare before the First World War (Allaz (2005)). Due to World War II, aircraft had been improved enough to be able to directly cross the Atlantic. Thanks to this, air transportation for cargo and passengers has experienced a huge growth since the 1970s. However, this growth was not steady and faced several setbacks due to recessions (1973-1975; 1980-1984; 1990-1991; the Asian Crisis of 1997; 2008-2009) or geopolitical instability (Gulf War of 1991; September 11 2001). The dramatic growth can also be explained by the increase in carried volume as well as in the average travelled distance. Moreover, the development of passenger services tends to induce freight demand as each additional plane usually offers additional cargo capacity (Rodrigue et al. (2013)).

Efficient and affordable air freight has contributed to many changes (Rodrigue et al. (2013)). Among others, one can first notice a dietary change by the increasing availability of new products or products in seasons during which they would not normally be available. Second, changes in retail can be observed. For instance, merchandise can be purchased online and shipped promptly by air transport. This trend will certainly continue to increase in the future due to the development of e-business. A third change occurs in manufacturing. Let us consider the example of a computer manufacturer depending on the global shipment of various components in the manufacturing and assembly processes. The increased importance of time-based competition ensures that the specific benefits of air cargo augur well for the future growth of air transportation.

According to the International Air Transport Association (IATA) (IATA (2016a)) and Air Transport Action Group (ATAG) (ATAG (2016)), 51.3

million metric tons of goods were transported by airlines in 2014, which represents more than 35% of global trade by value, but less than 1% of world trade by volume. Indeed, goods shipped by air are typically very high value commodities that are often perishable or time-sensitive. This is equivalent to $6.8 trillion worth of goods annually, or $18.6 billion worth of goods every day. Air freight is particularly suitable for supporting "just-in-time" production and distribution strategies with low inventory levels. Air cargo also has a niche market for emergency situations where the fast delivery of supplies prevails over cost issues (Rodrigue et al. (2013)).

To transport luggage, freight and mail, special containers are used. These are called Unit Load Devices (ULD) and can be described as an assembly of components consisting of a container or of a pallet covered with a net, whose purpose is to provide standardised size units for individual pieces of baggage or cargo, and to allow for rapid loading and unloading (Limbourg et al. (2012)). ULDs may have specific shapes to fit inside aircraft, such as truncated rectangular parallelepipeds. Two common ULDs are illustrated in Figure 1.1.

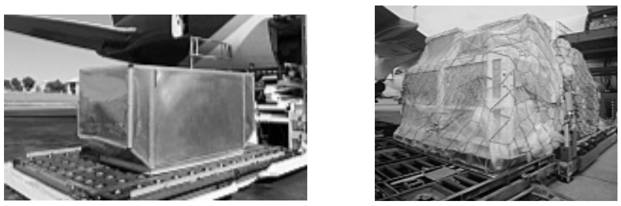


Figure 1.1: Different shapes of ULDs

Good packing of these ULDs is crucial for different reasons. First, a correct and stable loading of the ULDs prevents damage to their contents. In particular, the possible fragility of some boxes has to be taken into account. Second, ULDs are the primary cause of aircraft damage among all ground operations equipment. For this reason, IATA develops and maintains standards and procedures concerning the specifications, handling, restraint and maintenance of ULDs (IATA (2016b)). For instance, an unbalanced ULD can lead to mistakes in the centre of gravity calculation of the aircraft and to instability of the cargo. This shows how important it is to correctly pack the contents of these ULDs. Third, with around 900,000 aircraft ULDs in service representing a replacement value of over $1 billion, ULDs are expensive assets that require correct handling. Every year, the total cost of both repair and loss of aircraft ULDs is estimated to be about $300 million, excluding flight delays and cancellations due to their unavailability, and aircraft damages caused by improper ULD handling (IATA (2016b)). Therefore, efficiently using the volume inside each ULD is crucial in order to reduce the number required. For these reasons, the method of loading packages into the ULDs represents a key element for the safety of the cargo and of the aircraft, as well as for the economy of airline companies. However, in practice, this loading phase is manually achieved with no strict guarantee that the constraints are met and without ensuring that the volume inside the ULDs is correctly exploited.

The subject of this thesis is to solve the problem of packing a set of cuboid[1] boxes into containers of various shapes without wasting loading space. There are few identical boxes and they all have to be loaded. With regard to the containers, this thesis deals with the specific case of air cargo. In this context, containers are ULDs and thus, there are only several types of available bins. The aim is to select the best set of ULDs to pack all the boxes achieving a minimum unused volume. In this way, the volume of the ULDs is not wasted. Depending on the chosen application, other objectives can be defined such as minimising the costs of the selected ULDs. At the same time, a formal description of the proposed solution, called a *loading pattern*, is provided. Obviously, this work can be extended to many other packing applications. For instance, ULDs can be replaced by simple Euro pallets or by the loading volume of vehicles.

This problem is a combinatorial optimisation problem which belongs to the family of Cutting and Packing problems (C&P). As will be detailed in the next chapter, this has been labelled as a three-dimensional Multiple Bin Size Bin Packing Problem using the typology defined by Wäscher et al. (2007). Since our problem is a packing problem in three dimensions, it therefore also belongs to the family of Container Loading Problems according to the definition given in Bortfeldt and Wäscher (2013).

One of the main contribution of this thesis is the set of constraints taken into account. In the final loading pattern, all the boxes have to be packed without overlap and lie within the ULD. The weight capacity of each loaded ULD has to be respected. The boxes can orthogonally rotate, i.e., the edges of the boxes have to be either parallel or perpendicular to those of the ULDs. Sometimes only a limited number of orientations are allowed due to the contents of the boxes. For the same reason, some boxes may be fragile and thus are not able to support other boxes. Each box of the loading pattern has to be correctly supported to be stable. In air transportation, when ULDs are

[1]a regular six sided solid form

packed inside the airplane, the centre of gravity is computed assuming each ULD has a centre of gravity close to the geometrical centre of its basis, i.e., the weight distribution is uniform. This is therefore included in our problem as a hard constraint. All these constraints as well as those not considered in this thesis will be extensively explained in Chapter 2. This thesis is the first work studying the three-dimensional Multiple Bin Size Bin Packing Problem that takes into account all these constraints simultaneously.

The second important contribution of this thesis is the set of methods used to solve this problem. In Chapter 3, a Mixed Integer Linear Programming (MILP) formulation for this specific three-dimensional Multiple Bin Size Bin Packing Problem is developed. Such complete linear formulation cannot be found in the existing literature and the proposed research is the first to take up this challenge. As expected, considering this problem is NP-hard, this formulation cannot be efficiently solved. For this reason, Chapter 4 makes use of this model to apply three matheuristic methods, namely the Relax-and-Fix, the Insert-and-Fix and the Fractional Relax-and-Fix heuristics. Matheuristics combine exact approaches and heuristics. This is the first time that these three matheuristics are extended to packing problems. In order to compare the efficiency of these matheuristics, Chapter 5 proposes a tailored two-phase constructive heuristic. This heuristic is specially designed for the specific problem studied in this thesis and is thus developed to take into account all its features.

The third contribution is a practical one. Until now, there have been no benchmark instances for the problem considered in this thesis. Several box instances have been created on the basis of a box data set which stems from a real world case. These instances are available online to allow other researchers to compare their results. Information about these data sets is provided in Chapter 6 and they are used in computational experiments to tune the parameters of each method, to make a sensitivity analysis and then to compare all the methods. Finally, Chapter 7 of this thesis will draw some conclusions and give some insights about future work.

The outline of this thesis is represented in Figure 1.2.

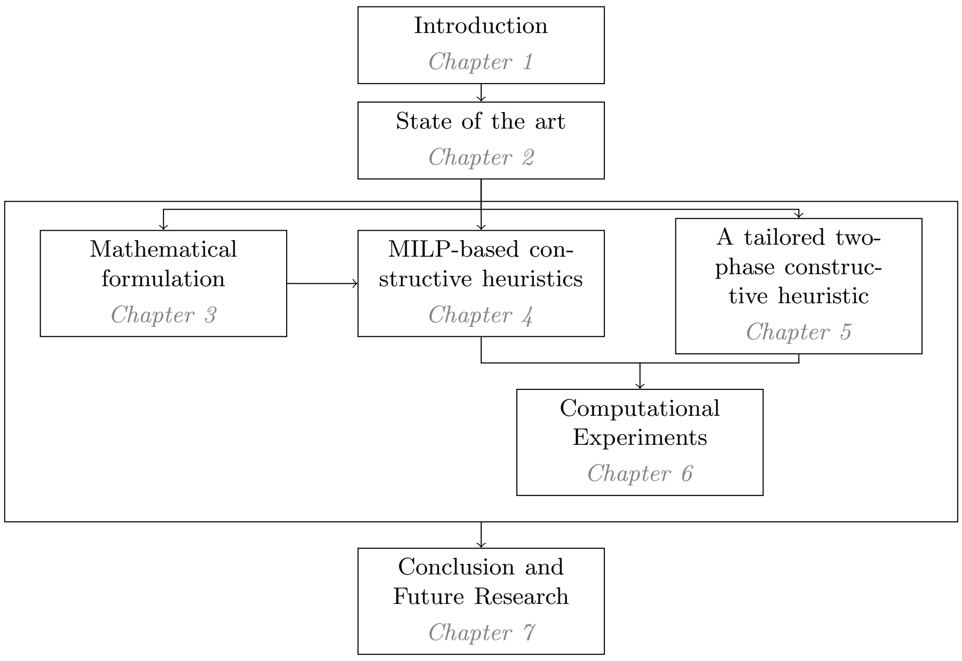

Introduction
Chapter 1
State of the art
Chapter 2
Mathematical formulation
Chapter 3
MILP-based constructive heuristics
Chapter 4
A tailored two-phase constructive heuristic
Chapter 5
Computational Experiments
Chapter 6
Conclusion and Future Research
Chapter 7

Figure 1.2: Thesis outline

Chapter 2

State of the art

The present chapter aims to review the literature around the Cutting and Packing (C&P) problems. More precisely, typology from Wäscher et al. (2007) is described in order to emphasise the differences between the types of problems in Section 2.1. Some descriptions of the existing constraints based on Bortfeldt and Wäscher (2013) are provided in Section 2.2. The contribution of this thesis is highlighted within this section. Afterwards, several insights about the methodologies used in C&P problems are drawn up in Section 2.3. In Section 2.4, some related problems are presented to show the importance of C&P problems in the operations research area. Finally, the problem definition is provided in Section 2.5. The following literature review is not exhaustive but aims to represent a broad view of the state-of-the-art.

The current position in the thesis outline is shown in bold in Figure 2.1.

2.1 Typology of Cutting and Packing problems

Cutting and Packing problems have plenty of applications and have been considerably studied during the last 30 years. In order to unify definitions and notations and thus also facilitate the communication between researchers in the field, a typology has been developed in Dyckhoff (1990) and later improved in Wäscher et al. (2007). Moreover, this typology aims to identify possible blank spots. According to Wäscher et al. (2007), C&P problems have a similar structure which can be summed up as follows:

> *Given are two sets of elements, namely a set of large objects (input, supply) and a set of small items (output, demand), which are defined exhaustively in one, two, three or more geometric dimensions. Select some or all small items, group them into one*

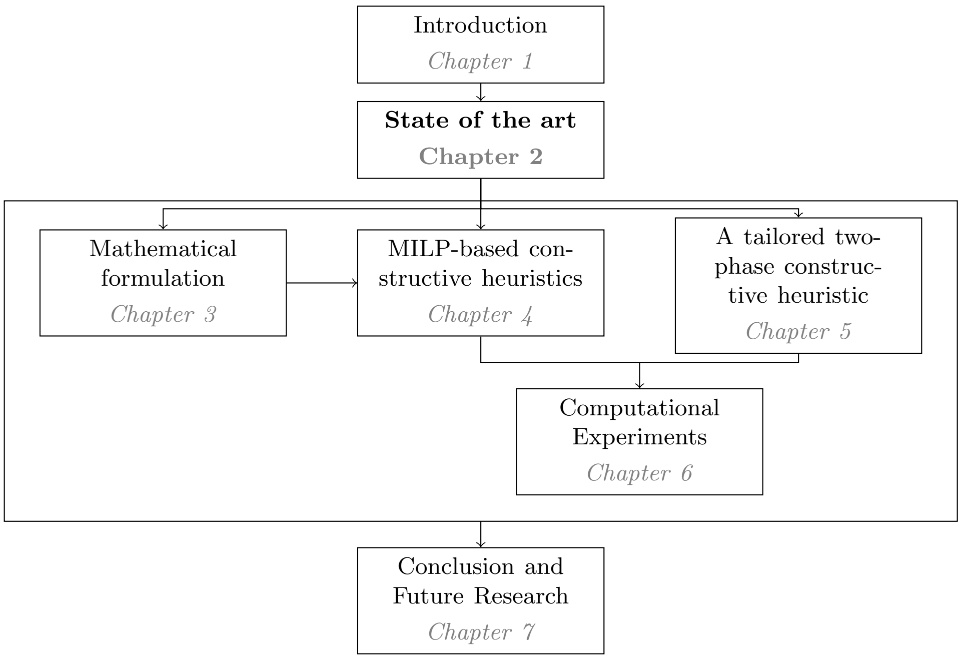

Introduction
Chapter 1
State of the art
Chapter 2
Mathematical formulation
Chapter 3
MILP-based constructive heuristics
Chapter 4
A tailored two-phase constructive heuristic
Chapter 5
Computational Experiments
Chapter 6
Conclusion and Future Research
Chapter 7

Figure 2.1: Current position in the thesis outline

or more subsets and assign each of the resulting subsets to one of the large objects such that the geometric conditions hold, i.e., the small items of each subset have to be laid out on the corresponding large object such that

- *all small items of the subset lie entirely within the large object and*
- *the small items do not overlap,*

and a given (single-dimensional or multi-dimensional) objective function is optimised.

The large objects may be real containers but also pallets which can be filled or the loading space of a truck for example.

The improved typology uses five criteria to classify the problems:

1. **Kind of assignment**: two objectives may exist: the output maximisation and the input minimisation. In the case of output maximisation, the set of large objects is not sufficient to accommodate the set of small items. A selection among the small items has thus to be done. On the contrary, if the aim is the input minimisation, then the set of large

objects is sufficient and all the small items are assigned to a selection of the large objects.

2. **Assortment of small items**: the typology considers three possibilities. There can be identical small items, a weakly heterogeneous assortment of small items (they can be grouped into relatively few classes for which the items are of identical size and shape) or a strongly heterogeneous assortment of small items (only a few elements are of identical size and shape).

3. **Assortment of large objects**: there can be only one large object or several. If there are several large objects, as for the small items, there are three possibilities: they are identical, there is a weakly heterogeneous assortment or a strongly heterogeneous assortment.

4. **Dimensionality**: the problems can be considered in one, two or three dimensions.

5. **Shape of small items**: the typology distinguishes the regular small items (rectangles, circles, boxes etc.) and irregular items.

The first two criteria define what is called the *basic problem types*, adding the third criterion clarifies the *intermediate problem types.* Considering all the five criteria gives the *refined problem types.* Replacing the assumptions with different ones (e.g., multiple objectives, stochastic problems, higher dimensions, etc.) leads to problem types which will be considered as *problem variants.* More details about each criterion and deeper classification are provided in Wäscher et al. (2007).

Considering the two first criteria, the typology defines six basic problem types which can be represented with a hierarchical structure as shown in Figure 2.2. Note that for the input minimisation, the large objects can have all their dimensions fixed or it may be possible to consider that one has a variable dimension. In this case, the problem consists in fixing the value of this variable dimension. As has been explained in the previous chapter, the aim of this thesis is to provide a method to pack a set of different cuboid boxes into a selection of ULDs. Thus, it deals with an input minimisation where all dimensions are fixed and the assortment of boxes is strongly heterogeneous. Therefore, as shown in Figure 2.2, the problem appears to be a Bin Packing Problem (BPP). The difference with the Cutting Stock Problem is the assortment of small items: the symmetry of small items is taken into account to solve those problems.

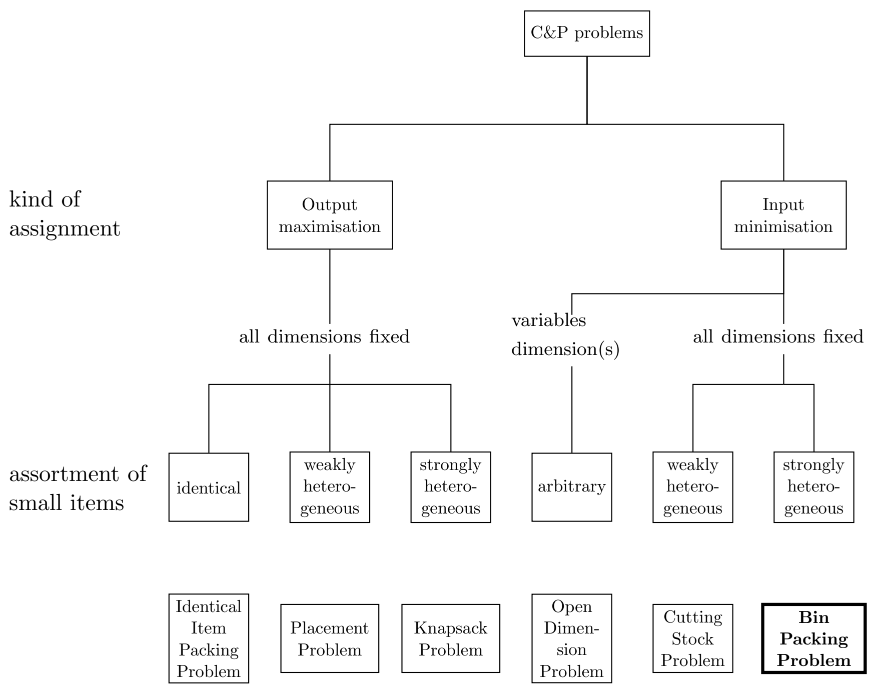


Figure 2.2: Basic Cutting and Packing problem types (from Wäscher et al. (2007))

For each basic problem type, several intermediate types are defined depending on the assortment of large objects. For the BPP, the intermediate types are provided in Figure 2.3. Since there are few types of ULDs, the intermediate type of our problem is a **Multiple Bin Size Bin Packing Problem** (MBSBPP). With the five criteria of the typology, the thesis deals with a three-dimensional rectangular MBSBPP.

Bin Packing Problem

- identical large objects → Single Bin Size Bin Packing Problem
- weakly heterogeneous assortment of large objects → **Multiple Bin Size Bin Packing Problem**
- strongly heterogeneous assortment of large objects → Residual Bin Packing Problem

Figure 2.3: Intermediate types of the Bin Packing Problem

Bortfeldt and Wäscher (2013) define *Container Loading Problems* as C&P problems in three dimensions, where small items are called cargo and large objects are called containers. This work also corresponds to this definition.

2.2 Constraints in Container Loading Problems

In the following, the geometric and the specific constraints are distinguished. The typology from Wäscher et al. (2007) does not consider the specific constraints of the problem, which represent one distinctive feature of the work.

2.2.1 Geometric constraints

The geometric constraints include that all boxes lie entirely within the ULDs, they do not overlap and are assumed to be placed orthogonally, i.e., the edges of the boxes have to be either parallel or perpendicular to those of the ULDs.

To lie entirely within the ULDs does not seem complicated but, in this thesis, ULDs may have a special shape to fit in the fuselage of the aircraft: some of them look like parallelepipeds that have been cut as shown in Figure 2.4. In Figure 2.4, one can see an aircraft cross-section where three ULDs are represented; two of these are on the lower deck and the third on the main deck. It is clear that the boxes must lie within the ULDs and therefore one should pay attention to their particular shapes (e.g., Chan et al. (2006)).

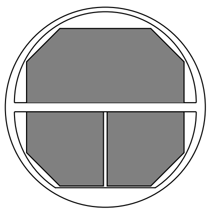


Figure 2.4: Special shapes of some ULDs to fit in the fuselage: ULDs are represented in gray

This uncommon container shape makes the problem a variant of the MB-SBPP. However, if this feature is dropped, the work can be extended to more general applications in fields other than air transportation.

2.2.2 Specific constraints

Twenty-two years ago, Bischoff and Ratcliff (1995) stated the lack of considered constraints in most papers about Container Loading Problems. Later, Bortfeldt and Wäscher (2013), on the basis of the paper by Bischoff and Ratcliff (1995), present a broad set of additional constraints that might be encountered in practical packing situations. Each type of constraint is explained in the following, and the selection of those taken into account in this thesis is clarified.

Container weight limit Usually, containers have a maximal weight limit, thus items can be loaded as long as this weight capacity is not exceeded. This type of constraint is quite common in the literature (e.g., Fraser and George (1994), Terno et al. (2000), Chan et al. (2006), Ceschia and Schaerf (2013)) and is particularly important when there are some high density items.

In our case, ULDs indeed have a maximum weight capacity and thus, the sum of the weights of the packed boxes cannot exceed this limit.

Weight distribution constraints in the containers Weight distribution constraints, also called load balance constraints, ensure that the weight of the cargo inside a container is evenly distributed as much as possible across the floor. A balanced cargo reduces the risk of shifting when the container is moving and thus lowers the risks of accidents. Some operations such as crane lifting may even become impossible if the weight is too unevenly distributed (Bischoff and Ratcliff (1995)). As done in Fraser and George (1994), Davies and Bischoff (1999), Jin et al. (2003), Techanitisawad and Tangwiwatwong

(2004), Chan et al. (2006), Moon and Nguyen (2013), Costa and Captivo (2016), Trivella and Pisinger (2016) among others, the centre of gravity (CG) of the load should be close to the geometrical mid-point of the container floor. This type of constraint can also be met in road transportation: the loaded truck has to respect a precise distribution of the cargo over the axles of the vehicle (Lim et al. (2013), Pollaris et al. (2015), Alonso et al. (2017)). Legislation about axle weight limits varies between countries and more details for European countries can be found in International Transport Forum (2016).

In the context of air cargo, once the ULDs have been filled, they are loaded into a compartmentalised cargo aircraft with some technical and safety constraints. The ULDs are loaded in such a way that the CG of the loaded plane is as close as possible to a recommended position determined by safety and fuel economy considerations and that a weight limit is satisfied at each inch slice of the aircraft (e.g., Amiouny et al. (1992), Mongeau and Bès (2003), Limbourg et al. (2012), Vancroonenburg et al. (2014), Lurkin and Schyns (2015)). According to the control and loading manual of some airline companies such as Boeing (Boeing (2008)), the CG of each ULD must lie in a determined area (depending on the ULD type) around the geometrical centre of the ULD and below a maximum height. This implies some uniformity in the weight distribution inside the ULDs. Based on this, the CG of each ULD is considered as a point in the centre of the position occupied to calculate the CG of the plane and to ensure some weight constraints. Therefore, the uniform weight distribution constraints are crucial and have to be taken into account in this thesis.

Loading priorities for the items In the case of an output maximisation problem, a subset of the small items is loaded and the others are left behind. In practice, some of the available items can have higher priority than the loading of others (Junqueira et al. (2012)), for example because of delivery deadlines or for limited shelf life products. Some loading priorities can thus be defined.

This type of constraint is not considered in this thesis since this deals with an input minimisation problem, i.e., all the boxes have to be packed in the studied case.

Orientation constraints for the items Boxes are assumed to be placed orthogonally, i.e., the edges of the boxes have to be either parallel or perpendicular to those of the containers. If the box can rotate and if each dimension can be in a vertical position, then six orientations are possible as shown in Figure 2.5. In practice, however, some boxes may not rotate in all directions

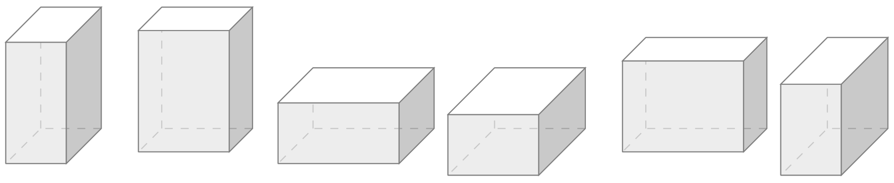


Figure 2.5: Six possible orientations

because of their contents. These constraints are called *orientation constraints* (Fraser and George (1994), Chen et al. (1995), Terno et al. (2000), Jin et al. (2003), Techanitisawad and Tangwiwatwong (2004), Lin et al. (2006), Chan et al. (2006), Almeida and Figueiredo (2010), Junqueira et al. (2012), Ceschia and Schaerf (2013)).

In this thesis, it is considered that some dimensions/edges may not be in a vertical position. Every time a dimension cannot be vertical, two orientations are no longer possible. A parameter is thus defined for each dimension to describe whether it can be in a vertical position or not.

Stacking constraints Also called load-bearing constraints, this type of constraint describes how many boxes can be placed on top of each other. More generally, load bearing strength refers to the maximum pressure that can be applied over the top face of a box without damaging it. How much pressure or weight a box can hold depends on the material and the construction of the boxes. Different strategies have been developed to manage this feature of the cargo (Terno et al. (2000), Techanitisawad and Tangwiwatwong (2004), Lin et al. (2006), Junqueira et al. (2012), Ceschia and Schaerf (2013)). This constraint is quite important in practice because it prevents damage to products contained in a fragile box.

In this work, a box is said to be fragile if no box can be placed on its top face.

Complete shipment constraints As for the loading priority constraints, the complete shipment constraints appear only with the output maximisation objective. Since there is a selection of small items, some are left behind. This type of constraint states that if one item of a subset is loaded, then all other items of the same subset have to be packed as well. Inversely, if one box of a subset cannot be packed, then no item of the same subset can be loaded (Eley (2003)). This kind of constraint may arise, for example, when parts of a piece of furniture are packed separately and have to be assembled on site at a customer's location.

This constraint is not considered in this thesis as all the boxes must be packed.

Allocation constraints This type of constraint arises when there are several containers in the problem (Tsai et al. (1993), Terno et al. (2000), Eley (2003), Chan et al. (2006), Almeida and Figueiredo (2010)). One can distinguish between the connectivity and the separation constraints. The connectivity constraints demand that items of a particular subset go into the same container, for example because they go to the same destination or because a customer wants to receive their order in a single consignment. Conversely, the separation constraints require that some items are accommodated in different containers for safety reasons. For example, food and perfumery have to be separated during shipping.

These constraints are not considered in this thesis because they represent particular cases. However, they could be an interesting extension of this work in the future.

Positioning constraints These constraints limit the location of items within the container either in absolute or in relative terms (Terno et al. (2000), Lin et al. (2006)). On the one hand, absolute positioning constraints specify where items should or should not be located within the container. For instance, volatile liquids or explosives should be packed near the opening of the bins so that they can be accessed and removed quickly if necessary. On the other hand, relative positioning constraints state whether items should or should not be located close to each other. For instance, items which alter the quality of other items (like food and petrol) must not be placed next to each other.

Note that *multi-drop* constraints are a combination of absolute and relative positioning constraints. These constraints arise when a Vehicle Routing Problem is combined with the packing problem and thus subsets of items go to different customers. The arrangement of the subsets of items should reflect the sequence according to which they will be delivered in order to avoid unnecessary unloading and reloading operations (e.g., Junqueira et al. (2012), Ceschia and Schaerf (2013), Alvarez-Martínez et al. (2015)).

This type of constraint is not considered here, because the journey is direct and thus all the ULDs are unloaded at the destination. Nevertheless, as for the allocation constraints, it could be a possible extension of this thesis.

Stability constraints Load stability is one of the most important types of constraint. Indeed, unstable loads may result in damaged cargo and even in injuries of personnel during handling operations. Cargo stability involves the vertical (or static) and the horizontal (or dynamic) stability (Terno et al. (2000), Jin et al. (2003), Techanitisawad and Tangwiwatwong (2004), Lin et al. (2006), Chan et al. (2006), Junqueira et al. (2012), Ceschia and Schaerf (2013)). For the sake of vertical stability, the bottom side of each box needs to be supported by the top face of other boxes or by the container floor. This constraint is also called *static stability* as it deals with static containers. The vertical stability excludes floating boxes. Deeper analysis of static stability can be found in Ramos et al. (2016). The horizontal stability refers to the capacity of the box to withstand the inertia of its own body when being moved. The boxes remain in their position with respect to x and y axes, hence the name *horizontal stability.*

This thesis only considers vertical stability because horizontal stability could be obtained by adding a special sheet increasing the friction coefficient, by adding foam to fill in holes between boxes or by binding unstable boxes.

Complexity constraints The proposed loading patterns sometimes have to be easy enough for the loading personnel to be able to visualise quickly. Moreover, more automatic packing technologies are not always suitable for complex cargo arrangement and thus may require extra labour. This type of constraint covers these limitations of human and technical resources (de Queiroz et al. (2012)). The most frequently considered complexity constraint is the guillotine cutting constraint. A packing is called *guillotineable* if it can be obtained by a series of cuts parallel to the container faces (Liu et al. (2014)).

However, as explained in Bortfeldt (2012), this kind of constraints is not always appropriate in practice since it may reduce the stability of the cargo when being transported. For this reason, these will not be considered in this thesis.

Summary In order to measure how often the different constraints are considered, Bortfeldt (2012) listed 163 papers that are publicly available and published between 1980 and the end of 2011 in English in international journals, edited volumes and conference proceedings. Among these 163 papers, only 12 (i.e., 7.4%) addressed the MBSBPP. The authors also present the number of papers in which constraint types have been addressed. The percentages are represented in Figure 2.6.

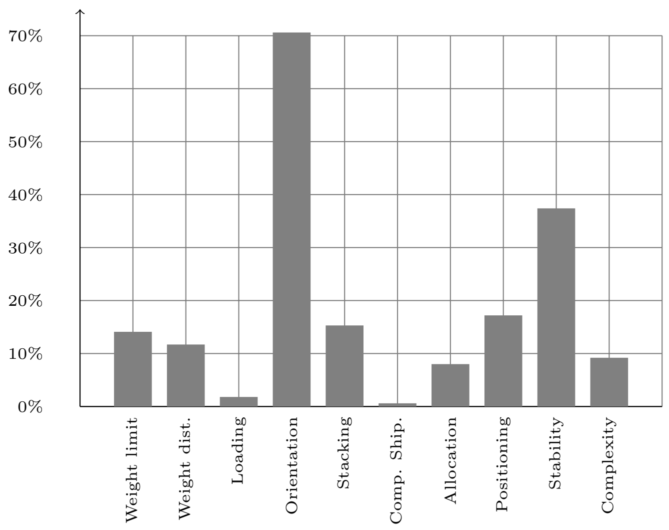


Figure 2.6: Percentage of papers in which constraint types have been addressed (population size = 163) from Bortfeldt (2012)

2.3 Algorithms in Container Loading Problems

Zhao et al. (2016) present a review of the different solution methodologies and a comparison of algorithm performance across the literature. The authors made a distinction between placement and improvement heuristics, and exact methods.

Placement heuristics are commonly known as constructive heuristics. They are used to decide how to put the boxes in the container, either to generate an initial solution, or as an embedded method in a more general approach. If the set of boxes is weakly heterogeneous, boxes of a same type can be arranged together in rows or columns, leading to *wall building* (George and Robinson (1980), Moura and Oliveira (2005)) and *layer building* (Bischoff and Ratcliff (1995), Costa and Captivo (2016)) algorithms. Conversely, if the mix of boxes is strongly heterogeneous, boxes will be placed one at a time. In addition to the decision of how to pack the boxes, one has to decide which box (or box type) has to be packed at each step. To answer this question, Zhao et al. (2016) make the distinction between static predetermined ordering (Crainic et al. (2008)) and dynamic ordering (Eley (2003)). The former uses some sorting criteria such as the volume or the height of the box, whereas the latter is based on the unused volume remaining after the next placement. Most placement heuristics model available spaces to decide where to pack a box. Martello et al. (2000) and Crainic et al. (2008) choose to identify placement points. These are the candidate placement positions for placing the unpacked boxes. This will be applied in Chapter 5.

Improvement heuristics aim to enhance solutions provided by placement heuristics. In general, they define and explore different neighbourhood structures (among others, Techanitisawad and Tangwiwatwong (2004), Parreño et al. (2010), Ceschia and Schaerf (2013)).

Zhao et al. (2016) highlight that the number of exact algorithms in 3D container loading, e.g.,Tsai et al. (1993), Chen et al. (1995), Junqueira et al. (2012), is small compared to the number of developed heuristics. According to Zhao et al. (2016), two elements may justify this observation: first, the difficulty in representing possible patterns or practical packing constraints and second, the expected large computational times considering the complexity of the problem.

2.4 Related problems

Cutting and Packing problems can be applied in many different fields and sometimes combined with other optimisation problems. Here are three examples.

Routing Problem The packing problem can be considered in combination with routing problems with loading constraints, which is called a *three-dimensional Loading Capacitated Vehicle Routing Problem* in the literature (e.g., Gendreau et al. (2006), Iori et al. (2007), Junqueira and Morabito (2015), Côté et al. (2017)). The idea is the following: a set of goods, packed into boxes, has to be delivered to different customers. The objective is to find minimum cost delivery routes for a fleet of identical vehicles that, leaving a depot, visit all the customers only once and then come back to the depot. With respect to the packing of the goods, the cargo has to respect the typical geometric C&P constraints as well as the multi-drop constraints explained in Section 2.2.2. Some other constraints can also be added such as the stability of the cargo. The challenge is to simultaneously optimise the planning of the vehicles' routes and the cargo arrangement inside the vehicles.

Air Cargo Loading Problem As briefly explained in the weight distribution constraints, a cargo aircraft generally contains multiple decks, each one being partitioned into distinct positions. A position is simply a particular aircraft space that accommodates exactly one ULD and can only accept some specific types of ULDs, depending on their contour, type and weight. The structure of the Boeing 747 is shown in Figure 2.7. Aircraft loading is subject to strict safety constraints, with respect to the stress imposed on the structure of the plane.

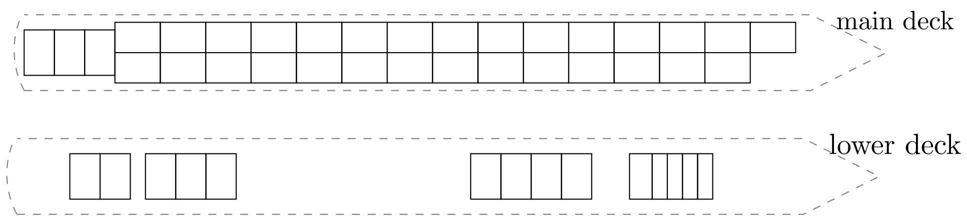


Figure 2.7: Possible positions for the ULDs inside a Boeing 747

As explained in Limbourg et al. (2012), Vancroonenburg et al. (2014), Lurkin and Schyns (2015) among others, the plane must be balanced longitudinally and transversally, the loading must respect a weight limit at each one inch slice of the aircraft. Moreover, the position of the centre of gravity of the loaded plane highly influences the manoeuvrability but also the fuel consumption of the aircraft. Because of the weight and CG constraints, the loading problem is sometimes called the *Weight and Balance problem.* The aim is that all the ULDs are loaded and the weight restrictions are respected so that the fuel consumption is minimised.

Because of the typical positions, this problem is not actually a Bin Packing Problem as it can seem, but it is an Assignment Problem as shown in Lurkin and Schyns (2015). However, typical constraints of C&P problems can be applied, for example, the allocation constraints for hazardous goods (Lurkin and Schyns (2015)).

Master Bay Plan Problem The Master Bay Plan Problem consists of determining how to stow a set of containers of different types into available locations of a container ship, with respect to some structural and operational constraints, related to both the containers and the ship, while minimising the time required for loading all containers on board (e.g., Sciomachen and Tanfani (2003), Ambrosino et al. (2004)). This problem has several similarities with the Air Cargo Loading Problem such as the compartmentalisation of the container ship and the structural constraints, typical to the Weight and Balance problems.

2.5 Problem definition

The aim of this thesis is to solve the MBSBPP with the following constraints:

- each box has to be assigned to exactly one bin,
- each box respects the limits of the bin, including the possible special shape of the ULDs,
- boxes do not overlap,
- the total weight of the boxes inside a ULD does not exceed its maximum capacity,
- orthogonal placement of the boxes including possible orientation constraints,
- boxes are vertically stable,
- fragility of the boxes is taken into account and
- the weight is uniformly distributed in the loaded ULDs.

Among the 12 papers addressing MBSBPP listed in Bortfeldt and Wäscher (2013), most proposed heuristic methods, which is natural since C&P problems are NP-hard combinatorial optimisation problems (Garey and Johnson (1979)). Chen et al. (1995), Westerlund et al. (2005) and Westerlund et al. (2007) propose a linear formulation for the MBSBPP with geometric constraints only. Jin et al. (2003) propose a similar linear formulation but use it only for small instances. For large scale instances, the authors develop a composite constructive algorithm: the first part based on tabu search metaheuristics assigns items to bins and the second part uses sub-volume based heuristics for packing a single bin. Almeida and Figueiredo (2010) propose a non-linear formulation for the MBSBPP where boxes cannot rotate, but focus on constructive heuristics based on placement points from Martello et al. (2000). As in Jin et al. (2003), the heuristic developed in Lin et al. (2006) is split into two steps: the box assignment to bins and then the packing itself. In the genetic algorithm proposed in Techanitisawad and Tangwiwatwong (2004), there is a first phase for the container selection and a second phase for the packing. The remaining papers addressing MBSBPP (Fraser and George (1994), Arenales and Morabito (1997), Brunetta and Grégoire (2005), de Queiroz et al. (2012), Ceschia and Schaerf (2013) propose several types of improvement heuristics. The approaches chosen for each paper are summarised in Table 2.1.

Table 2.1: Constraints considered in publications on MBSBPP (H=heuristic, E=exact)

Publications	Approach	weight limit	weight dist.	orientation	stacking	allocation	positioning	stability	complexity
Fraser and George (1994)	H	×	×	×					
Chen et al. (1995)	E		×	×					
Arenales and Morabito (1997)	H								
Jin et al. (2003)	E/H			×				×	
Techanitisawad and Tangwiwatwong (2004)	H		×	×	×			×	
Westerlund et al. (2005)	E								
Brunetta and Grégoire (2005)	H								
Lin et al. (2006)	H			×	×		×	×	
Westerlund et al. (2007)	E								
Almeida and Figueiredo (2010)	H			×		×			
de Queiroz et al. (2012)	H			×					×
Ceschia and Schaerf (2013)	H	×		×	×		×	×	
Thesis	E/H	×	×	×	×			×	

Only few of the 12 papers consider specific constraints. The formulations from Chen et al. (1995), Westerlund et al. (2005, 2007) take into account the geometric constraints and the rotations of the boxes but do not consider the container weight limit or any other specific constraints. The formulation presented in Chapter 3 is therefore the first to deal with all these specific constraints. The two papers presenting constructive heuristics as well as all the remaining papers consider only a subset of the specific constraints inherent to the topic of this thesis. Table 2.1 shows the 12 papers dealing with MBSBPP and the specific constraints taken into account in addition to the geometric constraints.

Our approach differs from the existing works by the set of constraints integrated but also by other factors like the type of problem, its representation, its method of resolution and its dimensionality. The aim of this thesis is to provide a rich and realistic resolution method for this extended MBSBPP. In the literature, the linear formulation from Chen et al. (1995) is used as a basis for the formulation presented in Chapter 3. The extension of placement points developed in Crainic et al. (2008) are used in the tailored two-phase constructive heuristic built in Chapter 5. Regarding the application field, Chan et al. (2006) also deal with air transportation and therefore with ULDs as well. They propose a two-phase heuristic for the Multiple Stock-Size Cutting Stock Problem (the assortment of boxes is weakly heterogeneous) and do not consider the stacking constraints.

As a conclusion, to the best of my knowledge, this work is the first to propose a linear mathematical formulation, to apply the Relax-and-Fix, Insert-and-Fix and Fractional Relax-and-Fix matheuristics and to develop a tailored two-phase constructive heuristic taking into account all the mentioned constraints.

Chapter 3

Mathematical formulation

The third chapter proposes a Mixed Integer Linear programming formulation for this specific three-dimensional Multiple Bin Size Bin Packing Problem described in the previous chapter. This chapter is an extended version of the methodology presented in *Paquay, C., M. Schyns, and S. Limbourg (2016). A mixed integer programming formulation for the three-dimensional Bin Packing Problem deriving from an air cargo application. International Transactions in Operational Research 23 (1-2), 187-213.* In Section 3.1, the set of parameters is presented. Variables used for the geometric constraints are described in Section 3.2. The objective function is provided in Section 3.3. Section 3.4 contains the geometric and specific constraints, as well as the variables required for the specific constraints. Results for small instances and the influence of the specific constraints are presented in Section 3.5 and finally, several areas for improvements are introduced in Section 3.6. This work is the first to propose a unique linear formulation for all these constraints together. Moreover, special container shapes are handled in this formulation.

The current position in the thesis outline is shown in bold in Figure 3.1.

3.1 Parameters

A set of n rectangular boxes of dimensions $l_i \times w_i \times h_i$ and weight m_i ($i \in \{1, ..., n\}$) has to be packed into m available ULDs of dimensions $L_j \times W_j \times H_j$, a maximal capacity, also called maximum gross weight, C_j and a volume V_j ($j \in \{1, ..., m\}$) while minimising the unused volume. All these numbers are assumed integer even if it means changing the scale, i.e., considering very small size units. In this thesis, dimensions are considered in millimetres.

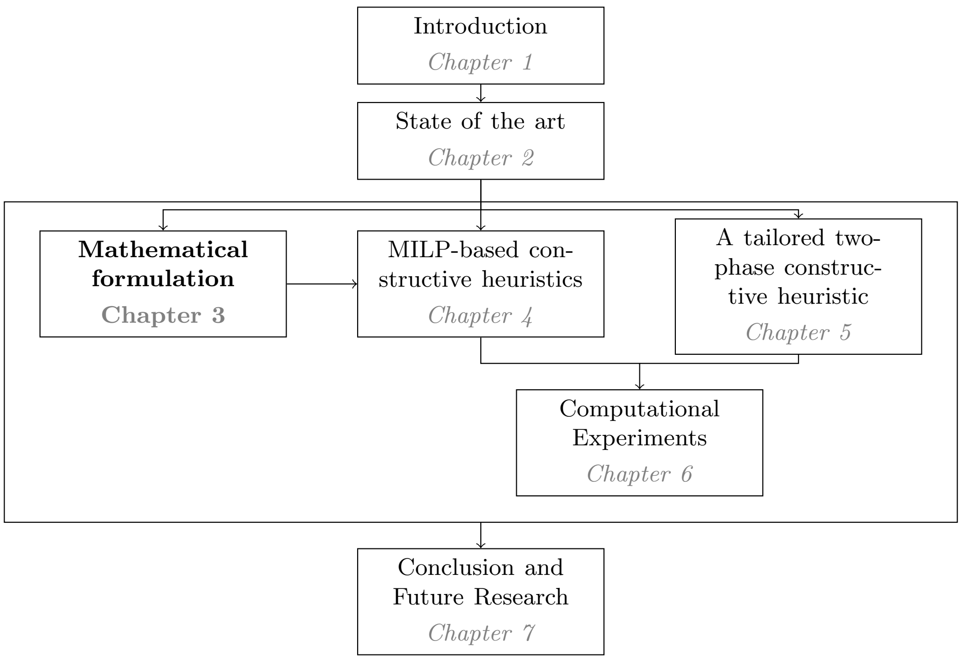

Introduction
Chapter 1
State of the art
Chapter 2
Mathematical formulation
Chapter 3
MILP-based constructive heuristics
Chapter 4
A tailored two-phase constructive heuristic
Chapter 5
Computational Experiments
Chapter 6
Conclusion and Future Research
Chapter 7

Figure 3.1: Current position in the thesis outline

These parameters are referred to as: $\forall i \in \{1, ..., n\},\ j \in \{1, ..., m\}$

n	Total number of boxes to be packed,	
$l_i \times w_i \times h_i$	Length $\times$ width $\times$ height of box i,	$\forall i$,
m_i	Weight of box i,	$\forall i$,
m	Total number of available ULDs,	
$L_j \times W_j \times H_j$	Length $\times$ width $\times$ height of ULD j,	$\forall j$,
C_j	Maximum gross weight of ULD j,	$\forall j$,
V_j	Volume of ULD j,	$\forall j$.

The volume of the ULDs can be deduced from the dimensions most of the time. However, when the ULD has a special shape as explained in the previous chapter, the value is not easy to compute.

Hereinafter, the subscripts relate to indices and the superscripts relate to fixed objects. Moreover, index j denotes the ULDs ($j \in \{1, ..., m\}$) while indices i and k denote the boxes ($i, k \in \{1, ..., n\}$).

Based on these parameters, we can define:

$$L = \max_{j \in \{1,...,m\}} L_j, \qquad W = \max_{j \in \{1,...,m\}} W_j, \qquad H = \max_{j \in \{1,...,m\}} H_j.$$

To ensure at least one feasible solution, some conditions are assumed to be satisfied: e.g., the weight of each box is supposed to be less than or equal to the maximum capacity of the bins: $m_i \leq \max_j C_j \quad \forall i \in \{1, ..., n\}$.

3.2 Variables

The situation can be represented in the three-dimensional geometric space. Without loss of generality, the coordinate system is placed with its origin on the front left bottom vertex of the ULDs and the axes such that the length L_j (resp. width W_j, height H_j) of the ULD j lies on the x-axis (resp. y-axis, z-axis) $\forall j \in \{1, ..., m\}$. A representation is given in Figure 3.2.

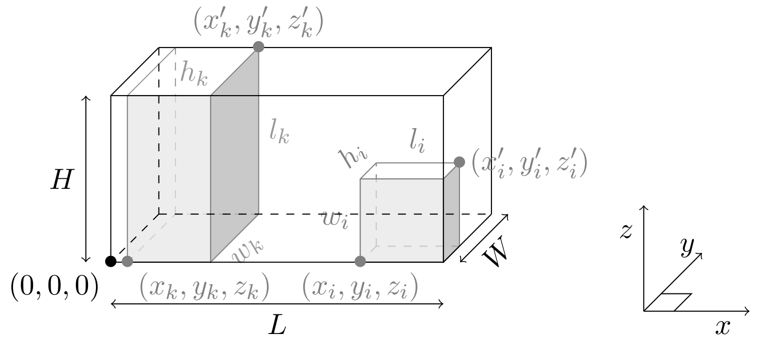


Figure 3.2: Representation of some parameters and variables: the ULD is shown in black lines, the boxes i and k are shown in grey, the coordinate system is on the right

ULDs with a special shape are considered as full parallelepipeds which have been cut. Thus, the origin is still placed at the front left bottom vertex as shown in Figure 3.3. More details about how to express these cuts are provided further.

The variables used for the geometric constraints and objective function are defined as follows:

$$p_{ij} = \begin{cases} 1 & \text{if box } i \text{ is in ULD } j, \\ 0 & \text{otherwise,} \end{cases} \qquad \forall i, j,$$

$$u_j = \begin{cases} 1 & \text{if ULD } j \text{ is used,} \\ 0 & \text{otherwise,} \end{cases} \qquad \forall j,$$

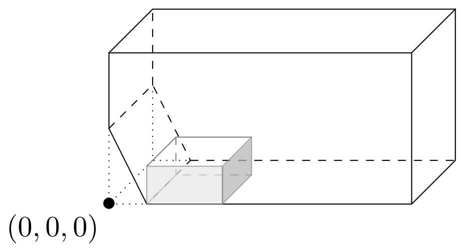


Figure 3.3: Coordinate system for ULDs with a special shape

(x_i, y_i, z_i)	Location of the front left bottom vertex of box i,	$\forall i$,
(x'_i, y'_i, z'_i)	Location of the rear right top vertex of box i,	$\forall i$,

$$r_{iab} = \begin{cases} 1 & \text{if the side } b \text{ of box } i \text{ is along the } a\text{-axis}, \\ 0 & \text{otherwise}, \end{cases} \qquad \forall i,$$

$$x^p_{ik} = \begin{cases} 1 & \text{if box } i \text{ is on the right of box } k (x'_k \leq x_i), \\ 0 & \text{otherwise}(x_i < x'_k), \end{cases} \qquad \forall i,$$

$$y^p_{ik} = \begin{cases} 1 & \text{if box } i \text{ is behind box } k (y'_k \leq y_i), \\ 0 & \text{otherwise}(y_i < y'_k), \end{cases} \qquad \forall i,$$

$$z^p_{ik} = \begin{cases} 1 & \text{if box } i \text{ is above } k (z'_k \leq z_i), \\ 0 & \text{otherwise}(z_i < z'_k), \end{cases} \qquad \forall i.$$

Note that the variables (x_i, y_i, z_i) and (x'_i, y'_i, z'_i) are also assumed integer. This assumption is reasonable because, in practice, the position of a box is described with a finite precision. These variables describe the position of box i inside a ULD. They are represented in Figure 3.2.

