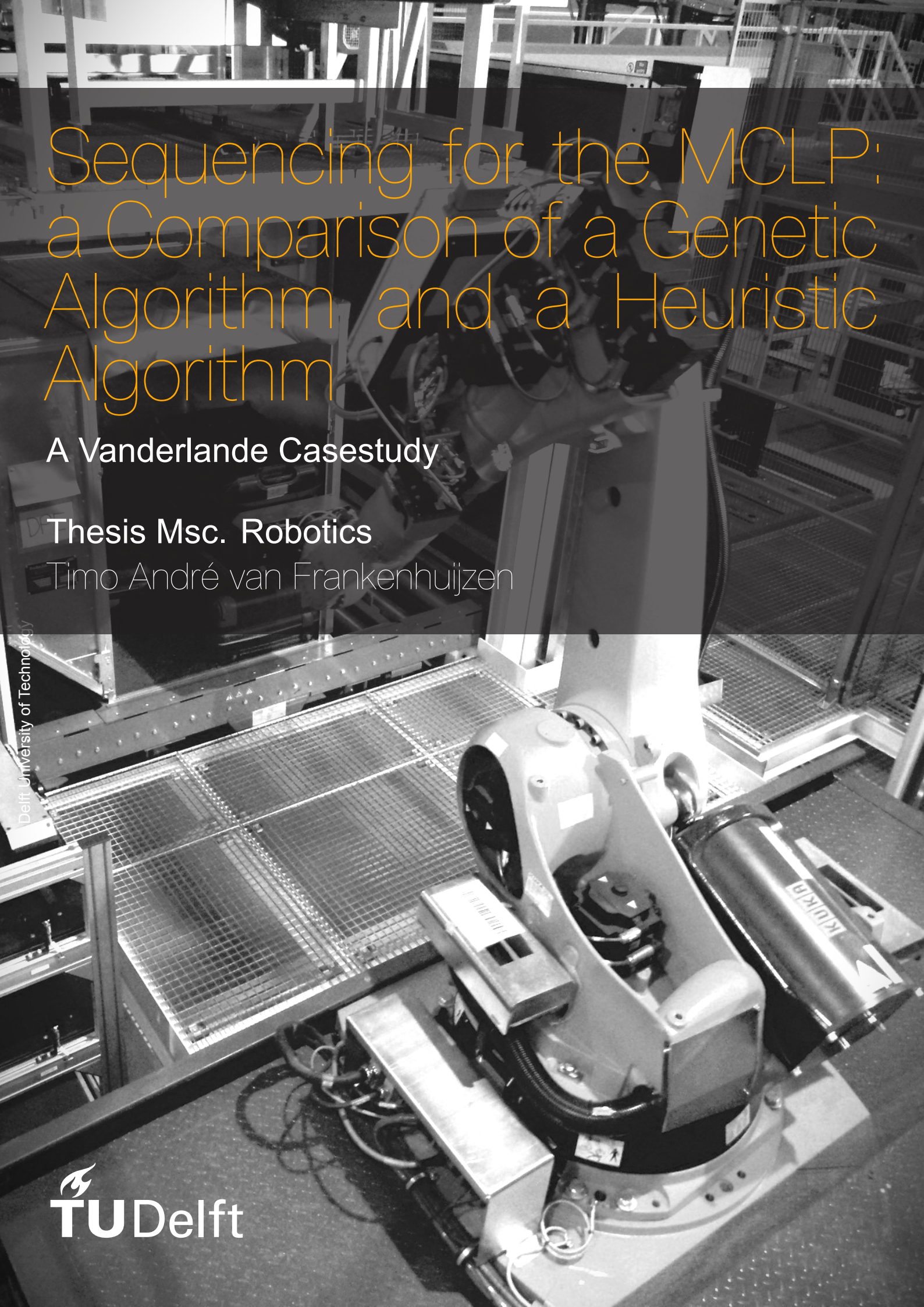


Sequencing for the MCLP: a Comparison of a Genetic Algorithm and a Heuristic Algorithm

A Vanderlande Casestudy

Thesis Msc. Robotics

Timo André van Frankenhuijzen

Delft University of Technology

TU Delft

Sequencing for the MCLP: a Comparison of a Genetic Algorithm and a Heuristic Algorithm

A Vanderlande Casestudy

by

Timo André van Frankenhuijzen

Student Name	Student Number
Timo van Frankenhuijzen	4483448

Supervisor:	Prof. dr. ir. J. (Hans) Hellendoorn
Company Supervisor:	ir. S. (Stijn) de Looijer
Institution:	Delft University of Technology
Place:	Vanderlande Industries, Veghel
Project Duration:	January, 2022 - August, 2022

Cover Image: ULD Loading Robot at Heathrow Airport

TU Delft

Abstract

Labour shortage in the logistic sector and the poor work conditions as a baggage handler are a major and highly relevant problem for airports around the world. A solution is provided by Vanderlande Industies: a baggage loading robot. Since the efficiency of the stacking process is sufficeint, operators that overlook the robot must intervene often. To minimse these interventions, better packings have to be made. To make better packings, Vanderlande Industries can adapt the sequence in which the items are presented to the robot. However, literature and the company are not aware of the effects of sequencing. Thereby in this thesis, we conducted research into sequencing items in various ways for the Multiple Container Loading Problem (MLCP). We attempted to find out which methods enhance the fill rate of the packings. In order to understand the academic background of the MCLP, we first looked at the literature. Additionally, state-of-the-art techniques have been investigated. A trip to Schiphol has been undertaken to observe the real