Since the boxes can rotate orthogonally, variables r_{iab} are introduced to describe the orientation of box i inside a ULD. Index b indicates the side of the box, i.e., $b \in \{l := 1, w := 2, h := 3\}$, whereas a indicates the axis, i.e., $a \in \{x := 1, y := 2, z := 3\}$. Variables r_{iab} specify which side of box i is along which axis. For example, these variables are equal to

$$\begin{array}{llllll} r_{i11} = 1 & r_{i12} = 0 & r_{i13} = 0 & r_{k11} = 0 & r_{k12} = 0 & r_{k13} = 1 \\ r_{i21} = 0 & r_{i22} = 0 & r_{i23} = 1 & r_{k21} = 0 & r_{k22} = 1 & r_{k23} = 0 \\ r_{i31} = 0 & r_{i32} = 1 & r_{i33} = 0 & r_{k31} = 1 & r_{k32} = 0 & r_{k33} = 0 \end{array}$$

in Figure 3.2.

To ensure that there is no overlap, we need to know the relative position of two boxes. To this purpose, the variable x^p_{ik} (resp. y^p_{ik}, z^p_{ik}) is equal to 1 if box i is on the right (resp. behind, above) of box k. These variables describe all the situations. Indeed, for instance if box i is on the left of box k, it means that box k is on the right of box i and then $x^p_{ki} = 1$.

In Figure 3.2, one has

x^p	i	k
i	0	1
k	0	0

y^p	i	k
i	0	0
k	0	0

Even if the definition of the z^p_{ik} is the same as x^p_{ik} and y^p_{ik}, we will see in Section 3.4.1 they are not fully determined since it is not necessary. Indeed, only half of the definition will be guaranteed by the constraints: if $z^p_{ik} = 1$, then we are sure that $z'_k \leq z_i$. On the contrary, if $z^p_{ik} = 0$, then we have no information.

3.3 Objective function

The objective function consists in minimising the unused volume of the selected ULDs

$$\sum_{j=1}^{m} u_j V_j - \sum_{i=1}^{n} l_i \ w_i \ h_i. \tag{3.1}$$

Since l_i, w_i, h_i are parameters that are initially given, the term $\sum_{i=1}^{n} l_i \ w_i \ h_i$ is a constant. Therefore, the volume of the used ULDs is minimised:

$$\sum_{j=1}^{m} u_j V_j. \tag{3.2}$$

As mentioned already, V_j represents the volume of the ULD j. However, other objective functions could be easily considered. For instance, if V_j represents the cost of ULD j, then the objective function would become a cost minimisation.

3.4 Constraints

As explained in Section 2.2, constraints can be split into geometric and specific ones.

3.4.1 Geometric constraints

The geometric constraints of the model are as follows: $\forall a, b \in \{1, 2, 3\}$

$$\sum_{i=1}^{n} m_i \; p_{ij} \leq C_j \; u_j, \qquad \forall j, \tag{3.3}$$

$$\sum_{j=1}^{m} p_{ij} = 1, \qquad \forall i, \tag{3.4}$$

$$x'_i \leq \sum_{j=1}^{m} L_j \; p_{ij}, \qquad \forall i, \tag{3.5}$$

$$y'_i \leq \sum_{j=1}^{m} W_j \; p_{ij}, \qquad \forall i, \tag{3.6}$$

$$z'_i \leq \sum_{j=1}^{m} H_j \; p_{ij}, \qquad \forall i, \tag{3.7}$$

$$x'_i - x_i = r_{i11} \; l_i + r_{i12} \; w_i + r_{i13} \; h_i, \qquad \forall i, \tag{3.8}$$

$$y'_i - y_i = r_{i21} \; l_i + r_{i22} \; w_i + r_{i23} \; h_i, \qquad \forall i, \tag{3.9}$$

$$z'_i - z_i = r_{i31} \; l_i + r_{i32} \; w_i + r_{i33} \; h_i, \qquad \forall i, \tag{3.10}$$

$$\sum_{a=1}^{3} r_{iab} = 1, \qquad \forall i, b, \tag{3.11}$$

$$\sum_{b=1}^{3} r_{iab} = 1, \qquad \forall i, a. \tag{3.12}$$

The maximum capacity of each ULD j cannot be exceeded, which is ensured by constraints (3.3). This set of constraints, in conjunction with the minimisation of the objective function, fully determines the values of variables u_j. Constraints (3.4) verify that each box is allocated to exactly one ULD. Constraints (3.5)-(3.7) ensure that the boxes do not exceed their ULD size. Constraints (3.8)-(3.12) describe that the boxes can rotate orthogonally in the ULD. Note that (3.8)-(3.10) imply $x_i < x'_i, \; y_i < y'_i, \; z_i < z'_i$.

The following constraints ensure that there is no overlap, i.e., two boxes cannot occupy a same portion of the space:

$$x^p_{ik} + x^p_{ki} + y^p_{ik} + y^p_{ki} + z^p_{ik} + z^p_{ki} \geq (p_{ij} + p_{kj}) - 1, \qquad \forall i, k, j, \tag{3.13}$$

$$x'_k \leq x_i + (1 - x^p_{ik}) \; L, \qquad \forall i, k, \tag{3.14}$$

$$y'_k \leq y_i + (1 - y^p_{ik}) \; W, \qquad \forall i, k, \tag{3.15}$$

$$z'_k \leq z_i + (1 - z^p_{ik})\ H, \qquad \forall i,k. \tag{3.16}$$

When variables $x^p_{ik}, x^p_{ki}, y^p_{ik}, y^p_{ki}, z^p_{ik}$ or z^p_{ki} equal 1, the two boxes i and k do not overlap along one of the axes. To prevent having two boxes occupying a same portion of space, it is sufficient to allow no overlap along at least one of the axes, i.e., at least one of these variables must equal 1. It leads to constraints (3.13). An overlap can happen only if two boxes are in the same bin, which is expressed by the right-hand side of constraints (3.13).

To describe additional variables for the vertical stability in Section 3.4.2, a unequivocal definition of variables x^p_{ik} and y^p_{ik} will be necessary. For this reason, constraints (3.17) and (3.18) are added to the model. In this way, constraints (3.14)-(3.18) ensure that $x^p_{ik} = 1$ if and only if $x_i \geq x'_k$ and $y^p_{ik} = 1$ if and only if $y_i \geq y'_k$:

$$x_i + 1 \leq x'_k + x^p_{ik}\ L, \qquad \forall i,k, \tag{3.17}$$

$$y_i + 1 \leq y'_k + y^p_{ik}\ W, \qquad \forall i,k. \tag{3.18}$$

Constraints (3.17) and (3.18) would need to be adapted if coordinate variables x_i, x'_i, y_i, y'_i were continuous variables.

Note that the parameters L, W, H are used in these constraints because we do not know in which ULD the boxes i and k have been packed.

3.4.2 Specific constraints

As mentioned in the previous chapter, applying the MBSBPP to the real world situations implies some specific constraints.

Orientation constraints Some boxes may not rotate in all directions because of their contents; for instance, some products may not turn upside-down. To this purpose, some new parameters are introduced for each box i:

$$l^+_i = \begin{cases} 1 & \text{if the length of box } i \text{ could be in a vertical position,} \\ 0 & \text{otherwise,} \end{cases}$$

$$w^+_i = \begin{cases} 1 & \text{if the width of box } i \text{ could be in a vertical position,} \\ 0 & \text{otherwise,} \end{cases}$$

$$h^+_i = \begin{cases} 1 & \text{if the height of box } i \text{ could be in a vertical position,} \\ 0 & \text{otherwise.} \end{cases}$$

If all these parameters are set to one, each box is free to rotate in any direction. More precisely, constraints (3.8)-(3.12) allow in this case six orientations for each box. If one parameter is set to zero, two of these six orientations are forbidden. For example, only the four orientations depicted in Figure 3.4 should remain feasible with l_i^+ set to 0.

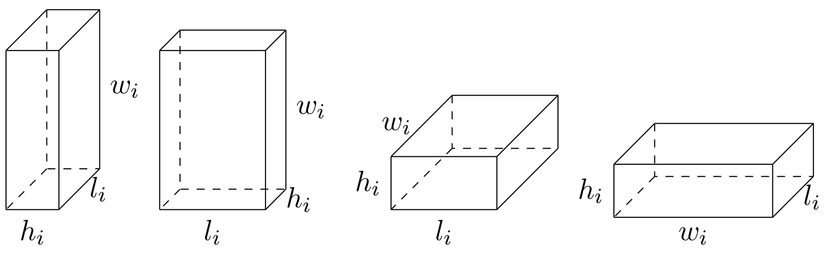


Figure 3.4: Possible orientations for box i if its length could not be along the z-axis ($l_i^+ = 0$)

Likewise, if two parameters equal 0, only two orientations of the box remain possible. Unless at least one of these parameters equals one, there is no possible orientation. Keeping in mind that variables r_{i3b} describe which side of box i is along the z-axis, i.e., determine the value of z_i', constraints (3.19)-(3.21) come naturally

$$r_{i31} \leq l_i^+, \qquad \forall i, \tag{3.19}$$
$$r_{i32} \leq w_i^+, \qquad \forall i, \tag{3.20}$$
$$r_{i33} \leq h_i^+, \qquad \forall i. \tag{3.21}$$

Special shapes of the ULDs As explained in the previous chapter, some ULDs may have a special shape to fit into the fuselage of the aircraft. These special ULDs can be described as full parallelepipeds whose one or several corners have been cut. There exist four possible cuts for a ULD. Each cut can be described by a linear equation and the front left bottom vertex of the ULD before truncation lies at the origin of the coordinate system as shown in Figure 3.5. To describe these cuts, eight new parameters (two for each cut σ, $\sigma \in \{1, ..., 4\}$) are added for each bin j: $a_{\sigma j},\ b_{\sigma j}\ \in \mathbb{R}^+$ as shown in Figure 3.5.

To lie inside these types of ULDs, each box has to satisfy constraints

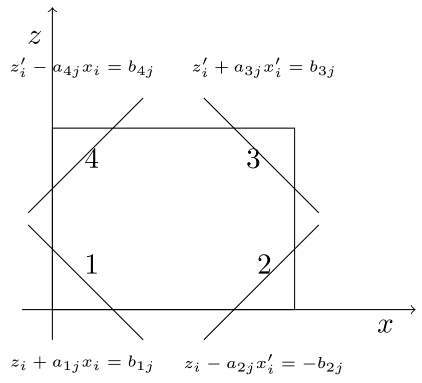

z
$z'_i - a_{4j}x_i = b_{4j}$
$z'_i + a_{3j}x'_i = b_{3j}$
4
3
1
2
x
$z_i + a_{1j}x_i = b_{1j}$
$z_i - a_{2j}x'_i = -b_{2j}$

Figure 3.5: Possible cuts (projection on the XZ plane)

associated to the cuts of the bin it is put in:

$$\text{cut 1: } z_i + a_{1j}x_i \geq b_{1j} - M(1-p_{ij}), \qquad \forall i,j, \tag{3.22}$$
$$\text{cut 2: } z_i - a_{2j}x'_i \geq -b_{2j} - M(1-p_{ij}), \qquad \forall i,j, \tag{3.23}$$
$$\text{cut 3: } z'_i + a_{3j}x'_i \leq b_{3j} + M(1-p_{ij}), \qquad \forall i,j, \tag{3.24}$$
$$\text{cut 4: } z'_i - a_{4j}x_i \leq b_{4j} + M(1-p_{ij}), \qquad \forall i,j. \tag{3.25}$$

The value of M has to be computed such that, when $p_{ij} = 0$, these inequations are unconstrainted. For this reason,

- for the cut 1: if $p_{ij} = 0$, then $z + a_{1j}x - b_{1j} \geq -M$ for all values of z and x. In particular, $\min_{x,z}(z + a_{1j}x - b_{1j}) = -b_{1j} \geq -M$. Thus, $M \geq b_{1j}$.

- for the cut 2: if $p_{ij} = 0$, then $z - a_{2j}x' + b_{2j} \geq -M$ for all values of z and x'. In particular, $\min_{x',z}(z - a_{2j}x' + b_{2j}) = -a_{2j}L + b_{2j} \geq -M$. Thus, $M \geq a_{2j}L - b_{2j}$.

- for the cut 3: if $p_{ij} = 0$, then $z' + a_{3j}x' - b_{3j} \leq M$ for all values of z' and x'. In particular, $\max_{x',z'}(z' + a_{3j}x' - b_{3j}) = H + a_{3j}L - b_{3j} \leq M$. Thus, $M \geq H + a_{3j}L - b_{3j}$.

- for the cut 4: if $p_{ij} = 0$, then $z' - a_{4j}x - b_{4j} \leq M$ for all values of z' and x. In particular, $\max_{x,z'}(z' - a_{4j}x - b_{4j}) = H - b_{4j} \leq M$. Thus, $M \geq H - b_{4j}$.

Therefore, M is equal to $\max\{\max_j b_{1j}, \max_j a_{2j}L - b_{2j}, \max_j H + a_{3j}L - b_{3j}, \max_j H - b_{4j}\}$. Another possibility would be to consider different values of M for the different constraints.

This part of the formulation could be easily extended to any bins which are convex polytopes.

Note that cuts 1 and 2 will influence the vertical stability since a box can be supported by these cuts as shown in Figure 3.6. This will be explained in the following.

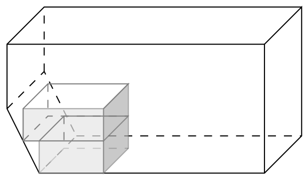


Figure 3.6: A box supported by another box and an inclined wall

Vertical stability As explained in Section 2.2.2, the bottom face of each box has to be supported by the top face of other boxes, by a cut or by the ULD floor. Thus, the boxes are not displaced with respect to z-axis. Especially, the vertical stability excludes floating boxes.

In this model, we do more than just verify if a box is supported. Physically speaking, an object is stable if its centre of gravity (CG) lies within its support base. The weight inside the boxes is assumed to be uniform, therefore the CG corresponds to the geometric centre of the box. Boxes are also assumed rigid and unbreakable. By definition, the support base is the convex hull of all the contact points. The stability constraints of this model rely on this idea: if a box is not on the ground, then the four vertices of its base must be supported. In this way, the application point of the applied forces will always lie in the support base because the support base is the entire box base. This is the condition for vertical stability (Ramos et al. (2016)). This assumption is technically sufficient but in practice, an improved stability could be achieved by ensuring that the base is better supported, for instance, by forcing a given percentage of its area to be supported. Note that supporting the four vertices is a restrictive condition and it may be possible to obtain a stable pattern with only three supported vertices. To achieve the stability, with the help of some variables, we determine whether a box is on the ground and whether a vertex of the bottom face is correctly supported. A vertex of the bottom face of a box i is correctly supported if there is another box k that has the suitable height to support the vertex, i.e., the coordinate along the z-axis of the top face of box k equals the coordinate along the z-axis of the bottom face of box i ($z'_k = z_i$), and with a particular overlap of the projections of these two boxes on the XY plane. To define this overlap, we consider that the vertices of box i are assigned to a number in the following way: $(x_i, y_i) := 1$, $(x'_i, y_i) := 2$, $(x'_i, y'_i) := 3$ and $(x_i, y'_i) := 4$. As shown in Figure 3.7, the vertex 1 (resp. 2, 3, 4) is supported if there exists a box k in the same bin, with the suitable height, such that

$$x_k \le x_i < x'_k \quad \text{and} \quad y_k \le y_i < y'_k \tag{3.26}$$
$$\text{(resp. } x_k < x'_i \le x'_k \quad \text{and} \quad y_k \le y_i < y'_k, \tag{3.27}$$
$$x_k < x'_i \le x'_k \quad \text{and} \quad y_k < y'_i \le y'_k, \tag{3.28}$$
$$x_k \le x_i < x'_k \quad \text{and} \quad y_k < y'_i \le y'_k). \tag{3.29}$$

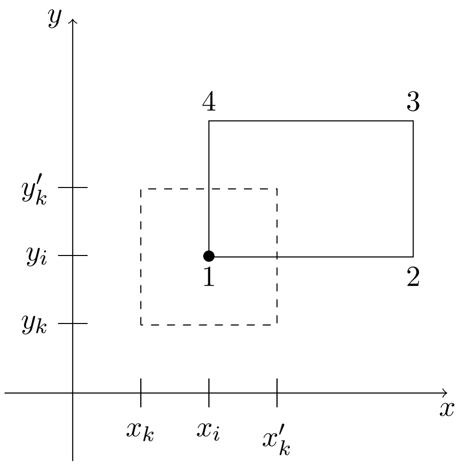


Figure 3.7: Vertex (x_i, y_i) from box i (in solid lines) is supported by the box k (in dashed lines) (projection on the XY plane)

For this purpose, new variables are introduced:

$$g_i = \begin{cases} 1 & \text{if box } i \text{ is on the ground} (z_i = 0), \\ 0 & \text{otherwise,} \end{cases} \qquad \forall i,$$

$$h_{ik} = \begin{cases} 0 & \text{if box } k \text{ has the suitable height to support box } i \\ & (z_i = z'_k), \\ 1 & \text{otherwise,} \end{cases} \qquad \forall i, k,$$

$$o_{ik} = \begin{cases} 0 & \text{if the projections on the } XY \text{ plane of the boxes} \\ & \quad i \text{ and } k \text{ have a non-empty intersection,} \\ 1 & \text{otherwise,} \end{cases} \qquad \forall i, k,$$

$$s_{ik} = \begin{cases} 1 & \text{if box } k \text{ supports box } i \text{ and are in the same bin,} \\ 0 & \text{otherwise,} \end{cases} \qquad \forall i, k,$$

$$\eta^1_{ik} = \begin{cases} 0 & \text{if } x_k \leq x_i, \\ 1 & \text{otherwise,} \end{cases} \qquad \forall i, k,$$

$$\eta^2_{ik} = \begin{cases} 0 & \text{if } y_k \leq y_i, \\ 1 & \text{otherwise,} \end{cases} \qquad \forall i, k,$$

$$\eta^3_{ik} = \begin{cases} 0 & \text{if } x'_i \leq x'_k, \\ 1 & \text{otherwise,} \end{cases} \qquad \forall i, k,$$

$$\eta^4_{ik} = \begin{cases} 0 & \text{if } y'_i \leq y'_k, \\ 1 & \text{otherwise,} \end{cases} \qquad \forall i, k,$$

$$\beta^l_{ik} = \begin{cases} 1 & \text{if vertex } l \text{ of box } i \text{ base is supported by} \\ & \quad \text{box } k, \\ 0 & \text{otherwise,} \end{cases} \qquad \forall i, k, l,$$

$$\gamma^1_i = \begin{cases} 1 & \text{if box } i \text{ lays on cut 1 of the bin in which} \\ & \quad \text{it lies,} \\ 0 & \text{otherwise,} \end{cases} \qquad \forall i,$$

$$\gamma^2_i = \begin{cases} 1 & \text{if box } i \text{ lays on cut 2 of the bin in which} \\ & \quad \text{it lies,} \\ 0 & \text{otherwise,} \end{cases} \qquad \forall i,$$

$i, k \in \{1, ..., n\}$, $l \in \{1, ..., 4\}$.

As developed previously, stability constraints could be written as follows

$$\sum_{l=1}^{4} \sum_{k=1}^{n} \beta^l_{ik} + 2\gamma^1_i + 2\gamma^2_i \geq 4(1 - g_i) \quad \forall i \in \{1, ..., n\}. \tag{3.30}$$

If box i is not on the ground, constraints (3.30) ensure that the four base vertices are supported, either by another box k, or by an inclined wall. Indeed, if $\sum_{k=1}^{n} \beta^l_{ik} = 1$, then it means that there is a box k supporting vertex l. Besides, if a box i relies on a cut, then two vertices on the same edge are supported. If we want to require only three supported vertices, then the factor 4 in the right-hand side of constraints (3.30) has to be equal to 3.

To define these new variables, constraints (3.31)-(3.52) are added:

$$
\begin{aligned}
z_i &\leq (1-g_i)\ H, && \forall i, && (3.31)\\
z'_k - z_i &\leq v_{ik}, && \forall i,k, && (3.32)\\
z_i - z'_k &\leq v_{ik}, && \forall i,k, && (3.33)\\
v_{ik} &\leq z'_k - z_i + 2H(1-m_{ik}), && \forall i,k, && (3.34)\\
v_{ik} &\leq z_i - z'_k + 2Hm_{ik}, && \forall i,k, && (3.35)\\
h_{ik} &\leq v_{ik}, && \forall i,k, && (3.36)\\
v_{ik} &\leq h_{ik}\ H, && \forall i,k, && (3.37)\\
o_{ik} \leq x^p_{ik} + x^p_{ki} + y^p_{ik} + y^p_{ki} &\leq 2o_{ik}, && \forall i,k, && (3.38)\\
(1-s_{ik}) \leq h_{ik} + o_{ik} &\leq 2(1-s_{ik}), && \forall i,k, && (3.39)\\
p_{ij} - p_{kj} &\leq 1 - s_{ik}, && \forall i,j,k, && (3.40)\\
p_{kj} - p_{ij} &\leq 1 - s_{ik}, && \forall i,j,k, && (3.41)\\
\beta^l_{ik} &\leq s_{ik}, && \forall i,k,l, && (3.42)\\
x_k &\leq x_i + \eta^1_{ik}\ L, && \forall i,k, && (3.43)\\
y_k &\leq y_i + \eta^2_{ik}\ W, && \forall i,k, && (3.44)\\
x'_i &\leq x'_k + \eta^3_{ik}\ L, && \forall i,k, && (3.45)\\
y'_i &\leq y'_k + \eta^4_{ik}\ W, && \forall i,k, && (3.46)\\
\eta^1_{ik} + \eta^2_{ik} &\leq 2(1-\beta^1_{ik}), && \forall i,k, && (3.47)\\
\eta^2_{ik} + \eta^3_{ik} &\leq 2(1-\beta^2_{ik}), && \forall i,k, && (3.48)\\
\eta^3_{ik} + \eta^4_{ik} &\leq 2(1-\beta^3_{ik}), && \forall i,k, && (3.49)\\
\eta^1_{ik} + \eta^4_{ik} &\leq 2(1-\beta^4_{ik}), && \forall i,k, && (3.50)\\
(1-\gamma^1_i)M &\geq z_i + a_{1j}x_i - b_{1j} - (1-p_{ij})M, && \forall i,j, && (3.51)\\
(1-\gamma^2_i)M &\geq z_i - a_{2j}x'_i + b_{2j} - (1-p_{ij})M, && \forall i,j, && (3.52)
\end{aligned}
$$

$i,k \in \{1,...,n\}, j \in \{1,...,m\}, l \in \{1,...,4\}$.

By constraints (3.31), if g_i equals 1, then box i is on the ground. Constraints (3.32)-(3.37) define variables h_{ik} by using v_{ik} which represent the absolute value $|z'_k - z_i|$ and m_{ik} which is equal to 1 if $z'_k \geq z_i$ and 0 otherwise.

Constraints (3.38) are based on the fact that boxes i and k share a part of their orthogonal projection on the XY plane if $x^p_{ik}+x^p_{ki}+y^p_{ik}+y^p_{ki} = 0$. A full determination of variables o_{ik} is required in the hereinafter. If the bottom face of box i is supported by the top face of a box k, it implies $h_{ik}+o_{ik} = 0$. This is represented by constraints (3.39). A full determination of variables s_{ik} is also required. A box i can be supported by a box k only if these two boxes are in the same bin, which is guaranteed by constraints (3.40)-(3.41). Constraints (3.42) certify that a box k supports one vertex of the base of box i only if this one is supported by box k, i.e., if $s_{ik} = 1$. Constraints (3.43)-(3.46) are similar to constraints (3.17) and (3.18). Constraints (3.47)-(3.50) ensure the second part of equations (3.26)-(3.29) is satisfied to define a supported vertex. Constraints (3.51) and (3.52) express that a box i is supported by a cut if this box satisfies the cut at equality. In practice, x_i, z_i, x'_i and z'_i are integer numbers and most of the time, the equations of the cuts 1 and 2 do not hold points with integer coordinates. For this reason, constraints (3.51) and (3.52) are softened by introducing a precision ε, which can represent the thickness of the ULD material for example, such that $\gamma^1 = 1$ if $z_i + a_1 x_i \in [b_1 - \varepsilon_1, b_1 + \varepsilon_1]$ and $\gamma^2 = 1$ if $z_i - a_2 x'_i \in [b_2 - \varepsilon_2, b_2 + \varepsilon_2]$. To make it reasonable, a deviation of 0.45 units (here millimetres) is allowed along the x and z-axes. Thus, ε_1 (resp. ε_2) is set to $0.45(1 + a_1)$ (resp. $0.45(1 + a_2)$). Several constraints have to be adapted:

- (3.22) becomes $z_i + a_{1j} x_i \geq b_{1j} - \varepsilon_1 - M(1 - p_{ij}), \forall i, j$,
- (3.23) becomes $z_i - a_{2j} x'_i \geq -b_{2j} - \varepsilon_2 - M(1 - p_{ij}), \forall i, j$,
- (3.51) becomes $(1 - \gamma_i^1)M + \varepsilon_1 \geq z_i + a_{1j} x_i - b_{1j} - (1 - p_{ij})M, \forall i, j$,
- (3.52) becomes $(1 - \gamma_i^2)M + \varepsilon_2 \geq z_i - a_{2j} x'_i + b_{2j} - (1 - p_{ij})M, \forall i, j$.

The two sets of constraints (3.51) and (3.52) require an update of the value of M. Indeed, if p_{ij} or γ_i^1 are equal to 0, then the inequations must be unconstrained, which is ensured if $M \geq \max(z + a_{1j}x - b_{1j}) = H + a_{1j}L - b_{1j}$ and $M \geq \max(z - a_{2j}x' - b_{2j}) = H + b_{2j}$. Thus, M is equal to

$$M = \max\{\max_j b_{1j}, \max_j a_{2j}L - b_{2j}, \max_j H + a_{3j}L - b_{3j}, \max_j H - b_{4j}, \\ \max_j H + a_{1j}L - b_{1j}, \max_j H + b_{2j}\}. \quad (3.53)$$

If the ULD j is a parallelepiped that has no cut, then the value of γ_i^1 and γ_i^2 should equal 0 for all the boxes i located in ULD j. For this reason, the following constraints are added:

$$p_{ij} + \gamma_i^1 \leq a_{1j} + b_{1j} + 1 \qquad \forall i, j, \qquad (3.54)$$
$$p_{ij} + \gamma_i^2 \leq a_{2j} + b_{2j} + 1 \qquad \forall i, j. \qquad (3.55)$$

Fragility As explained above, some boxes can be fragile, that is they cannot support boxes on their top face. This can be caused by the nature of the contents of these boxes. To express that a box i is fragile, a new set of parameters is introduced:

$$f_i = \begin{cases} 1 & \text{if box } i \text{ is fragile} \\ 0 & \text{otherwise} \end{cases} \qquad \forall i \in \{1, ..., n\}.$$

Constraints (3.56) ensure that if a box is fragile, then it does not support any other box on its top face:

$$\sum_{i=1}^{n} s_{ik} \leq n(1 - f_k) \qquad \forall k \in \{1, ..., n\}. \tag{3.56}$$

Indeed, the term $\sum_{i=1}^{n} s_{ik}$ represents the number of boxes supported by box k and thus should be equal to 0 if box k is fragile.

Weight distribution As said in the definition of the problem, we would like to ensure some uniformity about weight distribution. More exactly, according to the control and loading manual of Boeing (Boeing (2008)), the CG of the ULDs must lie within a specific area. Horizontally, this area is defined around the geometric centre of the ULD base. Vertically, the CG must lie below a given level.

Physically speaking, it is known that:

1. *a system of particles moves as if the resultant external force were applied to a single particle of mass M (mass of the system) located at its centre of gravity.*

2. *the x coordinate of the centre of gravity of n particles is defined to be*

$$x_{CG} = \frac{\sum_{i=1}^{n} m_i x_i}{\sum_{i=1}^{n} m_i} \tag{3.57}$$

where x_i is the x-coordinate of the i^{th} particle and m_i its mass.

This means that each box i can be considered as a point located at the coordinate $(\frac{x_i+x'_i}{2}, \frac{y_i+y'_i}{2}, \frac{z_i+z'_i}{2})$ of mass m_i, because the weight is uniformly distributed in the boxes. The second quotation states that the CG of the contents of ULD j is located at the coordinate

$$\frac{1}{\sum\limits_{i|p_{ij}=1} m_i} \left(\sum_{i|p_{ij}=1} m_i(\frac{x_i + x'_i}{2}), \sum_{i|p_{ij}=1} m_i(\frac{y_i + y'_i}{2}), \sum_{i|p_{ij}=1} m_i(\frac{z_i + z'_i}{2}) \right)$$
$$:= (x_{CG_j}, y_{CG_j}, z_{CG_j}), \tag{3.58}$$

the sums being applied only to the boxes i inside ULD j.

Here is the approach for the x-axis. Since $\frac{L_j}{2}$ is the x-coordinate of the geometric centre of ULD j, we want x_{CG_j} to lie in the neighbourhood of $\frac{L_j}{2}$. To define the allowable range of the x_{CG_j}, a new parameter α_j^L depending on the type of ULD and on the type of aircraft is introduced. Then, x_{CG_j} must fall into the interval $[\frac{L_j}{2} - \alpha_j^L, \frac{L_j}{2} + \alpha_j^L]$. To select only the boxes which are in bin j, some new real variables are introduced

$$X_{ij} \equiv p_{ij} \left(\frac{x_i + x_i'}{2} \right).$$

According to this definition, they have to satisfy the following constraints

$$X_{ij} \leq L\ p_{ij}, \qquad \forall i, j, \tag{3.59}$$

$$X_{ij} \leq \frac{x_i + x_i'}{2}, \qquad \forall i, j, \tag{3.60}$$

$$X_{ij} \geq \frac{x_i + x_i'}{2} - L\ (1 - p_{ij}), \qquad \forall i, j, \tag{3.61}$$

which are linear. To ensure that x_{CG_j} is in the neighbourhood of $L_j/2$, constraints (3.62) are added

$$\left(\frac{L_j}{2} - \alpha_j^L \right) \leq \frac{\sum_{i=1}^n X_{ij}\ m_i}{(\sum_{i=1}^n m_i p_{ij})} \leq \left(\frac{L_j}{2} + \alpha_j^L \right) \forall j. \tag{3.62}$$

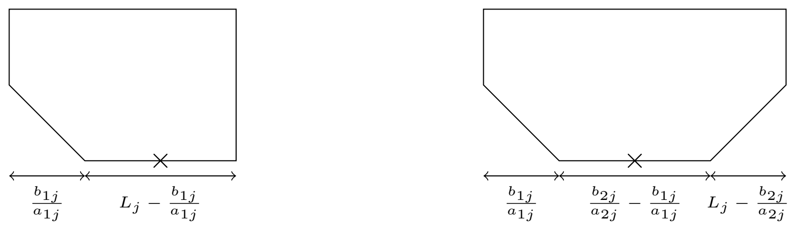
For the ULDs with a type 1 or type 2 cut, the constraint (3.62) has to be adapted. Indeed, x_{CG_j} must lie close to the geometrical centre of the base of the ULD. If the ULD has a type 1 cut, the centre is not $L_j/2$, but

$$\frac{b_{1j}}{a_{1j}} + \frac{1}{2} \left(L_j - \frac{b_{1j}}{a_{1j}} \right) = \frac{L_j}{2} + \frac{b_{1j}}{2a_{1j}}$$

as shown in Figure 3.8. Similarly, if the ULD has type 1 and type 2 cuts, the centre becomes

$$\frac{b_{1j}}{a_{1j}} + \frac{1}{2} \left(\frac{b_{2j}}{a_{2j}} - \frac{b_{1j}}{a_{1j}} \right) = \frac{1}{2} \left(\frac{b_{2j}}{a_{2j}} + \frac{b_{1j}}{a_{1j}} \right).$$

Note that the value of α_j^L remains the same since it is linked to the type of the ULD.



$\frac{b_{1j}}{a_{1j}}$
$L_j - \frac{b_{1j}}{a_{1j}}$
$\frac{b_{1j}}{a_{1j}}$
$\frac{b_{2j}}{a_{2j}} - \frac{b_{1j}}{a_{1j}}$
$L_j - \frac{b_{2j}}{a_{2j}}$

Figure 3.8: If the ULD has a type 1 or type 2 cut, then x_{CG_j} must lie close to the geometrical centre of the ULD base

The weight distribution along the y-axis can be managed in the same way. A parameter α_j^W is introduced and y_{CG_j} must lie within the interval $[\frac{W_j}{2} - \alpha_j^W, \frac{W_j}{2} + \alpha_j^W]$. Therefore, the real variables Y_{ij} defined as

$$Y_{ij} \equiv p_{ij} \left(\frac{y_i + y_i'}{2} \right)$$

are introduced. The corresponding constraints, similar to (3.59)-(3.62), are

$$\begin{aligned}
& Y_{ij} \leq W \; p_{ij}, && \forall i, j, && (3.63)\\
& Y_{ij} \leq \frac{y_i + y_i'}{2}, && \forall i, j, && (3.64)\\
& Y_{ij} \geq \frac{y_i + y_i'}{2} - W \; (1 - p_{ij}), && \forall i, j, && (3.65)\\
& \left(\frac{W_j}{2} - \alpha_j^W \right) \leq \frac{\sum_{i=1}^{n} Y_{ij} \; m_i}{(\sum_{i=1}^{n} m_i p_{ij})} \leq \left(\frac{W_j}{2} + \alpha_j^W \right), && \forall j. && (3.66)
\end{aligned}$$

About the weight distribution along the z-axis, the reasoning is the same except that the CG has to lie below a given ceiling. Then, a parameter α_j^H, which is the maximal value of z_{CG_j}, is introduced. This means that z_{CG_j} must lie within the interval $[0, \alpha_j^H]$. Therefore, the real variables Z_{ij} defined as

$$Z_{ij} \equiv p_{ij} \left(\frac{z_i + z_i'}{2} \right)$$

are introduced. The corresponding constraints, similar to (3.59)-(3.62), are

$$\begin{aligned}
& Z_{ij} \leq H \; p_{ij}, && \forall i, j, && (3.67)\\
& Z_{ij} \leq \frac{z_i + z_i'}{2}, && \forall i, j, && (3.68)\\
& Z_{ij} \geq \frac{z_i + z_i'}{2} - H \; (1 - p_{ij}), && \forall i, j, && (3.69)
\end{aligned}$$

$$0 \leq \frac{\sum_{i=1}^{n} Z_{ij}\ m_i}{(\sum_{i=1}^{n} m_i p_{ij})} \leq \alpha_j^H\ , \qquad \forall j. \tag{3.70}$$

The weight distribution constraint along the z-axis can become more restrictive for the ULDs with a cut of type 1 or 2. Note that this constraint may prevent the ULDs from having a high filling rate. More precisely, as can be seen in Chapter 6, α_j^H has a value around 40%-50% of the height of the ULDs considered in this thesis.

Variables X_{ij}, Y_{ij} and Z_{ij} are the only non-integer variables in the formulation.

3.5 Preliminary results

In order to check the validity of the presented formulation and to test the impact of the specific constraints on the solution values and the computational times, the formulation has been tested on sets of small preliminary instances. All tests were performed on a workstation with 32.0 GB RAM and an Intel Xeon processor E5-2620 v4 running 64-bit Windows 10 Pro. The formulations were implemented in Java using IBM ILOG CPLEX 12.6 library as Branch-and-Bound (B&B) solver with default parametrisation. The computation time has been limited to one hour for every run. The samples have 30 instances containing 6, 7, 8, 9 and 10 boxes each. A set of six distinct ULDs is proposed to pack the sets of boxes. More details about the data are provided in Section 6.1.

First, the complete formulation with all the specific constraints has been tested for all the sample sizes in Section 3.5.1. Then, the influence of the fragility constraint, the orientation constraint, the special shape of the ULDs and the uniform weight distribution constraint has been measured on instances from 6 to 9 boxes in Section 3.5.2.

3.5.1 Complete formulation

The formulation has been solved at optimality for the 30 instances with 6, 7 and 8 boxes, for 27 instances with 9 boxes and 15 instances with 10 boxes. Because of the limited computational time, it has been suboptimally solved for 3 instances (over 30) with 9 boxes, with an average CPLEX GAP[1] equal to 48.28%, and suboptimally solved for 15 instances (over 30) with 10 boxes, with an average CPLEX GAP equal to 39.56%. Information about

[1]The CPLEX GAP is the relative difference between the objective value of the best feasible solution and the best-known lower bound.

the computational times, the number of used ULDs and their filling rates is provided in Table 3.1.

Table 3.1: Results of the complete formulation resolution (calculations are made over the 30 instances)

Sample size	Times [s.]			# ULDs	Filling rates [%]		
	Avg.	Min.	Max.	Avg.	Avg.	Min.	Max.
$n = 6$	7.73	0.47	139.25	1.10	33.28	6.78	53.43
$n = 7$	30.59	0.85	517.37	1.23	39.07	7.58	72.09
$n = 8$	30.59	1.04	274.26	1.27	39.93	6.46	59.20
$n = 9$	570.99	7.57	3600.00	1.33	36.21	8.71	58.49
$n = 10$	2085.58	28.99	3600.00	1.53	40.63	10.77	67.12

Sometimes the formulation requires large amounts of computational time to solve instances even with a small number of boxes. This computation time becomes a limitation with instances with more than 8 boxes. This observation will be useful in Chapter 6 for determining the ranges of the matheuristic parameters.

3.5.2 Influence of the specific constraints

In order to observe the influence of the specific constraints, several adapted versions of the initial formulation have been created. In each version, one specific constraint is dropped and the adapted formulation is then solved on the same small preliminary instances from 6 to 9 boxes as in Section 3.5.1. Four versions are built: the first version ignores the fragility constraint, the second does not take into account the orientation constraints, the third does not consider the cuts of the ULDs (ULDs are then full rectangular parallelepipeds) and the fourth drops the uniformity of the weight distribution. Figure 3.9 shows the average computational times and the average objective function values over the 30 instances of each sample size for the different versions and the complete formulation.

The four adapted formulations are able to optimally solve all the 30 instances with 6, 7 and 8 boxes. The adapted formulation without the fragility constraint is able to optimally solve 26 instances and suboptimally solve the remaining 4 instances with 9 boxes (average CPLEX GAP equal to 51.13%). The formulation without the orientation constraint is able to optimally solve 27 instances, suboptimally solve 2 instances (average CPLEX GAP equal to

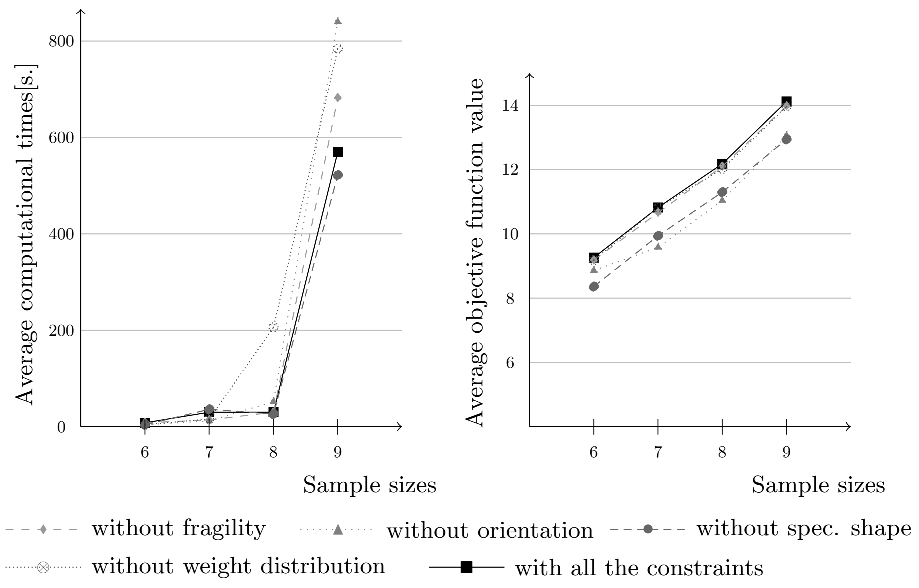


Figure 3.9: Influence of the specific constraints on the computational times and the objective function values

36.38%) and fails to solve the remaining instance with 9 boxes. The formulation without the special shapes of the ULDs is able to optimally solve 28 instances, suboptimally solve one instance (CPLEX GAP equal to 18.68%) and fails to solve the remaining instance with 9 boxes. The adapted formulation without the weight distribution constraint is able to optimally solve 26 instances with 9 boxes and suboptimally solve the remaining 4 instances (average CPLEX GAP equal to 49.44%). The two instances for which the formulation without the orientation constraint and the formulation without the special shapes of the ULDs are unable to find a feasible solution in one hour of computational time are ignored in the average calculations for the instances with 9 boxes, i.e., only 28 instances of 9 boxes are considered in Figure 3.9.

When the orientation constraint is dropped, every box is free to orthogonally rotate in all the directions. This means that the number of possible loading patterns increases and this tends to increase the average computational times to optimally solve the problem as well. However, it may also create better solutions with respect to the objective function as shown in Figure 3.9. The same reasoning can be applied to the adapted formulation without the fragility constraint. Without this constraint, the number of possible loading patterns also increases and this may explain why the formulation needs a larger amount of time to be solved optimally. However, the number of patterns seems to be smaller than without the orientation constraint, leading to a smaller average computational time than without the previous adapted formulation and thus also a smaller improvement of the objective function values. The formulation without the weight distribution constraint is, on average, slower than the complete formulation, surely for the same reason as previously. The average objective function values are slightly improved in comparison to the complete formulation. This improvement can perhaps be more noteworthy if the number of boxes to be packed is larger since the weight distribution constraint becomes restrictive when the volume to be packed is more important.

Conversely, the consideration of the special shapes of the ULDs seems to increase the average amount of computational time needed to solve the problems. The average objective function values are smaller when the special shapes are dropped. This observation is clearly expected: this adapted formulation drops the constraints with the cuts and the support they offered but the parameter describing the volume (or the cost) of the ULDs have not been adjusted. The program is thus able to pack more boxes in a ULD with the same volume (or cost). Even if this parameter is updated, then the proposed ULDs are not identical to those proposed to the other original and adapted versions and therefore, the average function values can still not be compared to the others.

3.6 Areas for improvement

In Section 3.4.2, the fragility of boxes has been taken into account in the formulation. As an extension, the possibility for boxes to be especially dense could also be considered and thus those boxes should not be supported by other boxes. A parameter d_i equal to 1 indicates that box i cannot be supported by other boxes. The constraints for this situation would be:

$$\sum_{i=1}^{n} s_{ki} \leq n(1 - d_k) \qquad \forall k \in \{1, ..., n\}. \tag{3.71}$$

This fragility management can be easily extended. We can consider that a fragile box can support other boxes only if these are also fragile. In this case, for the possible values of parameters f_i and f_k, variables s_{ik} and s_{ki} can take the following values:

f_i	f_k	s_{ik}	s_{ki}	
0	0	0 or 1	0 or 1	
0	1	0	0 or 1	box k cannot support box i
1	0	0 or 1	0	box i cannot support box k
1	1	0 or 1	0 or 1	

This can be achieved by ensuring that

$$s_{ik} f_k \leq f_i \quad \forall i, k.$$

One step further, we can imagine that the parameter f_i is a real parameter describing the capacity of supporting boxes. This parameter would be influenced by the material of the boxes, the orientation of the boxes (as in Ceschia and Schaerf (2013)) as well as the density of its contents. Consider that a box with a small value of f_i is able to support a lot of weight while a large value means the box cannot support heavy items. We use this new parameter definition as follows: a box k can support a box i if and only if $f_k \leq f_i$. In this way, we can prevent very dense boxes to be supported by less dense boxes. This can be achieved with the same set of constraints $s_{ik} f_k \leq f_i$ for all i, k where f_k and f_i are real numbers. Similarly, the heuristic proposed by Techanitisawad and Tangwiwatwong (2004) envisages items with higher density to be packed below items of lower density.

Chapter 4

Several MILP-based constructive heuristics

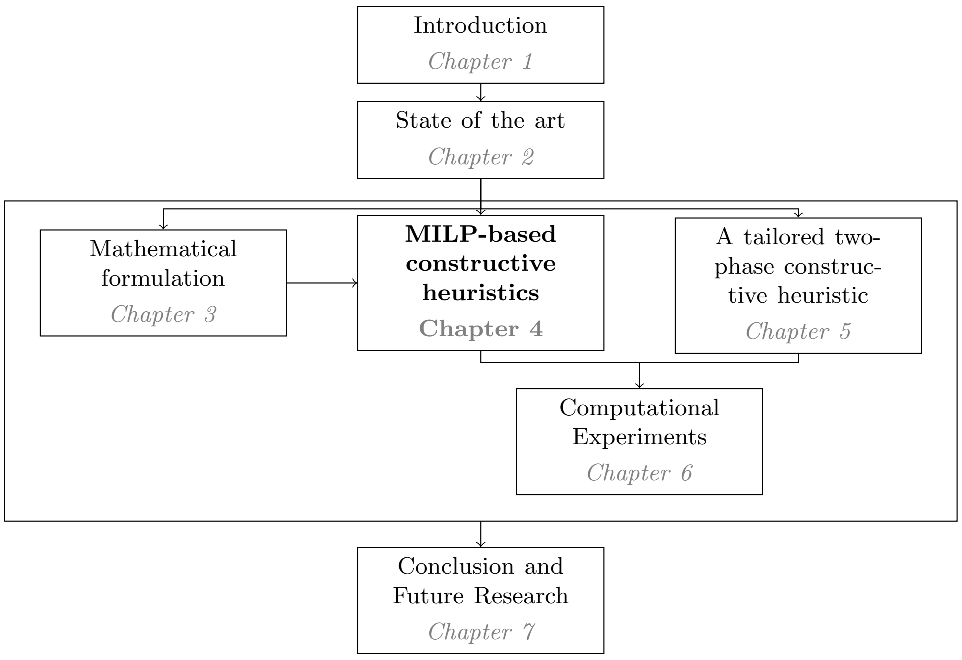
The MILP formulation provided in Chapter 3 includes many integer variables, which partially explains the slowness of its resolution. Therefore, this fourth chapter presents three constructive matheuristics able to use the potential offered by the mathematical formulation combined to heuristic methods: the Relax-and-Fix in Section 4.1, the Insert-and-Fix in Section 4.2 and the Fractional Relax-and-Fix algorithms in Section 4.3. They are iterative procedures which decompose a large-scale MILP problem into several easier subproblems in order to quickly get an initial feasible solution for the original problem or to compute bounds on the optimal value. Two specificities of the developed matheuristics are presented in Section 4.4. Results of preliminary experimentations are given in Section 4.5. Finally, a direction for future research is provided in Section 4.6.

These algorithms were originally created to solve lot-sizing problems and have never been applied to Bin Packing Problems. The application to the specific three-dimensional MBSBPP described in Chapter 2 is not straightforward and needs a deep modification. Thus, the purpose of this fourth chapter is to adapt these methods to quickly find a feasible solution.

The methodology presented in this chapter has been submitted for publication in March 2017 as *Paquay, C., S. Limbourg, M. Schyns, and J.F. Oliveira (2017). MIP-based constructive heuristics for the three-dimensional bin-packing problem with transportation constraints.*

For the sake of clarity, the different sets of variables are called *families* hereinafter. For instance, u_j family stands for u_j variables for all $j \in \{1, ..., m\}$.

The current position in the thesis outline is shown in bold in Figure 4.1.



Introduction
Chapter 1
State of the art
Chapter 2
Mathematical formulation
Chapter 3
MILP-based constructive heuristics
Chapter 4
A tailored two-phase constructive heuristic
Chapter 5
Computational Experiments
Chapter 6
Conclusion and Future Research
Chapter 7

Figure 4.1: Current position in the thesis outline

4.1 Relax-and-Fix heuristic

The Relax-and-Fix (R&F) methodology decomposes a large-scale MILP problem into several easier subproblems, namely by relaxing the integrality restriction of some variables, in order to reduce the computational times.

4.1.1 Literature review

The R&F was originally introduced to solve lot-sizing problems as explained in Pochet and Wolsey (2006) but has also been used in different areas. For instance, Kelly and Mann (2004) apply the R&F procedure to develop a flowsheet decomposition heuristic. This strategy allows to reduce the time spent to find good integer-feasible solutions when solving closed-shop scheduling problems found in the process industries. The basic idea of their algorithm is to assign units and vessels into equipment-groups. Then they iteratively solve each group using a separate call to a MILP solver and fix the binary variables to the best solution found. After that, they proceed to the next group in the sequence which is primarily determined by the material-flow-path. They also consider the auxiliary procedure of dropping, hiding or ignoring certain constraints. Beraldi et al. (2008) deal with an identical parallel machine lot-sizing and scheduling problem with sequence-dependent set-up costs. They use the R&F methodology with four distinct partitions of the set of variables and develop two types of rolling-horizon heuristic. The authors base their procedures on a compact formulation relying on the hypotheses of identical machines. They conclude that the relax-and-fix procedure outperforms the rolling-horizon heuristics. Ferreira et al. (2010) present a MILP model for the integrated lot-sizing and sequencing problem found in typical Brazilian regional small-scale soft drink plants. Considering the possibility of creating sub-models that are smaller and easier, they apply the R&F methodology. They propose different strategies for fixing the variables and explore at the same time the distinct configurations of the CPLEX system. Finally, they measure the quality of these strategies with a performance profile proposed by Dolan and Moré (2002). Oliveira et al. (2014) propose a R&F heuristic procedure to solve a specific vehicle-reservation assignment model. In order to reduce the potential infeasibility of a subproblem of the R&F heuristic, they include a constraint based on local branching that enables and controls changes between iterations. Baena et al. (2015) adapt the R&F methodology to the Controlled Tabular Adjustment (CTA) which is a technique for tabular data protection. CTA can be formulated as a MILP problem, but the size of the data is too important to get a solution quickly. The authors customized the R&F heuristic to get a good initial solution and lower bounds. They also add a backtrack mechanism in the algorithm to avoid some infeasible iterations. Furthermore, they combine their R&F with a block coordinate descent heuristic. To choose the best parameters, the performance profile defined in Dolan and Moré (2002) is used. To our knowledge, the above mentioned articles present a good coverage of applications of the R&F heuristics.

4.1.2 Methodology

The first in-depth explanation of the R&F methodology can be found in Pochet and Wolsey (2006) for logistics problems. A description is as follows. The original MILP

$$\begin{aligned} \min \quad & cx + fy \\ \text{s.t.} \quad & Ax + By \geq b \\ & x \in \mathbb{R}^n_+ \\ & y \in \{0,1\}^p \end{aligned} \tag{4.1}$$

has to be solved. Let $Q = \{1, ..., p\}$ be the index set of the y variables. These variables can be integer instead of binary variables. The authors partition the index set of integer variables y into R disjoints subsets $Q^1, ..., Q^R$ of decreasing importance. They also introduce some auxiliary subsets $U^r \subseteq Q^{r+1} \cup \ldots \cup Q^R$ for $r \in \{1, ..., R-1\}$. The methodology comprises sequentially solving R smaller MILPs to find a heuristic solution to the original MILP. These subproblems are denoted MILP^r with $1 \leq r \leq R$ and are as in (4.2) where $y_{\mathbb{j}}^{r-1}$ denotes the value of variable $y_{\mathbb{j}}$ obtained at iteration $r-1$. Note that the index $\mathbb{j}$ is different from the index j related to the ULDs.

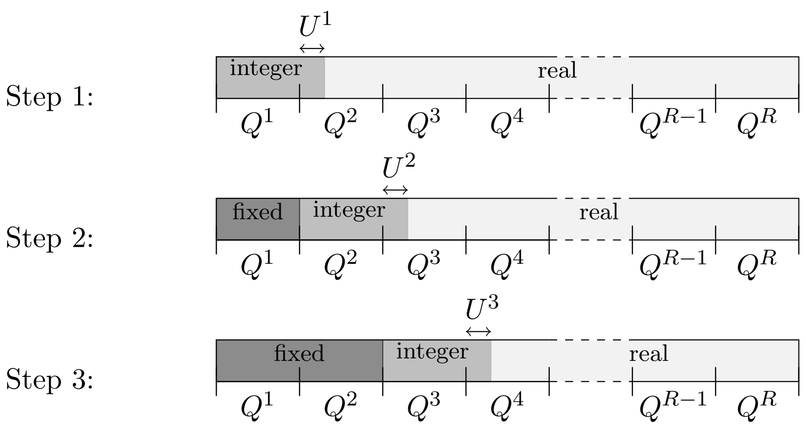
$$
\begin{aligned}
\min \quad & cx + fy & & (\text{MILP}^r) \\
\text{s.t.} \quad & Ax + By \geq b & & (4.2) \\
& x \in \mathbb{R}_+^n & & \\
& y_{\mathbb{j}} = y_{\mathbb{j}}^{r-1} \in \{0,1\} & \forall \mathbb{j} \in Q^1 \cup \cdots \cup Q^{r-1} & \\
& y_{\mathbb{j}} \in \{0,1\} & \forall \mathbb{j} \in Q^r \cup U^r & \\
& y_{\mathbb{j}} \in [0,1] & \forall \mathbb{j} \in Q \backslash (Q^1 \cup \cdots \cup Q^r \cup U^r) &
\end{aligned}
$$

The idea at iteration r is to fix values of the variables y with index in $Q^1 \cup \cdots \cup Q^{r-1}$ at their optimal values from the solution of the previous subproblem (MILP^{r-1}) and relax the integrality restriction for the remaining variables (with index in $Q \backslash (Q^1 \cup \cdots \cup Q^r \cup U^r)$). As can be seen, the subsets U^r make the algorithm less myopic as they represent sets of variables whose values are not kept for the next iteration but still have to be integer. An example of the types of the y variables during the first three iterations of the R&F heuristic is illustrated in Figure 4.2.

Therefore, MILP^R provides a heuristic solution for the original MILP (4.1) unless there exists an iteration r for which MILP^r is infeasible which means the heuristic failed.

This methodology is adapted to our MILP. The families X_{ij}, Y_{ij} and Z_{ij} represent the real variables x in (4.1), while all the other families represent the variables y. The first step is to partition the indices of the integer variables y into $Q^1, ..., Q^R$ subsets of decreasing importance. Their construction is box-oriented: at each iteration, the variables related to a subset of boxes are not relaxed, while the variables related to the rest of boxes have the integrality constraint relaxed. After solving this easier MILP, we keep the values of the integer variables relative to some boxes to fix them at the next iteration.

We define a *block*, denoted $\mathcal{B}$, as a selection of families of integer variables. Indeed, if all the integer variables relative to a subset of the boxes are relaxed, the solution obtained can be too far from the original problem and this causes the failure of the heuristic. Then, the parts Q^r and U^r are built by selecting elements relative to some boxes within each family of the block.



Step 1:
U^1
integer
real
Q^1 Q^2 Q^3 Q^4 Q^{R-1} Q^R
Step 2:
U^2
fixed
integer
real
Q^1 Q^2 Q^3 Q^4 Q^{R-1} Q^R
Step 3:
U^3
fixed
integer
real
Q^1 Q^2 Q^3 Q^4 Q^{R-1} Q^R

Figure 4.2: An example of the types of variables y during the first three iterations of a Relax-and-Fix algorithm

Note. During the first step, the variables from $Q^1 \cup U^1$ are integer while the others are relaxed. During the second step, the variables from Q^1 are fixed to the values obtained in step 1, the variables from $Q^2 \cup U^2$ are integer while the others are relaxed. During the third step, the variables from $Q^1 \cup Q^2$ are fixed to the values obtained in step 2, the variables from $Q^3 \cup U^3$ are integer while the others are relaxed.

In mathematical terms, the set of indices of the integer variables y, denoted $\mathcal{F}$, is divided into two parts:

- $\mathcal{F}_\mathcal{B}$ contains the indices of the variables of the families in the selected block
- $\mathcal{F}_{\neg\mathcal{B}} = \mathcal{F} \backslash \mathcal{F}_\mathcal{B}$ contains all the other indices.

Only the subset $\mathcal{F}_\mathcal{B}$ is partitioned in subsets $Q^1, ..., Q^R$ of decreasing importance. Then one has $\mathcal{F} = \mathcal{F}_\mathcal{B} \uplus \mathcal{F}_{\neg\mathcal{B}}$ and $\mathcal{F}_\mathcal{B} = Q^1 \uplus \cdots \uplus Q^R$. The sets $U^r \subseteq Q^{r+1} \cup \cdots \cup Q^R$ for $r \in \{1, ..., R-1\}$ are also introduced. They represent variables corresponding to some next boxes in the sequence which also satisfy the integrality condition. Based on these definitions, (4.2) is turned into the following MILP^r which is sequentially solved for $r \in \{1, ..., R\}$:

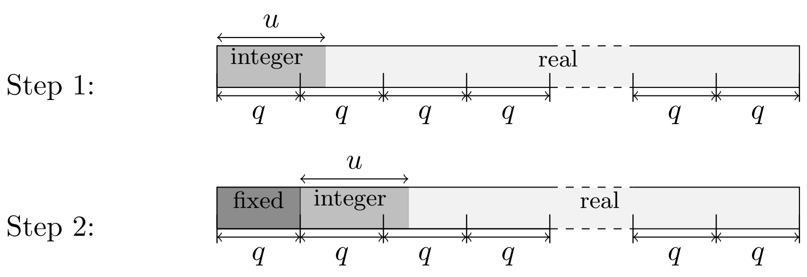
$$\begin{aligned} \min \quad & cx + fy && (\text{MILP}^r) \\ \text{s.t.} \quad & Ax + By \geq b \\ & x \in \mathbb{R}^n_+ \\ & y_\mathbb{j} \in \mathbb{Z}^+ && \forall \mathbb{j} \in \mathcal{F}_{\neg\mathcal{B}} \end{aligned}$$

$$\begin{aligned}
&y_{\mathbb{j}} = y_{\mathbb{j}}^{r-1} \in \mathbb{Z}^+ && \forall \mathbb{j} \in Q^1 \cup \cdots \cup Q^{r-1}\\
&y_{\mathbb{j}} \in \mathbb{Z}^+ && \forall \mathbb{j} \in Q^r \cup U^r\\
&y_{\mathbb{j}} \in \mathbb{R}^+ && \forall \mathbb{j} \in \mathcal{F}_{\mathcal{B}} \backslash (Q^1 \cup \cdots \cup Q^r \cup U^r).
\end{aligned}$$

In order to describe the subsets Q^r and U^r, two integers q and u, with $q < u$, are introduced. Because the subsets Q^1, ..., Q^R are of decreasing importance and because the largest boxes are the most difficult to pack, boxes are first sorted by decreasing volume. The subset Q^1 (resp. U^1) is then defined as the variables relative to the first q boxes (resp. from box $q+1$ to u) in the families of the block, Q^2 (resp. U^2) to the q next boxes (resp. from box $2q+1$ to $q+u$) in the sequence of boxes and so on. More generally, sets Q^r and U^r can be defined as the set of variables relative to the boxes:

Q^r	from box $(r-1) \times q + 1$	to $r \times q$
U^r	from box $r \times q + 1$	to $(r-1) \times q + u$
$Q^r \cup U^r$	from box $(r-1) \times q + 1$	to $(r-1) \times q + u$

where $r \in \{1, ..., R\}$ is the index of the iteration. At step r, the variables relative to the boxes from 1 to $(r-1) \times q$ are fixed to a previous value, the variables relative to the u next boxes (i.e., from $(r-1) \times q+1$ to $(r-1) \times q+u$) are integer, but among those, only the values of the variables relative to the first q boxes (i.e., from $(r-1) \times q + 1$ to $(r-1) \times q + q$) are kept for the next iteration. As a matter of fact, parameter u denotes the number of boxes whose variables are integer and to be determined and q the number of boxes whose variables are fixed. The first two steps of Figure 4.2 can thus be represented as follows:



Since there are a lot of families relative to boxes, there exist many potential blocks. Only the most representative and intuitive blocks are considered in this thesis. The chosen blocks are described in Table 4.1. The second column contains the name given to the blocks which are used hereunder. The variables considered in these blocks can be seen as primary variables. Indeed, the other variables are mainly intermediate variables depending on the values of the primary ones and defined to handle specific constraints.

Table 4.1: Blocks tested in the Relax-and-Fix heuristic

Families in the block	**Name of the block**
p_{ij}	p_{ij}
p_{ij}, x_i, y_i, z_i, x'_i, y'_i, z'_i	Coord & p_{ij}
x_i, y_i, z_i, x'_i, y'_i, z'_i	Coord
p_{ij}, x_i, y_i, z_i, x'_i, y'_i, z'_i, r_{iab}, g_i, γ^1_i, γ^2_i	All i

In practice, if the block *Coord* defined in Table 4.1 is examined, then the variables during the first two steps of the algorithm are as in Table 4.2.

Table 4.2: Types of the variables during the first two steps of the Relax-and-Fix heuristic with the block *Coord*

	Variables	**Types**	**Sets**
	X_{ij}, Y_{ij}, Z_{ij},$\forall i \in \{1, ..., n\}$, $\forall j \in \{1, ..., m\}$	Real	
	u_j, p_{ij}, r_{iab}, x^p_{ik}, y^p_{ik}, z^p_{ik}, g_i, h_{ik}, a_{ik}, m_{ik}, o_{ik}, s_{ik}, γ^1_i, γ^2_i, η^1_{ik}, η^2_{ik}, η^3_{ik}, η^4_{ik}, β^l_{ik}, $\forall i, k \in \{1, ..., n\}$, $\forall j \in \{1, ..., m\}$	Integer	$\mathcal{F}_{\neg\mathcal{B}}$
Step 1	x_1, y_1, z_1, x'_1, y'_1, z'_1,..., x_u, y_u, z_u, x'_u, y'_u, z'_u	Integer	$Q^1 \cup U^1$
	x_{u+1}, y_{u+1}, z_{u+1}, x'_{u+1}, y'_{u+1}, z'_{u+1},..., x_n, y_n, z_n, x'_n, y'_n, z'_n	Real	$\mathcal{F}_{\mathcal{B}} \backslash (Q^1 \cup U^1)$
Step 2	x_1, y_1, z_1, x'_1, y'_1, z'_1,..., x_q, y_q, z_q, x'_q, y'_q, z'_q	Fixed to previous value	Q^1
	x_{q+1}, y_{q+1}, z_{q+1}, x'_{q+1}, y'_{q+1}, z'_{q+1},..., x_{q+u}, y_{q+u}, z_{q+u}, x'_{q+u}, y'_{q+u}, z'_{q+u}	Integer	$Q^2 \cup U^2$
	x_{q+u+1}, y_{q+u+1}, z_{q+u+1}, x'_{q+u+1}, y'_{q+u+1}, z'_{q+u+1},..., x_n, y_n, z_n, x'_n, y'_n, z'_n	Real	$\mathcal{F}_{\mathcal{B}} \backslash (Q^1 \cup Q^2 \cup U^2)$

The stopping criterion is based on the number of steps. Indeed, the algorithm must stop as soon as $q \times r + u \geq n$, i.e., $r \geq \frac{n-u}{q}$.

Algorithm 1 represents the R&F approach. Here are some details about its content:

- `selVar[i]` represents all the variables of the block families relative to box i,

- `results[i]` contains the kept values of the selected variables relative to box i,

- r is the index of the step of the algorithm.

Algorithm 1 Relax-and-Fix heuristic on boxes

```
$r$=0
**while** $r < \frac{n-u}{q}$ **do**
  **for** $i = 1$ **to** $q \times r$ **do**
    selVar[i] $\leftarrow$ results[i]
  **end for**
  **for** $i = \min\{q \times r + u + 1, n\}$ **to** $n$ **do**
    selVar[i].integralityRestrictionRelaxed
  **end for**
  solveMILP
  **if** MILPHasASolution **then**
    **for** $i = q \times r + 1$ **to** $\min\{q \times r + q, n\}$ **do**
      results[i] $\leftarrow$ value(selVar[i])
    **end for**
    $r \leftarrow r + 1$
  **else**
    heuristicFailed - STOP
  **end if**
**end while**
HeuristicSolutionFound
```

4.2 Insert-and-Fix heuristic

The Insert-and-Fix (I&F) methodology uses an iterative procedure to build an initial solution. Introduced in Liberalino (2012), this algorithm is similarly based on the decomposition of a large-scale MILP into smaller subproblems. However, the I&F and the R&F heuristics are different in their construction of these subproblems. First, in the subproblems built by the I&F method, not all the variables are considered at every iteration. Variables are added step by step along the algorithm. The second difference is the absence of relaxation of the integrality restriction. At each iteration, all the considered variables are set to their final type. However, as in the R&F algorithm, at the end of each iteration the values of a subset are kept to be fixed at the next step.

We also adapt this methodology to our MILP as follows: let $\mathcal{F}$ denote the set of indices of **all** the variables x and y in MILP (4.1). $\mathcal{F}$ is decomposed into two parts: $\mathcal{F}_{\mathcal{B}}$ and $\mathcal{F}_{\neg\mathcal{B}}$. $\mathcal{F}_{\mathcal{B}}$ contains the indices of the variables which are introduced step by step during the algorithm, while $\mathcal{F}_{\neg\mathcal{B}}$ contains the indices of the variables which are present in the whole algorithm. $\mathcal{F}_{\mathcal{B}}$ may contain integer variables y and non-integer variables x. The families of integer (resp. non-integer) variables of $\mathcal{F}_{\mathcal{B}}$ are partitioned into parts Q^1, ..., Q^R (resp. W^1, ..., W^R) of decreasing importance. These cells Q^r and W^r for $r \in \{1, ..., R\}$ are added iteration after iteration. Some subsets $U^r \subseteq Q^{r+1} \cup \cdots \cup Q^R$ for $r \in \{1, ..., R-1\}$ are also introduced to make the algorithm more flexible. Then, the methodology consists of solving sequentially R MILPs, denoted MILP^r with $1 \leq r \leq R$, to find a heuristic solution to the original MILP (4.1):

$$\begin{aligned}
\min \quad & cx + fy && (\text{MILP}^r) \\
\text{s.t.} \quad & Ax + By \geq b && \qquad (4.3)\\
& x_{\mathbb{i}} \in \mathbb{R}^+ && \forall \mathbb{i} \in \mathcal{F}_{\neg\mathcal{B}} \\
& y_{\mathbb{j}} \in \mathbb{Z}^+ && \forall \mathbb{j} \in \mathcal{F}_{\neg\mathcal{B}} \\
& y_{\mathbb{j}} = y_{\mathbb{j}}^{r-1} \in \mathbb{Z}^+ && \forall \mathbb{j} \in Q^1 \cup \cdots \cup Q^{r-1} \\
& x_{\mathbb{i}} \in \mathbb{R}^+ && \forall \mathbb{i} \in W^1 \cup \cdots \cup W^r \\
& y_{\mathbb{j}} \in \mathbb{Z}^+ && \forall \mathbb{j} \in Q^r \cup U^r.
\end{aligned}$$

Note that the index $\mathbb{i}$ is different from the index i related to the boxes. At step r, integer variables with index in $Q^1 \cup \cdots \cup Q^{r-1}$ are fixed at their optimal values from the solution of the previous subproblem (MILP^{r-1}) and those with index in $Q^r \cup U^r$ are dealt with. Note that variables in $\mathcal{F}_{\mathcal{B}} \backslash (Q^1 \cup \cdots \cup Q^r \cup U^r \cup W^1 \cup \cdots \cup W^r)$ are totally ignored and there is no relaxation of the integrality restriction. An example of the integer variables y with indices in $\mathcal{F}_{\mathcal{B}}$ considered during the first three iterations of the I&F heuristic is illustrated in Figure 4.3.

In our MILP, the subset $\mathcal{F}_{\neg\mathcal{B}}$ only contains the family u_j because it does not correspond to boxes. This means that these variables are present along the whole algorithm. The subset $\mathcal{F}_{\mathcal{B}}$ holds the families of non-integer variables X_{ij}, Y_{ij}, Z_{ij} (split into $W^1, ..., W^R$) and all the remaining families of integer variables (split into $Q^1, ..., Q^R$).

The key part of the methodology is the definition of Q^r, U^r and W^r for all r. Their construction relies on the idea of inserting boxes, step by step, inside ULDs. Therefore, all the variables, integer or non-integer, are divided in such a way that each subset Q^r and W^r represent the variables inherent to several boxes. At the first step, the idea comprises introducing the first u boxes. The index set of these variables is the set $Q^1 \cup U^1$ for the integer

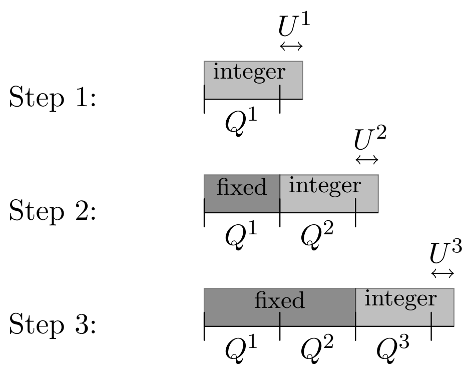


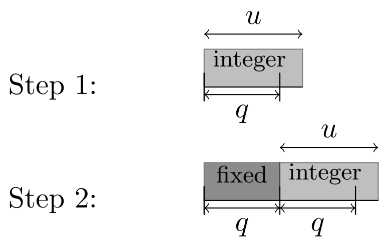
Figure 4.3: An example of the integer variables y with indices in $\mathcal{F}_\mathcal{B}$ considered during the first three iterations of an Insert-and-Fix heuristic

Note. During the first step, only the variables from $Q^1 \cup U^1$ exist and are integer while the other y variables from $\mathcal{F}_\mathcal{B}$ are ignored. During the second step, the variables from Q^1 are fixed to the values obtained in step 1, the variables from $Q^2 \cup U^2$ are considered and integer while the others are ignored. During the third step, the variables from $Q^1 \cup Q^2$ are fixed to the values obtained in step 2, the variables from $Q^3 \cup U^3$ are considered and integer while the others are still ignored.

variables and the W^1 for the non-integer variables: Q^1 (resp. U^1, W^1) is defined as the index set of the variables relative to the boxes from 1 to q (resp. from $q+1$ to u, from 1 to u), with $q < u$. More generally, sets Q^r, U^r and W^r can be defined as the set of variables relative to the boxes:

$$\begin{array}{rlll}
Q^r & \text{from box } (r-1)\times q+1 & \text{to } r\times q & \forall r, \\
U^r & \text{from box } r\times q+1 & \text{to } (r-1)\times q+u & \forall r, \\
Q^r\cup U^r & \text{from box } (r-1)\times q+1 & \text{to } (r-1)\times q+u & \forall r, \\
W^1 & \text{from box } 1 & \text{to } u, & \\
W^r & \text{from box } u+(r-2)\times q+1 & \text{to } (r-1)\times q+u & \forall r\geq 2,
\end{array}$$

where $r \in \{1, ..., R\}$ is the index of the iteration. As a matter of fact, the parameter u denotes the number of boxes whose variables are to be determined and q the number of boxes whose variables are fixed. The first two steps of Figure 4.3 can thus be represented as follows:



Another representation of the first two iterations of the I&F heuristic is shown in Table 4.3.

Table 4.3: Existing variables during the first two steps of the Insert-and-Fix heuristic

	Variables	**Types**	**Sets**
	$u_j \quad \forall j \in \{1, ..., m\}$	Integer	$\mathcal{F}_{\neg\mathcal{B}}$
Step 1	X_{ij}, Y_{ij}, Z_{ij},$\forall i \in \{1, ..., u\}$, $\forall j \in \{1, ..., m\}$	Real	W^1
	x_i, y_i, z_i, x'_i, y'_i, z'_i, a_{ik}, p_{ij}, r_{iab}, x^p_{ik}, y^p_{ik}, z^p_{ik}, g_i, h_{ik}, m_{ik}, o_{ik}, s_{ik}, γ^1_i, γ^2_i, η^1_{ik}, η^2_{ik}, η^3_{ik}, η^4_{ik}, β^l_{ik}, $\forall i, k \in \{1, ..., u\}$, $\forall j \in \{1, ..., m\}$	Integer	$Q^1 \cup U^1$
Step 2	x_i, y_i, z_i, x'_i, y'_i, z'_i, a_{ik}, p_{ij}, r_{iab}, x^p_{ik}, y^p_{ik}, z^p_{ik}, g_i, h_{ik}, m_{ik}, o_{ik}, s_{ik}, γ^1_i, γ^2_i, η^1_{ik}, η^2_{ik}, η^3_{ik}, η^4_{ik}, β^l_{ik}, $\forall i, k \in \{1, ..., q\}$, $\forall j \in \{1, ..., m\}$	Fixed to previous value	Q^1
	X_{ij}, Y_{ij}, Z_{ij},$\forall i \in \{1, ..., q+u\}$, $\forall j \in \{1, ..., m\}$	Real	$W^1 \cup W^2$
	x_i, y_i, z_i, x'_i, y'_i, z'_i, a_{ik}, p_{ij}, r_{iab}, x^p_{ik}, y^p_{ik}, z^p_{ik}, g_i, h_{ik}, m_{ik}, o_{ik}, s_{ik}, γ^1_i, γ^2_i, η^1_{ik}, η^2_{ik}, η^3_{ik}, η^4_{ik}, β^l_{ik}, $\forall i, k \in \{q+1, ..., q+u\}$, $\forall j \in \{1, ..., m\}$	Integer	$Q^2 \cup U^2$

We need to be careful with the weight distribution constraints. Indeed, in a subproblem, some constraints could become meaningless if other related variables have not yet been included. Therefore, those constraints are considered at each iteration for the boxes present at that moment of the algorithm. It is rather restrictive, but enforcing them all at the last step would tend to make the problem infeasible.

Algorithm 2 represents the I&F methodology. In this algorithm, `intVar[i]` (resp. `realVar[i]`) represents the integer (resp. non-integer) variables associated to box i and the method add(`intVar[i]`) and add(`realVar[i]`) consists of adding the variables and the constraints associated to box i.

Algorithm 2 Insert-and-Fix heuristic

```
$r$=0
while $r < \frac{n-u}{q}$ do
  for $i = 1$ to $q \times r$ do
    intVar[i] $\leftarrow$ results[i]
  end for
  for $i = q \times r + 1$ to $\min\{q \times r + u, n\}$ do
    add(intVar[i])
    add(realVar[i])
  end for
  solveMILP
  if MILPHasASolution then
    for $i = q \times r + 1$ to $\min\{q \times r + q, n\}$ do
      results[i] $\leftarrow$ value(intVar[i])
    end for
    $r \leftarrow r + 1$
  else
    heuristicFailed - STOP
  end if
end while
HeuristicSolutionFound
```

4.3 Fractional Relax-and-Fix heuristic

The Fractional Relax-and-Fix (FRF) methodology can be considered as a combination of the R&F and the I&F heuristics: it is an iterative procedure working on a subset of the original MILP (as in the I&F), but also on the relaxation of the integrality constraints (as in the R&F). This methodology was introduced in Pochet and Wolsey (2006) and in Liberalino (2012). The basic concept is identical to the I&F heuristic, that is, only a subset of the original model is solved iteratively. Variables, and constraints inherent to them, are added step by step along the algorithm. The difference is the relaxation of the integrality restriction on a part of the variables.

Here is how the FRF has been adapted to our MILP. From MILP (4.1), let $\mathcal{F}$ be the index set of the variables x and y. This set is partitioned into two parts: $\mathcal{F}_{\neg\mathcal{B}}$ contains the indices of the variables which are present in the whole algorithm while $\mathcal{F}_{\mathcal{B}}$ contains those of the families which successively appear as in the I&F methodology. The influence of the R&F appears in the $\mathcal{F}_{\mathcal{B}}$ set. $\mathcal{F}_{\mathcal{B}}$ contains the indices of families of integer and non-integer variables. From the integer families, several are selected to have the integrality restriction relaxed on a part of them. This selection of integer families is called a block as defined in the R&F heuristic. The blocks from Table 4.1 are also considered for this heuristic. The chosen block is divided to build the subsets $Q^1, ..., Q^R$. In mathematical notations, $\mathcal{F}_{\mathcal{B}}$ is divided into two parts:

- $\mathcal{F}_{\mathcal{B}}^b$ represents several families of integer variables on which the integrality restriction is relaxed at some stage of the algorithm. It contains the indices of the variables of the families in the selected block;
- $\mathcal{F}_{\mathcal{B}}^{\neg b}$ contains families of integer or real variables. This subset is not affected by the relaxation of the integrality constraints.

This partitioning is represented in Figure 4.4.

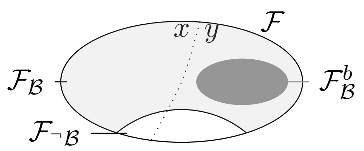

x
y
$\mathcal{F}$
$\mathcal{F}_{\mathcal{B}}$
$\mathcal{F}_{\mathcal{B}}^b$
$\mathcal{F}_{\neg\mathcal{B}}$

Figure 4.4: Partitioning of the variables in a Fractional Relax-and-Fix algorithm

As in the R&F heuristic, the only variables that are fixed during the algorithm are those previously affected by the relaxation of the integrality constraint. For this reason, only the variables of $\mathcal{F}_{B}^b$ can be fixed. The subset $\mathcal{F}_{\mathcal{B}}^b$ is divided into several parts: $Q^1 \cup \cdots \cup Q^R$ and some auxiliary subsets are introduced: $U^r \subseteq Q^{r+1} \cup \cdots \cup Q^R$ and $V^r \subseteq U^r$ for all $r \in \{1, ..., R\}$. The idea is to consider integer variables with index in $Q^r \cup U^r$ and to relax the integrality restriction on variables with index in $U^r \backslash V^r$. This adds more flexibility in the algorithm since the values of all the integer variables are not kept for the next iteration. On the other hand, $\mathcal{F}_{B}^{\neg b}$ is divided into subsets $W^1, ..., W^R$ which are also inserted step by step. Thus, one has

$$\begin{aligned}
\mathcal{F} &= \mathcal{F}_{\mathcal{B}} \cup \mathcal{F}_{\neg\mathcal{B}} \text{ where } \mathcal{F}_{\mathcal{B}} = \mathcal{F}_{\mathcal{B}}^b \cup \mathcal{F}_{\mathcal{B}}^{\neg b} \text{ and } \mathcal{F}_{\mathcal{B}}^b = Q^1 \cup \cdots \cup Q^R \\
U^r &\subseteq Q^{r+1} \cup \cdots \cup Q^R \quad \forall r \in \{1, ..., R\} \\
V^r &\subseteq U^r \quad \forall r \in \{1, ..., R\} \\
\mathcal{F}_{\mathcal{B}}^{\neg b} &= W^1 \cup \cdots \cup W^R.
\end{aligned}$$

This gives the following definition of MILP^r:

$$\min \quad cx + fy \qquad\qquad (\text{MILP}^r)$$

$$\begin{aligned}
\text{s.t.}\quad & Ax + By \geq b & & \qquad (4.4)\\
& x_{\mathbb{i}} \in \mathbb{R}^+ & \forall \mathbb{i} \in \mathcal{F}_{\neg\mathcal{B}} &\\
& y_{\mathbb{j}} \in \mathbb{Z}^+ & \forall \mathbb{j} \in \mathcal{F}_{\neg\mathcal{B}} &\\
& x_{\mathbb{i}} \in \mathbb{R}^+ & \forall \mathbb{i} \in W^1 \cup \cdots \cup W^r &\\
& y_{\mathbb{j}} \in \mathbb{Z}^+ & \forall \mathbb{j} \in W^1 \cup \cdots \cup W^r &\\
& y_{\mathbb{j}} = y_{\mathbb{j}}^{r-1} \in \mathbb{Z}^+ & \forall \mathbb{j} \in Q^1 \cup \cdots \cup Q^{r-1} &\\
& y_{\mathbb{j}} \in \mathbb{Z}^+ & \forall \mathbb{j} \in Q^r \cup V^r &\\
& y_{\mathbb{j}} \in \mathbb{R}^+ & \forall \mathbb{j} \in U^r \backslash V^r. &
\end{aligned}$$

At step r, the idea is to consider the variables with index in $W^1 \cup \cdots \cup W^r$ without relaxation, to fix the variables with index in $Q^1 \cup \cdots \cup Q^{r-1}$ at their optimal values from the solution of MILP^{r-1}, to add the variables with index in $Q^r \cup U^r$, but to relax the integrality restriction on variables with index in $U^r \backslash V^r$. All the variables with index in $\mathcal{F}_{\neg\mathcal{B}}$ exist and have the final type during the whole algorithm (no integrality constraint relaxed for the y variables). An example of the integer variables y with index in $\mathcal{F}_{\mathcal{B}}^b$ during the first three iterations of a FRF algorithm is shown in Figure 4.5.

During the application of the algorithm to our specific MILP, the set $\mathcal{F}_{\neg\mathcal{B}}$ contains the family u_j because these variables are present during the entire algorithm and do not depend on the index i. $\mathcal{F}_{\mathcal{B}}^b$ contains the families of the selected block while $\mathcal{F}_{\mathcal{B}}^{\neg b}$ holds all the remaining families.

To describe the subsets Q^r, U^r, V^r and W^r, three integers q, v and u, with $q < v < u$, are needed. At the first iteration, the idea is to insert the first u boxes of the sorted sequence, but relax some variables relative to the last $u - v$ boxes and then to keep the values of a selection of variables relative to the first q boxes. Specifically, we

- introduce the variables relative to box 1 to box u from the set $\mathcal{F}_{\mathcal{B}}^{\neg b}$ and $\mathcal{F}_{\mathcal{B}}^b$,
- relax the integrality restriction on variables from the set $\mathcal{F}_{\mathcal{B}}^b$ for the box $v + 1$ to u,
- keep the values of the variables from the set $\mathcal{F}_{\mathcal{B}}^b$ for the box 1 to q for the second iteration once the MILP is solved.

On the one hand, this means that the subset Q^1 (resp. U^1, V^1) is defined as the index set of the variables in the families of the block relative to the first q boxes (resp. from box $q+1$ to u, from $q+1$ to v) of the sorted sequence, then Q^2 (resp. U^2, V^2) corresponds to the variables relative to the q next boxes

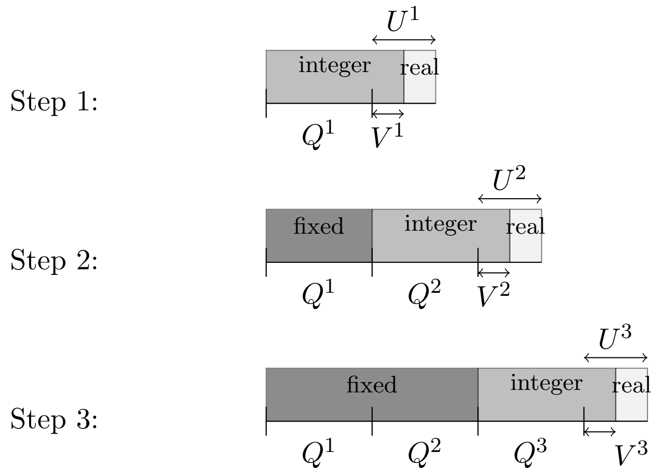


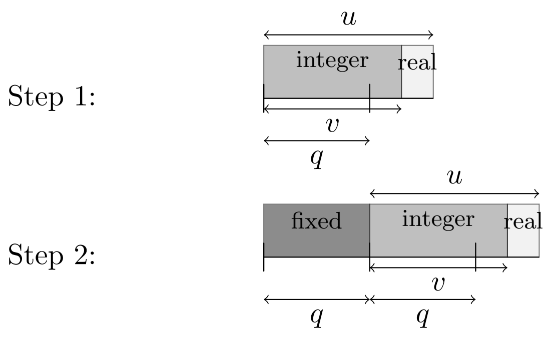
Figure 4.5: An example of the integer variables y with index in $\mathcal{F}^b_{\mathcal{B}}$ during the first three iterations of a Fractional Relax-and-Fix algorithm

Note. During the first step, only the variables from $Q^1 \cup U^1$ exist, $Q^1 \cup V^1$ are integer and $U^1 \backslash V^1$ are relaxed while the other y variables from $\mathcal{F}^b_{\mathcal{B}}$ are ignored. During the second step, the variables from Q^1 are fixed to the values obtained in step 1, the variables from $Q^2 \cup U^2$ are considered, $Q^2 \cup V^2$ are integer and $U^2 \backslash V^2$ are relaxed while the others are ignored. During the third step, the variables from $Q^1 \cup Q^2$ are fixed to the values obtained in step 2, the variables from $Q^3 \cup U^3$ are considered, $Q^3 \cup V^3$ are integer and $U^3 \backslash V^3$ are relaxed while the others are still ignored.

(resp. from box $2q+1$ to $q+u$, from box $2q+1$ to $q+v$) of the sequence and so on. On the other hand, W^1 (resp. W^2) contains the indices in $\mathcal{F}_{\mathcal{B}}^{\neg b}$ of the variables relative to the box 1 to box u (resp. $u+1$ to box $q+u$). More generally, the subsets Q^r, V^r, U^r and W^r can be defined as the set of variables relative to the boxes:

Q^r	from box $(r-1)\times q+1$	to $r \times q$	$\forall r \in \{1,...,R\}$
U^r	from box $r \times q+1$	to $(r-1)\times q+u$	$\forall r \in \{1,...,R\}$
V^r	from box $r \times q+1$	to $(r-1)\times q+v$	$\forall r \in \{1,...,R\}$
W^1	from box 1	to u	
W^r	from box $u+(r-2)\times q+1$	to $(r-1)\times q+u$	$\forall r \in \{2,...,R\}$.

As a matter of fact, parameter u denotes the number of boxes which are considered and whose variables are to be determined, v is the number of boxes whose variables have the integrality restriction to be satisfied and q the number of boxes whose variables are fixed. The first two steps of Figure 4.5 can thus be represented as follows:



Note that if $u=n$, then the FRF becomes a R&F heuristic and if $v=u$, it becomes a particular case of the I&F in which only a subset of variables relative to a box are fixed from one iteration to another.

In practice, if the block *Coord* defined in Table 4.1 is examined, then the variables during the first two steps of the algorithm are presented in Table 4.4.

The algorithm should stop as soon as $q \times r + v \geq n$, which means it runs as long as $r < \frac{n-v}{q}$.

Algorithm 3 represents the FRF heuristic. Here are more details about the new functions:

- `selVar[i]` represent all the variables of the families constituting the subset $\mathcal{F}_{\mathcal{B}}^{b}$ and

Table 4.4: The first two steps of the Fractional Relax-and-Fix heuristic on block *Coord*

	Variables	**Types**	**Sets**
	$u_j \quad \forall j \in \{1, ..., m\}$	Integer	$\mathcal{F}_{\neg \mathcal{B}}$
Step 1	X_{ij}, Y_{ij}, Z_{ij},$\forall i \in \{1, ..., u\}$, $\forall j \in \{1, ..., m\}$	Real	W^1
	p_{ij}, r_{iab}, x^p_{ik}, y^p_{ik}, z^p_{ik}, g_i, h_{ik}, a_{ik}, m_{ik}, o_{ik}, s_{ik}, γ^1_i, γ^2_i, η^1_{ik}, η^2_{ik}, η^3_{ik}, η^4_{ik}, β^l_{ik}, $\forall i, k \in \{1, ..., u\}$, $\forall j \in \{1, ..., m\}$	Integer	
	x_i, y_i, z_i, x'_i, y'_i, z'_i, $\forall i \in \{1, ..., v\}$	Integer	$Q^1 \cup V^1$
	x_i, y_i, z_i, x'_i, y'_i, z'_i, $\forall i \in \{v + 1, ..., v + u\}$	Real	$U^1 \backslash V^1$
Step 2	x_i, y_i, z_i, x'_i, y'_i, z'_i, $\forall i \in \{1, ..., q\}$	Fixed to previous value	Q^1
	X_{ij}, Y_{ij}, Z_{ij},$\forall i \in \{q + 1, ..., q + u\}$, $\forall j \in \{1, ..., m\}$	Real	$W^1 \cup W^2$
	p_{ij}, r_{iab}, x^p_{ik}, y^p_{ik}, z^p_{ik}, g_i, h_{ik}, a_{ik}, m_{ik}, o_{ik}, s_{ik}, γ^1_i, γ^2_i, η^1_{ik}, η^2_{ik}, η^3_{ik}, η^4_{ik}, β^l_{ik}, $\forall i, k \in \{1, ..., q + u\}$, $\forall j \in \{1, ..., m\}$	Integer	
	x_i, y_i, z_i, x'_i, y'_i, z'_i, $\forall i \in \{q + 1, ..., q + v\}$	Integer	$Q^2 \cup V^2$
	x_i, y_i, z_i, x'_i, y'_i, z'_i, $\forall i \in \{q + v + 1, ..., q + u\}$	Real	$U^2 \backslash V^2$

- `var[i]` represent all the variables of the families constituting the subset $\mathcal{F}_{\mathcal{B}}^{\neg b}$.

Algorithm 3 Fractional Relax-and-Fix on boxes

```
$r$=0
while $r < \frac{n-v}{q}$ do
    for $i = 1$ to $q \times r$ do
        selVar[i] $\leftarrow$ results[i]
    end for
    for $i = q \times r + 1$ to $\min\{q \times r + u, n\}$ do
        add(var[i])
        add(selVar[i])
    end for
    for $i = \min\{q \times r + v + 1, n\}$ to $\min\{q \times r + u, n\}$ do
        selVar[i].integralityRestrictionRelaxed
    end for
    solveMILP
    if MILPHasASolution then
        for $i = q \times r + 1$ to $\min\{q \times r + q, n\}$ do
            results[i] $\leftarrow$ value(selVar[i])
        end for
        $r \leftarrow r + 1$
    else
        heuristicFailed - STOP
    end if
end while
HeuristicSolutionFound
```

4.4 General remarks

Backtracking As in Baena et al. (2015), a backtrack process has been developed to overcome possible failures of the algorithms. Indeed, it may happen that a heuristic fails at one step, especially when the integrality restriction is removed and then reinserted. This does not mean the original problem is infeasible and this failure does not bring any information about the suitability of this algorithm, except that some variables cannot be fixed to those values at that step. Backtracking works as follows: if a failure occurs at iteration r, the MILP^r is solved again, but without fixing the integer variables with index in Q^{r-1}. If the heuristic fails again, then we backtrack once more, solving MILP^r without fixing the integer variables with index in Q^{r-1} and in Q^{r-2}. We can proceed that way $r-1$ times, ending with just a relaxation of the initial MILP, which is supposed to be feasible.

Additional ULDs One advantage of the matheuristics in comparison to a B&B resolution is the number of considered ULDs. When using the B&B technique, the number of proposed ULDs is fixed and initially given. The problem is that, on one hand, too many ULDs increase the computational times since the number of constraints and variables grows. On the other hand, too few ULDs can miss a potentially better loading. The heuristics developed here do not encounter this problem. Indeed, because they are iterative procedures, new ULDs can be added at each iteration. In detail, at the beginning of the procedure an initial ULD set is proposed and at the end of each iteration, if there is a (or several) new used ULD(s), then a copy of the new used ULD(s) is created and added to the list of proposed ULDs.

4.5 Preliminary results

4.5.1 Parameter ranges

These three types of matheuristics have different parameters which would need to be tuned in order to get the best possible solutions with respect to the objective function. The assignment of values to each parameter is called a *configuration*. This is achieved in Section 6.2 using `irace` parametrisation technique.

However, some preliminary experiments have been carried out on small samples of 30 instances containing 7, 8, 9 and 10 boxes. More details about the data will be provided in Section 6.1. The three types of constructive heuristics have been implemented in Java, using IBM ILOG CPLEX 12.5 library as B&B solver for the subproblems, under the default parametrisation. Tests were carried out on a personal computer (Windows 8, Intel Core i7, 2.40 GHz, 8.00 GB of RAM) and the time limit for each heuristic run is one hour. In this first test phase, the possibility to add ULDs during the runs is not considered.

Relax-and-Fix heuristic

Table 4.5 contains the results of several configurations of the R&F heuristics. As a reminder, n is the number of boxes for each instance, u is the number of boxes with unrelaxed integer variables and q is the number of boxes for which we keep the values of the variables for the next iterations. For each block, each value of n and each configuration, the average computational times, in seconds, over the 30 instances are provided. Note that it may happen that some instances cannot be solved within one hour. Let w denote the

Table 4.5: Average computational times, in seconds, over the 30 instances of each size for different configurations of the Relax-and-Fix heuristics

	$n=7$	$n=8$	$n=9$	$n=7$	$n=8$	$n=9$
		$p_{\mathtt{ij}}$			Coord & $p_{\mathtt{ij}}$	
$u=2;q=1$	$13.36^{(1)}$	73.83	$151.18^{(4)}$	73.31	66.64	$73.46^{(4)}$
$u=3;q=1$	$14.19^{(1)}$	$98.59^{(1)}$	$140.23^{(3)}$	$12.97^{(1)}$	112.19	$175.38^{(3)}$
$u=3;q=2$	40.66	$86.75^{(1)}$	$140.86^{(3)}$	$13.70^{(1)}$	133.45	$170.41^{(3)}$
$u=4;q=1$	$11.30^{(1)}$	55.61	$283.76^{(3)}$	$13.67^{(1)}$	$79.36^{(1)}$	$174.18^{(3)}$
$u=4;q=2$	$11.15^{(1)}$	45.97	$277.35^{(3)}$	$13.55^{(1)}$	$79.04^{(1)}$	$171.86^{(3)}$
$u=4;q=3$	$11.15^{(1)}$	45.99	$275.27^{(3)}$	$13.51^{(1)}$	173.08	$191.67^{(3)}$
		Coord			All $\mathtt{i}$	
$u=2;q=1$	93.05	51.87	$164.56^{(3)}$	$10.85^{(1)}$	107.00	$79.79^{(2)}$
$u=3;q=1$	54.66	71.63	$314.06^{(3)}$	$14.77^{(1)}$	56.49	$161.67^{(2)}$
$u=3;q=2$	51.67	69.81	$308.97^{(3)}$	$12.71^{(1)}$	$43.04^{(1)}$	$286.04^{(1)}$
$u=4;q=1$	55.36	57.97	$182.01^{(2)}$	$13.15^{(1)}$	225.51	$155.61^{(3)}$
$u=4;q=2$	52.99	57.26	$177.95^{(2)}$	$14.53^{(1)}$	45.76	$166.15^{(4)}$
$u=4;q=3$	53.78	56.94	$178.15^{(2)}$	$9.56^{(1)}$	237.81	$180.81^{(3)}$
B&B	$15.67^{(1)}$	64.43	$114.04^{(5)}$	$15.67^{(1)}$	64.43	$114.04^{(5)}$

Note. The notation $\cdot^{(w)}$ means that the heuristic with that configuration could not find a solution to the original problem within one hour for w out of the 30 instances. The average computational times are then calculated on the $30-w$ remaining instances. The last row contains the results obtained with a direct B&B resolution.

number of unsolved instances. In that case, the average computational time is calculated over the remaining $30-w$ solved instances.

As expected, the larger the value of n, the larger the computational times. From this table, we can already see that for several instances with 9 boxes, these heuristics may not find a feasible solution within one hour. Looking at the computational times of both the heuristic and the B&B (last row in Table 4.5), they are sometimes larger for the heuristics. It is useless to try configurations with larger values of u, the subproblems would then be more considerable and the computational times would tend to increase. Considering the long computational times, there seems little advantage in looking deeper at this method; apparently, it does not suit this kind of problem.

Insert-and-Fix heuristic

Table 4.6 shows the average computational times of the I&F algorithm. Obviously, computational time increases as the number of boxes increases or as the value of u increases. Still, the average computational time is rather reasonable and dramatically smaller than for the B&B, unlike the R&F heuris-

Table 4.6: Average computational times, in seconds, over the 30 instances of each size for different configurations of the Insert-and-Fix heuristics

	$n = 7$	$n = 8$	$n = 9$	$n = 10$
$u = 2; q = 1$	0.28	0.36	0.47	0.57
$u = 3; q = 1$	0.32	0.43	0.53	0.68
$u = 3; q = 2$	0.41	0.36	0.44	0.58
$u = 4; q = 1$	0.42	0.52	0.65	0.90
$u = 4; q = 2$	0.37	0.42	0.54	0.74
$u = 4; q = 3$	0.34	0.41	0.53	0.61
$u = 5; q = 1$	1.05	0.72	0.84	1.36
$u = 5; q = 2$	0.90	0.64	0.68	0.96
$u = 5; q = 3$	0.87	0.60	0.70	0.91
$u = 6; q = 1$	7.52	2.45	3.46	2.95
$u = 6; q = 2$	4.78	2.32	3.60	4.22
$u = 6; q = 3$	4.74	2.29	2.86	3.24
$u = 7; q = 1$		10.01	16.43	13.41
$u = 7; q = 2$		9.98	15.67	12.58
$u = 7; q = 3$		9.92	15.46	11.23
$u = 8; q = 1$			$209.92^{(1)}$	208.12
$u = 8; q = 2$			$208.76^{(1)}$	206.51
B&B	$15.67^{(1)}$	64.43	$114.04^{(5)}$	$519.89^{(13)}$

Note. The notation $\cdot^{(w)}$ means that the heuristic with that configuration could not find a solution to the original problem within one hour for w out of the 30 instances. The average computational times are then calculated on the $30 - w$ remaining instances. The last row contains the results obtained with a direct B&B resolution.

tics. However, some instances needed more than one hour to be solved for $u = 8$. Therefore, parameter u will be tuned in the range $[2, 7]$. Intuitively, the larger u, the closer to the original problem, and a small value of q allows more flexibility from one step to another potentially leading to better solutions. Moreover, a large value of q means that a lot of variables in each subproblem are fixed and it could increase the time to find a feasible solution. For these reasons, parameter q is considered in the range $[1, 3]$.

Fractional Relax-and-Fix heuristic

The FRF heuristics have three parameters for each configuration, compared with only two for the previous heuristics.

We can observe that the four FRF heuristics have reasonable computational times in comparison to times obtained with the B&B.

Almost the same conclusions may be drawn for the two blocks Coord and Coord & p_{ij} when the preliminary results in Table 4.7 and Table 4.8 are observed. One can see that some configurations have difficulties finding solutions within one hour even for 7 boxes. There is no configuration with $u = 8$ that works for the instances with 9 or 10 boxes. Therefore, the parameter u will be tuned in the range $[3, 7]$. The parameters v and q have to be such that $q < v < u$ and thus v will be in the range $[2, 6]$. As explained in the I&F parameters selection, the parameter q is considered in the range $[1, 3]$. About the blocks p_{ij} and All i, Table 4.7 and Table 4.8 show that they are not able to give a solution to the original problem within one hour of computational time even for a small number of boxes. Considering the long computational times, there is little advantage in looking deeper at these two methods, as for the R&F.

4.5.2 Influence of suboptimal resolution

As can be seen in Section 4.5.1, the computational times can become very large even for instances with a small number of boxes to pack. These important computational times are often due to the search of the optimal solution of each subproblem. A feasible solution can sometimes be quickly obtained, but its improvement or even just the proof of its optimality can be very time consuming. A method to speed up the process could be to accept suboptimal solutions for the subproblems.

For each instance, the whole algorithm has a maximum computational time of one hour. In this variant, every subproblem also has a maximum time limit, `timeLimitPerStep`, for its resolution, possibly leading to a suboptimal solution. For each subproblem resolution, one has:

- before the limit `timeLimitPerStep` is reached, the resolution may stop because:
  - an optimal solution has been found $\rightarrow$ kept and next step,
  - the subproblem is infeasible:
    - if there is still available time for computation, then the algorithm backtracks,
    - if there is no more available time, then the heuristic has failed;
- after `timeLimitPerStep`, the resolution is stopped:
  - if there is a feasible solution $\rightarrow$ kept and next step,
  - if there is no solution found,

Table 4.7: Average computational times, in seconds, over the 30 instances of each size for different configurations of the Fractional Relax-and-Fix heuristics with the blocks p_{ij} and Coord

	FRF p_{ij}				FRF Coord			
	$n = 7$	$n = 8$	$n = 9$	$n = 10$	$n = 7$	$n = 8$	$n = 9$	$n = 10$
$u = 3$								
$v = 2; q = 1$	0.45	0.67			0.44	0.52		
$u = 4$								
$v = 2; q = 1$	0.84	0.90			0.61	0.70		
$v = 3; q = 1$	0.91	0.88			0.60	0.61		
$v = 3; q = 2$	9.71	0.64			0.46	0.48		
$u = 5$								
$v = 2; q = 1$	2.58	3.81			1.73	1.07		
$v = 3; q = 1$	2.51	1.59			1.63	1.28		
$v = 3; q = 2$	1.09	1.34			1.08	1.01		
$v = 4; q = 1$	5.33	2.94			1.52	1.01		
$v = 4; q = 2$	1.00	1.64			1.25	0.78		
$v = 4; q = 3$	0.86	0.96			0.97	0.78		
$u = 6$								
$v = 3; q = 1$	4.44$^{(1)}$	54.47	11.52$^{(1)}$	117.82	17.70	3.78	3.95	4.88
$v = 3; q = 2$	4.28	5.62	13.45	41.03$^{(1)}$	4.33	2.86	3.16	3.29
$v = 4; q = 1$	3.47$^{(1)}$	6.47	13.94	9.06$^{(1)}$	8.39	4.18	5.66	9.10
$v = 4; q = 2$	6.03	5.72	40.43	6.86	5.24	2.97	3.62	5.53
$v = 4; q = 3$	5.96	4.01	3.10	11.67$^{(1)}$	5.08	2.90	3.25	3.77
$v = 5; q = 1$	3.10$^{(1)}$	6.07	4.41$^{(1)}$	5.03	8.79	3.29	10.48	4.59
$v = 5; q = 2$	5.35	3.69	2.72$^{(1)}$	4.04	4.24	2.85	3.66	2.88
$v = 5; q = 3$	5.24	3.12	2.68	3.30	4.31	2.63	9.88	2.76
$u = 7$								
$v = 4; q = 1$		22.05	57.73$^{(1)}$	39.56$^{(3)}$		10.34	102.68	18.26
$v = 4; q = 2$		13.81	14.74$^{(1)}$	60.89		9.82	11.50$^{(1)}$	44.69
$v = 4; q = 3$		13.64	13.45	75.26$^{(2)}$		9.84	11.83	16.64
$v = 5; q = 1$		24.44	29.62$^{(1)}$	23.64		8.95	30.09$^{(1)}$	25.03
$v = 5; q = 2$		11.77	32.51$^{(1)}$	16.34		8.31	133.61	31.80
$v = 5; q = 3$		15.26	35.42	20.60$^{(1)}$		8.21	19.77	13.68
$v = 6; q = 1$		19.47	12.28$^{(2)}$	74.65$^{(1)}$		20.11	67.32	19.17
$v = 6; q = 2$		13.06	8.42$^{(2)}$	70.55		19.36	39.89	14.67
$v = 6; q = 3$		10.40	9.37$^{(1)}$	15.06$^{(1)}$		18.98	11.15	12.10
$u = 8$								
$v = 4; q = 1$			94.14$^{(1)}$	151.61$^{(4)}$			100.76$^{(2)}$	91.46$^{(3)}$
$v = 4; q = 2$			91.41$^{(1)}$	315.58$^{(4)}$			162.17$^{(1)}$	304.32$^{(1)}$
$v = 4; q = 3$			96.90	137.37$^{(3)}$			117.33$^{(1)}$	299.22$^{(1)}$
$v = 5; q = 1$			163.92$^{(1)}$	327.76$^{(1)}$			161.73$^{(1)}$	194.21$^{(1)}$
$v = 5; q = 2$			112.69$^{(1)}$	196.12$^{(2)}$			158.93$^{(1)}$	185.37$^{(1)}$
$v = 5; q = 3$			149.59	256.67$^{(1)}$			157.27$^{(1)}$	185.15
$v = 6; q = 1$			184.79$^{(1)}$	416.88$^{(2)}$			96.92$^{(2)}$	327.45
$v = 6; q = 2$			78.99$^{(1)}$	245.51$^{(2)}$			60.58$^{(2)}$	302.45
$v = 6; q = 3$			70.98$^{(1)}$	262.57$^{(1)}$			59.72$^{(2)}$	302.63
B&B	15.67$^{(1)}$	64.43	114.04$^{(5)}$	519.89$^{(13)}$	15.67$^{(1)}$	64.43	114.04$^{(5)}$	519.89$^{(13)}$

Note. The notation $\cdot^{(w)}$ means that the heuristic with that configuration could not find a solution to the original problem within one hour for w out of the 30 instances. The average computational times are then calculated on the $30 - w$ remaining instances. The last row contains the results obtained with a direct B&B resolution.

Table 4.8: Average computational times, in seconds, over the 30 instances of each size for different configurations of the Fractional Relax-and-Fix heuristics with the blocks Coord & p_{ij} and All i

	FRF Coord & p_{ij}				FRF All i			
	$n=7$	$n=8$	$n=9$	$n=10$	$n=7$	$n=8$	$n=9$	$n=10$
$u=3$								
$v=2; q=1$	0.40	0.45			1.42	1.02		
$u=4$								
$v=2; q=1$	0.67	0.99			11.46	48.43		
$v=3; q=1$	0.56	0.57			2.04	1.49		
$v=3; q=2$	0.40	0.45			1.10	2.33		
$u=5$								
$v=2; q=1$	3.01	1.69			7.18	62.38		
$v=3; q=1$	1.34	1.06			8.20	21.86		
$v=3; q=2$	1.83	0.87			4.58	11.93		
$v=4; q=1$	1.44	0.96			4.52	10.25		
$v=4; q=2$	1.15	0.83			3.76	89.98		
$v=4; q=3$	0.97	0.98			5.44	14.01		
$u=6$								
$v=3; q=1$	48.24	3.31	4.67	5.35	17.87	21.41$^{(1)}$	109.66$^{(2)}$	93.31$^{(2)}$
$v=3; q=2$	7.21	3.30	10.72	4.69	8.55	159.95	32.53$^{(3)}$	81.01$^{(3)}$
$v=4; q=1$	10.90	4.56	4.26	4.78$^{(1)}$	33.47	12.68	23.84$^{(3)}$	21.06$^{(2)}$
$v=4; q=2$	4.46	5.83	3.79	54.64	9.70	16.20	58.41$^{(2)}$	91.92$^{(1)}$
$v=4; q=3$	4.41	4.02	3.83	36.96$^{(1)}$	7.50	19.15	82.69$^{(2)}$	23.68$^{(1)}$
$v=5; q=1$	5.62	3.40	14.24	3.66	9.08	111.03$^{(1)}$	15.71	9.35
$v=5; q=2$	3.37	3.14	2.78	3.28	5.29	16.82	7.54	19.40
$v=5; q=3$	3.39	3.03	18.26	3.69	4.84	15.04	6.05$^{(1)}$	129.79$^{(1)}$
$u=7$								
$v=4; q=1$		10.56	55.49	25.38		269.92	338.49$^{(1)}$	74.88$^{(2)}$
$v=4; q=2$		9.90	88.44	33.01		120.00	342.93$^{(2)}$	44.53$^{(4)}$
$v=4; q=3$		9.84	14.20	31.32$^{(1)}$		126.32	165.53$^{(1)}$	137.26$^{(6)}$
$v=5; q=1$		10.21	37.12	15.77		36.64	32.67$^{(3)}$	54.78$^{(3)}$
$v=5; q=2$		9.49	8.48$^{(1)}$	13.37		37.60	68.15$^{(2)}$	74.67$^{(4)}$
$v=5; q=3$		9.41	10.70	13.20		121.82	160.41$^{(1)}$	40.33$^{(6)}$
$v=6; q=1$		9.76	18.59$^{(1)}$	12.88		18.59	125.91$^{(2)}$	141.93
$v=6; q=2$		9.15	110.81	11.58		14.33	130.87$^{(2)}$	138.93
$v=6; q=3$		9.14	13.42	11.64		13.46	59.61$^{(1)}$	25.97$^{(1)}$
$u=8$								
$v=4; q=1$			65.81$^{(1)}$	240.27$^{(1)}$			112.70$^{(5)}$	242.88$^{(4)}$
$v=4; q=2$			89.75	202.99$^{(2)}$			278.30$^{(3)}$	159.11$^{(2)}$
$v=4; q=3$			89.01	185.43$^{(2)}$			157.58$^{(2)}$	82.78$^{(5)}$
$v=5; q=1$			106.39$^{(2)}$	180.47			57.06$^{(4)}$	101.18$^{(3)}$
$v=5; q=2$			104.27$^{(2)}$	174.98			179.11$^{(2)}$	424.19$^{(5)}$
$v=5; q=3$			103.81$^{(2)}$	175.48			140.94$^{(2)}$	156.84$^{(5)}$
$v=6; q=1$			107.66$^{(1)}$	194.75$^{(1)}$			200.80$^{(2)}$	100.59$^{(2)}$
$v=6; q=2$			155.49	192.55$^{(1)}$			119.04$^{(3)}$	127.41$^{(2)}$
$v=6; q=3$			130.19	192.32$^{(1)}$			130.34$^{(2)}$	280.14$^{(3)}$
B&B	15.67$^{(1)}$	64.43	114.04$^{(5)}$	519.89$^{(13)}$	15.67$^{(1)}$	64.43	114.04$^{(5)}$	519.89$^{(13)}$

Note. The notation $\cdot^{(w)}$ means that the heuristic with that configuration could not find a solution to the original problem within one hour for w out of the 30 instances. The average computational times are then calculated on the $30 - w$ remaining instances. The last row contains the results obtained with a direct B&B resolution.

* $*$ if there is still available time for computation, then the resolution continues until a feasible solution has been found or the time has elapsed,
* $*$ if there is no more available time, then the heuristic has failed.

The time limit per step is based on the parameters of the matheuristics. For the R&F and the I&F heuristics, the total number of steps in the algorithm is $\lceil \frac{n-u}{q} \rceil$. The available time, 60 minutes, is uniformly divided over the steps: each step has a maximum time limit equal to

$$\texttt{timeLimitPerStep} = \frac{60}{\lceil \frac{n-u}{q} \rceil}\text{minutes.} \tag{4.5}$$

This choice is reasonable because if parameter u is large, then the subproblems to solve are larger and the computational times can be more important. Conversely, if parameter u is small, there are a large number of small subproblems to be solved and a small amount of time can be sufficient to find a solution.

This extension has been applied to the R&F with the block p_{ij} and to the I&F and experiments have been carried out using IBM ILOG CPLEX 12.6 library as B&B solver with default parametrisation.

R&F with the block p_{ij} In Table 4.5, one can see that even the instances with 9 boxes can be unsolved within one hour of computational times regardless of the parameter values. The extension of the R&F with the block p_{ij} with a limited duration for the possibly suboptimal resolution of the subproblems is tested on the 30 instances with 9 boxes with the parameters $u = 5$ and $q = 1$. With this configuration, the original MILP is decomposed into 5 subproblems to be solved. With the equation (4.5), each subproblem has a maximum time limit equal to 720 seconds to be solved with the suboptimal R&F with the block p_{ij}. Table 4.9 shows the objective function values and the computational times (in seconds) for the original R&F and the suboptimal R&F with the block p_{ij}. The last column indicates the steps for which the subproblems have been suboptimally solved because of the time limit. One can see that the six instances unsolved with the original R&F have a solution with the suboptimal R&F. Out of the 24 remaining instances, for 19 instances, the resolutions are identical for both methods (each iteration naturally lasts less than 720 seconds) and for 5 instances, the resolution with the suboptimal R&F with the block p_{ij} is clearly faster but the same objective function values are obtained. One can also observe that the iteration that takes a lot of time is often the first iteration. The last row shows the average computational times over the 30 instances. The suboptimal R&F is faster than the original R&F and is then able to find a solution for all the instances.

I&F As can be seen in Table 4.6, the I&F can already have some difficulties to find solutions for sets of 9 boxes within one hour of computational times when $u = 8$. For smaller parameter values, the heuristic is fast and the improvement brought by this extension can be missed. For this reason, experiments are carried out with the configuration $u = 8$ and $q = 1$ on a set of 30 instances, each instance having 30 boxes to be packed. In this case, the algorithm splits the problem into 23 subproblems, i.e., 23 iterations. Considering the equation (4.5), each subproblem has 156 seconds to be solved with the suboptimal I&F heuristic. Table 4.10 shows the objective function values and the computational times (in seconds) for the original I&F and the suboptimal I&F. The last column indicates the steps for which the subproblems have been suboptimally solved because of the time limit. One can see that the three instances unsolved with the original I&F have a solution with the suboptimal I&F. Moreover, six instances have the same objective function values with both heuristics but faster with the suboptimal I&F than with the original I&F. Finally, only one instance has a larger objective function value with the suboptimal I&F than with the original I&F. The last row shows the average computational times over the 30 instances. The suboptimal I&F is clearly faster than the original I&F as expected.

Based on these results, the parameters for the suboptimal I&F may possibly take other values than those selected in Section 4.5.1. More computational experiments should be carried out to validate this assumption and to measure the loss of quality with such extension.

A hint to improve this alternative would be to change the parameters of CPLEX to emphasise the search of feasible solutions early in the B&B (`MIP emphasis switch`).

4.6 Areas for improvement

Box sorting In each heuristic, boxes are sorted by non-increasing volume. Other sorting operators could be applied to the list of boxes.

Table 4.9: Comparison of the original R&F and the suboptimal R&F with the block p_{ij}

Inst.	Original R&F		Suboptimal R&F		
	Obj. fct.	Duration [s.]	Obj. fct.	Duration [s.]	Suboptimal steps
1	5	79.92	5	74.91	
2	9.1	33.30	9.1	30.86	
3	-	3600.33	12.4	721.47	step 0
4	12.4	2460.87	12.4	722.32	step 0
5	31.1	19.84	31.1	17.80	
6	7.4	46.21	7.4	42.32	
7	-	3600.32	12.4	938.36	step 0
8	16.5	1352.80	16.5	740.32	step 0
9	12.4	425.55	12.4	400.76	
10	31.1	28.99	31.1	26.68	
11	-	3600.36	12.4	721.49	step 0
12	12.4	3038.51	12.4	743.40	step 0
13	9.1	110.76	9.1	101.60	
14	-	3600.34	16.5	1425.74	step 0
15	5	62.76	5	54.93	
16	-	3600.12	16.5	1249.29	step 0
17	7.4	26.55	7.4	25.12	
18	19.1	1927.65	19.1	1097.71	step 0
19	16.5	249.67	16.5	235.66	
20	7.4	31.57	7.4	29.17	
21	7.4	20.79	7.4	18.89	
22	31.1	16.12	31.1	14.76	
23	31.1	3.24	31.1	2.73	
24	31.1	29.57	31.1	26.32	
25	12.4	262.30	12.4	251.39	
26	7.4	2902.12	7.4	734.60	step 0
27	-	3600.29	7.4	720.80	step 0
28	7.4	100.61	7.4	91.08	
29	31.1	157.63	31.1	148.93	
30	12.4	45.50	12.4	41.84	
Avg.		1167.82		381.71	

Table 4.10: Comparison of the original I&F and the suboptimal I&F

Inst.	Original I&F		Suboptimal I&F		
	Obj. fct.	Duration [s.]	Obj. fct.	Duration [s.]	Suboptimal steps
1	31.1	14.74	31.1	14.24	
2	38.5	47.02	38.5	45.58	
3	31.1	21.91	31.1	20.70	
4	50	48.23	50	46.90	
5	23.9	624.04	23.9	423.73	steps 3, 5
6	26.3	235.55	26.3	233.58	
7	31.3	339.03	31.3	330.24	step 5
8	28.9	377.91	28.9	230.21	step 4
9	28	22.92	28	21.80	
10	-	3599.55	23.9	1228.62	steps 1-3, 5-8
11	18.9	35.99	18.9	33.35	
12	23.9	2268.50	26.3	395.15	steps 4, 5
13	28	26.82	28	24.76	
14	18.9	33.42	18.9	32.35	
15	62.2	77.49	62.2	76.00	
16	-	3596.99	41.1	1041.79	steps 6-11
17	45.4	1878.14	45.4	521.69	steps 2, 4
18	43.5	91.33	43.5	85.74	
19	-	3597.00	36.1	374.01	steps 7, 8
20	62.2	22.06	62.2	19.75	
21	26.3	409.04	26.3	246.96	step 5
22	37.8	895.15	37.8	617.49	steps 2-4
23	50	27.90	50	25.76	
24	18.9	48.04	18.9	44.88	
25	18.9	72.53	18.9	69.73	
26	23.9	71.97	23.9	68.36	
27	31.1	36.13	31.1	33.93	
28	27.4	146.44	27.4	138.70	
29	18.9	35.41	18.9	33.45	
30	23.9	58.72	23.9	55.30	
Avg.		625.33		217.82	

Chapter 5

A tailored two-phase constructive heuristic

The matheuristics developed in the previous chapter provide solutions in shorter computational times than those obtained with the B&B resolution of the linear formulation from Chapter 3. However, these matheuristics are still limited for instances with a large number of boxes as will be shown in Chapter 6. Therefore, this fifth chapter proposes a fast algorithm to build an initial solution for the specific three-dimensional MBSBPP described in Chapter 2. This algorithm is composed of two main phases. First, a packing algorithm, described in Section 5.1, provides a loading pattern for a given set of boxes and a given ULD type. This packing process is based on the Extreme Points introduced in Crainic et al. (2008). The different types of proposed ULDs and their selection represent the second phase of the algorithm, using the packing algorithm as a subroutine (Section 5.2). Finally, among the feasible loading patterns created during the second phase, the pattern with the minimum used volume is selected and if necessary is enhanced in terms of weight distribution as explained in Section 5.3. Section 5.4 provides several areas for improvement for this method.

The methodology presented in this fifth chapter has been submitted for publication in March 2017 as *Paquay, C., S. Limbourg and M. Schyns (2017). A tailored two-phase constructive heuristic for the three-dimensional Multiple Bin Size Bin Packing Problem with transportation constraints.*

The current position in the thesis outline is shown in bold in Figure 5.1.

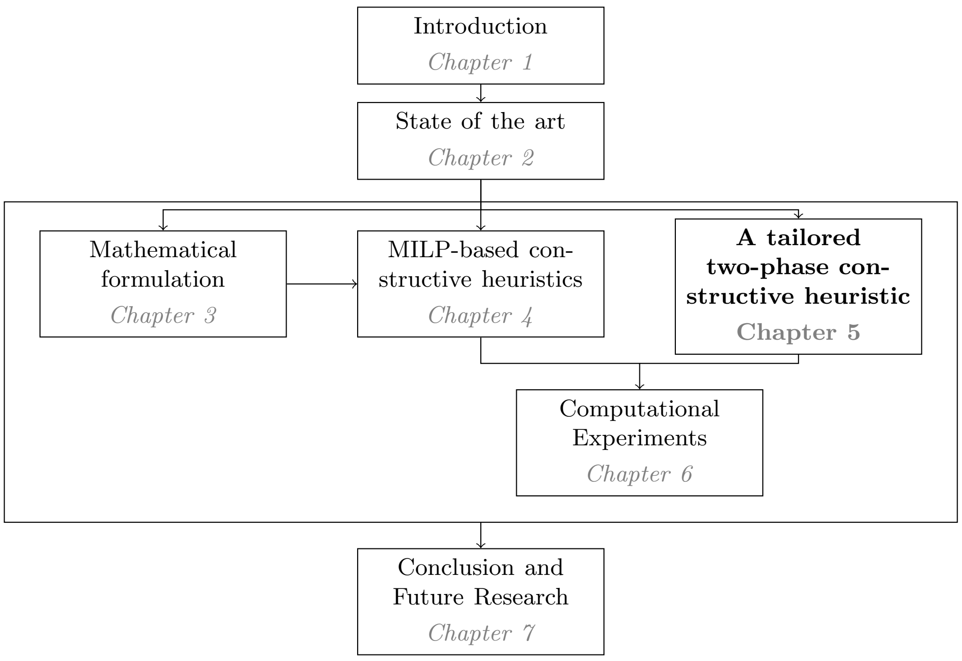

Introduction
Chapter 1
State of the art
Chapter 2
Mathematical formulation
Chapter 3
MILP-based constructive heuristics
Chapter 4
A tailored two-phase constructive heuristic
Chapter 5
Computational Experiments
Chapter 6
Conclusion and Future Research
Chapter 7

Figure 5.1: Current position in the thesis outline

5.1 Packing algorithm: Phase 1

In this section, an algorithm is developed to provide a loading pattern for a given set of boxes and a given ULD type. The ULD is available in an unlimited quantity and the aim is to use its volume in the most efficient manner.

Among the possible constructive heuristics, two algorithms represent a promising perspective as they showed interesting results in terms of worst-case performance ratio for the one-dimensional Bin Packing Problem (Martello and Toth (1990)). The First Fit Decreasing (FFD) algorithm first sorts the list of items by non-increasing size and then considers each item one after another. It assigns each item to the lowest indexed bin into which it fits. If there is no such bin, then a new bin is created. The Best Fit Decreasing (BFD) algorithm assigns an item to the feasible bin (if any) having the smallest residual capacity. If there is no such bin, then a new bin is created.

However, the adaptation of these algorithms for the three-dimensional case is far from simple: several rules for sorting the items and the bins exist and placing a box in a bin can also be achieved in different ways. This challenge was taken up by Crainic et al. (2008) who considered the three-dimensional Single Bin Size Bin Packing Problem in which items cannot be rotated or overlap. They raised another important point: the choice of placement points is a major challenge in multi-dimensional C&P problems. Indeed, the space utilisation and the solution quality are highly influenced by the item-positioning rule. This issue is particularly crucial and difficult to manage for three-dimensional situations. For this reason, Crainic et al. (2008) developed an *Extreme Point* (EP)-based rule for packing items inside a three-dimensional container. The EPs represent the relevant possible positions to accommodate items and are an extension of the *Corner Points* introduced in Martello et al. (2000). This new EP rule has been created to be efficient with regard to computational effort and volume utilisation. After extensive experiments, the authors selected a BFD algorithm with a specific sorting rule for the items as well as an EP selection rule.

The packing algorithm proposed in this chapter is based on the EP defined in Crainic et al. (2008). However, several extensions are provided to consider all the constraints of the specific MBSBPP studied in this thesis. Namely, the rotations, the possible fragility and the stability of the boxes as well as the particular shape of some ULDs are taken into account.

As explained in Chapter 3, each box i has a length l_i, a width w_i, a height h_i, and a weight m_i, all these dimensions being integers. The position of the box i in the loading pattern is described by its front left bottom vertex (x_i, y_i, z_i) and its top right rear vertex (x'_i, y'_i, z'_i) in a system as described in Figure 5.2. In this section, identical ULDs are available to pack the set of boxes.

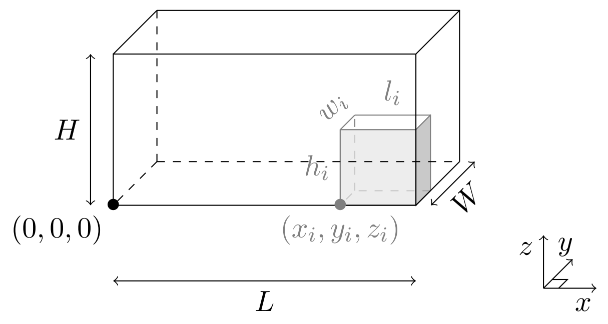
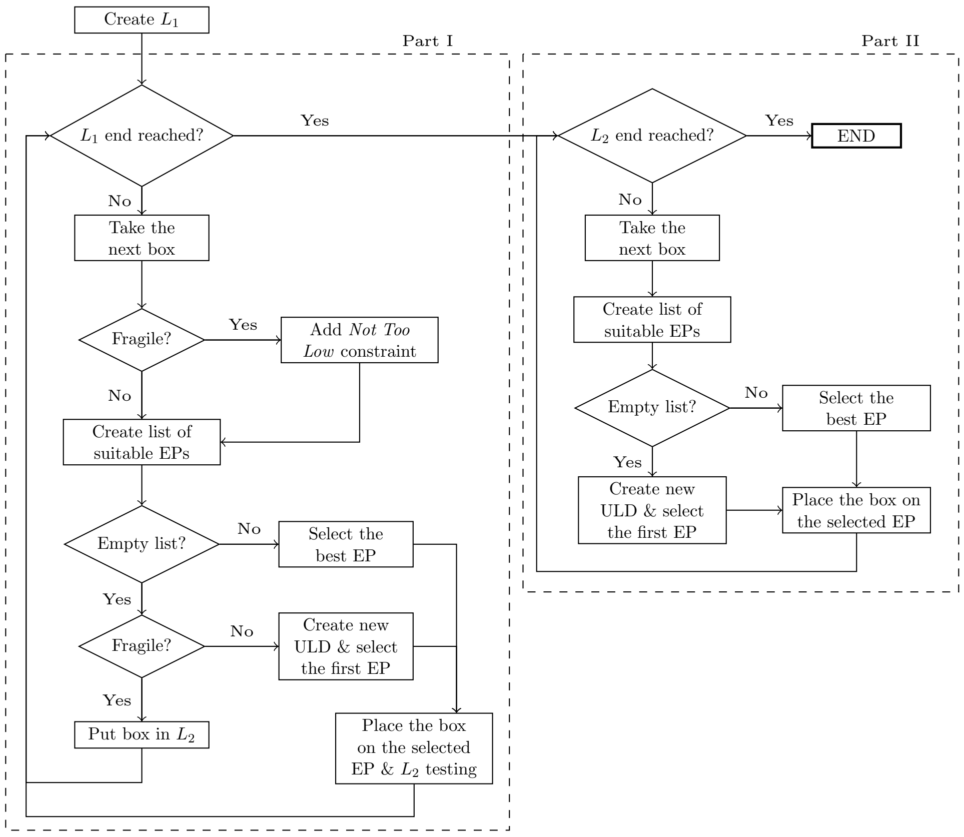


Figure 5.2: The coordinate system associated to a bin and the coordinates of a box i

The possible fragility of the boxes needs special care in this algorithm. As a reminder, no other box can be packed on top of a fragile box. Therefore, if the top face of a fragile box is low in the ULD, then a consequent volume above this becomes unusable for packing other boxes. For this reason, two lists of boxes are used and the packing algorithm is divided into two major parts as represented in Figure 5.3. L_1 is the main list containing all the boxes with their best orientation. The best orientation of a non-fragile box is selected with the Clustered Area-Height rule from Crainic et al. (2008). If the box is fragile, the best orientation is the orientation with the minimum top face area in order to minimise the unusable space. This criterion leads to two possible orientations. If the considered ULD type has a length larger (resp. smaller) than the width, then the box orientation with the largest length (resp. width) is selected. First, the boxes from L_1 are handled one after another. During this first part, a fragile box cannot be packed too low in a ULD because it would lead to a waste of volume above its top face. This is called the *Not Too Low* constraints in the following. If a fragile box cannot be packed, then it is added to L_2 (Part I). Second, once all the boxes from L_1 have been packed or assigned to L_2, all the fragile boxes from L_2 are loaded at any possible height (Part II).

To select the location of a box, possible interesting positions are described by the EPs. Thus, at any time of the procedure, a global list contains all the EPs existing in all the ULDs. At each iteration, a box (from L_1 or L_2) is examined and tried to be packed. To select the position for packing the box, each EP of the global list is tested to check whether the box with this orientation can be placed with its left front bottom vertex on that EP while satisfying several constraints. If it is possible, the EP is suitable for that box. From the list of suitable EPs, the best EP according to a merit function defined in Section 5.1.2 is selected to accommodate the considered box. However, if the list of suitable EPs is empty, which means that the current box cannot be packed anywhere, then a new bin is created unless the box is fragile, this box would then be added to L_2. Indeed, opening a new ULD and packing a fragile box as the first box would lead to a important waste of volume. Therefore, these unpacked fragile boxes are differently handled. Every time a box is packed, new EPs are generated since new positions become conceivable. These new EPs can be suitable for the fragile boxes already assigned to L_2. For this reason, these EPs are analysed for the current boxes of L_2 and used to accommodate any of these if possible. This process is called L_2 *testing* as mentioned in Figure 5.3 and is explained in more detail in Figure 5.7.

Each part of the algorithm is now detailed.



Create L_1
Part I
Part II
L_1 end reached?
Yes
L_2 end reached?
Yes
END
No
Take the next box
No
Take the next box
Create list of suitable EPs
Fragile?
Yes
Add *Not Too Low* constraint
No
Create list of suitable EPs
Empty list?
No
Select the best EP
Yes
Create new ULD & select the first EP
Place the box on the selected EP
Empty list?
No
Select the best EP
Yes
Fragile?
No
Create new ULD & select the first EP
Yes
Put box in L_2
Place the box on the selected EP & L_2 testing

Figure 5.3: Packing algorithm

5.1.1 Creating L_1

Building the list The first step of the packing algorithm consists in building the list of boxes to take into account the possibility of orthogonal rotations and the possible fragile boxes. In more detail, a box initially has six different orientations, but it may happen that it is not authorised to be packed in some orientations due to its contents as explained in Chapter 2. Some orientations may not be feasible because the box would not fit in the proposed ULD type.

To check if a box with a given orientation is feasible for a ULD type, this oriented box is tried to be packed on the *first EP* of an empty ULD. The *first EP* of a ULD is located on the front left bottom vertex, that is $(0, 0, 0)$ if the ULD has no type 1 cut, $(\frac{b_1}{a_1}, 0, 0)$ otherwise[1]. Each of the authorised orientations is analysed and checked for whether the box with this orientation can be packed on the first EP. A list with all the authorised and feasible orientations, i.e., the possible oriented boxes, is created. This list thus contains between zero and six elements. If the list is empty, it means that no orientation is acceptable and it is thus impossible to pack the box even in an empty ULD. This box (with all its orientations) is kept for a different ULD type. If there is only one oriented box in the list, then this oriented box is added to L_1. If the list has more than one oriented box, then the best orientation, which depends on the fragility of the box, is selected and the corresponding oriented box is added to L_1. If the box is fragile, the best orientation is the orientation with the minimum top face area. This criterion leads to two possible orientations. If the considered ULD type has a length larger (resp. smaller) than the width, then the orientation with the largest length (resp. width) is selected as previously explained. If the box is not fragile, the best orientation is obtained with the *Clustered Area-Height* sorting rule from Crainic et al. (2008).

The *Clustered Area-Height* rule works as follows: the set of boxes to pack is partitioned into clusters, each box being assigned to a cluster according to the size of its base area. Each cluster $A_{j,\delta}$ corresponds to a proportion δ, with $\delta \in [0.01, 1]$, of the bin base area $L \times W$:

$$A_{j,\delta} = \{\text{boxes } i \mid l_i \times w_i \in](j-1) \times L \times W \times \delta, j \times L \times W \times \delta]\}.$$

The value of δ represents the proportion of the base area. The number of clusters is thus equal to $\lceil \frac{1}{\delta} \rceil$.

For example, if $\delta = 0.3$, then the set of boxes is partitioned into four different clusters:

[1]As described in Section 3.4.2, a type 1 cut can be written as $z_i + a_1 x_i = b_1$.

- $A_{1,30} = \{\text{boxes } i \mid l_i \times w_i \in]0, 0.3L \times W]\}$,
- $A_{2,30} = \{\text{boxes } i \mid l_i \times w_i \in]0.3L \times W, 0.6L \times W]\}$,
- $A_{3,30} = \{\text{boxes } i \mid l_i \times w_i \in]0.6L \times W, 0.9L \times W]\}$ and
- $A_{4,30} = \{\text{boxes } i \mid l_i \times w_i \in]0.9L \times W, L \times W]\}$.

If a box has a base whose area represents less than 30% of the base area of the ULD, then this box is assigned to $A_{1,30}$. Note that a box with a base area larger than $L \times W$ cannot fit in this type of ULD. Thus, the cluster $A_{4,30}$ has an upper limit equal to $L \times W$.

Inside each cluster, items are then sorted by non-increasing value of their height h_i. If two boxes have the same base area and the same height, then, if the considered ULD type has a length larger (resp. smaller) than the width, the box with the largest length (resp. width) is considered first. Finally, the clusters are sorted by decreasing value of j, which implies that the cluster with the boxes with the largest bases is considered first.

In Crainic et al. (2008), the value of δ has been tuned. However, since a lot of other constraints are added, the value of δ has to be calibrated to optimise the objective function of this specific algorithm in the experiments, which is addressed in Section 6.2.3.

The best feasible orientation is thus selected and added to the list L_1.

Sorting the list Once L_1 is created, the sequence according to which the boxes are considered has still to be decided. Several rules to sort list L_1 may be applied. For instance, boxes can be sorted by non-increasing volume or by non-increasing values of their base area. Crainic et al. (2008) state that the Clustered Area-Height is the sorting rule leading to the minimum number of selected bins. This rule is thus applied to sort L_1.

5.1.2 Part I

Each box of list L_1 is considered and packed if possible, one after another. Thus, during the packing algorithm, one tries to pack a given box while other boxes have already been packed.

Extreme Point Suitability For a given box, a list with all the suitable EPs in all ULDs that are already used is created by testing all the available EPs. An EP is *suitable* for a box i with a given orientation if, once the box i is packed with its front left bottom vertex on the EP:

1. the box i does not overlap other boxes previously packed,
2. the box i lies within the limits of the ULD (in particular, box i respects the possible special shape of the ULD),
3. the ULD weight capacity is still respected,
4. the box i does not lay on fragile boxes previously packed,
5. the centre of gravity of the set of loaded boxes does not exceed the given upper limit α^H (see Section 3.4.2 for a reminder) and
6. the box i and the previously packed boxes are stable.

These conditions are easily checked with the coordinates of the new and the already packed boxes (see Chapter 3 for more details). Note that the stability of the boxes is handled in the same way as in the previous chapters, ensuring that the four base vertices are supported.

If the box is fragile, an additional constraint is considered during the first part of the packing algorithm. Nothing can be packed on top of a fragile box, meaning that the volume above is lost. For this reason, during the first part of the packing algorithm, an EP (x_{EP}, y_{EP}, z_{EP}) is suitable if, once the box i is packed with its front left bottom vertex on this EP, the top face of the box has a height at least equal to a given percentage β of the ULD height:

$$z_{EP} + h_i \geq \beta H.$$

This constraint is the *Not Too Low* constraint in Figure 5.3. The value of β is tuned using a parametrisation technique in Section 6.2.3.

Selection of the best Extreme Point Choosing the best EP is a difficult task and many criteria can be imagined in order to optimise different objectives. For instance, if the packing has to be as compact as possible, then the EP leading to the minimum moment of inertia would be selected. Another criterion could be to minimise the maximum height of the packed boxes. This selection is achieved with a merit function as defined in Crainic et al. (2008). Two distinct merit functions are now presented.

In Crainic et al. (2008), the authors aim to minimise the number of bins used to pack the boxes, all the bins being identical. Their best merit function is the function maximising the utilisation of the EP's Residual Space. The *Residual Space* (RS) of a given EP is defined as the free volume available around an EP. It is composed of three components RS_x, RS_y and RS_z, each component describing the free space along each direction. When the EP

(x_{EP}, y_{EP}, z_{EP}) is created in ULD j, the RS of this new EP is equal to the distance from the EP to the side of the bin along each axis:

$$RS_x = L_j - x_{EP};\ RS_y = W_j - y_{EP};\ RS_z = H_j - z_{EP}.$$

Then, every time a box is assigned to ULD j, the RS components of the EPs in this ULD are updated. However, the EP generation implemented in this algorithm is extended as explained further in this section and thus the initial RS may be different. In the present work, in order to compute the RS_x of a new EP (x_{EP}, y_{EP}, z_{EP}), one focuses on boxes i packed in ULD j with their front left bottom vertex on (x_i, y_i, z_i) such that $x_{EP} \leq x_i$, $y_i \leq y_{EP} \leq y_i + w_i$ and $z_i \leq z_{EP} \leq z_i + h_i$ as shown in Figure 5.4. If only one box i exists, then $RS_x = x_i - x_{EP}$. If several boxes i exist, then $RS_x = \min_i(x_i - x_{EP})$. If none exist, then $RS_x = L_j - x_{EP}$. The same reasoning is applied for the two other directions. Moreover, if the ULD j has some cuts, L_j or H_j have to be adapted.

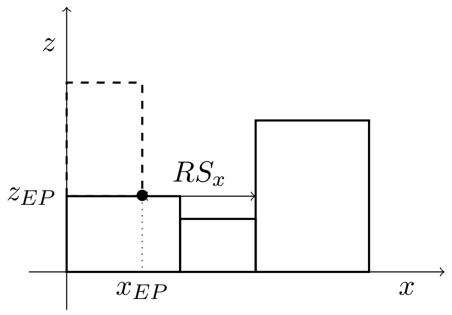


Figure 5.4: Computation of RS_x. Plane cut in $y = y_{EP}$

For a box i to be packed, the merit function from Crainic et al. (2008) selects the EP that minimises the difference between its RS and the box dimensions:

$$MF1 = (RS_x - l_i) + (RS_y - w_i) + (RS_z - h_i).$$

This selection is close to the selection achieved in the common Best Fit algorithm.

Since the ULD weight distribution is taken into account in this work, another specific merit function is also introduced. Indeed, one particular constraint is that the weight of the boxes packed inside each ULD should be rather uniformly distributed on the XY plane. As a reminder, the position of the CG of the boxes packed in ULD j is calculated as follows:

$$CG_j = \left(\frac{\sum\limits_{i\ in\ j} m_i \frac{2x_i + l_i}{2}}{\sum\limits_{i\ in\ j} m_i}, \frac{\sum\limits_{i\ in\ j} m_i \frac{2y_i + w_i}{2}}{\sum\limits_{i\ in\ j} m_i}, \frac{\sum\limits_{i\ in\ j} m_i \frac{2z_i + h_i}{2}}{\sum\limits_{i\ in\ j} m_i} \right).$$

As explained in Chapter 3, in order to measure this uniformity, the CG of the packed boxes (x_{CG}, y_{CG}, z_{CG}) is compared to the geometrical centre of the ULD base (x_M, y_M, z_M). For this reason, an additional merit function that computes the deviation between the two elements is defined:

$$MF2 = \sqrt{(x_{CG} - x_M)^2 + (y_{CG} - y_M)^2},$$

This merit function is a first step towards a uniform weight distribution. Note that the constraint relative to the height of the CG is included in the definition of suitable EPs. Among these two merit functions, the best function with respect to the objective function minimisation is determined with a parametrisation technique in Section 6.2.3.

The suitable EPs which have the smallest merit function value are selected. However, since the CG of the packed boxes cannot lie above a determined height, the selection is refined to take this into account. The suitable EPs are sorted by increasing value of the merit function and then among the best EPs (the precise number, called `bestHeight`, is tuned in Section 6.2.3), the EP that, if chosen, would lead to the lowest CG of the packed boxes is kept. When selecting the best EPs, the EPs leading to a CG lying in the authorised area are favoured. In practice, the CG has a maximum allowed deviation which should be respected. If there is no feasible EP with respect to weight distribution, then other EPs can be chosen.

Placing the box on the selected EP and L_2 testing Once the box is placed with its front left bottom vertex on the best EP, new EPs are generated and one tries to pack some fragile boxes from L_2 on these new points.

Placement

The considered box is assigned to the ULD containing the selected EP with the position described by the EP. The selected EP is removed from the EP list. It may happen that placing a box prevents other existing EPs from accommodating a box. Therefore, each EP lying in the same ULD is checked and deleted if unusable.

EP Generation

When a box i is placed with its front left bottom vertex on the point (x_i, y_i, z_i) with its opposite vertex on (x'_i, y'_i, z'_i), then new EPs are generated. This generation process is based on the procedure developed by Crainic et al. (2008). The three initial points (x'_i, y_i, z_i) , (x_i, y'_i, z_i) and (x_i, y_i, z'_i) represent the EP sources (and also new EPs themselves) and each point is projected along two directions until reaching a previously packed box or a container side to create new EPs as shown in Figure 5.5.

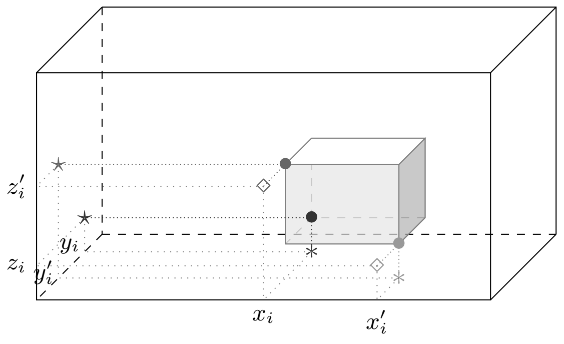


Figure 5.5: Projection of the source points of box i: the symbol $\star$ (resp. $\diamond$, $*$) represents the projection on the YZ plane (resp. XZ plane, XY plane)

The EP generation process is extended in this work as follows. Consider three types of boxes: one very long, another very wide and a third very high. Each of these three boxes is packed on a source point and is then pushed as far as possible along the projection direction, until it reaches either a previously packed box or the side of the ULD. The position of the front left bottom vertex of this virtual box defines a new EP. However, since the four vertices of each box have to be supported, the projection of the point (x_i, y_i, z'_i) does not lead to usable EPs. For this reason, this point is not projected unless there is a type 1 cut which can also support the box as shown in Figure 5.6.

The detailed generation process is provided in appendix A.1.

L_2 testing

A list of new EPs has been generated and cleared to avoid duplicates and unusable EPs. Before adding this list to the global list of EPs, these new proposed positions are tested to accommodate fragile boxes from L_2. In more detail, for each box from L_2, the new EPs are tested looking for suitable EPs (with the *Not Too Low* constraint). If there is one (or more) suitable EP, then

- the fragile box is packed on the (best, if several, as defined previously) EP,

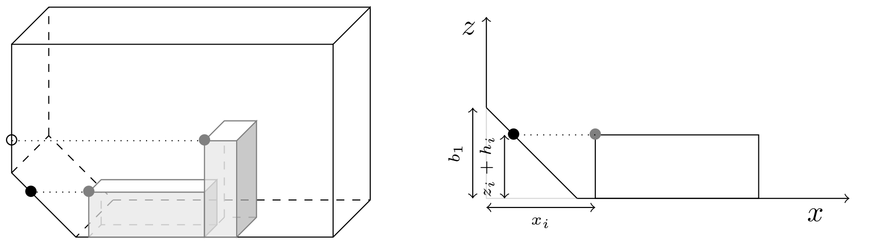
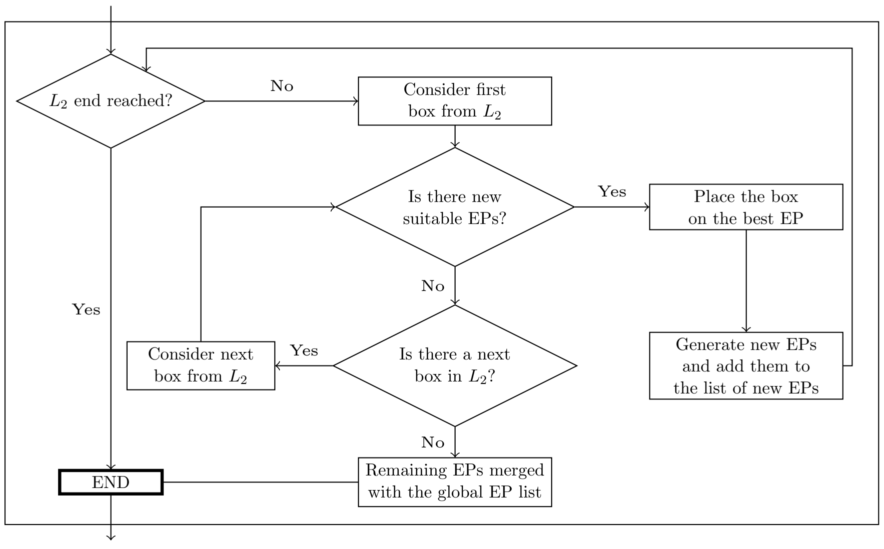


Figure 5.6: Projection of (x_i, y_i, z_i') along the x-axis in a ULD with a cut 1

- the selected new EP is removed from the list of available EPs and
- new EPs are generated with the same method and added to the list of new EPs.

Every time a box from L_2 is packed, this box is removed from L_2 and the beginning of L_2 is started again since new EPs are available. If a box from L_2 cannot be packed on any of the new EPs, then the next box from L_2 is considered. This process stops when boxes from L_2 can no longer be packed on these new EPs. The final list of new EPs is then merged with the global EP list. This step of the algorithm is depicted in Figure 5.7. Note that boxes from L_2 are also naturally sorted with the Clustered Area-Height rule since every time a box from L_1 cannot be packed, it is added at the end of L_2.



L_2 end reached?
No
Consider first box from L_2
Is there new suitable EPs?
Yes
Place the box on the best EP
No
Yes
Consider next box from L_2
Yes
Is there a next box in L_2?
Generate new EPs and add them to the list of new EPs
No
END
Remaining EPs merged with the global EP list

Figure 5.7: L_2 testing

5.1.3 Part II

At this point of the algorithm, every box from L_1 has been either packed in a ULD, or, if it was fragile and not possible to pack without opening a new ULD (maybe due to the *Not Too Low* constraint), assigned to the list L_2. In Part II, the *Not Too Low* constraint is dropped and all the fragile boxes from L_2 have to be loaded. The process is now identical to that of Part I: for every box, the global list of EPs is considered and the suitable ones are kept. The best suitable EP, if several, is selected with the same criterion as previously. If there is no suitable EP, then a new ULD is created to accommodate the current fragile box.

5.2 Multiple ULDs: Phase 2

In the packing algorithm, a given set of boxes is loaded in a given ULD type available in an unlimited amount. In practice, several types of ULDs exist and can be selected, each type available in a determined quantity. Therefore, the algorithm has to be extended in order to take this possibility into account.

In Kang and Park (2003), the existence of several types of bins is considered for the one-dimensional BPP. The authors assume that the items are sorted by non-increasing size and so are the bins. They extend the First-Fit Decreasing algorithm in the following way: they first build a feasible solution by packing all the items in the largest bin type using a traditional First-Fit algorithm. In order to obtain a different feasible solution, they repack the items of the last bin of the previous solution in the next largest bin type, still with the same First-Fit algorithm. They repeat this process until it is no longer possible to repack. An example is given in Figure 5.8. Out of all the obtained feasible solutions, the authors select the best solution with respect to the objective function value as the final solution.

It is not straightforward to apply the method from Kang and Park (2003) to the studied problem because of the difficulty of sorting items/boxes and bins/ULDs in three dimensions. In this work, boxes are sorted with the Clustered Area-Height rule as explained in Section 5.1.1 while ULD types are sorted by non-increasing volume. There are few ULD types and they all have a different volume. In the current case, an easy adaptation is to first pack (with the packing algorithm from Section 5.1) all the boxes into the largest ULD type (called type 1 ULDs). A list of type 1 ULDs that contain all the boxes is thus obtained. Then, the used ULDs of the same type are sorted by non-increasing occupied volume. Indeed, since the last ULD is the ULD whose boxes will be repacked, the less volume there is to repack the

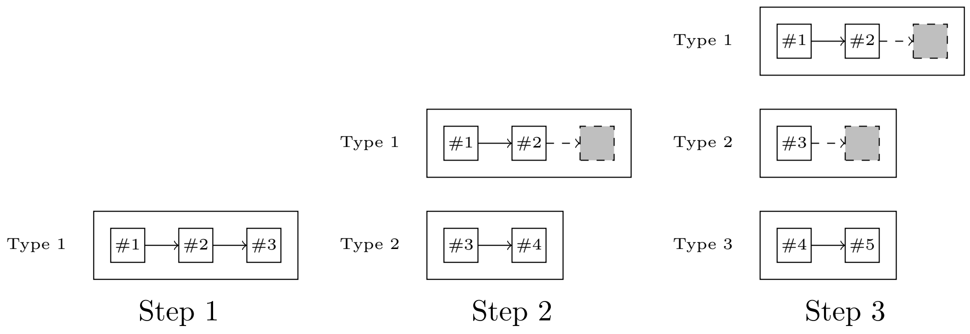


Figure 5.8: Example of three steps of the algorithm from Kang and Park (2003)

more likely that it is possible to repack into a smaller ULD type. Then, the boxes that are assigned to the last ULD of the list, which is the least loaded ULD, are repacked in the next ULD type. A list of type 2 ULDs is obtained, sorted and the boxes accommodated in the last element are repacked in type 3 ULDs. The process is repeated until all ULD types have been tested. Finally, the solution with the minimum used volume is selected.

This adaptation of the method is not likely to achieve good quality solutions since only the last ULD of each type is removed to be repacked. If at the first step the obtained solution has three ULDs of the first type, a solution with less than two type 1 ULDs cannot be reached. For this reason, the method is extended: instead of removing only the last ULD, all the ULDs of a given type are repacked into another smaller ULD type if it is possible. During the first (resp. second, third, etc.) iteration, boxes are packed in type 1 (resp. type 2, type 3, etc.) ULDs. Each iteration is decomposed into several steps, each step leading to a distinct loading pattern. From one iteration to another, all the patterns of the previous iteration are considered and the boxes placed in ULDs of the last type are repacked into a new ULD type. Moreover, it would seem natural to sort the ULDs by non-increasing occupied volume to increase the chance to repack into a smaller ULD type. However, some preliminary experiments show that the least loaded ULDs often have boxes with special dimensions (for instance, a very long box) and these boxes may be hard or even impossible to repack into smaller ULD. For this reason, sorting by non-increasing or non-decreasing occupied volume is a choice made using `irace` parametrisation technique in Section 6.2.3.

In more detail, the first iteration consists in packing all the boxes in the first ULD type, type 1, which creates the first loading pattern. These ULDs are sorted by non-increasing/non-decreasing occupied volume depending on the decision taken in Section 6.2.3. In the second iteration, the first step repacks the boxes from the last type 1 ULD into type 2 ULDs, the second step repacks the boxes from the last two type 1 ULDs into type 2 ULDs. During the third iteration, among the patterns obtained in the second iteration, the boxes in type 2 ULDs are repacked into type 3 ULDs, etc. An example is shown in Figure 5.9.

This procedure can lead to a large number of potential patterns but not all these patterns are feasible because some boxes may not fit in certain ULD types. In other words, some patterns may have unpacked boxes. When all the patterns have been created, the feasible pattern with the minimum volume is retained. The incomplete patterns are then analysed. For every incomplete pattern with an occupied volume less than the current minimum volume, the unpacked boxes may be loaded in two different ways. A first method is to pack the unpacked boxes in the largest ULD type, that is, type 1 ULD. If some of the unpacked boxes do not fit in this ULD type, then the second largest ULD type is considered. This process is repeated until every initially unpacked box is loaded. A feasible pattern is obtained with this method since each box is assumed to fit in at least one ULD. Once the pattern has been completed, its objective function value is compared to the value of the current best solution to determine the new best pattern. A second completion method could be to consider all the EPs still available in all the ULDs and to try to load the unpacked boxes, regardless of the chosen orientation, no matter the EP. Some preliminary experiments have been carried out on 7 samples of 30 instances containing from 10 to 100 boxes with given parameters[2]. More details about the data and deeper experiments will be provided in Section 6.1. Results for the second completion method are shown in Table 5.1. In this table, the first row shows the number of incomplete patterns for all 30 instances that have an objective function value smaller than the current best pattern. The second completion method is applied to these patterns and the second row indicates the number of patterns that have been successfully completed. The last row shows the number of instances, out of the 30, that have not been solved. This means that for those instances, all the patterns were incomplete and the second completion method was not able to complete any of them. The second completion method is very time consuming because of the number of patterns and of available EPs, especially for instances with a large number of boxes as shown on the boxplots in Figure 5.10. Moreover, the objective function values obtained are not always better than those obtained with

[2]without parametrisation: $\delta = 0.01, \beta = 80\%, MF2$, `bestHeight`=5, `uldOccupation-Sort`=2 (see Section 6.2.3 for more details about the parameters)

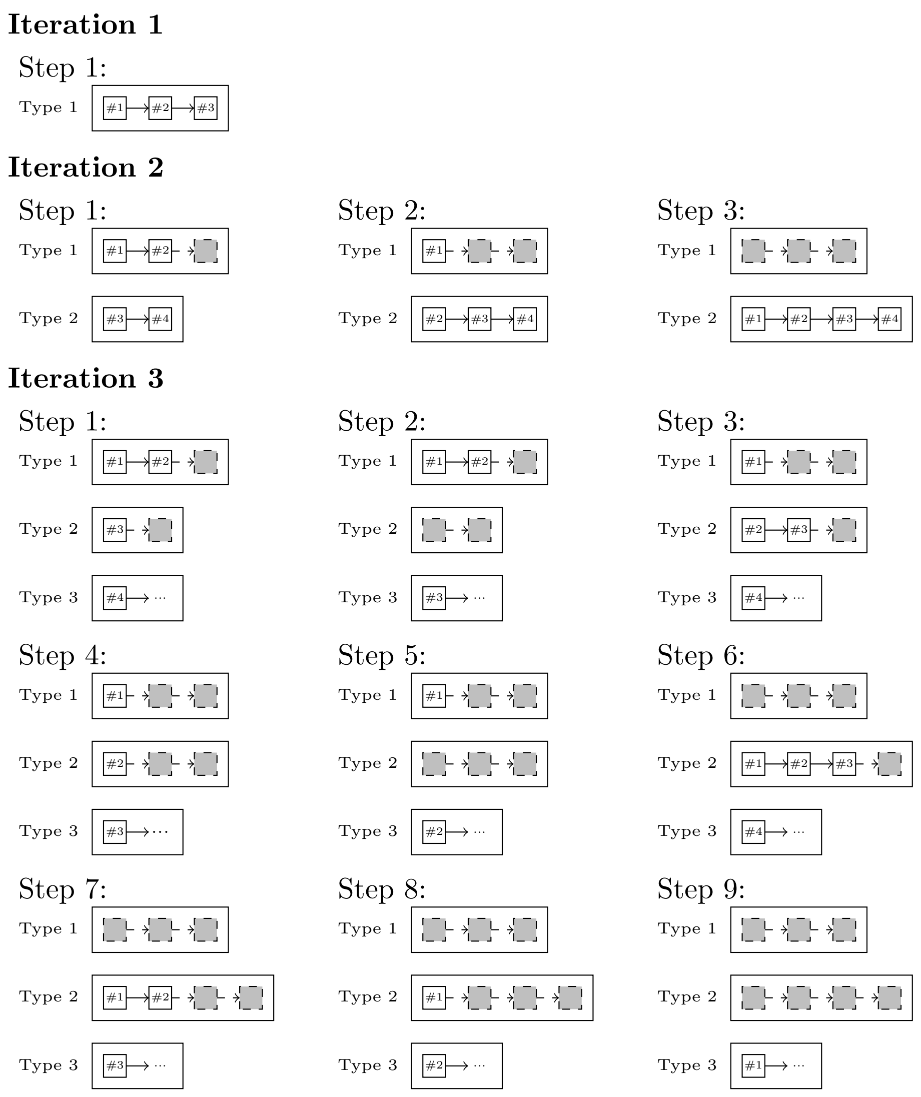
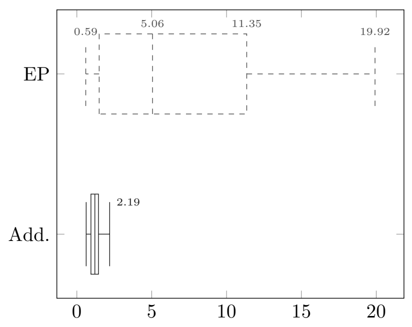


Figure 5.9: Creation of loading patterns by considering different ULD types

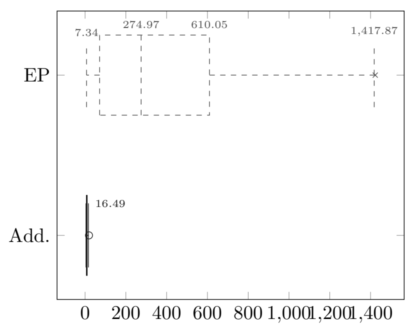
Note. In the first iteration, boxes are packed in type 1 ULDs. During the second iteration, the first step repacks the boxes from the last ULD of type 1 in two ULDs of type 2, the second step repacks the boxes from type 1 ULDs #2 and #3 and the third step repacks all the boxes in type 2 ULDs only. In the third iteration, steps 1 and 2 are based on the pattern obtained in the first step of iteration 2, steps 3, 4 and 5 are based on the pattern obtained in the second step of iteration 2 and steps 6, 7, 8 and 9 are based on the third pattern obtained in the third step of iteration 2.

Table 5.1: Analysis of the second completion method using the available EPs to complete patterns (each sample size holds 30 instances of n boxes)

	$n = 10$	$n = 20$	$n = 30$	$n = 40$	$n = 50$	$n = 60$	$n = 100$
# completion attempts	69	175	594	925	2134	2958	24627
# successful completion	7	5	1	3	5	9	8
# unsolved instances	0	3	3	4	3	4	12



(a) Comparison of the computational times, in seconds, for the 30 instances of 60 boxes for the adding ULDs strategy in full lines and the use of available EPs in dashed lines



(b) Comparison of the computational times, in seconds, for the 30 instances of 100 boxes for the adding ULDs strategy in full lines and the use of available EPs in dashed lines

Figure 5.10: Comparison between adding ULDs or using available remaining EPs for unpacked boxes of each pattern

the first method, as shown in Table 5.2. Table 5.2 shows the comparisons between the objective function values of the patterns obtained with the two completion methods for every instance. For example, for the 30 instances of 50 boxes, 3 instances have not been solved with the second completion method (based on the available EPs), 21 instances have a final pattern with the same objective function value for the two methods, 4 instances have a pattern with a larger objective function value with the first method than with the second method and inversely for the remaining 2 instances (3+21+4+2=30). The first completion method is thus preferable for the present heuristic.

The methodology described in this section can be relevant in the case of an actual application. In practice, if only one aircraft/flight is available to pack the whole set of boxes, a limited number of ULDs of each type are provided.

Table 5.2: Comparison of the objective function values of the two completion methods (each sample has 30 instances of n boxes)

Sample size	# unsolved instances with EP method	Adding=EP	Adding >EP	Adding <EP
$n = 50$	3	21	4	2
$n = 60$	4	21	2	3
$n = 100$	12	9	3	6

Note. Adding represents the completion method based on the addition of new ULDs while EP represents the completion method based on the use of available EPs in all the ULDs of the pattern.

This method can be extended by considering only the given number of each ULD and keeping the unpacked boxes for the following type of ULD. There should be enough ULDs to pack all the boxes, otherwise the nature of the problem would be different (it would become a Knapsack instead of a Bin Packing Problem).

5.3 Post process: balance improvement

After the first two phases of the algorithm, a packing with possibly distinct ULDs has been obtained. In this packing, all the constraints of the specific MBSBPP are ensured except the uniform weight distribution in the XY plane. As explained in the merit function definition in Section 5.1.2, a first step can be achieved when selecting the best EP for each box, but this does not ensure that the CG is in the allowed region around the geometric centre of the ULD base. For instance, preliminary results show that, over 300 instances with a number of boxes ranging from 10 to 100, 878 ULDs are used and 502 (i.e., 57.18%) of them have a CG out of the allowable region in at least one direction of the XY plane. Considering the choice of the first EP and the EP generation during the packing algorithm, the CG is likely to lie too far to the left or too far to the front of the geometrical centre of the ULD base area, which is confirmed by some preliminary experiments: over the 502 unbalanced ULDs, **70.12%** (resp. 0.60%, **86.45%**, 0.40%) has a CG located too far to the **left** (resp. right, **front**, rear). Post processing has to be conducted to improve the weight distribution in the unbalanced ULDs.

Table 5.3: Comparison between exact approach and the J&S operators for the weight distribution improvement (each sample size has 30 instances, n is the number of boxes in each instance)

	Before impr.	After exact approach		After J&S operators	
Sample size	# unbal./ #ULDs	# unbal./ #ULDs	Avg. dur. [s.]	# unbal./ #ULDs	Avg. dur. [s.] (max.)
$n = 10$	35/47	0/47	1.58	2/47	0.008 (0.113)
$n = 20$	30/55	4/55	$489.33^{(4)}$	2/55	0.004 (0.044)
$n = 30$	53/86	3/86	$136.52^{(2)}$	5/86	0.007 (0.284)
$n = 40$	55/92	6/92	$329.58^{(5)}$	5/92	0.030 (1.381)
$n = 50$	50/101	2/101	$144.90^{(2)}$	5/101	0.002 (0.014)
$n = 60$	54/113	3/113	$202.75^{(2)}$	7/113	0.007 (0.317)

Note. The ULDs already balanced are not considered in the average duration calculation. The notation $\cdot^{(w)}$ means that the B&B could not find a solution within one hour for w out of the 30 instances. For the J&S operators, the maximum durations are provided in brackets.

5.3.1 Exact method

A first method is to adapt the mathematical formulation developed in Chapter 3. In the present situation, the aim is no longer the selection of ULDs but the achievement of a uniform weight distribution in a specific ULD with a given set of boxes to be packed. The chosen objective is to minimise the height of the CG of the set of boxes. The detailed linear formulation can be found in Appendix A.2. Solving optimally this model is sometimes time consuming and since a feasible but not necessarily optimal solution is sufficient, the algorithm can stop as soon as the height of the CG is less than or equal to the maximum authorised value α_H. As preliminary experiments, this formulation has been implemented in Java, using IBM ILOG CPLEX 12.6 library as B&B solver under the default parametrisation with a maximum of one hour computational time. If no solution is found within one hour, the studied loading pattern remains unchanged. As shown in Table 5.3, the exact approach is very time consuming, especially compared to the approaches developed in the next sections, making this exact method inappropriate in practice. In this table, the notation $\cdot^{(w)}$ means that the B&B could not find a solution within one hour for w out of the 30 instances. All the unbalanced ULDs that B&B was not able to solve hold at least 18 boxes. The average durations are calculated over the unbalanced ULDs:

$$\frac{\text{total time spent for balance improvement}}{\#\text{unbalanced ULDs}}.$$

5.3.2 Shifts

The second method proposes two *shift* operators to improve the balance along the x and y-axes. For the ULDs presenting a CG located too far to the left, the boxes are pushed as much as possible to the right until the CG reaches the threshold of the allowed region. The maximum possible global shift is first applied to all the boxes, i.e., the maximum possible overall shift is computed and applied to all the boxes at once. A *global* shift means that if the shift is increased by one more unit, then the ULD limit constraint becomes violated. If the threshold is still not achieved, then each box is considered separately and an adapted shift is set up until a constraint is violated or the threshold is reached. An *adapted* shift means that the shift does not have a general value but is specific to each box. Note that special attention is required to manage the right (resp. left) shift in ULDs with cuts of type 2 or 3 (resp. 1 or 4). In order to increase the chance of possible adapted shift, the boxes i of the ULD are considered by non-increasing x'_i (resp. y'_i) values in the case of a right (resp. back) shift and by non-decreasing x_i (resp. y_i) values in the case of a left (resp. front) shift. Ties are broken sorting by non-increasing value of their front left bottom vertex height (z_i) and finally by non-increasing value of their density. An example of the application of shift operators to two unbalanced ULDs is represented in Figure 5.11.

The CG of the set of packed boxes is tested for each direction and the deviations are analysed in this order: CG too far to the left, front, right and rear. Every time the answer is positive, shift operators are applied. The next deviation is then tested and the same process is applied until the balance is reached or if there is no more improvement.

Some preliminary experiments are presented in Table 5.4 and showed that among the 353 ULDs having a CG too far to the left, only 19 of them cannot be corrected with the shift operators, which represents a reduction of 94.60%, while 394 out of the 434 ULDs no longer have a CG too far to the front (reduction of 90.78%). Conversely, the shift operators do not seem to work on the CG too far to the rear and on the right since the number of deviations remains identical. However, the global number of unbalanced ULDs has significantly decreased: 64 out of 878 ULDs (7.29%) still have a CG outside the allowable range.

A lot of patterns have been improved enough to reach a feasible solution. However, the remaining 7.29% of unbalanced ULDs are not easy to fix, there are often heavy boxes that cannot be shifted further. Therefore, a third method has been developed.

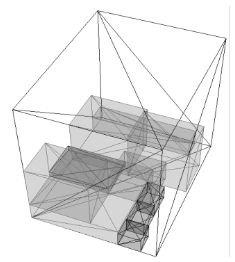
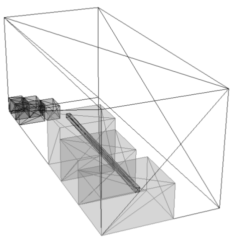
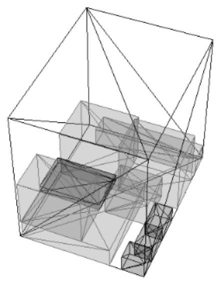
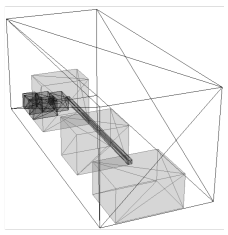
Before right and rear shifts	After right and rear shifts
 	 

Figure 5.11: Two examples of shift operators applications

Table 5.4: Reductions in the number of CG deviations among the 878 used ULDs by applying the shift operators only, the jump operator only and the J&S combination (percentages are computed over the 878 used ULDs)

	Number of	Deviations			
	unbal. ULDs	Left	Right	Front	Rear
Initial	502	352	3	434	2
	(57.18%)	(40.09%)	(0.34%)	(49.43%)	(0.23%)
After shifts	64	19	3	40	2
	(7.29%)	(2.16%)	(0.34%)	(4.56%)	(0.23%)
Reduction	87.25%	94.60%	0.00%	90.78%	0.00%
After jumps only	458	278	2	397	4
	(52.16%)	(31.66%)	(0.23%)	(45.22%)	(0.46%)
Reduction	8.76%	21.02%	33.33%	8.53%	-100.00%
After J&S	52	15	2	35	2
	(5.92%)	(1.71%)	(0.23%)	(3.99%)	(0.23%)
Reduction	89.64%	95.74%	33.33%	91.94%	0.00%

5.3.3 Jumps

The idea of the *jump* operator is to take one or several boxes from one side of an unbalanced ULD and to pack these boxes (make them jump) to the other side of the ULD. For instance, if the CG of the ULD is too far to the left, then one tries to repack boxes from the left-hand side to the right-hand side of the ULD in order to move the CG to the right. Note that the movements leading to a CG too far to the right are not allowed. Moreover, if the weight was balanced in the other direction, movements leading to an unbalanced loading along this other direction are also rejected.

In more detail, if the CG is too far to the left, then the boxes assigned to this ULD are considered by non-decreasing value of their front left bottom vertex abscissa (x_i) because the more on the left the boxes are, the more important is their influence on the CG. Ties are broken sorting by non-increasing value of their front left bottom vertex height (z_i) and finally by non-increasing value of their density. The loading pattern has still to be stable when the box to be jumped is removed from its current position and the highest boxes are less likely to support other boxes. Finally, the denser the boxes, the larger the impact on the CG.

To repack the boxes, the available EPs of the ULD produced during the packing algorithm are analysed. The idea is to repack the leftmost boxes on rightmost EPs to move the CG as much as possible. Box i can jump from (x_i, y_i, z_i) to a new EP (x_{EP}, y_{EP}, z_{EP}) if several conditions are met:

1. the loading pattern is still feasible, in particular still stable, if box i is removed from its current position,
2. only the EPs with $x_{EP} > x_i$ are helpful and thus considered since a balance improvement is expected,
3. only the suitable (as defined in the beginning of Section 5.1.2 but without the *Not Too Low* constraint) EPs are considered,
4. if the CG was in the allowable area along the y-axis before box i removal, it has still to be in the allowable area after repacking box i on the new EP and
5. after repacking, the CG is not too far to the right (on the other side of the allowable region).

If several EPs satisfy these conditions, the rightmost one is selected to accommodate box i. Ties are broken selecting the EP with the smallest residual space. Afterwards, the selected EP is removed from the list as well as the potentially unusable other EPs. New EPs are generated and added to the list of available EPs. If the CG is still on the left-hand side of the allowable region, then the next box of the sorted list is considered for a jump.

Naturally, the jump operator can be applied in the other directions: a jump from the right-hand side to the left-hand side, from the front to the rear and inversely. An example of jump operator applications is shown in Figure 5.12.

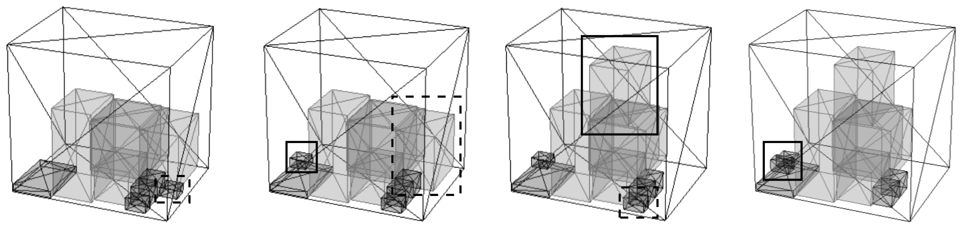


Figure 5.12: An example of jump operator applications to fix a CG too far to the right: the former positions are circled in dashed lines, while the new positions (after jumping) are circled in full lines

As for the shift operators, the CG of the set of packed boxes is tested for each direction and the deviations are analysed in the same order: CG too far to the left, front, right and rear. Every time the answer is positive, the jump operator is applied. The next deviation is then tested and the same process is applied until the balance is reached or if there is no more improvement. The same preliminary tests have been carried out and are presented in Table 5.4. As can be seen, the jump operator is more efficient for the right deviation but shows an unexpected behaviour for the rear deviations. Looking closely at these two new rear deviations, after the second phase of the algorithm, the loading patterns were showing a left and a front deviation. While improving the CG position along the x-axis, the CG which was too far to the front became too far to the rear and since the balance was obtained for the x-axis, jumps leading to a loss of this new gained balance have been rejected and thus the balance has been improved along the y-axis but not enough for the CG to be in the authorised area.

5.3.4 Combination of jump and shift operators

The jump and shift (J&S) operators can be combined into the same process. In practice, the jump operator is applied first. The jump operator is less accurate than the shift operators because the CG does not move one unit by one unit. Moreover, after applying shift operators, the residual space of some EPs may be reduced and thus these EPs would not be able to accommodate some boxes. In practice, the CG of the set of packed boxes is tested for each direction and the deviations are analysed in this order: CG too far to the left, front, right and rear. Every time the answer is positive, jump is applied. If no box is still able to jump and the CG is still not in the allowable region for the considered direction, then the shift operators are applied. The next deviation is then tested and the same process is applied. This means that jump operators may be applied after shifts in another direction. It is therefore crucial to keep the EPs and their respective residual space up-to-date during the whole process. It may happen that deviations are not corrected at the first application of J&S operators. However, the ULD may have another deviation whose correction leads to other movements of the boxes. For this reason, unless boxes are no longer moving, a loop over the different deviations is put in place. In order to avoid infinite loops, the CGs already met are memorised and compared to the current CG. If a same CG is obtained twice, then the algorithm stops. Indeed, it may happen that when trying to correct two deviations (one along the x-axis and one along the y-axis), boxes are moved in one direction first and then in another direction, which can lead to infinite loop. The whole process is illustrated in Algorithm 4. In this algorithm, $x_{CG_{min}}$ represents the minimum authorised value for x_{CG}, $\frac{L_j}{2}-\alpha_j^L$ (or adapted if there is a type 1 or type 2 cut), $x_{CG_{max}}$ the maximum authorised value for x_{CG}, $\frac{L_j}{2}+\alpha_j^L$ (or adapted if there is a type 1 or type 2 cut) and identically for $y_{CG_{min}}$ and $y_{CG_{max}}$. Some preliminary experiments presented in Table 5.4 show that the improvement is considerable. Figure 5.13 shows the average percentage of unbalanced ULDs per sample size (there are 30 instances for each size).

In conclusion, after applying the J&S operators as explained in Algorithm 4, 17 ULDs out of 878, i.e., around 1.94%, still have an unbalanced weight along the x-axis (too far to the left or right) and 37 ULDs out of 878, i.e., around 4.21%, still have an unbalanced weight along the y-axis (too far to the front or rear). The 54 remaining deviations have an average value of 9.01% (9.01% on average for the x deviations and 9.01% on average for the y deviations). The calculation of the deviation percentage along the x-axis for a CG outside the allowed region is computed as follows:

$$\frac{\min\{|x_{CG} - x_{CG_{min}}|, |x_{CG} - x_{CG_{max}}|\}}{0.5(L - (x_{CG_{max}} - x_{CG_{min}}))} \tag{5.1}$$

where $(L - (x_{CG_{max}} - x_{CG_{min}}))$ represents the size of the allowed region. Out of the 54 remaining deviations, 43 deviations can be corrected using the exact

Algorithm 4 Weight Distribution Improvement

```
doesSomethingChange ← **true**
**while** doesSomethingChange **do**
    **if** $(x_{CG} < x_{CG_{min}}) \,||\, (x_{CG} > x_{CG_{max}})$ **then**
        Jump operator application
        **if** $(x_{CG} < x_{CG_{min}}) \,||\, (x_{CG} > x_{CG_{max}})$ **then**
            shift operator application
        **end if**
    **end if**
    **if** $(y_{CG} < y_{CG_{min}}) || (y_{CG} > y_{CG_{max}})$ **then**
        Jump operator application
        **if** $(y_{CG} < y_{CG_{min}}) || (y_{CG} > y_{CG_{max}})$ **then**
            shift operator application
        **end if**
    **end if**
    doesSomethingChange ← $((x_{CG}, y_{CG}) \neq$ any previous$(x_{CG}, y_{CG}))$
**end while**
```

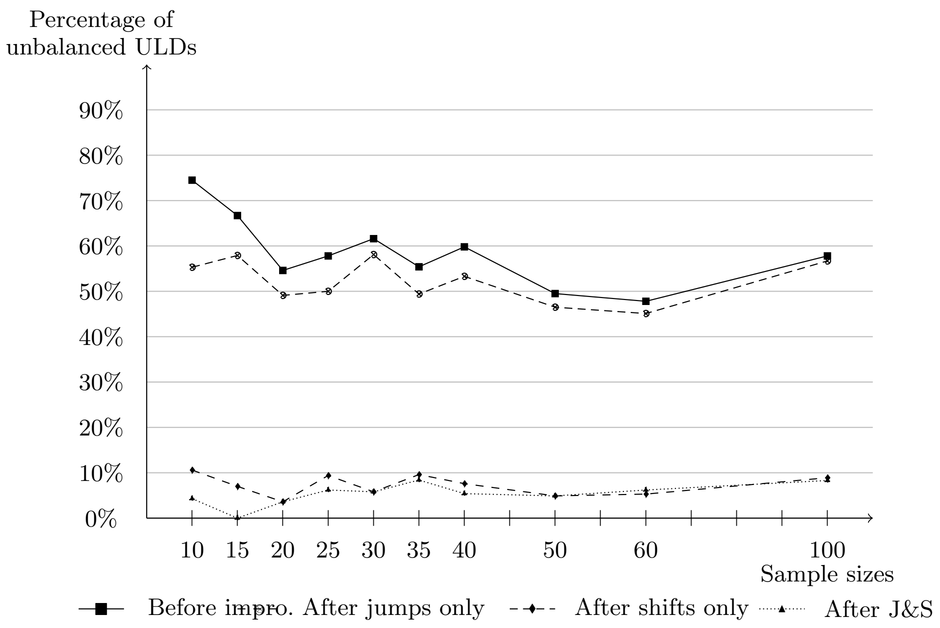


Figure 5.13: Average percentage of unbalanced ULDs per sample size (each sample size has 30 instances)

approach developed in Section 5.3.1, 11 deviations would need more than one hour of computational time to be corrected and the last deviation cannot be corrected in such a loading pattern.

In comparison to the exact approach developed in Section 5.3.1, the J&S has two main advantages. First, even if the ULD is eventually unbalanced, the J&S operators improve the CG position whereas the exact approach achieves a balanced ULD if it is able to or leaves the loading pattern unchanged. In particular, it seems that, in practice, this constraint is not strict and the unbalanced ULDs could still be loaded in the airplane. The second advantage of the J&S operators is its relative speed as shown in Table 5.3. Note that the ratios from Table 5.3 before improvement after J&S operator application can be found in Figure 5.13.

5.4 Areas for improvement

In this section, several hints are given for future research.

Density consideration The fragility feature of the boxes plays a key role in the packing algorithm in Section 5.1. This algorithm does not take into account the possible very dense boxes of the pattern. An extension of this packing algorithm could take this density into account and attempts to take advantage of this characteristic.

Orientation selection The orientation of each box is determined at the beginning of the algorithm and is then fixed. An area for improvement could be to consider the possibility of rotating the boxes along the whole algorithm. For instance, in the packing algorithm, if there is no suitable EP for a box, then another orientation of this box can be tried before creating a new ULD (in the case of a non-fragile box). Another possibility would be to test different orientations of a box when repacking during the application of the jump operator.

Combination of exact and J&S approaches In the preliminary experiments, it appears that the exact approach for weight distribution improvement solved with B&B can take time if the treated ULD holds more than 18 boxes, but also has very good results. More tests could be carried out to precise this number of boxes. Then, a method could combine the exact and the J&S approaches depending on the number of boxes in the ULD to be treated.

Multiple boxes shifts When applying the shift operators, some packings of boxes may be movable in practice but not authorised in the algorithm because of the stability constraints. As shown in Figure 5.14, boxes can be combined such that if one of the boxes is shifted by one unit, then the four vertices are no longer supported, which leads to an unstable loading. Thus, none of these boxes can be moved and the whole box combination remains at the same location. To improve the shift operators, it could be useful to identify these kinds of combinations and to shift them all together.

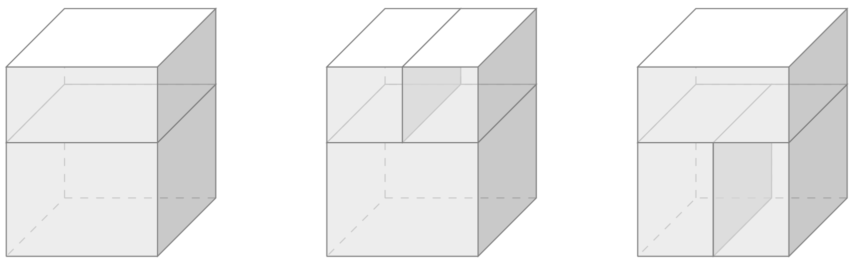


Figure 5.14: Unmovable box combinations: the first two combinations cannot be shifted in any direction, while the last combination cannot be shifted along the y-axis

Deterioration in jump application Another hint for improvement relates to the authorised jumps. As mentioned above, a box i can jump from (x_i, y_i, z_i) to a new EP (x_{EP}, y_{EP}, z_{EP}) if, among other conditions, the CG was in the allowable area along the y-axis before box i removal, it has still to be in the allowable area after repacking box i on the new EP. This condition can be relaxed and a slight deterioration smaller than a given percentage for the deviation along the other axis could be accepted in the hope that it can be corrected later.

Chapter 6

Computational experiments

This sixth chapter addresses the computational experiments. First, data sets are described in Section 6.1. These sets come from a real world case and are meant for different purposes: a group of instances is used to give some insights about each heuristic (preliminary data sets), a second group is used to parametrise the heuristics (training data sets) and a third group is used to measure the quality and to compare the efficiency of the heuristics (final data sets). Second, in Section 6.2, the parametrisation method is explained and applied to the I&F and FRF matheuristics from Chapter 4 and to the tailored two-phase heuristic from Chapter 5 using training instances. A sensitivity analysis is carried out for several parameters in Section 6.3. Finally, the trained heuristics are tested on the final data set, analysed separately in Section 6.4 and compared on this basis in Section 6.5.

All tests were performed on a workstation with 32.0 GB RAM and an Intel Xeon processor E5-2620 v4 running 64-bit Windows 10 Pro. Codes were implemented in Java, and CPLEX 12.6 library was used as B&B solver.

The current position of this chapter in the thesis outline is shown in bold in Figure 6.1.

6.1 Data sets

6.1.1 ULDs

Among the ULDs that can be loaded in a Boeing 777, which is the most common aircraft for cargo transportation, six common types are selected: three for the lower deck and three for the main deck. These ULDs are described in Table 6.1. More details about these ULDs can be found in Boeing (2008). This set of ULDs constitutes a representative sample as all shapes

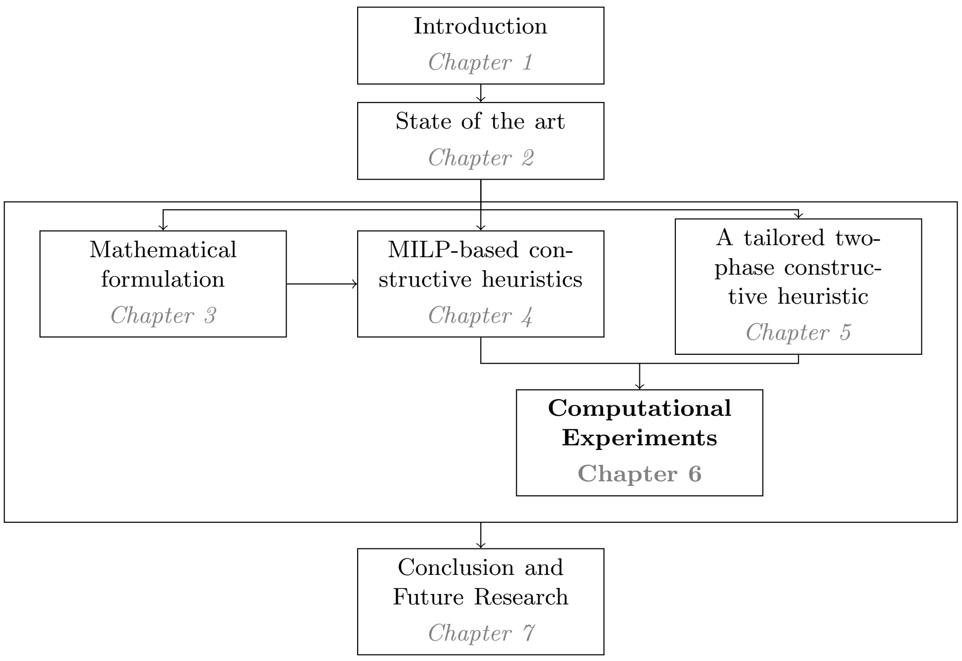

Introduction
Chapter 1
State of the art
Chapter 2
Mathematical formulation
Chapter 3
MILP-based constructive heuristics
Chapter 4
A tailored two-phase constructive heuristic
Chapter 5
Computational Experiments
Chapter 6
Conclusion and Future Research
Chapter 7

Figure 6.1: Current position in the thesis outline

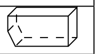
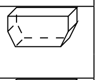
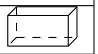
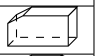
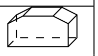
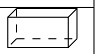
are proposed.

6.1.2 Boxes

As can be seen in Bortfeldt (2012) and Zhao et al. (2016), there are no benchmark instances for the three-dimensional MBSBPP. The closest instances are those from Ivancic et al. (1989) which were designed for a Multiple Stock-Size Cutting Stock Problem (see Figure 2.2). For this reason, new data sets were needed in order to represent the specificity of the problem studied in this thesis.

For the computational experiments, a box data set which stems from a real world case is considered. It contains information about the dimensions and weight for 562 rectangular boxes. The main features of these boxes are given in Table 6.2. Unfortunately, the initial data set does not include the authorised orientations and fragility characteristics. Therefore, data manipulation is set up to add these missing parameters. In more detail, if a box is too heavy (>50kg), it is assumed to be too much work to turn it, the parameters l^+, w^+ are thus assumed to be 0. The parameter h^+ equals 1 for all the boxes because the initial orientation of the boxes is supposed feasible. For

Table 6.1: Description of the parameters of the proposed ULDs

IATA Code	Dimensions $L \times W \times H$ [mm]	Cap. [kg]	Vol. [m^3]	$\alpha^L;\alpha^W;\alpha^H$ [mm]	Equation of the cuts	Shapes
			Lower deck - containers			
LD1	2337×1534×1626	1518	5	157;152;864	cut 1: $z + \frac{826}{775}x = \frac{640150}{775}$	
LD6	4064×1534×1626	2945	9.1	318;152;864	cut 1: $z + \frac{422}{444}x = \frac{187368}{444}$ cut 2: $z - \frac{422}{445}x = -\frac{1527218}{445}$	
LD11	3175×1534×1626	2991	7.4	318;152;864	None	
			Main deck - pallets			
PA	2235×3175×2997	5890	20	224;318;1219	cut 4: $z - \frac{747}{1056}x = \frac{2376000}{1056}$	
PG	2438×6058×2438	10840	31.1	244;302;1219	cut 3: $z + \frac{1054}{660}x = \frac{3483092}{660}$ cut 4: $z - \frac{1054}{660}x = \frac{913440}{660}$	
PM	2438×3175×2997	6680	18.9	244;318;1219	None	

the stacking constraint, a box which has a density lower than 0.05 kg/dm^3 is considered as fragile. After applying these two rules, all the parameters are available for each box.

Table 6.2: Information about the initial data set

	length [mm]	width [mm]	height [mm]	weight [kg]	volume [dm^3]	density [kg/dm^3]
Range	[130;5250]	[100;1720]	[40;1900]	[1;1983]	[2.2;5832.96]	[0.01;5.62]
Average	927.65	620.18	618.89	106.55	571.57	0.24
St. dev.	625.91	309.72	407.78	177.02	682.75	0.31

Number of distinct boxes (types):	199
Average number of identical boxes:	2.82

Description of the data sets Instances with different sizes are interesting in order to analyse the behaviour of each technique. Therefore, a data process is used to generate box samples of different sizes ranging from 6 to 100 boxes. The size of the considered samples depends on the tested heuristic. For instance, the matheuristics may be very slow if the number of boxes is large. Small instances are thus tested in the preliminary experimentations. For each sample size, a set of 30 instances is built by random selection from the original data set. Instance sets are available from `http://hdl.handle.net/2268/206856`.

- Preliminary data sets: 30 instances with 6, 7, 8, 9 and 10 boxes each (i.e., 120 instances) have been generated for preliminary elimination of some matheuristics in Chapter 4, 30 instances with 15, 20, 25, 30, 35, 40, 50, 60 and 100 boxes each have been generated and added to the set of 30 instances with 10 boxes (i.e., 300 instances) to analyse several leads in Chapter 5;

- Training data sets: in order to tune the parameters of each technique in Section 6.2, a set of representative instances has to be provided. To this purpose, 30 instances with 10, 15, 20, 25 and 30 boxes each (i.e., 150 instances) have been generated;
- Final data sets: for the sensitivity analysis in Section 6.3 and for testing of all the techniques in Sections 6.4 and 6.5, 30 instances with 10, 20, 30, ..., 100 boxes each (i.e., 300 instances) have been generated.

6.2 Parametrisation with `irace`

The software package `irace` provides an automatic configuration tool for tuning optimisation algorithms, that is, automatically finding the most appropriate settings of an algorithm given a set of instances of a problem, saving the effort that is normally required in manual tuning (López-Ibáñez et al. (2016)). During the tuning phase, a set of training instances representative of a particular problem has to be provided to choose the best algorithm configuration. The selected algorithm configuration can then be used to solve new instances of the same problem.

In more detail, let be a parametrised algorithm with N^{param} parameters which can take different values. A *configuration* of the algorithm $\theta = \{x_1, ..., x_{N^{\text{param}}}\}$ is a unique assignment of values to parameters. The set of all the configurations of the algorithm is denoted Θ. A *racing method for selection*, or *race*, aims to find a good configuration in Θ through a sequence of steps. As explained in Birattari et al. (2002) and López-Ibáñez et al. (2016), along the racing method, if sufficient evidence is gathered that some candidate configurations perform statistically worse than at least another configuration, such candidates are discarded and the procedure is iterated over the remaining surviving configurations. The elimination of inferior candidates speeds up the procedure and allows a more reliable evaluation of the promising configurations.

To evaluate the performance, a set of training instances to be sampled is provided, as well as a cost function. The cost function, $\mathcal{C}(\theta, i)$ assigns a value to each configuration θ when applied to a single problem instance i. This returned value can be the computational time or the value of the objective function as it is the case in this work.

To discard a configuration, statistical test is performed on the values provided by the cost function of several instances. In `irace`, the default test is the non-parametric Friedman's two-way analysis of variance by ranks and its associated post-hoc analysis described in Conover (1999). However, the paired t-test is an alternative implemented in `irace`. Both tests use a default significance level of 0.05. Friedman's test is recommended when tuning for solution quality and is thus used in the following analysis.

The race is applied until reaching a minimum number of surviving configurations, a maximum number of instances that have been used or a pre-defined computational budget. This budget can be a computational time or a number of experiments, where an experiment is the application of a configuration to an instance. An example of racing method from López-Ibáñez et al. (2016) is shown in Figure 6.2.

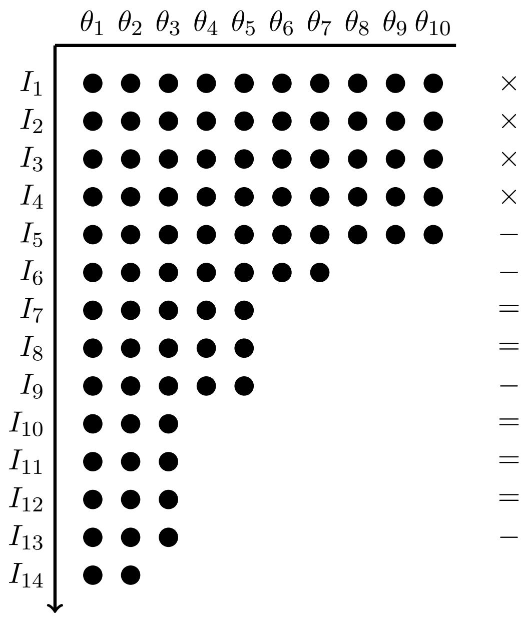

θ_1 θ_2 θ_3 θ_4 θ_5 θ_6 θ_7 θ_8 θ_9 θ_{10}
I_1 ×
I_2 ×
I_3 ×
I_4 ×
I_5 −
I_6 −
I_7 =
I_8 =
I_9 −
I_{10} =
I_{11} =
I_{12} =
I_{13} −
I_{14}

Figure 6.2: Racing for automatic algorithm configuration (López-Ibáñez et al. (2016))

Note. Each node is the evaluation of one configuration on one instance. '×' means that no statistical test is performed, '-' means that the test discarded at least one configuration, '=' means that the test did not discard any configuration.

In iterated racing, each configurable parameter is associated to a sampling distribution which is independent of the sample distributions of the other parameters. More details about these sampling distributions can be found in López-Ibáñez et al. (2016). *Iterated racing* is a method for automatic configuration which can be decomposed into three steps:

1. sampling the new configurations according to a particular distribution,
2. selecting the best configurations from the newly sampled configurations by means of racing (as explained here-above),
3. updating the sampling distribution of each configurable parameter in order to bias the sampling towards the best configuration. The update biases the distributions to increase the probability of sampling the parameter values in the best configurations found so far.

The three steps are repeated until a termination criterion is met.

`irace` needs to get a number, which is the objective function value in this case, for every run in order to assess the quality of the configuration. Since the matheuristics have a limited authorised duration of one hour, sometimes a solution may not be found within this time. In that case, the matheuristic returns a large number, $n \times \max_j V_j$ where n is the number of boxes in the instance and V_j the volume of ULD j, to represent the poor results.

6.2.1 Insert-and-Fix

The following two parameters need calibration:

Parameters	Types	Range
u	Integer	[2,7]
q	Integer	[1,3]

As a reminder, parameter u denotes the number of boxes whose variables have to be determined and q the number of boxes whose variables are fixed. The configurations with $q \geq u$ are thus forbidden.

With a maximum number of 1000 experimentations, `irace` states that the best configuration is $u = 7$ and $q = 2$. These values will be interpreted and analysed through a sensitivity analysis in Section 6.3.1.

6.2.2 Fractional Relax-and-Fix

The following three parameters need calibration:

Parameters	Types	Range
u	Integer	[3,7]
v	Integer	[2,6]
q	Integer	[1,3]

Parameter u denotes the number of boxes which are considered and whose variables are to be determined, v is the number of boxes whose variables have the integrality restriction to be satisfied and q the number of boxes whose variables are fixed. The configurations with $q \geq v$ or $v \geq u$ are thus forbidden.

- block Coord: with a maximum number of 1000 experimentations, `irace` states that the best parameter combination is $u = 7$, $v = 5$ and $q = 1$. These values will be interpreted and analysed through a sensitivity analysis in Section 6.3.2;
- block Coord & p_{ij}: with a maximum number of 1000 experimentations, `irace` states that the best parameter combination is $u = 7$, $v = 6$ and $q = 1$.

6.2.3 A tailored two-phase constructive heuristic

At different stages of the tailored two-phase constructive heuristic, several options are possible and choices have to be made in order to optimise the value of the objective function. For this reason, the following six parameters need calibration:

Parameters	Types	Range
β	Real	[0.50,1.00]
`bestHeight`	Integer	[1,5]
merit function	Integer	[1,2]
`uldOcptSort`	Integer	[1,2]
`boxSort`	Integer	[1,2]
δ	Integer	[1,99]

Parameter β represents the ratio of the ULD height that the top face of a fragile box has to reach during the first part of the packing algorithm with the *Not Too Low* constraints (see page 94). Values tested for β are limited to two digits. Parameter `bestHeight` represents the number of first Extreme Points (EP) considered for selecting the best one with respect to the centre of gravity (CG) height (see page 96). Two merit functions have been proposed (see page 95) to select the best EP: the first function, MF1, uses the residual space (RS) around the EP and the second, MF2, uses the distance between the CG and the geometrical centre of the ULD in the XY plane. Parameter `uldOcptSort` describes the ordering of the loaded ULDs to be repacked in the second phase of the heuristic. If `uldOcptSort`=1, then the ULDs are sorted by non-increasing occupied volume, while if `uldOcptSort`=2, they are sorted by non-decreasing occupied volume (see page 100). Parameter `boxSort` is equal to 1 if boxes are sorted according to the Clustered Area-Height before being packed and is equal to 2 if boxes are sorted by decreasing base area. The parameter δ is the percentage used to define the Clustered Area-Height sorting rule (see page 92). Therefore, if `boxSort`=2, then δ is not addressed (represented by N.A. in Table 6.3).

After 5000 experimentations, `irace` states that the four best configurations are those presented in Table 6.3 (the first configuration has the best mean performance).

Table 6.3: Best configurations found by `irace` for the tailored two-phase constructive heuristic parameters (if `boxSort`=2, then the value of δ is not addressed - N.A.)

	β	`bestHeight`	merit function	`uldOcptSort`	`boxSort`	δ
#1	0.68	**1**	**MF1**	**2**	**2**	**N.A.**
#2	0.50	1	MF1	2	2	N.A.
#3	0.60	1	MF1	2	2	N.A.
#4	0.64	1	MF1	2	2	N.A.

These four configurations differ on the β value only. A reason could be the small influence of parameter β. This, among others, will be analysed in a sensitivity analysis in the next section. An interpretation of each tuned parameter value will be provided as well.

Note that the default parameters of `irace` have been modified for the tailored two-phase constructive heuristic because the first runs showed that the scenarios were heterogeneous: the algorithm with a given configuration had an inconsistent performance on different instances, that is some configurations perform well for a subset of the instances while they perform poorly for other subset. In other words, it seems that no configuration performs significantly better than the others. For this reason, two `irace` parameters have been modified to not prematurely discard configurations. The parameter that specifies how many instances are evaluated between elimination tests has been increased as well as the parameter that specifies how many instances are evaluated before the first elimination test. Considering the increase of experimentations required in each race, the maximum number of experimentations has been set to 5000.

Considering the speed of the tailored two-phase heuristic and its capacity to solve instances with a large number of boxes, `irace` has also run with larger training instances (from 10 to 100 boxes) and the same best configuration is obtained, except for β (which is equal to 0.53) but as mentioned already this parameter does not have a significant impact on the solution quality.

6.3 Sensitivity analysis

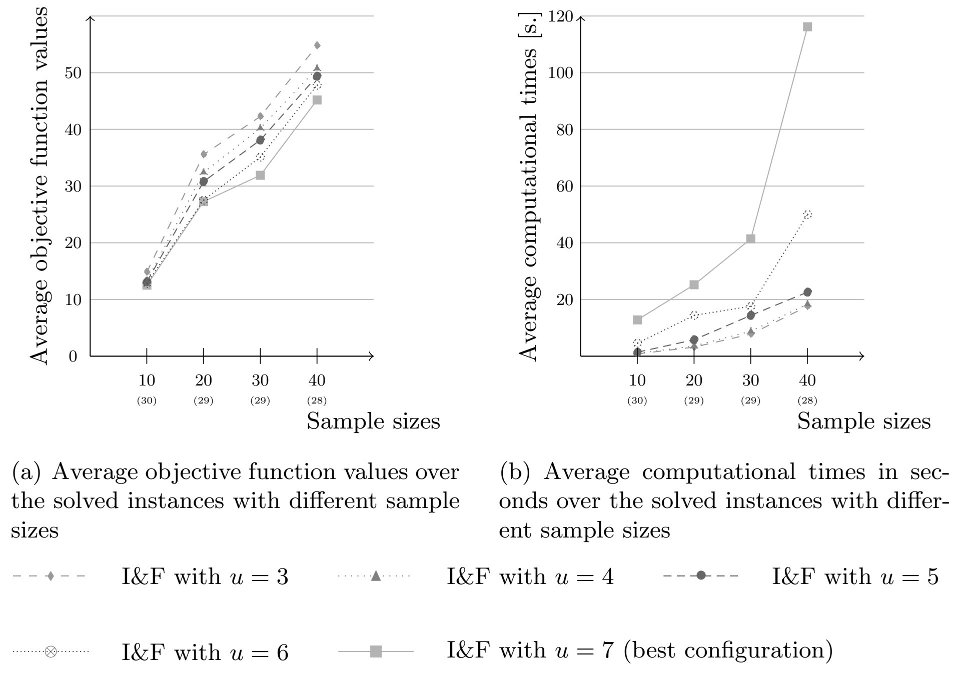
In the previous section, all parameters of each heuristic have been reminded and tuned with the `irace` package. It would be interesting to see the impact of the variation of each parameter in terms of solution quality. It is also an opportunity to verify the choices made by `irace`. In particular, `irace` used instances with a number of boxes ranging from 10 to 30. This section checks whether the decisions proposed by `irace` are still valid for larger instances, especially for the tailored two-phase heuristic which is able to solve larger instances than those of the training data set. For every heuristic, different values for one parameter are tested on the final data sets defined above while the other parameters are set to the values of the best configuration found by `irace`. Note that for every experiment, the matheuristics have a running time limited to one hour.

6.3.1 Insert-and-Fix

In this section, parameters u and q are set to different values to measure their impact on the solution quality. Associated computational times are also analysed.

Parameter u

Parameter u represents the number of boxes which have to be packed at each step, i.e., whose variables have to be determined. In this section, parameter u will range from 3 to 7 while parameter q is set to 2, its optimal value according to `irace`. Average objective function values and average computational times are shown in Figure 6.3. Both pieces of information are calculated over the instances which have been successfully solved by the five configurations. More precisely, the configuration with $u = 6$ failed for one instance with 20 boxes



(a) Average objective function values over the solved instances with different sample sizes

(b) Average computational times in seconds over the solved instances with different sample sizes

Figure 6.3: Comparison of different values of u for the I&F heuristic for different sample sizes (each sample size has 30 instances)

Note. The average objective function values and computational times are calculated over the instances which have been solved by all the five configurations. The number of such instances is indicated below the sample sizes along the x-axis.

and for two instances with 40 boxes, while the configuration with $u = 7$ failed for one instance with 30 boxes and for the same two instances with 40 boxes.

It can be seen on the left-hand side of Figure 6.3 that the solution quality increases, i.e., the average objective function values are smaller, if parameter u has a larger value. This behaviour can be expected: if u is small, then few boxes are packed at each step and this can lead to numerous selections of small ULDs, while when u is large, a broader view of the problem is considered and fewer large ULDs may be used. The average computational times for every configuration are represented on the right-hand side of Figure 6.3. One can notice that the larger the value of u, the larger the computational times. Once again, this observation can be expected: the larger u, the closer to the original MILP. This means that on one hand, the solution quality is improved, but on the other hand, each subproblem takes a larger amount of time to be solved. This represents the typical trade-off existing between computational times and solution quality.

Parameter q

Parameter q denotes the number of boxes whose variables are fixed at each iteration. In this section, parameter q will take values 1, 2 and 3, while parameter u is fixed to 7, its optimal value according to `irace`. Table 6.4 shows three pieces of information for each value of q and for each sample size: first, the number of unsolved instances, second the average computational times calculated over the instances solved by the three configurations and third the average objective function values also calculated over the instances solved by the three configurations. In Table 6.4, one can see that the configuration with $q = 1$ gives the best average objective function value in general, but this configuration takes a large amount of time and has thus more unsolved instances. This slowness can be expected since, if q is small, the MILP to be solved has less fixed variables and thus more decisions have to be taken in the subproblems. This may explain why `irace` discarded the configuration $u = 7$ and $q = 1$ in its experimentations. As a reminder, when a configuration was not able to solve an instance within one hour, it returned a large number to indicate to `irace` that it did not perform well. The difference between the configurations with $q = 2$ and $q = 3$ with respect to the average objective function values is not clear as shown in Table 6.4. The average objective function value is slightly better for $q = 2$ than for $q = 3$ for instances with 10, 30 and 40, which may explain the decision of `irace`. However, this behaviour is not noticed for larger instances. It seems that the configuration with $q = 2$ is generally faster than the configuration with $q = 3$, but may encounter some problems with some instances for which it is not able to find a feasible solution within on hour. The influence of parameter q seems minor with respect to the solution quality.

6.3.2 Fractional Relax-and-Fix with Coord

In this section, the impact of the three parameters u, v and q is analysed. For this purpose, each parameter will take different values while the other parameters are fixed to the value found by `irace`.

Table 6.4: Comparison of different values of q for the I&F heuristic for different sample sizes (each sample size has 30 instances)

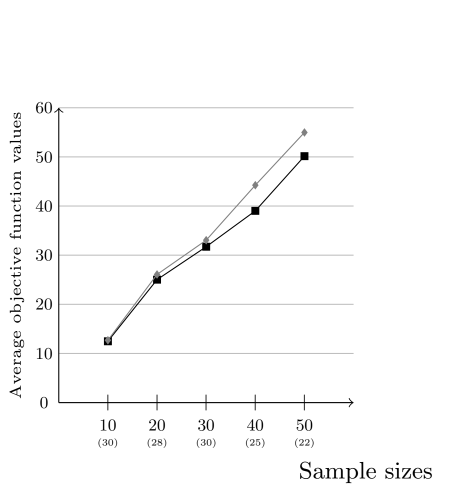
	$n = 10$	$n = 20$	$n = 30$	$n = 40$	$n = 50$	$n = 60$
	# unsolved instances					
$q = 1$	0	1	3	3	2	1
$q = 2$ (best)	0	0	1	2	2	1
$q = 3$	0	1	0	2	1	0
	Average computational times [s.]					
$q = 1$	17.38	38.85	93.12	249.74	311.27	453.85
$q = 2$ (best)	12.73	25.25	40.58	115.22	106.32	158.82
$q = 3$	14.44	12.83	67.50	175.12	195.82	192.34
	Average objective function values					
$q = 1$	12.31	26.26	30.92	42.38	56.69	72.83
$q = 2$ (best)	12.52	26.92	32.68	44.10	57.40	77.15
$q = 3$	12.69	26.50	32.86	44.80	56.57	74.77

Note. The average objective function values and computational times in seconds are calculated over instances which have been solved by the three configurations.

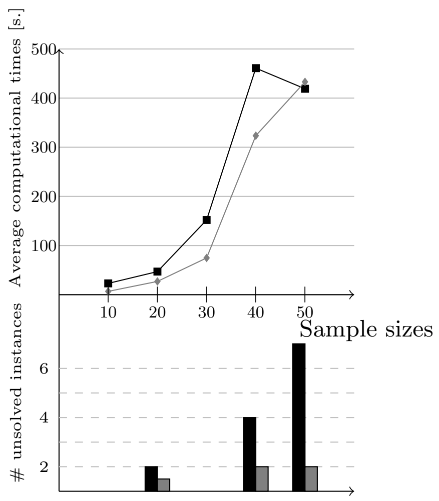
Parameter u

As for the I&F heuristic, parameter u describes the number of boxes whose variables have to be determined. The solution quality and the computational times are analysed when parameter u is equal to 6 and 7 (optimal according to `irace`), while parameter v is set to 5 and q to 1.

Figure 6.4 shows the average objective function values (left-hand side) as well as information about the computational times (right-hand side) for these two configurations for different sample sizes. The average objective function values and average computational times are computed over the instances solved by the two configurations. The number of these instances is indicated on the x-axis of the plot on the left-hand side. One can see that the average objective function values are smaller with the configuration with $u = 7$ as announced by `irace`. However, this configuration takes a large amount of time on average to solve the instances as indicated on the right plot in Figure 6.4. For this reason, the number of unsolved instances within one hour computational time is also larger for the configuration with $u = 7$ than for the configuration with $u = 6$. As for the sensitivity of parameter u for the I&F heuristic, this observation can be expected: the larger u, the larger the number of variables in the subproblems. This leads to a better solution quality, but requires a larger amount of computational time.



(a) Average objective function values over the solved instances with different sample sizes



(b) Average computational times in seconds over the solved instances with different sample sizes

◆ FRF Coord with $u = 6$ ■ FRF Coord with $u = 7$

Figure 6.4: Comparison of different values of u for the FRF heuristic with the block Coord for different sample sizes (each sample size has 30 instances)

Note. The average objective function values and computational times are calculated over the instances which have been solved by the two configurations. The number of such instances is indicated below the sample sizes along the x-axis on the left plot.

Parameter v

The influence of parameter v is analysed in this section. The objective function values and the computational times are observed when parameter v ranges from 2 to 6, while parameter $u = 7$ and parameter $q = 1$. As a reminder, parameter v controls the number of boxes whose coordinate variables are integer, the coordinate variables related to boxes from $v + 1$ to u have the integrality restriction relaxed. As a consequence, parameter v influences the number of steps in the algorithm since the FRF heuristic stops as soon as $q \times \#\text{step} + v \geq n$ as explained in Section 4.3. Moreover, parameter v also impacts on the difficulty of the subproblems since it controls the number of integer variables. If v is small, then it leads to a large number of steps but the subproblems to be solved are easier while if v is large, there are fewer iterations but they are more difficult to solve.

Table 6.5 shows three pieces of information for each value of v and for each sample size: first, the number of unsolved instances, second the average computational times calculated over the instances solved by the five configurations and third the average objective function values also calculated over the instances solved by all the configurations. The number of instances solved by the five configurations is indicated in Table 6.5 below the number of unsolved instances for each configuration. Looking at the first part of Table 6.5, one can observe that some instances which cannot be solved differ from one configuration to another. As a consequence, only two thirds of the instances with 40 boxes can be solved by all the configurations, which reduces the possible comparisons. The configuration with $v = 4$ has a high number of failures for 30 boxes to be packed and configurations with $v = 3$ and $v = 5$ have one sixth of the instances with 40 boxes unsolved. In terms of solution quality, the configuration with $v = 2$ gives the best average objective function value for instances with 10 and 20 boxes but not for larger instances. Despite `irace` states the configuration with $v = 5$ is the best configuration in terms of solution quality, the results in Table 6.5 do not clearly indicate this tendency. Parameter v seems to have a minor influence on the solution quality.

Parameter q

In this section, parameter q ranges from 1 to 3, while parameter u is equal to 7 and parameter v is equal to 5. Parameter q describes the number of boxes whose variables are kept and fixed from one iteration to the next. As a consequence, parameter q also influences the complexity of the subproblems since it controls the number of variables to be determined. Parameter q has

Table 6.5: Comparison of different values of v for the FRF heuristic with the block Coord for different sample sizes (each sample size has 30 instances)

	$n = 10$	$n = 20$	$n = 30$	$n = 40$
	# unsolved instances			
$v = 2$	0	1	1	3
$v = 3$	0	1	0	5
$v = 4$	0	1	4	3
$v = 5$ (best)	0	2	0	4
$v = 6$	0	1	2	5
# solved instances	30	27	24	20
	Average computational times [s.]			
$v = 2$	24.55	26.43	153.45	398.94
$v = 3$	19.73	40.96	196.32	336.49
$v = 4$	21.91	27.88	174.83	301.31
$v = 5$ (best)	23.20	38.63	161.75	489.27
$v = 6$	30.18	71.89	93.44	290.70
	Average objective function values			
$v = 2$	12.39	23.81	31.79	40.03
$v = 3$	12.44	24.61	30.71	38.22
$v = 4$	12.44	24.30	31.65	39.17
$v = 5$ (best)	12.48	24.33	32.40	38.19
$v = 6$	12.44	24.76	32.23	39.36

also an impact on the number of steps of each configuration for the same reason as parameter v. As shown in Table 6.6, the average objective function values are smaller for configuration with $q = 1$ as predicted by `irace`. This is understandable since a small value of q gives the heuristic more flexibility from one iteration to another. However, the average duration is longer for this configuration which can be explained by an increase in the number of variables to be determined.

Table 6.6: Comparison of different values of q for the FRF heuristic with the block Coord for different sample sizes (each sample size has 30 instances)

	$n = 10$	$n = 20$	$n = 30$	$n = 40$
	# unsolved instances			
$q = 1$ (best)	0	2	0	4
$q = 2$	0	1	1	2
$q = 3$	0	3	0	3
# solved instances	30	27	29	24
	Average computational times [s.]			
$q = 1$ (best)	23.20	26.45	144.59	566.78
$q = 2$	18.28	18.73	84.57	203.88
$q = 3$	20.57	32.80	76.30	260.12
	Average objective function values			
$q = 1$ (best)	12.48	24.87	31.99	39.03
$q = 2$	12.56	26.50	31.45	40.57
$q = 3$	12.69	26.96	32.11	41.50

6.3.3 Fractional Relax-and-Fix with Coord & p_{ij}

Considering the values provided by `irace`, only the sensitivity of parameters v and q are analysed in this section.

Parameter v

To evaluate the sensitivity of parameter v, parameter u is set to 7 and parameter q to 1. Parameter v ranges from 2 to 6, where 6 is the value given by `irace`.

As for the previous FRF heuristic, Table 6.7 shows three pieces of information for each value of v and for each sample size: first, the number of unsolved instances, second the average computational times calculated over the instances solved by the five configurations and third the average objective

Table 6.7: Comparison of different values of v for the FRF heuristic with the block Coord & p_{ij} for different sample sizes (each sample size has 30 instances)

	$n = 10$	$n = 20$	$n = 30$	$n = 40$
	# unsolved instances			
$v = 2$	1	4	4	13
$v = 3$	0	3	3	7
$v = 4$	0	0	4	4
$v = 5$	0	0	0	4
$v = 6$ (best)	0	1	1	3
# solved instances	29	24	20	10
	Average computational times [s.]			
$v = 2$	30.21	138.51	224.40	629.69
$v = 3$	18.51	85.85	275.79	764.12
$v = 4$	20.99	54.60	157.50	317.50
$v = 5$	129.31	39.00	102.64	294.87
$v = 6$ (best)	**13.81**	**21.31**	**56.57**	**161.89**
	Average objective function values			
$v = 2$	12.69	28.05	35.91	46.68
$v = 3$	12.65	26.08	36.03	49.24
$v = 4$	12.68	25.93	34.33	46.74
$v = 5$	12.53	25.68	33.49	**44.41**
$v = 6$ (best)	**12.39**	**25.11**	**32.65**	44.66

function values also calculated over the instances solved by all the configurations. The number of instances solved by the five configurations is indicated in Table 6.7 below the number of unsolved instances for each configuration.

As in Table 6.5, one can observe in Table 6.7 that some instances which cannot be solved differ from one configuration to another. As a consequence, only one third of the instances with 40 boxes can be solved by all the configurations, which reduces the possible comparisons. One can see that the configuration with $v = 6$ is often the configuration that solves the largest number of instances for every sample size. Moreover, even among the instances solved by all the configurations, the configuration with $v = 6$ is the fastest configuration as indicated in bold in Table 6.7. A high value of v reduces the number of steps in the algorithm which may explain the speed of the configuration with $v = 6$. With respect to the solution quality, this configuration is also able to provide solutions with the highest average quality.

Parameter q

In this section, parameter q ranges from 1 to 3, while parameter u is equal to 7 and parameter v is equal to 6. As shown in Table 6.8, the average objective function values are smaller for configuration with $q = 1$ as predicted by `irace`. As for the other FRF heuristic, this may be expected since a small value of q gives the heuristic more flexibility from one iteration to another. However, unlike the other FRF heuristic, the average duration is not longer for the configuration with $v = 6$ and the number of unsolved instances is not larger for this configuration.

Table 6.8: Comparison of different values of q for the FRF heuristic with the block Coord & p_{ij} for different sample sizes (each sample size has 30 instances)

	$n = 10$	$n = 20$	$n = 30$	$n = 40$
	# unsolved instances			
$q = 1$ (best)	0	1	1	3
$q = 2$	0	1	0	5
$q = 3$	0	1	1	4
# solved instances	30	29	28	25
	Average computational times [s.]			
$q = 1$ (best)	14.88	49.91	79.09	262.80
$q = 2$	12.26	13.48	47.22	237.80
$q = 3$	16.10	87.69	52.60	147.46
	Average objective function values			
$q = 1$ (best)	**12.39**	**26.49**	**32.67**	**43.96**
$q = 2$	12.72	26.73	33.34	44.10
$q = 3$	12.78	27.63	34.17	45.28

6.3.4 A tailored two-phase constructive heuristic

The tailored two-phase constructive heuristic has six parameters whose values have been determined with `irace`. To analyse the sensitivity of each parameter, the values of the five other parameters are set to the values of the best configuration found by `irace` and presented in the first row of Table 6.3.

Parameter β

Parameter β has been introduced for the *Not Too Low* constraint of the packing algorithm and controls the minimum height that the top face of a fragile box has to reach during the first part of the packing algorithm. This parameter has been introduced to avoid to waste space on top of fragile boxes early in the algorithm. A high value means that the *Not Too Low* constraint is very restrictive and thus fragile boxes are more likely assigned to list L_2. Conversely, a small value of β means that the constraint is easily satisfied. The values of the different parameters of the tested configurations are:

β	`bestHeight`	merit function	`uldOcptSort`	`boxSort`	δ
[0.50,0.99]	1	MF1	2	2	N.A.

The differences between the results obtained with the different configurations are very small (most of the time smaller than 1%) and cannot be clearly seen on a plot unless a considerable scale is used. For this reason, the results are shown in Table 6.9.

Table 6.9: Average objective function values over the 30 instances for each sample size and for different values of β

	Number of boxes									
β	10	20	30	40	50	60	70	80	90	100
0.50	15.46	31.71	38.55	52.30	62.37	75.37	82.79	91.92	98.44	108.31
0.55	15.46	31.71	38.55	52.30	62.37	75.20	82.83	91.92	98.44	108.31
0.60	15.46	31.71	38.55	52.30	62.37	75.20	82.83	91.92	98.44	108.31
0.65	15.46	31.71	38.55	52.30	62.37	75.20	82.83	91.93	98.44	108.31
0.70	15.46	31.54	38.55	52.30	62.41	75.20	82.83	92.14	98.41	108.31
0.75	15.46	31.54	38.55	51.90	62.46	74.63	83.20	92.01	98.41	108.55
0.80	15.46	31.54	38.55	51.90	62.46	74.63	83.20	92.01	98.41	108.55
0.85	15.46	31.54	38.72	51.90	62.49	74.80	83.36	92.01	98.41	108.72
0.90	15.46	31.54	38.72	51.73	62.49	74.80	83.36	92.01	98.57	108.72
0.95	15.46	31.54	38.72	51.73	62.49	74.80	83.36	92.01	98.57	108.55
1.00	15.46	31.71	38.89	51.90	62.49	74.80	83.44	92.01	98.67	108.71

It seems that the solution quality can be slightly deteriorated for some sample sizes (e.g., 30, 50, 70 and 90) when β is close to 1. However, for the remaining values, it seems that there is no value distinctly better than the others. This parameter does not seem to have a noticeable influence on the solution quality. This observation explains the difficulty for `irace` to find a value for this parameter.

Parameter `bestHeight`

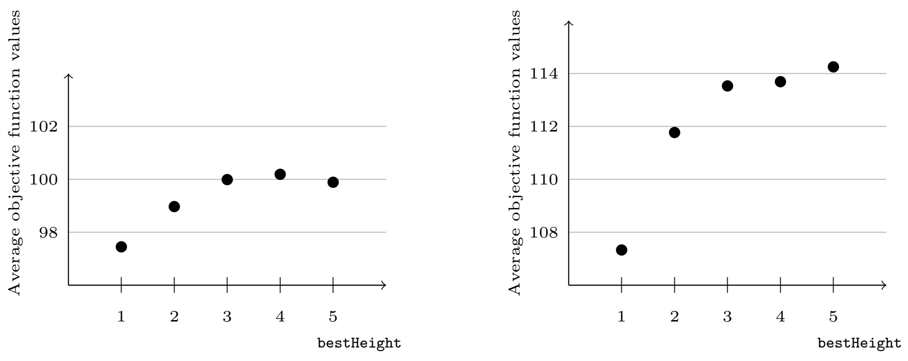
Parameter `bestHeight` controls the EP selection for accommodating a box during the packing algorithm. The merit function first sorts the EPs and among the first `bestHeight` ones, the EP leading to the lowest CG is selected. The aim is thus to keep the CG of the packing as low as possible. `irace` suggests to select the first EP regarding the merit function only. `irace` states that it is not necessary to keep and sort the first EPs with respect to the CG height. The values of the parameters of the tested configurations are:

β	`bestHeight`	merit function	`uldOcptSort`	`boxSort`	δ
0.68	**[2,5]**	MF1	2	2	N.A.

As shown in Table 6.10, the average objective function values tend to increase when the value of `bestHeight` increases, as predicted by `irace`. This behaviour seems clear from 1 to 3 but is less obvious when ranging from 3 to 5 as shown in Figure 6.5 for instances with 90 and 100 boxes.

Table 6.10: Average objective function values over the 30 instances for each sample size (n is the number of boxes in each sample) and for the different possible values of parameter `bestHeight`

	Number of boxes									
`bestHeight`	10	20	30	40	50	60	70	80	90	100
1 (best)	15.46	31.54	38.55	52.30	62.41	75.20	82.83	91.93	98.41	108.31
2	15.82	31.65	40.29	54.29	63.37	77.78	85.87	93.69	99.94	112.75
3	15.98	32.32	40.47	53.83	63.82	76.94	86.82	95.40	100.96	114.51
4	15.98	32.36	40.68	53.52	63.12	77.21	86.76	95.80	101.16	114.65
5	16.01	32.31	40.71	53.69	63.66	76.94	86.93	95.43	100.86	115.24



(a) Average objective function values over the 30 instances with 90 boxes

(b) Average objective function values over the 30 instances with 100 boxes

Figure 6.5: Average objective function values when parameter `bestHeight` ranges from 1 to 5

Merit function

The merit function is the function used to sort and then to select the EP for packing a box. MF1 is based on the RS around the EP while MF2 considers the distance between the CG and the geometrical centre of the ULD. According to `irace`, MF1 gets better solution quality than MF2.

The values of the parameters of the tested configurations are:

β	`bestHeight`	merit function	`uldOcptSort`	`boxSort`	δ
0.68	1	**MF2**	2	2	N.A.

A comparison of the average objective function values per sample size for these two merit functions is presented in Figure 6.6. A representation with the standard deviations can be found in Appendix A.3, Figure A.1.

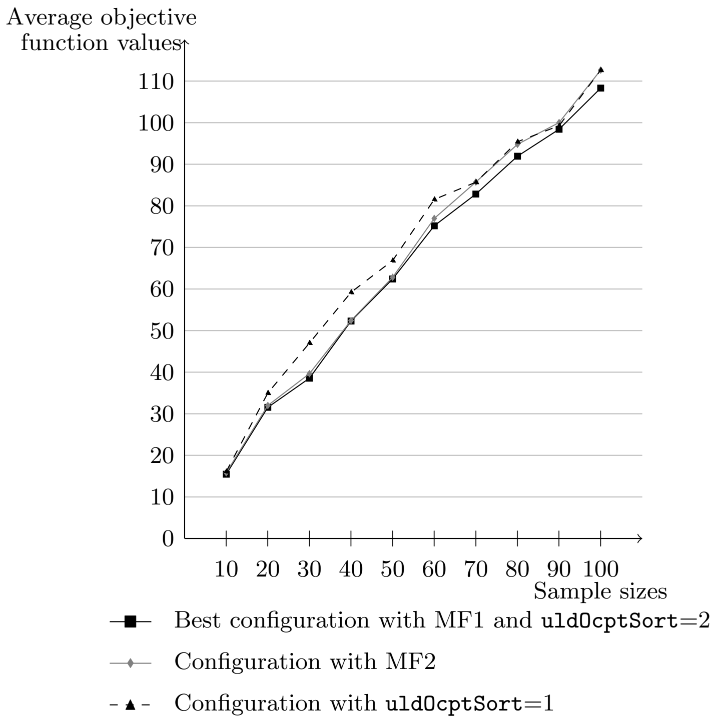


Figure 6.6: Comparison of the average objective function values per sample size for different configurations (each sample size has 30 instances)

As can be seen in Figure 6.6, the solution quality is slightly better with MF1 as stated by `irace`. This behaviour can be expected since the aim of the RS utilisation is to exploit the space in the best way, choosing the minimum volume able to accommodate the given box. However, the goal of the function MF2 is to minimise the distance between the CG and the geometric centre of the ULD. The weight distribution is improved with MF2 and the number of observed deviations is therefore reduced compared to the use of MF1, as shown in Figure 6.7. In Figure 6.7, it can be seen that MF1 has a larger number of unbalanced ULDs than with MF2 before the J&S operators are applied to improve weight distribution, and this is still true after J&S operator application. Therefore, MF2 fulfils its role with respect to the weight distribution.

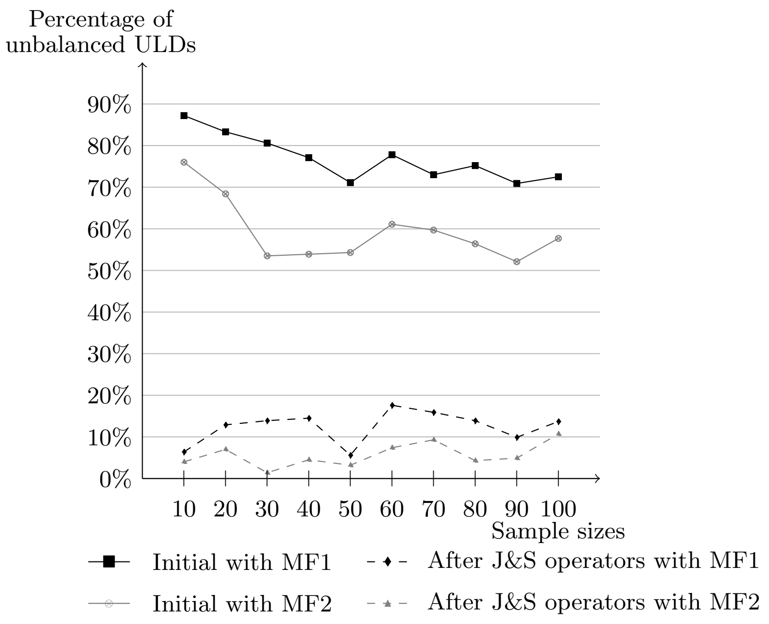


Figure 6.7: Average percentage of unbalanced ULDs per sample size (each sample size has 30 instances)

Parameter `uldOcptSort`

Parameter `uldOcptSort` controls the sequence according to which the ULDs are unpacked and repacked in the second phase of the tailored constructive heuristic. ULDs are considered either by non-increasing or by non-decreasing loaded volume. `irace` states that the solution quality is better if the ULDs are sorted by non-decreasing loaded volume.

The values of the parameters of the tested configurations are:

β	`bestHeight`	merit function	`uldOcptSort`	`boxSort`	δ
0.68	1	MF1	**1**	2	N.A.

The intuition based on the preliminary tests explained on page 100 turns out to be verified by `irace`. The average objective function values obtained with the configuration with `uldOcptSort`=1 is also represented in Figure 6.6. The configuration with `uldOcptSort`=1 seems to give larger objective function values on average. This behaviour may be explained by the fact that some ULDs have few packed boxes but with special shapes. These boxes can thus be hard to repack in smaller types of ULDs. The decision made by `irace` seems the most appropriate even for larger sample sizes.

Parameter `boxSort`

`irace` states that it is more efficient in terms of solution quality to sort the boxes to be packed by decreasing base area. To have an idea of the values obtained with the Clustered Area-Height as box sorting operator, different values of δ are tested. The values of the parameters of the tested configurations are:

β	`bestHeight`	merit function	`uldOcptSort`	`boxSort`	δ
0.68	1	MF1	2	**1**	**[10,90]**

As can be seen in Figure 6.8, the obtained results show that the larger δ, the larger the average objective function values. This observation can be expected: if δ is large, then there are few clusters among which some can hold a lot of boxes. Inside the clusters, boxes are sorted by decreasing height and this sorting is apparently less effective in terms of solution quality. This also explains that the results with $\delta = 10$ are close to those provided by a decreasing base area sorting. If δ is small, there are a lot of clusters, possibly containing few boxes. As explained on page 92, the clusters with the largest bases are considered first and thus the sorting tends to be a decreasing base area sorting. The objective function values obtained with $\delta = 30, 40, 50, 60, 70, 80$ and 90 are similar for all the size of samples as shown in Figure 6.9 and thus all of these configurations are not represented in Figure 6.8. Based on these observations, the choice made by `irace` seems the most appropriate, even for larger instances.

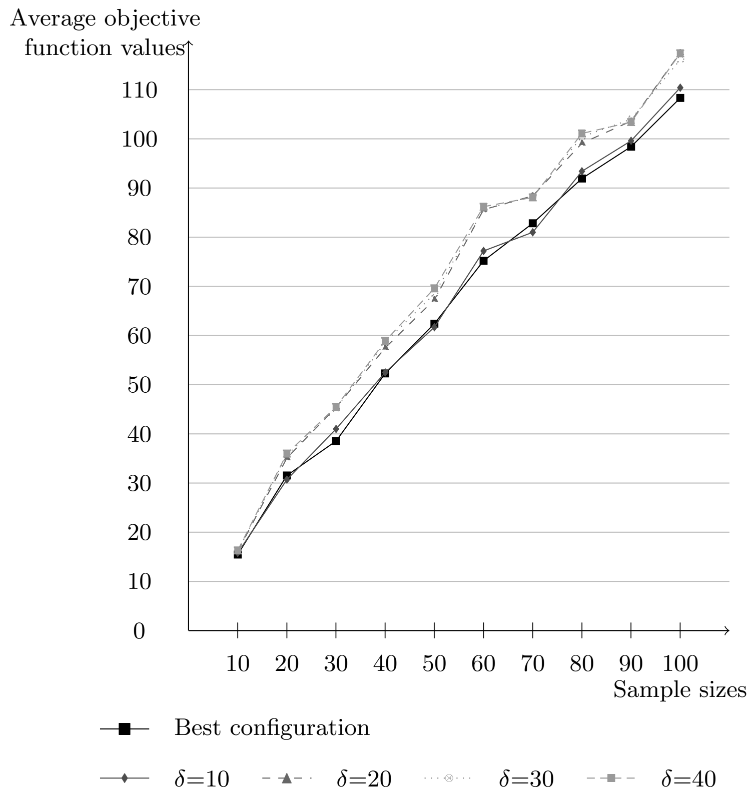
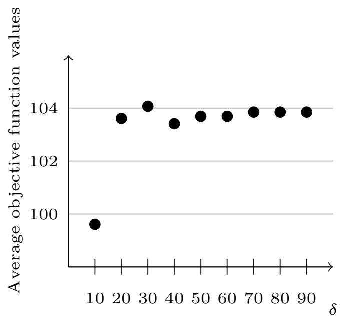
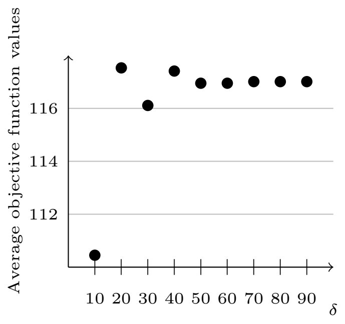


Figure 6.8: Comparison of the average objective function values per sample size for different configurations (each sample size has 30 instances)



(a) Average objective function values over the 30 instances with 90 boxes



(b) Average objective function values over the 30 instances with 100 boxes

Figure 6.9: Average objective function values when `boxSort`=1 and parameter δ takes value from 10 to 90

6.4 Results of the parametrised heuristics

In this section, the results of the experimentations carried out with the three matheuristics and the tailored two-phase constructive heuristics on the final data sets are analysed on different aspects: the computational times and the filling rates. The latter shows how efficiently the loading space is used inside the ULD. However, the constraint related to the height of the CG may prevent a high filling rate from happening. The maximum height for the CG is often around 50% of the ULD height. In practice, it is difficult to provide an average filling rate because it depends on the cargo and may fluctuate from one flight to another. It would seem that the average utilisation is around 55% and hardly ever above 80%. In particular, airlines will try to exploit the volume of ULDs, but it may happen that ULDs with low filling rates are loaded because it is unusual to keep the cargo for a next flight. In particular, cargo sent by plane is often time-sensitive.

6.4.1 Insert-and-Fix

Two elements about the solutions obtained with the I&F are observed and analysed in this section. First, a good indicator of the solution quality is the filling rate of the used ULDs. Second, the computational times are observed. An a priori disadvantage of the matheuristics developed in this thesis is the computational duration which tends to increase considerably when the number of boxes is larger. These two aspects are now analysed.

Note that backtracking has never been activated over all the experiments with the I&F.

Filling rate analysis

To represent the trend of the filling rates for each sample size, boxplots are drawn in Figure 6.10. In addition to the sample sizes, the number of ULDs used to solve each set of 30 instances is shown in brackets, next to the sample sizes, along the y-axis. For instance, 225 ULDs are used to pack the boxes of the 30 instances with 80 boxes. These numbers of ULDs thus represent the number of observations represented in the boxplots.

When looking closely at Figure 6.10, the size of the interquartile ranges does not seem to show a clear trend, but the ranges themselves slightly move to the right when the sample size increases from 10 to 30. This means that the middle 50% of observations has a filling rate growing with the number of boxes to be packed. The end of the right whisker takes larger values when the sample size increases from 10 to 30 and then remains almost constant for larger sizes. This remark can be expected since a minimum number of boxes or a minimum volume may be needed to achieve large filling rates. Note that for instances with at least 40 boxes, almost half of the ULDs have a filling rate greater than 50%. Moreover, one ULD used for an instance with 40 boxes reaches a filling rate of almost 80%. All these observations indicate that the I&F is able to correctly exploit loading space inside the ULDs.

However, for any sample size, there exist one or several ULDs with low filling rates as shown in Figure 6.10. The length of the right whisker seems rather constant, but the number of outliers on the left increases with the sample size, which is typical of a distribution with a negative skewness. This is not surprising. It may happen that at the end of the algorithm, the last boxes, i.e., the smallest boxes since they are sorted by decreasing volume, cannot be packed in already used ULDs and thus lead to the opening of a new ULD.

Computational times

Average computational times needed to solve the final instances with the I&F are shown in Figure 6.11. The upper part of the figure shows the average times which are computed over the instances that have been solved in less than one hour computational time for each sample size. As a reminder, each run of the I&F algorithm has a time limit of one hour to solve each instance. If the computation lasts one hour, then at the end of that duration, the algorithms stops and either returns a feasible (but suboptimal) solution

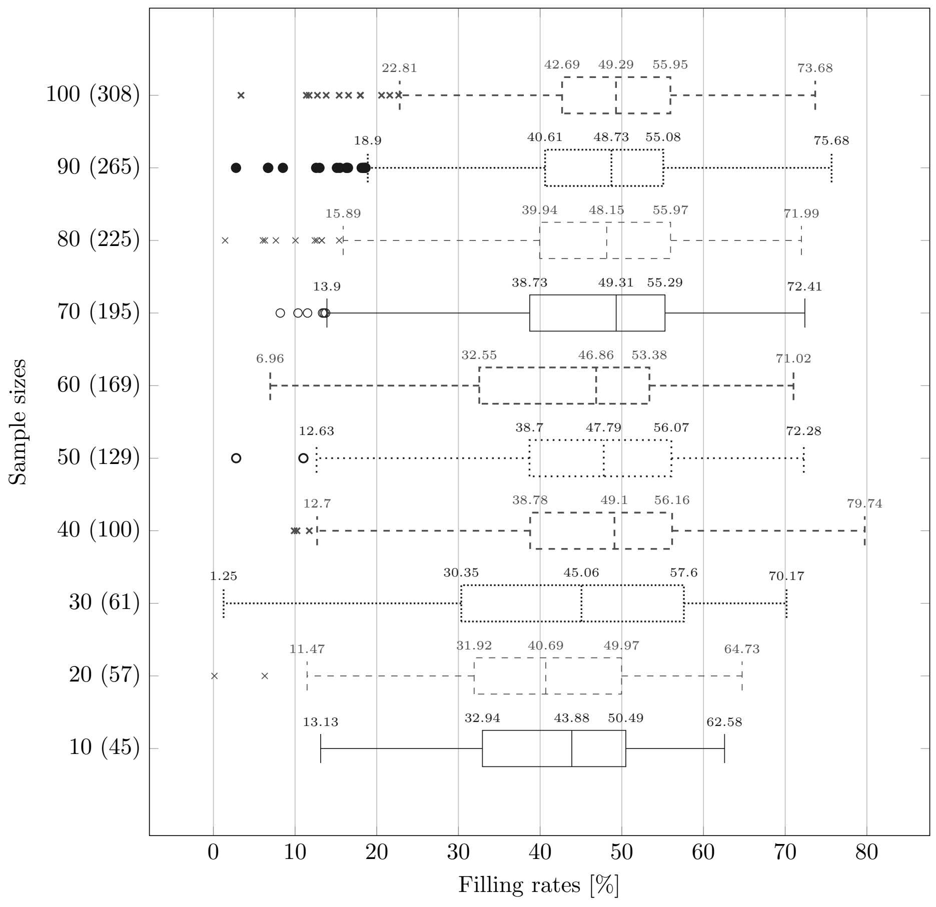


Figure 6.10: Filling rates in percent per sample size (each sample size has 30 instances) for the I&F heuristic. The number in brackets denotes the number of ULDs used for all 30 instances of each sample size

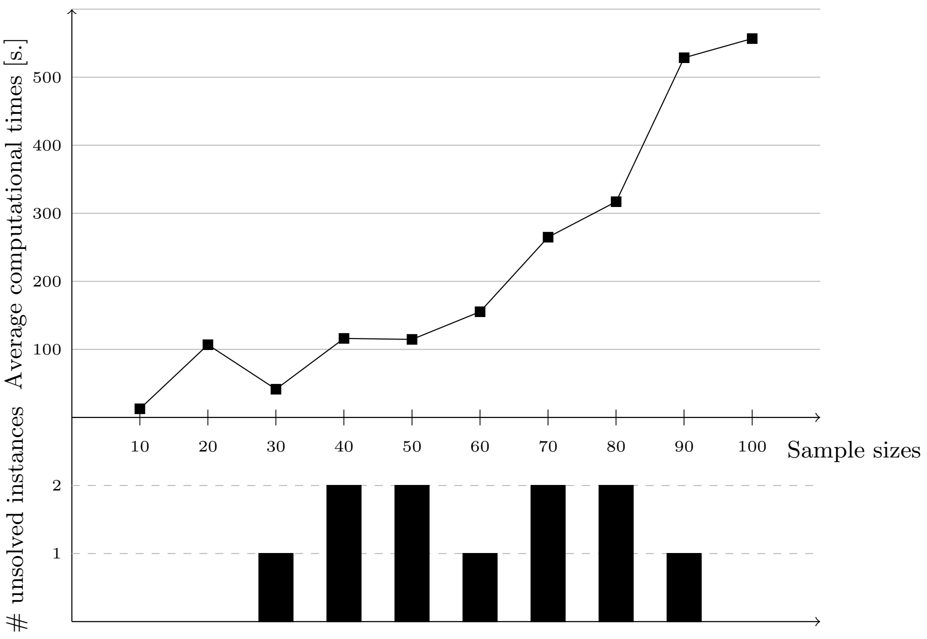


Figure 6.11: Average computational times in seconds and number of unsolved instances (out of 30) for the Insert-and-Fix heuristic for different sample sizes (the average is calculated over the solved instances)

or returns nothing if no feasible solution has been found. The number of unsolved instances is shown in the lower part of Figure 6.11. It can be seen that the times needed to solve the instances clearly increases when the number of boxes to be packed, i.e., the sample size, increases as well. This trend even becomes a problem for some instances since they cannot be solved within one hour. For instance, two instances out of the 30 with 80 boxes have not been solved in one hour, the average computational time is then calculated over the 28 solved instances.

6.4.2 Fractional Relax-and-Fix with the block Coord

As for the I&F heuristic, the filling rates of the ULDs used by the FRF heuristic with the block Coord and the computational times needed for the instances to be solved are analysed in this section. It will be shown that this method needs large amounts of time to solve instances with 50 boxes already. For this reason, only a subset of the final instances has been tested. To justify this decision, the computational times analysis is first presented hereinafter.

The filling rates of the used ULDs are then observed for the solved instances.

Note that backtracking has never been activated over all the experiments with the Fractional Relax-and-Fix with the block Coord.

Computational times

Figure 6.12 shows average computational times needed to solve the final data sets with the FRF heuristic with the block Coord for each sample size. As for the I&F, the upper part of Figure 6.12 shows the average times which are computed over the instances that have been solved within one hour computational time. The number of unsolved instances for each sample size is shown in the lower part of Figure 6.12. It can be seen that the times required to solve the instances and the number of unsolved instances clearly increase when the number of boxes to be packed increases as well. This trend even becomes an issue for some instances since they cannot be solved within one hour as shown in the lower part of Figure 6.12. Since the FRF heuristic with the block Coord is not able to find a feasible solution to seven instances out of the 30 with 50 boxes, the larger sample sizes have not been tested for this heuristic.

Filling rate analysis

The filling rate gives information about how efficiently the loading space of the ULDs is used. Figure 6.13 shows the boxplots of the filling rates of the ULDs. The observations are the used ULDs for all the solved instances. The number of these observations is indicated next to the sample sizes along the y-axis. In comparison to the I&F filling rates, only the instances with 10 to 50 boxes have been tested. There are thus less boxplots represented in Figure 6.13. Moreover, even for those sample sizes, some instances have not been solved, which means that the number of observations is also smaller for each sample size than in the case of the I&F. These two elements lead to less information about the distribution and thus to a less obvious tendency about the behaviour of the filling rates. However, it seems that the interquartile ranges and the right whiskers are slightly shifting to the right when the number of boxes to be packed is larger. This means that the 75% of the observed filling rates increase with the sample sizes. One can see that reasonable filling rates can be reached. Especially, one ULD used to solve an instance with 40 boxes is able to reach a filling rate of 82.14%. The FRF heuristic with the block Coord is thus able to use the space efficiently inside the ULDs but unfortunately is very time consuming. Because of this slowness, almost one fourth of the instances with 50 boxes have not been solved in one hour.

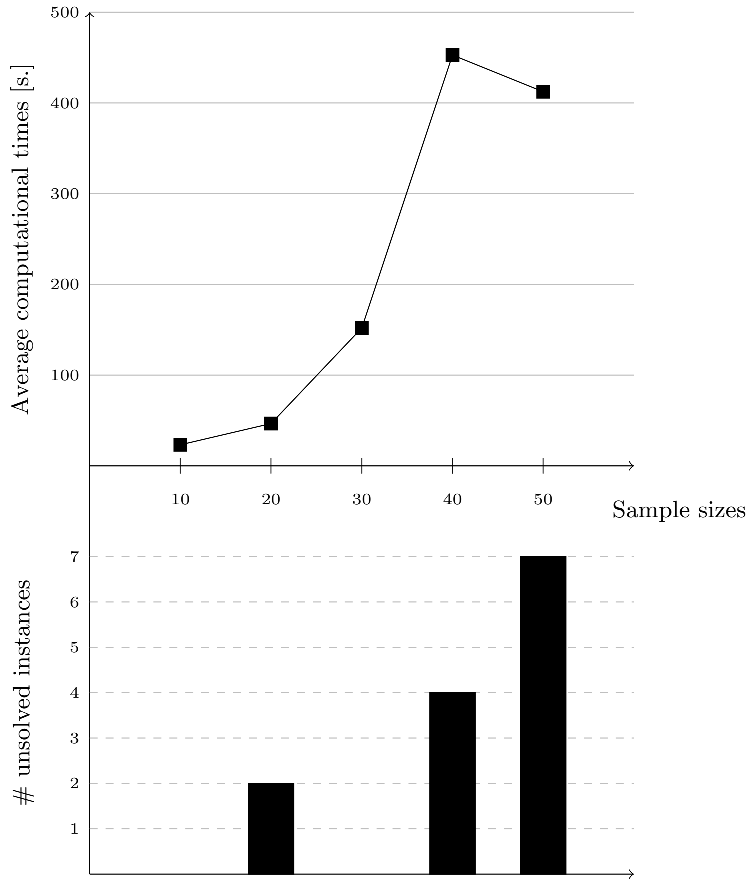


Figure 6.12: Average computational times in seconds and number of unsolved instances (out of 30) for the Fractional Relax-and-Fix heuristic with the block Coord for different sample sizes (the average is calculated over the solved instances)

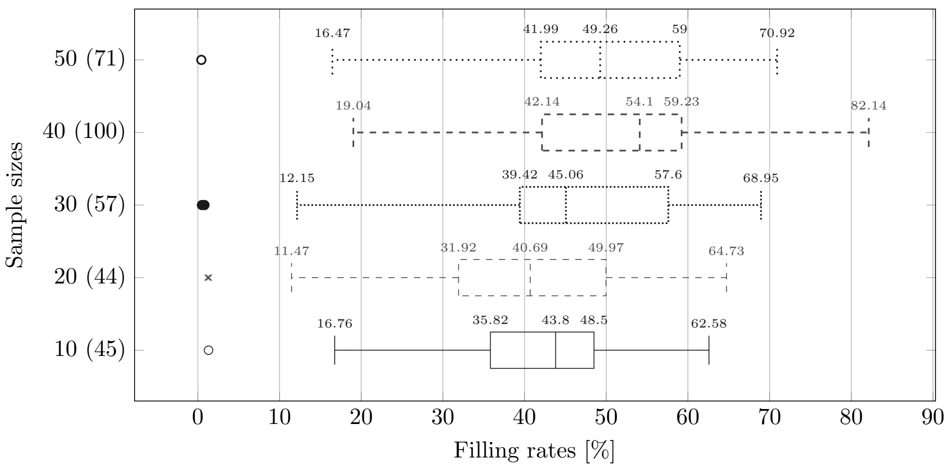


Figure 6.13: Filling rates in percent per sample size (each sample size has 30 instances) for the FRF heuristic with the block Coord. The number in brackets denotes the number of ULDs used for all 30 instances of each sample size

6.4.3 Fractional Relax-and-Fix with the block Coord & p_{ij}

As for the I&F and the FRF with the block Coord heuristics, both computational times and filling rates are analysed for the FRF heuristic with the block Coord & p_{ij}. It will be seen that the computational times explode when the number of boxes to pack increases. For this reason, the instances with 100 boxes have not been tested with the FRF heuristic with the block Coord & p_{ij}. To convince the reader, these computational durations are first presented and are followed by the filling rate analysis. Finally, the number of backtracks is provided for each sample size.

Computational times

Figure 6.14 presents the average computational times and the number of solved instances for each sample size for the FRF heuristic with the block Coord & p_{ij}. As can be seen, both elements are smaller than those of the FRF heuristic with the block Coord. For this reason, instances with up to 90 boxes have been tested with the FRF heuristic with the block Coord & p_{ij}. Despite this improved speed, computational times explode when the sample size becomes more important. This leads to an increase in the number of unsolved instances as shown in the lower part of Figure 6.14. Especially, 40% of the instances with 90 boxes have not been solved within one hour computational time.

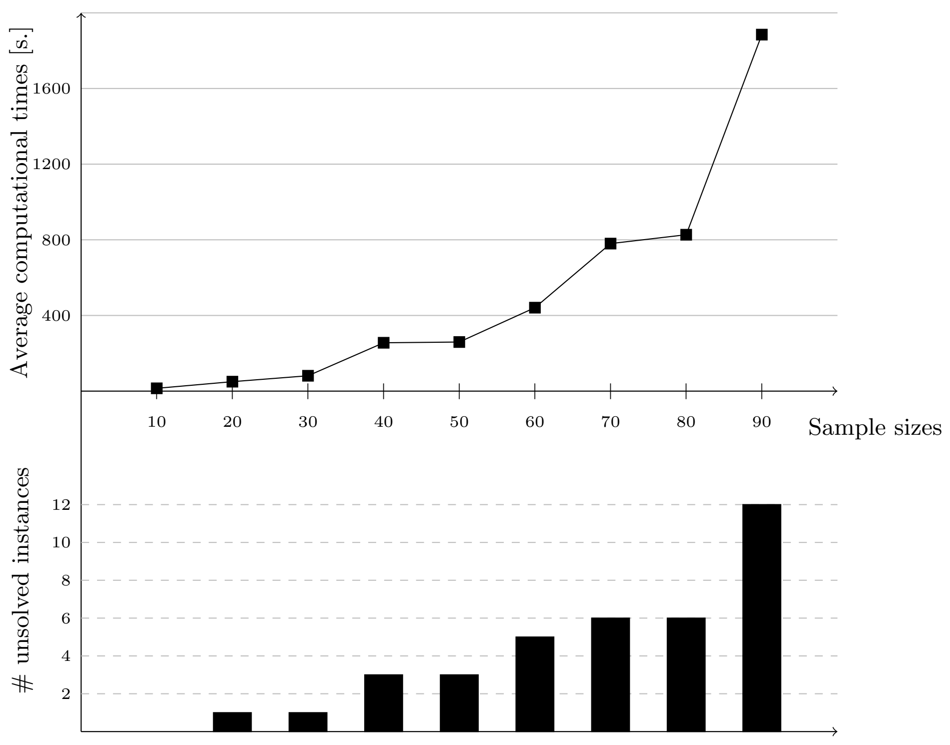


Figure 6.14: Average computational times and number of unsolved instances (out of 30) for the Fractional Relax-and-Fix with the block Coord & p_{ij} heuristic for the different sample sizes (the average computational times are calculated over the solved instances)

Filling rate analysis

The filling rate is an indicator of the solution quality. The filling rates of the used ULDs for each sample size are represented by boxplots in Figure 6.15. The number of observations is indicated in brackets, next to the sample sizes, along the y-axis.

Unlike the I&F and the FRF with the block Coord heuristics, the boxplots of this heuristic do not seem to move to the right when the sample size increases. The boxplots have almost the same median for all the sample sizes and they almost all show a distribution with a negative skewness. This last observation indicates that a lot of ULDs have a good filling rate.

At the first glance, there are fewer ULDs with a low filling rate in the

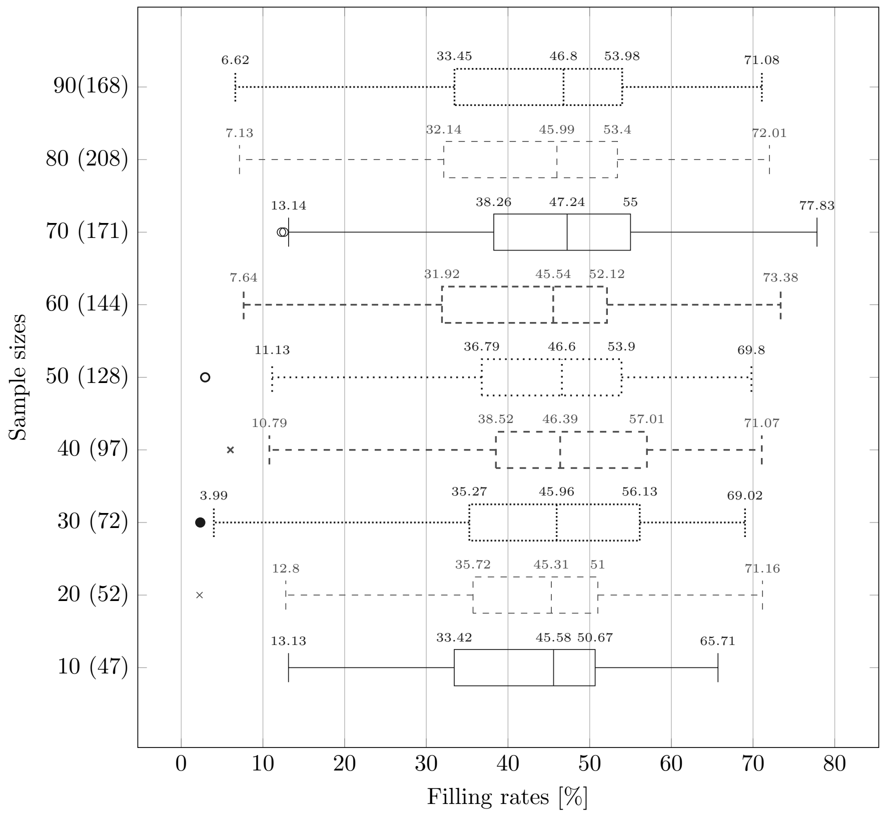


Figure 6.15: Filling rates in percent per sample size (each sample size has 30 instances) for the Fractional Relax-and-Fix with the block Coord & p_{ij} heuristic. The number in brackets denotes the number of ULDs used for all 30 instances of each sample size

solutions of the FRF heuristic with the block Coord & p_{ij} than for the I&F heuristic. The comparisons of the quality between the different methods is presented in Section 6.5.

Backtracking

Backtracking has been activated for several applications of the FRF heuristic with the block Coord & p_{ij}. More precisely, Table 6.11 shows the number of instances, over 30, for which backtracking has been applied for each sample size in the second column. The third column provides information about the number of times backtracking has been applied for the corresponding instances. For instance, 1 instance with 20 (resp. 40) boxes requires backtracking and it has been applied once (resp. 5 times) in the algorithm and 5 instances with 60 boxes required backtracking, it has been applied 3.8 times for each instance on average, with a minimum of 2 times and a maximum of 5 times.

Table 6.11: Number of backtracking application for the FRF heuristic with the block Coord & p_{ij}

Sample size	# instances	#backtracks [min, max], avg
$n = 10$	0	-
$n = 20$	1	$\{1\}$
$n = 30$	0	-
$n = 40$	1	$\{5\}$
$n = 50$	1	$\{1\}$
$n = 60$	5	$[2, 5]$, 3.8
$n = 70$	6	$[1, 8]$, 2.83
$n = 80$	3	$[2, 7]$, 4.67
$n = 90$	11	$[1, 17]$, 5.09

6.4.4 A tailored two-phase constructive heuristic

As for the MILP-based constructive heuristics, a good indicator of the solution quality is the filling rate of the used ULDs. Another important point for the tailored two-phase constructive heuristic is the weight distribution inside the loaded ULDs as explained in Section 5.3. This constructive heuristic has been developed to quickly find a feasible solution to the problem. The computational times are thus also observed.

Filling rate analysis

A good indicator to measure whether the space is efficiently exploited is the filling rates of all the loaded ULDs. To represent the global tendency of these filling rates, boxplots are drawn with all the ULDs used for the 30 instances of each sample size in Figure 6.16. In addition to the size of the samples, the number of used ULDs is provided along the y-axis.

In Figure 6.16, one can see that the larger the number of boxes, the higher the filling rates of the ULDs can be. In particular, half of the ULDs used with the instances of 100 boxes have a filling rate of at least 50%, one ULD even reaches a filling rate of 72.78%. This observation is noteworthy especially as the constraints ensuring that the height of the CG is below a given limit may prevent a too high filling rate. The tailored two-phase constructive heuristic is thus able to achieve ULDs with important filling rates.

The observation distribution tends to become asymmetrical when the number of boxes in the samples increases. For instance, the boxplot for the samples with 90 boxes is more left-skewed than the boxplot for instances with 10 boxes. The 25% of observations of the right part of the interquartile ranges (between the median and the third quartile) have more or less the same size, no matter the size of the samples, whereas the 25% of the left part of the interquartile ranges become more spread out when the sample size increases. In addition to the interquartile range itself, the left whisker encompasses a wide spread of values when the sample size is increasing. The observations seem to present a distribution with a negative skewness, i.e., with a tail on the left, when the sample has a large number of boxes. In other words, it seems that no matter the number of boxes in the instances to be solved, there are always ULDs with very low filling rates. This can be due to boxes with special dimensions (a very long box for example) or simply because some small boxes have to be packed at the end of the algorithm but not enough space remains in the ULDs that are already loaded.

CG deviation analysis

Figure 6.7 shows the percentage of unbalanced ULDs per sample size before and after the application of the J&S operators (in black). The percentage of unbalanced ULDs is initially very high especially for the instances with a small number of boxes. This is due to packing starting from the front left bottom vertex. If few boxes are packed, then the weight distribution is not

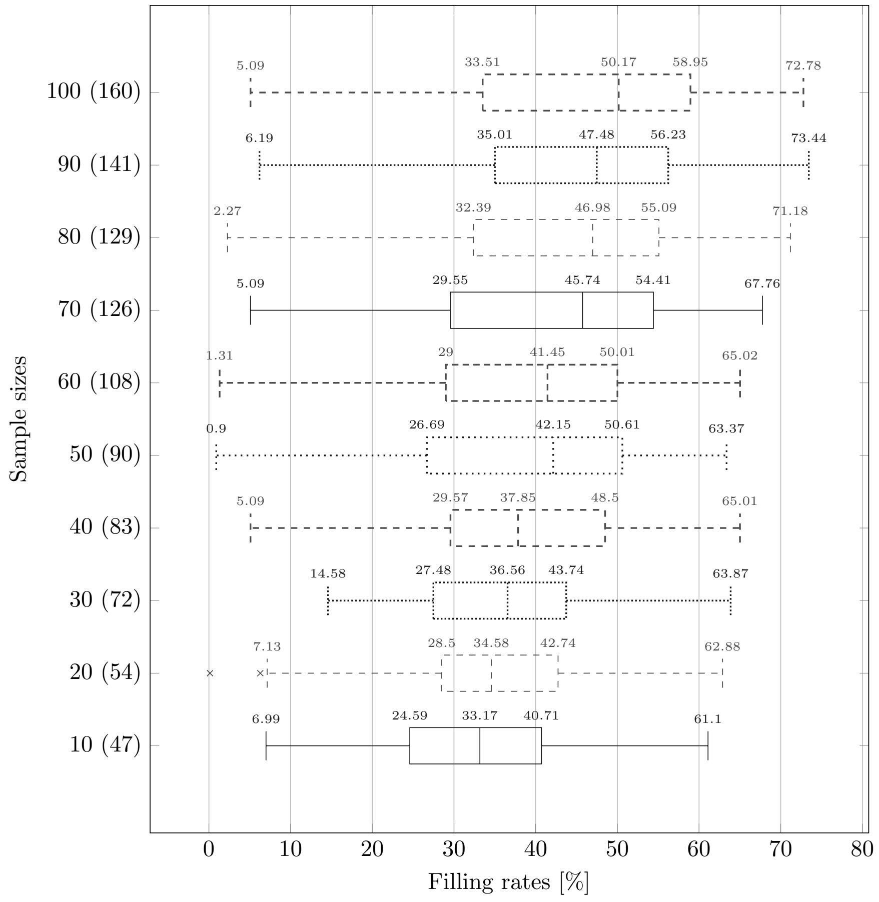


Figure 6.16: Filling rates in percent per sample size (each sample size has 30 instances) for the tailored two-phase constructive heuristic. The number in brackets denotes the number of ULDs used for all 30 instances of each sample size

likely to be improved naturally by packing more boxes in this ULD. The improvements brought by the J&S operators, shown in dashed in Figure 6.7, are considerable but maybe not as expected with preliminary experimentations in Figure 5.13. Results represented in Figure 6.7 suggest that the combination of MF2 and the J&S operators is the key for a low percentage of unbalanced ULDs.

Looking closely at the deviations separately, the packing frame algorithm starts the packing at the front left bottom vertex as explained in Section 5.3. Because of this, the number of CGs initially too far to the left or front is high, as shown in Figure 6.17. In this figure, the number of deviations before and after J&S operator application can be observed, highlighting the weight distribution improvement. The percentages of deviations for each sample size can be found in Figures A.2 and A.3 in Appendix A.3.2.

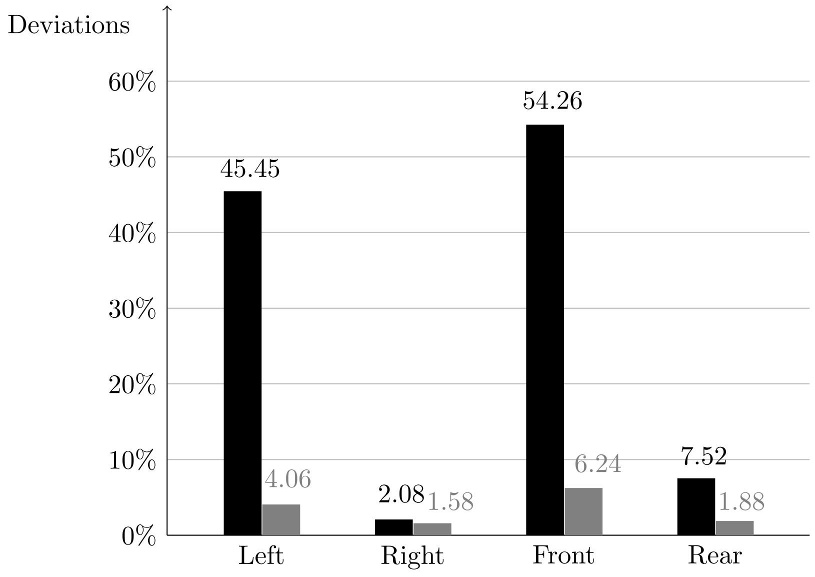


Figure 6.17: Percentages of deviations between the CG and the allowable area among the 1010 used ULDs (in black before and in gray after J&S operator application)

Table 6.12 summarises the reductions for each CG deviation.

The remaining deviations can be further analysed as in Section 5.3.4. Among the 130 ULDs unbalanced after the application of the J&S operators, 43.85% of these ULDs have an unbalanced weight along the x-axis, whereas 63.08% still have a CG out of the allowable region along the y-axis. The

Table 6.12: Reductions in the number of CG deviations among the 1010 used ULDs before and after the application of J&S operators

	Number of unbalanced ULDs	Deviations Left	Right	Front	Rear
Before J&S	761 (75.35%)	459 (45.45%)	21 (2.08%)	548 (54.26%)	76 (7.52%)
After J&S	130 (12.87%)	41 (4.06%)	16 (1.58%)	63 (6.24%)	19 (1.88%)
reduction	82.92%	91.07%	23.81%	88.50%	75.00%

former have an average x-deviation of 8.41% and the latter have an average y-deviation of 8.42%. These numbers have been calculated with Equation (5.1).

Computational time

The main advantage of the tailored two-phase constructive heuristic is its impressive speed. Indeed, even for the instances with 100 boxes to be packed, the maximum computational duration does not exceed 12 seconds for any instance.

6.5 Global comparison

In this section, a global comparison between the results obtained with the four parametrised methods is provided. First, the quality of the solutions for each sample size is compared, using Friedman's test to determine if the differences are significant. Second, the computational times and the number of unsolved instances are analysed and limitations are highlighted. Based on these elements, final conclusions are drawn for practical use.

Note that the complete results are provided in Appendix A.4.

6.5.1 Solution quality

Only the instances solved by all the considered methods are compared in the tests. Because of the limitations in terms of computational times, some methods have not found a solution to all the instances and some have not even been tested on all the instances. The solution quality is thus compared step by step, depending on the available results. The four methods are first compared on instances with a number of boxes ranging from 10 to 50 (Step 1). Then, the I&F, the FRF heuristic with the block Coord & p_{ij} and the tailored two-phase constructive heuristics are compared on instances with a number of boxes ranging from 60 to 80 (Step 2). The results of the FRF heuristic with the block Coord & p_{ij} on instances with 90 boxes are not compared to those obtained with the I&F and the tailored heuristics because a lot of instances have not been solved and it would reduce the number of comparisons which can be made. Finally, the I&F and the tailored heuristic are compared on instances with a number of boxes ranging from 90 to 100 (Step 3). A summary of the comparisons made at each step is presented in Table 6.13.

Table 6.13: Methods compared at each step based on sample sizes (n is the number of boxes in each instance)

	Sample sizes	I&F	FRF Coord	FRF Coord & p_{ij}	tailored
Step 1	$n \in \{10, ..., 50\}$	$\times$	$\times$	$\times$	$\times$
Step 2	$n \in \{60, ..., 80\}$	$\times$		$\times$	$\times$
Step 3	$n \in \{90, 100\}$	$\times$			$\times$

In order to compare the quality of the solutions provided by each method, Friedman's test has been performed in `R` to measure if there is a significant difference between the results. In the case of a positive answer, the post-hoc analysis using Conover test (Conover (1999)) is performed to highlight which pair of methods obtained significantly different solutions. In addition to these tests, the average objective function values computed over the instances solved by all the techniques (depending on the step) are plotted in Figure 6.18.

Step 1

This section shows the comparisons between the four methods on instances with a number of boxes ranging from 10 to 50.

Instances with 10 boxes The 30 instances with 10 boxes have been successfully solved by the four methods. Friedman's test is thus performed on the four series of 30 objective function values. A p-value equal to 6.885×10^{-10} is obtained meaning that there is a highly significant difference (p-value <0.01) among the four series of values. The post-hoc analysis for pairwise comparison gives the following results:

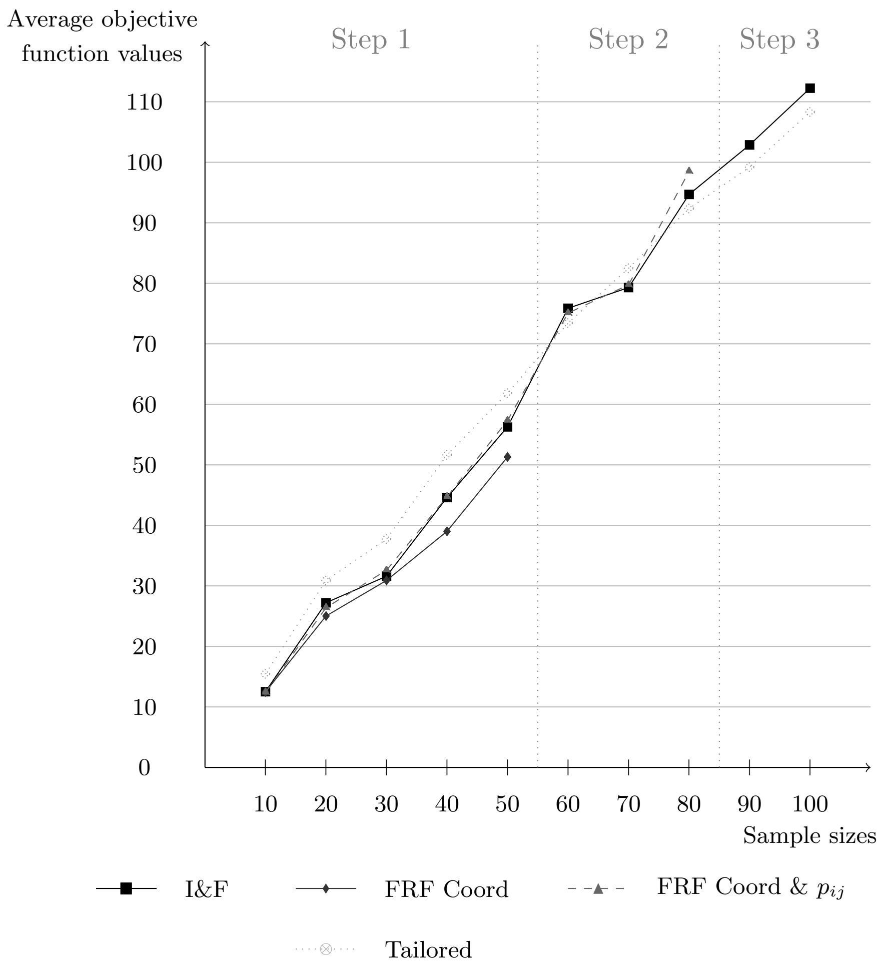


Figure 6.18: Comparison of the average objective function values for different sample sizes

	I&F	FRF Coord	FRF Coord & p_{ij}
FRF Coord	9×10^{-2}	-	-
FRF Coord & p_{ij}	3×10^{-2}	6.3×10^{-1}	-
Tailored	$<2 \times 10^{-16}$	$<2 \times 10^{-16}$	$<2 \times 10^{-16}$

One can observe that there is a highly significant difference between the tailored heuristic and the three matheuristics. Looking closely at the results in Table A.1, this difference is explained by larger values for the tailored heuristic than for each of the three matheuristics. This relation will be denoted as

$$\text{Tailored} > \text{I\&F}, \quad \text{Tailored} > \text{FRF with Coord}, \quad \text{Tailored} > \text{FRF with Coord \& } p_{ij}.$$

Another observation is the significant difference (p-value <0.05) between the I&F and the FRF heuristic with the block Coord & p_{ij} heuristics. Table A.1 shows that the latter performs better than the I&F for some instances:

$$\text{I\&F} > \text{FRF Coord \& } p_{ij}.$$

There is no significant difference neither between the I&F and the FRF heuristic with the block Coord, nor between the two FRF heuristics.

The average objective function values over the 30 instances of size 10 for each method are represented in Figure 6.18.

Instances with 20 boxes For a sample size of 20 boxes, the FRF heuristic with the block Coord failed for two instances and the other FRF heuristic also failed for one of these two. These two instances are thus ignored in the comparison. Friedman's test is performed on the four series of 28 objective function values. A p-value equal 8.275×10^{-5} is obtained meaning that there is a highly significant difference among the four series of values. The post-hoc analysis for pairwise comparison gives the following results:

	I&F	FRF Coord	FRF Coord & p_{ij}
FRF Coord	6.7×10^{-10}	-	-
FRF Coord & p_{ij}	1.6×10^{-2}	2×10^{-5}	-
Tailored	2.5×10^{-8}	$<2 \times 10^{-16}$	3.9×10^{-13}

All the p-values show significant differences between each pair of results. Looking closely at the results in Table A.1, these differences can be summarised as:

$$\text{Tailored} > \text{I\&F} > \text{FRF with Coord \& } p_{ij} > \text{FRF with Coord},$$

which means that the FRF heuristic with the block Coord significantly outperforms each of the other three methods.

The average objective function values over the 28 instances of size 20 for each method are represented in Figure 6.18.

Instances with 30 boxes For a sample size of 30 boxes, the I&F failed for one instance and the FRF heuristic with the block Coord & p_{ij} failed for another. These two instances are thus ignored in the comparison. Friedman's test is performed on the four series of 28 objective function values. A p-value equal to 2.354^{-8} is obtained meaning that there is a highly significant difference among the four series of values. The post-hoc analysis for pairwise comparison gives the following results:

	I&F	FRF Coord	FRF Coord & p_{ij}
FRF Coord	5.69×10^{-1}	-	-
FRF Coord & p_{ij}	1.31×10^{-1}	3.9×10^{-2}	-
Tailored	$<2 \times 10^{-16}$	$<2 \times 10^{-16}$	$<2 \times 10^{-16}$

Once again, the tailored two-phase constructive heuristic has larger objective function values than the three other methods. The p-value shows significant difference between the two FRF heuristics. When looking closely at the results in Table A.1, it appears that the one with the block Coord outperforms the one with the block Coord & p_{ij}.

The average objective function values over the 28 instances of size 30 for each method are represented in Figure 6.18.

Instances with 40 boxes The three matheuristics failed on one or several instances, leading to a set of 25 instances solved by all the four methods. Friedman's test provides a p-value equal to 3.808×10^{-7}. The post-hoc analysis for pairwise comparison gives the following results:

	I&F	FRF Coord	FRF Coord & p_{ij}
FRF Coord	2.5×10^{-8}	-	-
FRF Coord & p_{ij}	3×10^{-1}	3.1×10^{-10}	-
Tailored	$<2 \times 10^{-16}$	$<2 \times 10^{-16}$	9.4×10^{-15}

Once again, the tailored two-phase constructive heuristic has significantly larger objective function values than the other three methods. The p-value shows significant differences between the two FRF heuristics and between the I&F and the FRF heuristic with the block Coord. When looking closely at the results in Table A.1, one has

$$\text{I\&F} > \text{FRF with Coord} \quad \text{and} \quad \text{FRF with Coord \& } p_{ij} > \text{FRF with Coord.}$$

The average objective function values over the 25 instances of size 40 for each method are represented in Figure 6.18.

Instances with 50 boxes The three matheuristics failed on several instances each, leading to a set of 21 instances solved by the four methods. Friedman's test provides a p-value equal to 3.072×10^{-5}. The post-hoc analysis for pairwise comparison gives the following results:

	I&F	FRF Coord	FRF Coord & p_{ij}
FRF Coord	1.7×10^{-11}	-	-
FRF Coord & p_{ij}	2.67×10^{-2}	2.8×10^{-15}	-
Tailored	1.2×10^{-3}	$<2 \times 10^{-16}$	2.606×10^{-1}

Unlike the previous tests, the tailored two phase constructive heuristic has significantly larger objective function values than the I&F and the FRF heuristic with the block Coord, but no conclusion can be drawn between the tailored and the other FRF heuristic. The p-value shows significant differences between the two FRF heuristics and between the I&F and the FRF heuristic with the block Coord. When looking closely at the results in Table A.1, one has

$$\text{FRF with Coord} < \text{I\&F} < \text{FRF with Coord \& } p_{ij}.$$

The average objective function values over the 21 instances of size 50 for each method are represented in Figure 6.18.

The conclusion of this first step is that the FRF heuristic with the block Coord significantly outperforms the other methods for the considered sample sizes. However, as explained in the next section, this heuristic is also very slow in comparison to the other techniques. The FRF heuristic with the block Coord is thus recommended for instances with a reasonable size (up to 40).

Step 2

This section shows the comparisons between the I&F, the FRF with the block Coord & p_{ij} and the tailored heuristics on instances with a number of boxes ranging from 60 to 80.

Instances with 60 boxes The I&F failed on one instance and the FRF heuristic with the block Coord & p_{ij} failed on five other instances. Friedman's test is then performed on three series of 24 observations and returns a p-value equal to 0.6514, which does not indicate a significant difference. The average objective function values over the 24 instances of size 60 for the three methods are represented in Figure 6.18.

Instances with 70 boxes Both the I&F and the FRF with the block Coord & p_{ij} heuristics failed on several instances, leading to three series of 23 observations on which Friedman's test is performed. The test returns a p-value equal to 1 and thus no conclusion can be drawn.

The average objective function values over the 23 instances of size 70 for the three methods are represented in Figure 6.18.

Instances with 80 boxes Both the I&F and the FRF with the block Coord & p_{ij} heuristics failed on several instances, leading to three series of 22 observations on which Friedman's test is performed. The test returns a p-value equal to 0.129 and thus no conclusion can be drawn.

The average objective function values over the 22 instances of size 80 for the three methods are represented in Figure 6.18.

Contrary to the first step, no clear conclusion can be drawn after step 2. More precisely, no significant difference can be observed for each of the three sample sizes. Moreover, looking closely at Figure 6.18, none of the three methods seems to outperform the others for the three sample sizes. However, the solution quality provided by the tailored two-phase constructive heuristic is now in the range of the quality provided by the two matheuristics. Considering this last observation, the tailored two-phase constructive heuristic may be recommended for instances with a number of boxes ranging from 60 to 80 boxes. Its solution quality is reasonable and its computational times are remarkable as shown in Section 6.5.2.

Step 3

This section shows the comparisons between the I&F and the tailored heuristics on instances with a number of boxes ranging from 90 to 100.

Instances with 90 boxes I&F failed for one instance, leading to two series of 29 observations to be compared with Friedman's test. The test returns a p-value equal to 0.04109, indicating a significant difference between the solutions found by the two methods. When looking closely at the results in Table A.1, one can see that the tailored two-phase heuristic outperforms the I&F. The average objective function values over the 29 instances of size 90 for the two methods are represented in Figure 6.18 and show the same behaviour.

Instances with 100 boxes Both heuristics solved the 30 instances. Friedman's test is performed on the two series of 30 observations and returns a p-value equal to 0.4652. No significant difference can thus be concluded. Nevertheless, the average objective function values over the 30 instances of size 100 for the two methods represented in Figure 6.18 show that the tailored two-phase heuristic has better results on average.

As a conclusion to step 3, the tailored two-phase heuristic significantly outperforms the I&F heuristic for instances with 90 boxes and tends to do the same for instances with 100 boxes. The tailored heuristic is therefore recommended for instances with more than 90 boxes.

Note that the larger the sample size, the more the tailored heuristic is able to compete with matheuristics. A possible explanation of this observation is that the matheuristics locally optimise at each iteration and thus they may lose solution quality when the number of iterations becomes too important.

6.5.2 Computational times and unsolved instances

In this section, the average computational times and the number of unsolved instances are presented in Table 6.14 for each sample size and for each method. The average durations for each method are computed over the instances solved by the considered method, ignoring whether this instance has been solved by the other techniques.

Looking at Table 6.14, one can see that the FRF heuristic with the block Coord is the slowest technique. The second slowest technique is the other FRF heuristic. Then comes the I&F heuristic which has more reasonable computational times and a smaller number of unsolved instances, even for large sample size. Finally, the tailored two-phase constructive heuristic has a dramatic speed, even for large sample size.

6.5.3 In practice

For a reasonable number of boxes to be packed, the FRF heuristic with the block Coord achieves good solution quality. However, for instances with more

Table 6.14: Average computational times and unsolved instances for the four techniques and for different sample sizes (each sample size has 30 instances)

	$n = 10$	$n = 20$	$n = 30$	$n = 40$	$n = 50$
	# unsolved instances				
I&F	0	0	1	2	2
FRF with Coord	0	2	0	4	7
FRF with Coord & p_{ij}	0	1	1	3	3
Tailored	0	0	0	0	0
	Average computational times [s.]				
I&F	12.73	106.88	41.48	116.14	114.70
FRF with Coord	23.20	46.63	151.97	452.76	412.33
FRF with Coord & p_{ij}	14.88	49.91	81.09	255.51	259.53
Tailored	0.02	0.06	0.16	0.37	0.72

	$n = 60$	$n = 70$	$n = 80$	$n = 90$	$n = 100$
	# unsolved instances				
I&F	1	2	2	1	0
FRF with Coord					
FRF with Coord & p_{ij}	5	6	6	12	
Tailored	0	0	0	0	0
	Average computational times [s.]				
I&F	155.35	264.91	317.07	528.64	556.78
FRF with Coord					
FRF with Coord & p_{ij}	441.09	780.16	826.80	1882.56	
Tailored	1.06	1.60	2.25	3.36	4.74

Note. The average computational times in seconds are calculated over the instances which have been solved, independently of the other techniques.

than 40 boxes, the computational times become very large and the tailored two-phase heuristic becomes more interesting. More experimentations would be necessary to clarify the precise size limit.

Chapter 7

Conclusion

The problem studied in this thesis is the packing of a set of different boxes into a set of bins of different types, known as the Multiple Bin Size Bin Packing Problem. This work focuses on air transportation in which bins are Unit Load Devices. Due to the meant application, several constraints met in practice have to be taken into account as explained in Chapter 2. These constraints make the problem very specific and not deeply studied in the literature. Chapter 3 aims to develop a Mixed Integer Linear Program formulation for the studied problem with all its constraints. However, the Bin Packing Problem is an NP-hard problem and limited results were obtained due to the problem complexity. Nevertheless, this formulation can be considered as a tool for developing matheuristics, which is achieved in Chapter 4. In Chapter 4, several matheuristics initially designed for lot-sizing applications have been adapted to suit to the problem considered in this thesis. Instances with a larger number of boxes can be solved but, for some instances, no solution can be found within one hour computational time. In Chapter 5, the priority is to develop a fast greedy algorithm. The tailored two-phase constructive heuristic is an improvement of the Extreme Points from Crainic et al. (2008). This constructive heuristic takes into account specific constraints but also manages the possibility of using different unit load device types. Chapter 6 presents the data sets, especially designed for the problem studied in this thesis. Chapter 6 also presents the training process that has been used for the matheuristic and the tailored constructive algorithm, as well as the final results that have been obtained with both types of algorithms. The recommended technique depends on the number of boxes to be packed, the matheuristics provide good quality solutions but are time consuming. Moreover, for a large number of boxes, the solutions provided by the tailored two-phase constructive heuristic compete with those returned by some matheuristics. The current position of this final chapter in the thesis outline is shown in bold in Figure 7.1.

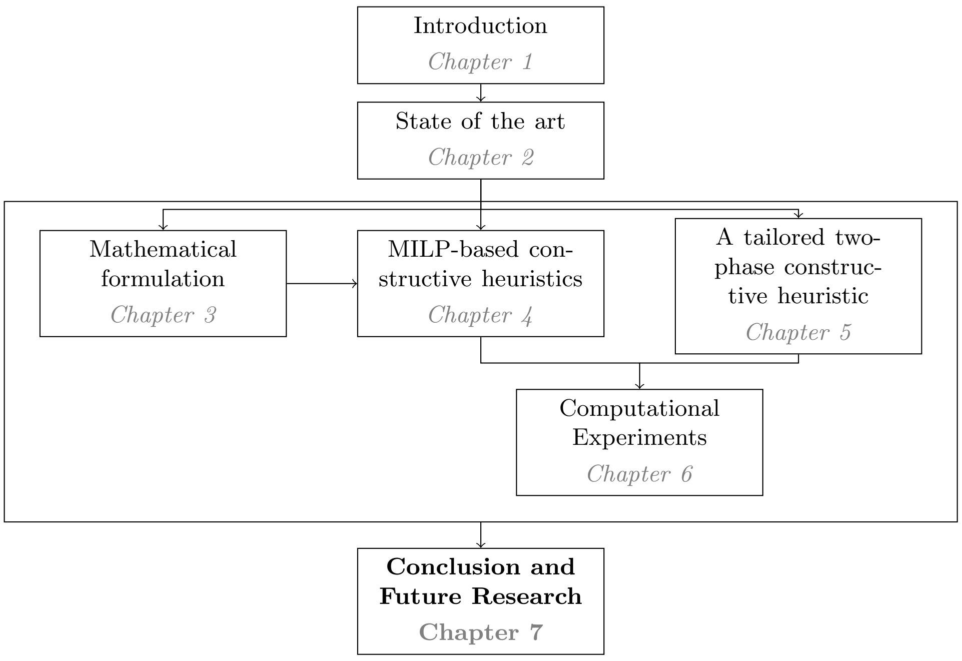

Introduction
Chapter 1
State of the art
Chapter 2
Mathematical formulation
Chapter 3
MILP-based constructive heuristics
Chapter 4
A tailored two-phase constructive heuristic
Chapter 5
Computational Experiments
Chapter 6
Conclusion and Future Research
Chapter 7

Figure 7.1: Current position in the thesis outline

Areas for improvement have been proposed at the end of each chapter of this thesis. However, more global directions for future research may be worthy of interest and are proposed hereinafter. These leads are presented starting from the possible improvements of precise points presented in this thesis and are then extended to broader and more general aspects.

Several hints for enhancement can be achieved by interactions between different chapters of this thesis. First, in Chapter 5, the exact approach for the weight distribution improvement can take a large amount of time to be solved with Branch-and-Bound for some instances. An extension could be to apply the Insert-and-Fix or Fractional Relax-and-Fix from Chapter 4 to this formulation. Second, in Chapter 6, the `irace` package has been presented to tune the parameters of the matheuristics and the tailored two-phase constructive heuristic. Another use of `irace` could be to train the Branch-and-Bound resolution with CPLEX as expressed in López-Ibáñez et al. (2016).

Several directions for future research are related to the constraints taken into account in this thesis.

The stability constraints can be handled differently. In particular, in Chapter 5, there is no linear constraint because of the formulation and another approach could be used. A lead would be to use the method developed by Ramos et al. (2016) for similar problems. This method is based on the first and third Newton's laws leading to forces and moments of forces. When a box is packed on top of other already packed boxes, the stability of the new box has to be verified as well as the stability of the other boxes. These verifications create systems which are statically indeterminate and thus hard to solve. Another point about the stability is that, in this thesis, it is assumed that a box can lay on an inclined wall of the unit load devices. However, in practice, the main purpose of the containers is not to support boxes but to provide protection.

A fourth possible direction for future research is to consider other constraints. As explained in Chapter 2, the allocation constraints are related to the items that should or should not travel together. For instance, some boxes have the same destination and have thus to be gathered in the same unit load device(s) if the flight has different stops. Conversely, some boxes may not be allowed to be transported in the same unit load device or next to each other because of their contents. These kinds of constraints can perhaps be easy to add in the linear formulation but the application to the tailored two-phase constructive heuristic may be less straightforward.

This thesis proposes several methods which are constructive heuristics. A possible extension would be to develop improvement heuristics to ameliorate the solutions found.

Another extension of this work may concern the sequence of boxes to be packed. In this thesis, the list of boxes is known and the sequence is based on sorting operators. An extension may be to consider random sequence of boxes. Similarly, another enhancement may be related to the a priori knowledge of the list of boxes to be packed. There exists an online version of the Bin Packing Problem in which the boxes have to be packed without knowing the boxes that have to be packed afterwards (Seiden (2001)).

The problem studied in this thesis is rich by its specificity. However, it can be easily extended to other applications. One special feature of the problem studied here is the assortment of the boxes, which is considered heterogeneous in our case. The case studied in this thesis is typical to air cargo. However, in practice, delivery companies such as TNT, FedEx or UPS may consider only few types of boxes making the assortment homogeneous. This leads to a Cutting Stock Problem as shown in Figure 2.2. The symmetries have then to be considered to speed up the process. A second example is related to the unit load devices. Instead of unit load devices, bins could be a fleet of trucks of possibly different sizes. If all the vehicles are identical, the objective would then become to minimise the number of needed trucks and thus to minimise the costs of the company but also on a larger scale, eventually reduce the congestion and the pollution. The constraint about the weight distribution has then to be adapted to satisfy the axle weight constraints met in truck loading (Pollaris et al. (2015)).

Appendix A

Appendix

A.1 EP generation

Here are the details of the generation process. In the following, index k denotes a previously packed box.

- The point (x'_i, y_i, z_i) is projected along
  - Y-axis direction:
    - Very long simulated box: $\max\{y'_k : y'_k \leq y_i, x_k \leq x'_i < x'_k, z_k \leq z_i < z'_k\}$
    - Very wide simulated box: $\max\{y'_k : y'_k \leq y_i, x'_i < x'_k, z_k \leq z_i < z'_k\}$
    - Very high simulated box: $\max\{y'_k : y'_k \leq y_i, x_k \leq x'_i < x'_k, z_i < z'_k\}$

    For each simulated box, if a k exists, then the generated EP is written as $(x'_i, \max y'_k, z_i)$. If none of the three simulated boxes have such k (no previously packed boxes have been met or $y_i = 0$), the generated EP is $(x'_i, 0, z_i)$ on the front side of the ULD.
  - Z-axis direction: this projection is considered only if $z_i \neq 0$ (if $z_i = 0$, only the projection along the Y-axis is considered)
    - Very long simulated box: $\max\{z'_k : z'_k \leq z_i, x_k \leq x'_i < x'_k, y_k \leq y_i < y'_k\}$
    - Very wide simulated box: $\max\{z'_k : z'_k \leq z_i, x'_i < x'_k, y_k \leq y_i < y'_k\}$
    - Very high simulated box: $\max\{z'_k : z'_k \leq z_i, x_k \leq x'_i < x'_k, y_i < y'_k\}$

    For each simulated box, if a k exists, then the generated EP will be written as $(x'_i, y_i, \max z'_k)$. If there is no such k for the three simulated boxes, the generated EP is $(x'_i, y_i, 0)$ if there is no type 1 cut [1]. If there is a type 1 cut, then
    - if $x'_i < \frac{b_1}{a_1}$, the generated EP is $(x'_i, y_i, b_1 - a_1 x'_i)$,

[1] As a reminder the type 1 cut is described by the equation $z + a_1 x = b_1$

    - if $x'_i \geq \frac{b_1}{a_1}$, the generated EP is also $(x'_i, y_i, 0)$.

- The point (x_i, y'_i, z_i) is projected along
  - X-axis direction
    - Very long simulated box: $\max\{x'_k : x'_k \leq x_i, y_k \leq y'_i < y'_k, z_k \leq z_i < z'_k\}$
    - Very wide simulated box: $\max\{x'_k : x'_k \leq x_i, y'_i < y'_k, z_k \leq z_i < z'_k\}$
    - Very high simulated box: $\max\{x'_k : x'_k \leq x_i, y_k \leq y'_i < y'_k, z_i < z'_k\}$

    For each simulated box, if a k exists, then the generated EP will be written as $(\max x'_k, y'_i, z_i)$. If there is no such k for the three simulated boxes, the generated EP is $(0, y'_i, z_i)$ if there is neither a type 1 cut nor a type 4 cut [2]. If there is a type 4 cut and/or a type 1 cut:
    - if $z_i \geq b_4$
      - if $x_i < \frac{H-b_4}{a_4}$, the generated EP is (x_i, y'_i, z_i)
      - if $x_i \geq \frac{H-b_4}{a_4}$, the generated EP is $([\frac{H-b_4}{a_4}], y'_i, z_i)$
    - if $b_1 \leq z_i < b_4$,
      - if $b_4 - z_i \geq \min_i h_i$ or if there is not type 4 cut, the generated EP is $(0, y'_i, z_i)$,
      - if $b_4 - z_i < \min_i h_i$ (i.e. even the box with the smallest height would not fit), the generated EP is (x_i, y'_i, z_i),
    - if $z_i < b_1$, the generated EP is $([\frac{c_1 - a_1 z_i}{b_1}], y'_i, z_i)$
  - Z-axis direction: this projection is considered only if $z_i \neq 0$ (if $z_i = 0$, only the projection along the X-axis is considered)
    - Very long simulated box: $\max\{z'_k : z'_k \leq z_i, y_k \leq y'_i < y'_k, x_k \leq x_i < x'_k\}$
    - Very wide simulated box: $\max\{z'_k : z'_k \leq z_i, y'_i < y'_k, x_k \leq x_i < x'_k\}$
    - Very high simulated box: $\max\{z'_k : z'_k \leq z_i, y_k \leq y'_i < y'_k, x_i < x'_k\}$

    For each simulated box, if a k exists, then the generated EP will be written as $(x_i, y'_i, \max z'_k)$. If there is no such k for the three simulated boxes, the generated EP is $(x_i, y'_i, 0)$ if there is no type 1 cut. If there is a type 1 cut, then
    - if $x_i < \frac{b_1}{a_1}$, the generated EP is $(x_i, y'_i, b_1 - a_1 x_i)$,
    - if $x_i \geq \frac{b_1}{a_1}$, the generated EP is also $(x_i, y'_i, 0)$.

- The point (x_i, y_i, z'_i) is projected along the X-axis direction. If there is no k such that $x'_k \leq x_i, y_k \leq y_i < y'_k, z_k \leq z'_i < z'_k$, if there is a type 1 cut and if $z'_i < b_1$ as shown on figure 5.6, the generated EP is $([\frac{b_1 - z'_i}{a_1}], y_i, z'_i)$.

[2]As a reminder the type 4 cut is described by the equation $z - a_4 x = b_4$

A.2 Mathematical formulation for balance improvement

Parameters

$i \in \{1, ..., n\}$

n	Total number of boxes to be packed,	
$l_i \times w_i \times h_i$	Length $\times$ width $\times$ height of box i,	$\forall i,$
m_i	Weight of box i,	$\forall i,$
$L \times W \times H$	Length $\times$ width $\times$ height of the ULD,	
C	Maximum gross weight of the ULD,	
V	Volume of the ULD,	
l_i^+	$= \begin{cases} 1 & \text{if the length of box } i \text{ could be vertical,} \\ 0 & \text{otherwise,} \end{cases}$	$\forall i,$
w_i^+	$= \begin{cases} 1 & \text{if the width of box } i \text{ could be vertical,} \\ 0 & \text{otherwise,} \end{cases}$	$\forall i,$
h_i^+	$= \begin{cases} 1 & \text{if the height of box } i \text{ could be vertical,} \\ 0 & \text{otherwise.} \end{cases}$	$\forall i,$
f_i	$= \begin{cases} 1 & \text{if box } i \text{ is fragile} \\ 0 & \text{otherwise} \end{cases}$	$\forall i,$
α^L(resp. α^W)	Accepted gap between the CG and the base geometrical centre along the x-axis (resp. y-axis),	
α^W	Maximum allowed height for the CG.	

Variables

$i, k \in \{1, ..., n\},\ a, b \in \{1, 2, 3\}$

(x_i, y_i, z_i)	Location of the front left bottom vertex of box i,	$\forall i,$
(x'_i, y'_i, z'_i)	Location of the rear right top vertex of box i,	$\forall i,$
r_{iab}	$= \begin{cases} 1 & \text{if the side } b \text{ of box } i \text{ is along the } a\text{-axis,} \\ 0 & \text{otherwise,} \end{cases}$	$\forall i,$
x_{ik}^p	$= \begin{cases} 1 & \text{if box } i \text{ is on the right of box } k (x'_k \leq x_i), \\ 0 & \text{otherwise} (x_i < x'_k), \end{cases}$	$\forall i,$

$$y_{ik}^p = \begin{cases} 1 & \text{if box } i \text{ is behind box } k(y_k' \leq y_i), \\ 0 & \text{otherwise}(y_i < y_k'), \end{cases} \qquad \forall i,$$

$$z_{ik}^p = \begin{cases} 1 & \text{if box } i \text{ is above } k(z_k' \leq z_i), \\ 0 & \text{otherwise}(z_i < z_k'), \end{cases} \qquad \forall i,$$

$$g_i = \begin{cases} 1 & \text{if box } i \text{ is on the ground}(z_i = 0), \\ 0 & \text{otherwise}, \end{cases} \qquad \forall i,$$

$$h_{ik} = \begin{cases} 0 & \text{if box } k \text{ has the suitable height to support} \\ & \text{ box } i \ (z_i = z_k'), \\ 1 & \text{otherwise}, \end{cases} \qquad \forall i, k,$$

$$o_{ik} = \begin{cases} 0 & \text{if the projections on the } XY \text{ plane of the} \\ & \quad \text{boxes } i \text{ and } k \text{ have a non-empty intersection}, \\ 1 & \text{otherwise}, \end{cases} \qquad \forall i, k,$$

$$s_{ik} = \begin{cases} 1 & \text{if box } k \text{ supports box } i, \\ 0 & \text{otherwise}, \end{cases} \qquad \forall i, k,$$

$$\eta_{ik}^1 = \begin{cases} 0 & \text{if } x_k \leq x_i, \\ 1 & \text{otherwise}, \end{cases} \qquad \forall i, k,$$

$$\eta_{ik}^2 = \begin{cases} 0 & \text{if } y_k \leq y_i, \\ 1 & \text{otherwise}, \end{cases} \qquad \forall i, k,$$

$$\eta_{ik}^3 = \begin{cases} 0 & \text{if } x_i' \leq x_k', \\ 1 & \text{otherwise}, \end{cases} \qquad \forall i, k,$$

$$\eta_{ik}^4 = \begin{cases} 0 & \text{if } y_i' \leq y_k', \\ 1 & \text{otherwise}, \end{cases} \qquad \forall i, k,$$

$$\beta_{ik}^l = \begin{cases} 1 & \text{if the vertex } l \text{ is supported by box } k, \\ 0 & \text{otherwise}, \end{cases} \qquad \forall i, k, l,$$

$$\gamma_i^1 = \begin{cases} 1 & \text{if the box } i \text{ lays on cut 1}, \\ 0 & \text{otherwise}, \end{cases} \qquad \forall i,$$

$$\gamma_i^2 = \begin{cases} 1 & \text{if the box } i \text{ lays on cut 2}, \\ 0 & \text{otherwise}, \end{cases} \qquad \forall i.$$

Objective function

$$\min \quad z_{CG} := \frac{\sum_{i=1}^{n} m_i \frac{z_i + z_i'}{2}}{\sum_{i=1}^{n} m_i}$$

Constraints

$i, k \in \{1, ..., n\},\ a, b \in \{1, 2, 3\}, l \in \{1, ..., 4\}$

$$\begin{aligned}
x_i' &\leq L, && \forall i,\\
y_i' &\leq W, && \forall i,\\
z_i' &\leq H, && \forall i,\\
x_i' - x_i &= r_{i11}\ l_i + r_{i12}\ w_i + r_{i13}\ h_i, && \forall i,\\
y_i' - y_i &= r_{i21}\ l_i + r_{i22}\ w_i + r_{i23}\ h_i, && \forall i,\\
z_i' - z_i &= r_{i31}\ l_i + r_{i32}\ w_i + r_{i33}\ h_i, && \forall i,\\
\sum_a r_{iab} &= 1, && \forall i, b,\\
\sum_b r_{iab} &= 1, && \forall i, a,\\
x_{ik}^p + x_{ki}^p + y_{ik}^p + y_{ki}^p + z_{ik}^p + z_{ki}^p &\geq 1, && \forall i, k,\\
x_k' &\leq x_i + (1 - x_{ik}^p)\ L, && \forall i, k,\\
x_i + 1 &\leq x_k' + x_{ik}^p\ L, && \forall i, k,\\
y_k' &\leq y_i + (1 - y_{ik}^p)\ W, && \forall i, k,\\
y_i + 1 &\leq y_k' + y_{ik}^p\ W, && \forall i, k,\\
z_k' &\leq z_i + (1 - z_{ik}^p)\ H, && \forall i, k,\\
r_{i31} &\leq l_i^+, && \forall i,\\
r_{i32} &\leq w_i^+, && \forall i,\\
r_{i33} &\leq h_i^+, && \forall i,\\
z_i + a_1 x_i &\geq b_1, && \forall i,\\
z_i - a_2 x_i' &\geq -b_2, && \forall i,\\
z_i' + a_3 x_i' &\leq b_3, && \forall i,\\
z_i' - a_4 x_i &\leq b_4, && \forall i,\\
\sum_{l=1}^{4} \sum_{k=1}^{n} \beta_{ik}^l + 2\gamma_i^1 + 2\gamma_i^2 &\geq 3(1 - g_i) && \forall i,
\end{aligned}$$

$$
\begin{aligned}
z_i &\le (1-g_i)\ H, && \forall i,\\
z'_k - z_i &\le v_{ik}, && \forall i,k,\\
z_i - z'_k &\le v_{ik}, && \forall i,k,\\
v_{ik} &\le z'_k - z_i + 2H(1-m_{ik}), && \forall i,k,\\
v_{ik} &\le z_i - z'_k + 2Hm_{ik}, && \forall i,k,\\
h_{ik} &\le v_{ik}, && \forall i,k,\\
v_{ik} &\le h_{ik}\ H, && \forall i,k,\\
o_{ik} \le x^p_{ik} + x^p_{ki} + y^p_{ik} + y^p_{ki} &\le 2o_{ik}, && \forall i,k,\\
(1-s_{ik}) \le h_{ik} + o_{ik} &\le 2(1-s_{ik}), && \forall i,k,\\
\beta^l_{ik} &\le s_{ik}, && \forall i,k,l,\\
\eta^1_{ik} + \eta^2_{ik} &\le 2(1-\beta^1_{ik}), && \forall i,k,\\
\eta^2_{ik} + \eta^3_{ik} &\le 2(1-\beta^2_{ik}), && \forall i,k,\\
\eta^3_{ik} + \eta^4_{ik} &\le 2(1-\beta^3_{ik}), && \forall i,k,\\
\eta^1_{ik} + \eta^4_{ik} &\le 2(1-\beta^4_{ik}), && \forall i,k,\\
x_k &\le x_i + \eta^1_{ik}\ L, && \forall i,k,\\
y_k &\le y_i + \eta^2_{ik}\ W, && \forall i,k,\\
x'_i &\le x'_k + \eta^3_{ik}\ L, && \forall i,k,\\
y'_i &\le y'_k + \eta^4_{ik}\ W, && \forall i,k,\\
(1-\gamma^1_i)M &\ge z_i + a_1x_i - b_1, && \forall i,\\
(1-\gamma^2_i)M &\ge z_i - a_2x'_i + b_2, && \forall i,\\
\gamma^1_i &\le a_1 + b_1 && \forall i,\\
\gamma^2_i &\le a_2 + b_2 && \forall i,\\
\sum_i s_{ik} &\le n(1-f_k) && \forall k,\\
\frac{L+a_1-a_2}{2} - \alpha^L \le \frac{\sum_{i=1}^n m_i(\frac{x_i+x'_i}{2})}{\sum_{i=1}^n m_i} &\le \frac{L+a_1-a_2}{2} + \alpha^L,\\
\frac{W}{2} - \alpha^W \le \frac{\sum_{i=1}^n m_i(\frac{y_i+y'_i}{2})}{\sum_{i=1}^n m_i} &\le \frac{W}{2} + \alpha^W.
\end{aligned}
$$

A.3 Results for the tailored two-phase constructive heuristic

A.3.1 Comparison of the average objective function values with standard deviations

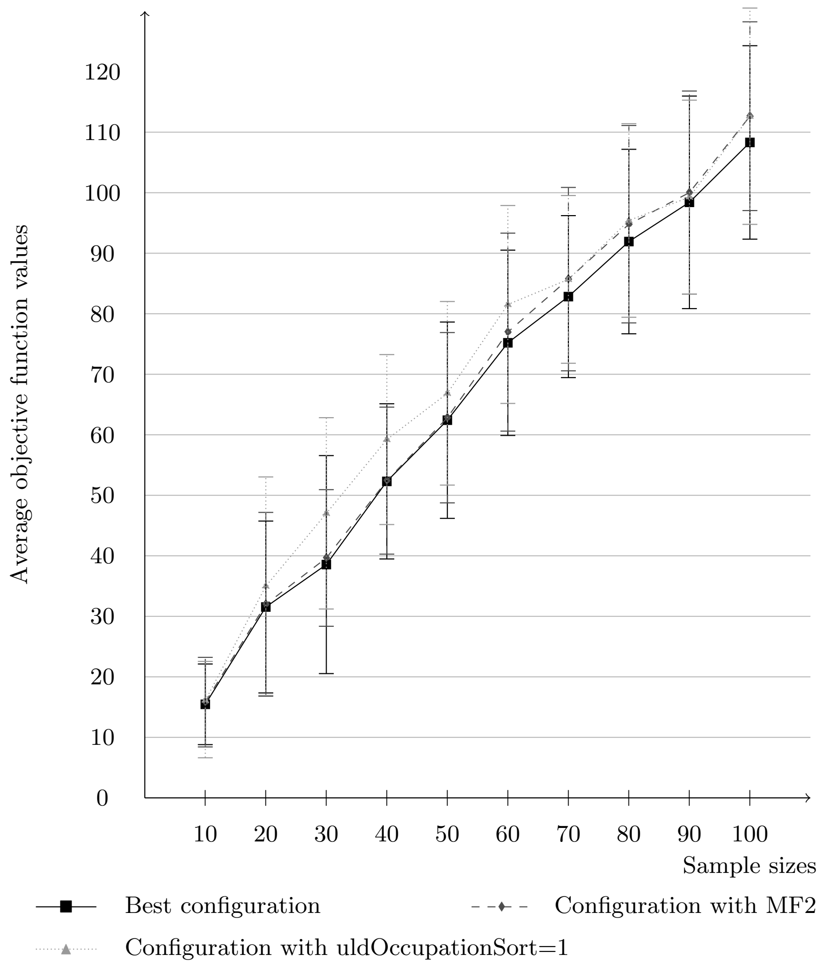


Figure A.1: Comparison of the average objective function values per sample size for different configurations for the tailored two-phase constructive heuristic with the standard deviations (each sample size has 30 instances) (detailed version of Figure 6.6)

A.3.2 Percentages of deviations for each sample size

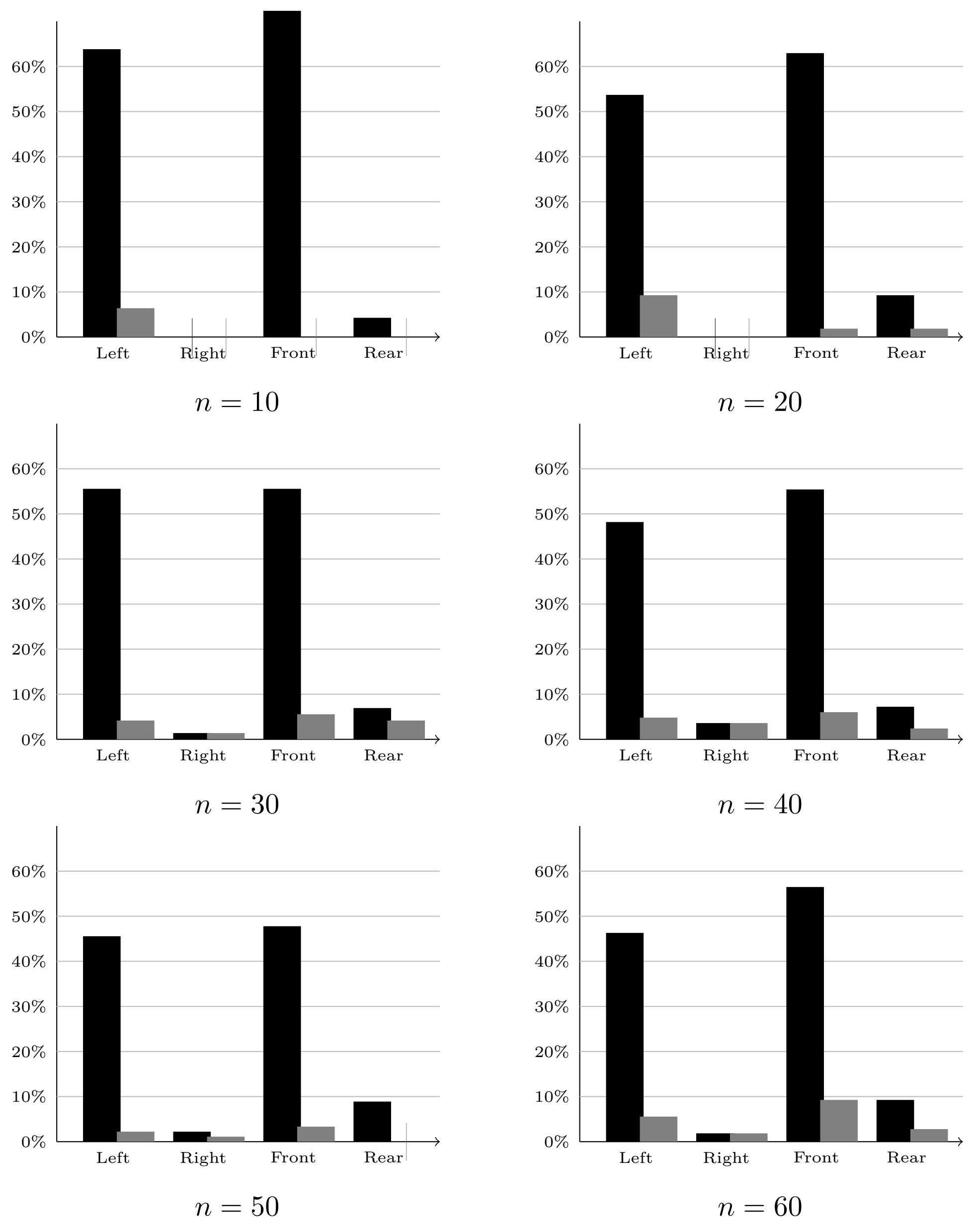


Figure A.2: Percentages of deviations for each sample size in black before the J&S operator application and in gray after J&S application (Part 1)

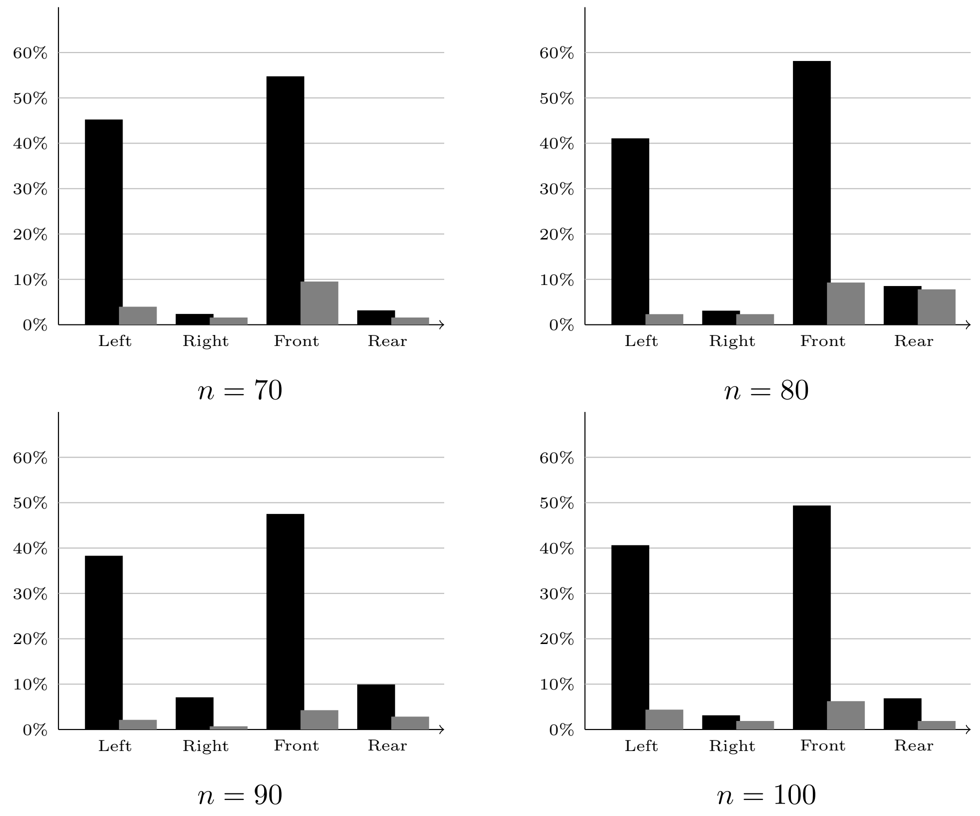


Figure A.3: Percentages of deviations for each sample size in black before the J&S operator application and in gray after J&S application (Part 2)

A.4 Global comparison of the four parametrised techniques

Table A.1: Objective function values and computational times of the four parametrised techniques for every instance

	I&F	FRF Coord	FRF Coord & p_{ij}	tailored	I&F	FRF Coord	FRF Coord & p_{ij}	tailored
	Objective function values				Computational times in milliseconds			
	$n = 10$							
1	21.5	21.5	21.5	31.1	5486	18358	10570	76
2	12.4	12.4	12.4	14.8	4443	17308	10415	45
3	14.8	16.5	14.8	18.9	4303	9751	9917	91
Continued on next page								

Table A.1: Objective function values and computational times of the four parametrised techniques

	I&F	FRF Coord	FRF Coord & p_{ij}	tailored	I&F	FRF Coord	FRF Coord & p_{ij}	tailored
4	12.4	12.4	12.4	14.8	16948	21370	15041	11
5	7.4	7.4	7.4	7.4	4280	11546	3768	10
6	31.1	31.1	31.1	31.1	1414	2477	1480	15
7	5	5	5	7.4	13563	13222	8983	25
8	7.4	7.4	7.4	12.4	14934	9719	5692	7
9	16.5	14.8	14.8	14.8	2788	6695	17904	56
10	18.9	18.9	19.8	18.9	5487	7241	27055	49
11	23.9	23.9	23.9	26.3	23842	22617	18110	149
12	12.4	12.4	12.4	12.4	69225	223166	34843	7
13	7.4	7.4	7.4	12.4	14015	3857	10262	8
14	7.4	7.4	7.4	12.4	10895	8639	4364	7
15	7.4	7.4	7.4	12.4	3073	7528	6833	13
16	7.4	7.4	7.4	12.4	17417	12946	6546	24
17	16.5	14.1	14.1	14.8	20786	25848	31530	8
18	12.4	12.4	12.4	14.8	16651	44501	12536	23
19	12.4	12.4	12.4	18.9	20402	8795	19097	12
20	7.4	7.4	7.4	7.4	3705	14841	9006	6
21	5	5	5	5	5940	3955	10286	6
22	14.8	16.5	14.8	18.9	4197	7195	19718	10
23	16.5	16.5	16.5	31.1	12204	18210	14550	9
24	14.8	14.1	14.1	14.8	25750	39456	46776	7
25	12.4	12.4	12.4	14.8	15584	89748	45691	7
26	18.9	18.9	18.9	18.9	1133	2133	1566	6
27	9.1	9.1	9.1	12.4	11710	14086	17262	11
28	7.4	7.4	7.4	12.4	11284	12948	13882	9
29	7.4	7.4	7.4	12.4	14487	15503	7961	14
30	7.4	7.4	7.4	7.4	5814	2411	4640	10
	$n = 20$							
1	18.9	18.9	18.9	23.9	10371	9911	11774	22
2	31.1	31.1	31.1	50	3453	6043	5028	27
3	18.9	18.9	18.9	18.9	4011	8604	5681	83
4	18.9	18.9	18.9	18.9	10857	12793	19893	72
5	35.4	26.3	33.7	37.8	10478	63833	27823	25
6	47.6	38.5	38.5	36.1	8518	15213	9894	21
7	16.5	16.5	16.5	36.1	4924	9528	22215	39
8	23.9	18.9	23.9	23.9	449769	8337	801290	144
9	7.4	7.4	7.4	7.4	12763	19560	20157	96
10	18.9		28.9	31.3	15881	3600000	39313	44
11	48.5	43.5	43.5	64.8	32432	262685	48620	28
12	28	28	31.1	38.5	5747	10313	8890	70

Continued on next page

Table A.1: Objective function values and computational times of the four parametrised techniques

	I&F	FRF Coord	FRF Coord & p_{ij}	tailored	I&F	FRF Coord	FRF Coord & p_{ij}	tailored
13	50	31.1	50	36.1	5499	13282	10280	91
14	18.9	18.9	18.9	26.3	10123	13139	9197	21
15	18.9	18.9	18.9	18.9	4647	8655	6899	89
16	26.5	21.5	21.5	31.1	37576	39316	80413	107
17	16.5	16.5	16.5	18.9	4908	12920	59650	175
18	18.9	18.9	18.9	18.9	6636	10259	11773	24
19	21.5	28.9	28.9	50	7486	591551	66471	51
20	31.1	31.1	31.1	41.1	11849	26818	19734	41
21	37.1			50	2474278	3600000	3600000	47
22	18.9	18.9	18.9	18.9	3574	7312	6673	26
23	62.4	57.4	59.1	68.9	9544	29701	17270	34
24	16.5	14.8	16.5	14.8	7959	22797	63239	30
25	18.9	18.9	18.9	18.9	9672	9579	9791	95
26	18.9	18.9	18.9	31.1	4086	7965	7229	31
27	31.1	31.1	31.1	31.1	5066	6088	6147	24
28	21.5	18.9	19.8	26.3	7503	23548	22971	20
29	18.9	18.9	18.9	18.9	8447	12132	14038	117
30	57.4	50	50	38.5	18369	43839	14955	20
	$n = 30$							
1	36.1	38.5	36.1	50	13657	43930	95969	226
2	18.9	18.9	18.9	18.9	13943	52313	16349	112
3	18.9	18.9	28.9	33.7	17343	46693	40029	127
4	57.4	50	57.4	60	18790	74782	62874	81
5	36.3	37.8	36.3	41.1	79132	133182	136986	68
6	36.1	40.2	36.1	50	28907	101652	29567	81
7		43.5	38.5	50	3600000	226542	220306	64
8	23.9	23.9	23.9	31.1	81563	326420	59524	594
9	26.3	18.9	18.9	31.3	58882	140101	28661	102
10	26.3	31.3	31.3	31.3	64506	197150	225012	121
11	31.1	31.1	31.1	31.1	11199	14680	13464	130
12	18.9	18.9	18.9	22.2	12996	58459	32795	116
13	26.3	33.7	26.3	37.8	44276	430941	86741	234
14	18.9	18.9	23.9	26.3	25784	156415	48518	679
15	28.9	26.3	26.3	33.7	31072	113111	176735	144
16	18.9	18.9	28.9	31.3	19153	21526	157148	135
17	36.1	38.5	36.1	50	14421	118162	29224	65
18	55	50	50	38.5	24458	37329	23816	68
19	50.2	37.8	42.8	33.7	26434	680629	93138	211
20	57.4	57.4	64.8	67.2	29719	71208	141404	65
21	31.3	31.3	31.3	37.8	32513	121181	65990	215

Continued on next page

Table A.1: Objective function values and computational times of the four parametrised techniques

	I&F	FRF Coord	FRF Coord	tailored	I&F	FRF Coord	FRF Coord	tailored
			& p_{ij}				& p_{ij}	
22	38.5	38.5	36.1	50	20507	75788	68407	248
23	18.9	18.9	18.9	26.3	15387	49643	26719	237
24	31.1	31.1	31.1	38.5	8911	15793	14543	85
25	26.3	23.9	28.9	31.3	75696	366115	78832	68
26	40.4	35.4	38	45.9	85497	170271	166351	280
27	40.4	42.8		50	80395	515941	3600000	58
28	26.3	26.3	31.3	37.8	244309	130246	103877	59
29	18.9	18.9	28.9	33.7	13768	53300	93480	173
30	31.1	31.1	31.1	36.1	9782	15689	15157	72
	$n = 40$							
1	64.8	57.4	66.5	50	41830	874137	290111	123
2	31.3	33	31.3	37.8	204967	1044964	109807	164
3	43.5	43.5	53.5	62.2	40903	159417	1354943	201
4	32.4	31.3	31.3	37.8	39107	120344	101972	203
5	62.2	36.1	40.2	38.5	92549	174236	168341	187
6	31.3	26.3	26.3	38.7	155887	325230	134889	297
7	46.1	37.8		56.7	384926	256039	3600000	156
8	31.3	33.7	38.7	33.7	74009	494130	282090	219
9	51.1	41.1	48.5	56.7	83420	242540	128798	94
10	58.5	45.2	58.6	62.4	132157	392160	632179	125
11				37.8	3600000	3600000	3600000	844
12	74.6	38.5	77	67.2	140905	102882	160351	814
13	42.8	35.4	38.7	50	40602	81289	158953	210
14	23.9	23.9	23.9	34.8	647785	321646	180415	203
15	50.9	50	50.9	68.9	30408	170981	56230	166
16	38.5	40.2	48.5	67.2	34355	152464	90813	598
17	62.2	38.5	62.2	50	102230	79092	83719	191
18	36.1	38.5	36.1	55	68136	591594	75959	301
19	38.7	37.8	38.7	45.2	90032	443280	230466	185
20				62.2	3600000	3600000	3600000	261
21	46.1		45.4	45.2	148017	3600000	305640	165
22	33.7	33.7	33.7	50	51062	178338	130549	161
23	58.3	53.3	55.9	69.6	90680	498624	159471	212
24	31.3	31.3	26.3	33.7	101949	2093354	325727	538
25	45.9	43.5	45.9	55	48685	1400959	448682	867
26	55.9	50.9	64.8	73.9	115803	211585	110353	160
27	45.9	50	45.9	56.7	44667	133457	187879	252
28	26.3	26.3	26.3	33.7	70199	1090084	446895	334
29	58.3		52.6	76.3	132644	3600000	258341	264
30	43.5	36.1	48.5	62.2	44051	138960	285121	2747

Continued on next page

Table A.1: Objective function values and computational times of the four parametrised techniques

	I&F	FRF Coord	FRF Coord	tailored	I&F	FRF Coord	FRF Coord	tailored
			& p_{ij}				& p_{ij}	
	$n = 50$							
1	60.9	45.2	63.5	56.7	90117	360568	426596	356
2	57.4	50	50	62.2	97110	192157	251608	1068
3	69.8	64.8	66.5	81.1	53893	282368	154952	483
4		45.2	38.7	46.3	3600000	526972	215186	334
5	68.1	64.8	73.1	81.1	87424	272088	338822	636
6	65.7	64.8	62.4	100	48367	950022	286598	701
7	38.7	37.8	55.2	42.8	69883	181386	299323	583
8	82.2		76.5	64.8	98021	3600000	234351	257
9	46.1	43.5	48.5	62.4	95338	322331	267772	3813
10	60	50	62.4	50	37292	84008	85567	2449
11	58.3	57.4	60	73.9	49683	544389	185901	316
12	58.3	53.3	60.9	68.9	53199	1545241	263636	496
13	38.7	35.4	43	50	43846	261234	465915	461
14	60		60	69.6	56760	3600000	265938	389
15	45.2	33.7	46.9	37.8	84860	261335	244483	651
16	62.2	62.2	67.2	68.9	45558	89686	73135	363
17	55.2	54.3	55.2	68.9	146142	874312	336198	269
18	88.7	78	78	77	103404	201583	379129	888
19	45.2			45.2	340889	3600000	3600000	349
20	77	71.3	77	68.9	60050	442105	104690	436
21				36.1	3600000	3600000	3600000	653
22	50.9	45.2	47.6	62.2	76722	310991	179775	603
23	65.7		65.7	81.1	78999	3600000	264666	814
24	53.3	43.5	45.9	57.8	77247	342259	136586	528
25	43.5	45.2		67.2	90852	139750	3600000	471
26	53.3		60.9	93.5	936665	3600000	301989	700
27	41.1	38.7	49.5	37.8	60335	639106	350755	1028
28	43.5	45.9	45.9	52.6	68170	406161	232430	555
29	63.3		60.7	69.6	104173	3600000	464913	491
30	42.8	37.8	45.2	37.8	56470	253558	196490	598
	$n = 60$							
1	95.9		88.5	100	263330		118296	810
2	89.4		96.8	81.1	256980		397927	529
3	98.7		81.3	81.1	137590		310585	1095
4	72.2		83.9	75.6	117550		352052	494
5	81.3		69.8	69.6	144515		243827	1963
6	74.1		73.4	68.9	103727		420114	547
7	106.8			100	114987		3600000	715
8	91.8		98.5	100	127701		294048	706

Continued on next page

Table A.1: Objective function values and computational times of the four parametrised techniques

	I&F	FRF Coord	FRF Coord & p_{ij}	tailored	I&F	FRF Coord	FRF Coord & p_{ij}	tailored
9	64.8		50	53.3	126360		98363	2831
10	91.8		89.4	81.1	304282		400761	512
11	115.7		93.5	83.7	206738		263030	392
12	78.9		75.6	69.6	116228		386513	630
13	67.6		65	74.6	178683		481466	880
14	75.5		99.2	80.6	110850		220810	620
15	38.7		38.7	37.8	196243		340804	2510
16	77.2			81.1	208790		3600000	860
17	78.1		79.8	93.5	95783		536591	987
18	63.3		63.3	67.2	123450		509431	2614
19			77.4	93.5	3600000		2177525	1798
20	55.9			67.2	107059		3600000	1468
21	72.2		74.8	86.1	116351		247141	1050
22	50		55.9	57.4	58050		471619	977
23	43.5		43.5	67.2	73278		691124	1369
24	60			64.1	85488		3600000	491
25	69.6		50.9	50	110928		215534	2113
26	70.7		70.7	68.9	104723		168346	500
27	48.5		50.2	50.2	158692		744235	839
28	91.8		108.3	87.8	144565		591070	427
29	89.4			87.8	327318		3600000	530
30	96.8		100.9	77	284771		345959	630
	$n = 70$							
1			83.7	96.3	3600000		1743330	931
2	58.5		55.2	56.7	1160879		1308138	2018
3	60.7		74.1	85	122419		502804	1081
4	75.7			75.6	289424		3600000	2217
5	100.9		105.9	100	411380		617417	916
6	65		65	71.5	1169553		433326	1681
7	111.6		108.5	106.7	180583		540414	1141
8	58.3		58.3	77	132604		319936	2076
9	71.5			83	423004		3600000	840
10	85.5		84.8	87.8	158406		759984	961
11	108.3		105.9	98.3	199940		1158406	1393
12	68.9		74.8	87.8	287578		855068	1204
13	89.6		96.1	81.1	131131		1100843	1126
14	101.1		88.9	83.7	131796		690663	878
15	58.3		70	68.9	144004		383706	1190
16	85.5		88.1	100	135477		404703	1466
17	60.2		56.9	81.1	146133		969334	7141

Continued on next page

Table A.1: Objective function values and computational times of the four parametrised techniques

	I&F	FRF Coord	FRF Coord	tailored	I&F	FRF Coord	FRF Coord	tailored
			& p_{ij}				& p_{ij}	
18	100.5			96.3	221312		3600000	781
19	84.4		84.4	83.7	140680		437714	1177
20	100.9		86.1	87.8	131462		237800	989
21	75.7		77.4	75.6	299184		1315911	951
22	55.9			64.1	182216		3600000	934
23	85.5		85.5	83.7	177645		713061	1097
24	75.5		79.8	93.3	253960		1355698	1724
25	46.1		47.8	50.2	158560		686412	5337
26	58.3			76.3	109481		3600000	1694
27	106.1		103.5	100.2	160513		677298	686
28	60.7		65.7	70	221510		1032950	2237
29				95.9	3600000		3600000	797
30	76.5		71.5	67.2	136635		478996	1227
	$n = 80$							
1	65.7		70.7	74.6	232091		1065197	5910
2	94.4		108.5	105	238187		1152719	2316
3	79.6		79.6	81.1	223943		424319	7743
4	108.3		120.7	95.9	238448		717511	2827
5	106.8		96.1	88.5	314114		958162	1897
6			104.6	114.8	3600000		1218944	1085
7	85.7			75.6	453989		3600000	1469
8			105.9	100	3600000		953297	2420
9	75.7			68.9	354879		3600000	1170
10	80.5		80.5	98.3	292587		886882	1384
11	74.1			104.8	891720		3600000	1769
12	103.3		123.1	106.7	244988		1224007	1358
13	86.1		95.2	87.8	273383		507794	1308
14	73.3			56.7	311623		3600000	2294
15	84.6		86.3	81.1	256017		549954	2229
16	79.6		92.2	100.7	211181		550319	1347
17	119.8		122.4	87.8	182714		524370	1624
18	110.9		110.9	87.8	864549		886649	1942
19	94.4		103.7	81.1	203084		739932	1365
20	114.2		115.2	81.3	371487		1148528	1182
21	98.5		97.8	81.1	256763		924338	1756
22	103.5		111.1	118.9	219580		1083230	1787
23	114.2			111.7	334602		3600000	3089
24	119		116.6	100	314192		888750	1268
25	73.9		83	87.8	231608		825831	1696
26	89.6		93.7	77	315064		612969	2234
Continued on next page								

Table A.1: Objective function values and computational times of the four parametrised techniques

	I&F	FRF Coord	FRF Coord & p_{ij}	tailored	I&F	FRF Coord	FRF Coord & p_{ij}	tailored
27	68.1		68.1	81.1	266356		464929	3250
28	102		84.6	102.6	241510		715984	1945
29	96.1			93.5	227646		3600000	3613
30	100.3		107	125.6	311775		818666	2198
	$n = 90$							
1	96.3		91.3	98.3	421786		837132	4699
2	82.9		82.2	100	329215		1354399	1936
3	118.4			111.3	3573586		3600000	2421
4	92			94.6	281166		3600000	2178
5	129		125.7	125.6	341812		2468876	3397
6	106.6		98.5	95.2	238540		2346069	1925
7	129.6		142.2	118.9	503368		3642109	2728
8	70.7			76.3	557243		3600000	3311
9	86.3		94.6	81.1	382637		1945501	2066
10	114		120.7	102.6	402698		1859322	4516
11	128.1			137.8	359484		3600000	11396
12	80.5		87.2	93.3	276855		2486053	3249
13	87			81.1	289443		3600000	15784
14	89.6			81.1	448377		3600000	3279
15	123.3		124.1	118.9	384415		2577854	3076
16	79.8		98.9	75.6	656716		1189905	1958
17	97.2		86.3	75.6	287800		2154917	2382
18	85.7			75.6	402373		3600000	2240
19				75.6	3600000		3600000	3477
20	106.8			105.7	364230		3600000	1751
21	102		98.7	106.7	283314		1603455	2156
22	115.9		98.5	124.8	407397		1529694	2922
23	82.2		96.3	78.9	565707		1858829	2237
24	94.4			88.5	301200		3600000	1956
25	124.8		148.9	106.7	778578		1971733	1923
26	108.3			81.1	446129		3600000	2635
27	110.9		100.9	106.7	275305		1309747	1878
28	126.4		92	100	614625		1551650	2115
29	88.9			122.6	512472		3600000	2604
30	125.7		106.8	112.2	644020		1241925	2607
	$n = 100$							
1	71.5			77	357386			12591
2	111.8			100	575752			2546
3	118.1			121.5	447500			3781
4	122.4			117.2	599841			6477

Continued on next page

Table A.1: Objective function values and computational times of the four parametrised techniques

	I&F	FRF Coord	FRF Coord & p_{ij}	tailored	I&F	FRF Coord	FRF Coord & p_{ij}	tailored
5	99.4			112.2	1121816			5875
6	104.2			100	343906			4503
7	152.5			139.2	867656			3059
8	88.7			93.9	474238			3624
9	98.5			111.7	415899			2902
10	110.1			118.9	429409			4587
11	104.6			91.1	595499			3729
12	128.9			115.2	451418			2957
13	147.9			144.1	699924			3286
14	96.1			92.8	396523			3207
15	140.3			100	404272			4142
16	120.7			108.5	574143			5743
17	110.3			111.7	513851			2660
18	116.8			87.8	677997			5024
19	89.6			92.8	356536			5398
20	92.2			102.4	383852			10287
21	102			100.7	537759			4343
22	120.7			131.1	1338171			6642
23	145.5			116.7	546330			2296
24	104.4			75.6	473979			4879
25	105.1			112.2	595769			4465
26	118.5			116.7	441146			3259
27	130.5			115.9	465232			9241
28	95.5			111.7	580033			3847
29	104.4			100	549996			4003
30	115.9			130.6	487489			2856

Bibliography

Air Transport Action Group (2016). Facts & Figures. [Online; accessed 18-August-2016]. `http://www.atag.org/facts-and-figures.html`.

Allaz, C. (2005). *History of air cargo and airmail from the 18th century.* Christopher Foyle Publishing.

Almeida, A. D. and M. B. Figueiredo (2010). A particular approach for the three-dimensional packing problem with additional constraints. *Computers & Operations Research 37*(11), 1968–1976.

Alonso, M., R. Alvarez-Valdes, M. Iori, F. Parreno, and J. Tamarit (2017). Mathematical models for multicontainer loading problems. *Omega 66*, 106–117.

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

Ambrosino, D., A. Sciomachen, and E. Tanfani (2004). Stowing a containership: the master bay plan problem. *Transportation Research Part A: Policy and Practice 38*(2), 81 – 99.

Amiouny, S. V., J. J. Bartholdi, J. H. Vande Vate, and J. Zhang (1992). Balanced loading. *Operations Research 40*(2), 238–246.

Arenales, M. and R. Morabito (1997). An overview of and/or-graph approaches to cutting and packing problems. *Decision Making under Conditions of Uncertainly*, 207–224.

Baena, D., J. Castro, and J. A. González (2015). Fix-and-relax approaches for controlled tabular adjustment. *Computers & Operations Research 58*, 41–52.

Beraldi, P., G. Ghiani, A. Grieco, and E. Guerriero (2008). Rolling-horizon and fix-and-relax heuristics for the parallel machine lot-sizing and scheduling problem with sequence-dependent set-up costs. *Computers & Operations Research 35*(11), 3644 – 3656.

Birattari, M., T. Stützle, L. Paquete, and K. Varrentrapp (2002). A racing algorithm for configuring metaheuristics. In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, pp. 11–18. Morgan Kaufmann Publishers Inc.

Bischoff, E. E. and M. S. W. Ratcliff (1995). Issues in the development of approaches to container loading. *Omega 23*(4), 377–390.

Boeing (2008). Weight and balance control and loading manual – model 777f.

Bortfeldt, A. (2012). A hybrid algorithm for the capacitated vehicle routing problem with three-dimensional loading constraints. *Computers & Operations Research 39*(9), 2248 – 2257.

Bortfeldt, A. and G. Wäscher (2013). Constraints in container loading - A State-of-the-Art Review. *European Journal of Operational Research 229*, 1–20.

Brunetta, L. and P. Grégoire (2005). A general purpose algorithm for three-dimensional packing. *INFORMS Journal on Computing 17*(3), 328–338.

Ceschia, S. and A. Schaerf (2013). Local search for a multi-drop multi-container loading problem. *Journal of Heuristics 19*(2), 275–294.

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

Chen, C., S. Lee, and Q. Shen (1995). An analytical model for the container loading problem. *European Journal of Operational Research 80*, 68–76.

Conover, W. (1999). *Pratical nonparametric statistics* (thid ed.). New York: John Wiley & Sons.

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

Côté, J., G. Guastaroba, and M. Speranza (2017). The value of integrating loading and routing. *European Journal of Operational Research 257*(1), 87–105.

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

Davies, A. and E. Bischoff (1999). Weight distribution considerations in container loading. *European Journal of Operational Research 114*, 209–527.

de Queiroz, T. A., F. K. Miyazawa, Y. Wakabayashi, and E. C. Xavier (2012). Algorithms for 3d guillotine cutting problems: Unbounded knapsack, cutting stock and strip packing. *Computers & Operations Research 39*(2), 200–212.

Dolan, E. D. and J. J. Moré (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming 91*(2), 201–213.

Dyckhoff, H. (1990). A typology of cutting and packing problems. *European Journal of Operational Research 44*, 145–159.

Eley, M. (2003). A bottleneck assignment approach to the multiple container loading problem. *OR Spectrum 25*(1), 45–60.

Ferreira, D., R. Morabito, and S. Rangel (2010). Relax and fix heuristics to solve one-stage one-machine lot-scheduling models for small-scale soft drink plants. *Computers & Operations Research 37*(4), 684 – 691.

Fraser, H. J. and J. A. George (1994). Integrated container loading software for pulp and paper industry. *European Journal of Operational Research 77*(3), 466–474.

Garey, M. and D. Johnson (1979). *Computers and Intractability : A guide to the theory of NP-Completeness*. San Francisco: W.H. Freeman.

Gendreau, M., M. Iori, G. Laporte, and S. Martello (2006). A tabu search algorithm for a routing and container loading problem. *Transportation Science 40*, 342–350.

George, J. A. and D. F. Robinson (1980). A heuristic for packing boxes into a container. *Computers & Operations Research 7*(3), 147–156.

International Air Transport Association (2016a). Air Cargo - Enabling global trade. [Online; accessed 18-August-2016]. `www.iata.org/whatwedo/cargo/Pages/index.aspx`.

International Air Transport Association (2016b). IATA Cargo Strategy. [Online; accessed 18-August-2016]. `www.iata.org/whatwedo/cargo/Documents/cargo-strategy.pdf`.

International Transport Forum (2016). Permissible maximum weights of lorries in europe. [Online; accessed 22-August-2016]. `http://www.itf-oecd.org/permissible-maximum-weights-lorries-europe`.

Iori, M., J.-J. Salazar-GonzÃălez, and D. Vigo (2007). An exact approach for the vehicle routing problem with two-dimensional loading constraints. *Transportation Science 41*(2), 253–264.

Ivancic, N., K. Mathur, and B. Mohanty (1989). An integer-programming based heuristic approach to the three-dimensional packing problem. *Journal of Manufacturing and Operations Management* (2), 268–289.

Jin, Z., T. Ito, and K. Ohna (2003). A three-dimensional bin packing problem and its practical algorithm. *JSME International Journal Series C : Mechanical Systems, Machine Elements and Manufacturing*(46), 60–66.

Junqueira, L. and R. Morabito (2015). Heuristic algorithms for a three-dimensional loading capacitated vehicle routing problem in a carrier. *Computers & Industrial Engineering 88*, 110–130.

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

Kang, J. and S. Park (2003). Algorithms for the variable sized bin packing problem. *European Journal of Operational Research 147*(2), 365–372.

Kelly, J. D. and J. L. Mann (2004). Flowsheet decomposition heuristic for scheduling: a relax-and-fix method. *Computers & Chemical Engineering 28*(11), 2193 – 2200.

Liberalino, H. (2012). *Problèmes de Production avec Transport des Composants.* Ph. D. thesis, Université Blaise Pascal - Clermont-Ferrand II.

Lim, A., H. Ma, C. Qiu, and W. Zhu (2013). The single container loading problem with axle weight constraints. *International Journal of Production Economics 144*(1), 358–369.

Limbourg, S., M. Schyns, and G. Laporte (2012). Automatic aircraft cargo load planning. *Journal of the Operational Research Society 63*(0), 1271–1283.

Lin, J.-L., C.-H. Chang, and J.-Y. Yang (2006). A study of optimal system for multiple-constraint multiple-container packing problems. In M. Ali and R. Dapoigny (Eds.), *Advances in Applied Artificial Intelligence*, Volume 4031 of *Lecture Notes in Computer Science*, pp. 1200–1210. Springer Berlin Heidelberg.

Liu, S., W. Tan, Z. Xu, and X. Liu (2014). A tree search algorithm for the container loading problem. *Computers & Industrial Engineering 75*, 20–30.

López-Ibáñez, M., J. Dubois-Lacoste, L. P. Cáceres, M. Birattari, and T. Stützle (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives 3*, 43–58.

Lurkin, V. and M. Schyns (2015). The airline container loading problem with pickup and delivery. *European Journal of Operational Research 244*(3), 955–965.

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

Martello, S. and P. Toth (1990). *Knapsack Problems: Algorithms and Computer Implementations*. New York, NY, USA: John Wiley & Sons, Inc.

Mongeau, M. and C. Bès (2003). Optimization of aircraft container loading. *IEEE Transactions on Aerospace and Electronic Systems 39*, 140–150.

Moon, I. and T. Nguyen (2013). Container packing problem with balance constraints. *OR Spectrum 36*, 837–878.

Moura, A. and J. F. Oliveira (2005). A GRASP approach to the container-loading problem. *IEEE Intelligent Systems 20*(4), 50–57.

Oliveira, B. B., M. A. Carravilla, J. F. Oliveira, and F. M. Toledo (2014). A relax-and-fix-based algorithm for the vehicle-reservation assignment problem in a car rental company. *European Journal of Operational Research 237*(2), 729 – 737.

Parreño, F., R. Alvarez-Valdés, J. Oliveira, and J. M. Tamarit (2010). Neighborhood structures for the container loading problem: a vns implementation. *Journal of Heuristics 16*(1), 1–22.

Pochet, Y. and L. A. Wolsey (2006). *Production Planning by Mixed Integer Programming*. Springer.

Pollaris, H., K. Braekers, A. Caris, G. K. Janssens, and S. Limbourg (2015). Vehicle routing problems with loading constraints: state-of-the-art and future directions. *OR Spectrum 37*(2), 297–330.

Ramos, A. G., J. F. Oliveira, J. F. Gonçalves, and M. P. Lopes (2016). A container loading algorithm with static mechanical equilibrium stability constraints. *Transportation Research Part B: Methodological 91*, 565–581.

Ramos, A. G., J. F. Oliveira, and M. P. Lopes (2016). A physical packing sequence algorithm for the container loading problem with static mechanical equilibrium conditions. *International Transactions in Operational Research 23*(1-2), 215–238.

Rodrigue, J.-P., C. Comtois, and B. Slack (2013). *The geography of transport systems*. Routledge.

Sciomachen, A. and E. Tanfani (2003). The master bay plan problem: a solution method based on its connection to the three-dimensional bin packing problem. *IMA Journal of Management Mathematics 14*(3), 251–269.

Seiden, S. S. (2001). *On the Online Bin Packing Problem*, pp. 237–248. Berlin, Heidelberg: Springer Berlin Heidelberg.

Techanitisawad, A. and P. Tangwiwatwong (2004). A GA-based heuristic for the interrelated container selection loading problems. *Industrial Engineering and Management systems 3*, 22–37.

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

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

Tsai, R., E. Malstrom, and W. Kuo (1993). Three dimensional palletization of mixed box sizes. *IIE Transactions 25*(4), 64–75.

Vancroonenburg, W., J. Verstichel, K. Tavernier, and G. Vanden Berghe (2014). Automatic air cargo selection and weight balancing: a mixed integer programming approach. *Transportation Research E, Logistics and Transportation Review 65*, 70–83.

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

Westerlund, J., L. G. Papageorgiou, and T. Westerlund (2005). A problem formulation for optimal mixed-sized box packing. *Computer Aided Chemical Engineering 20*, 913–918.

Westerlund, J., L. G. Papageorgiou, and T. Westerlund (2007). A MILP model for N-dimensional allocation. *Computers & Chemical Engineering 31*(12), 1702–1714.

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

Index

List of Figures

List of Tables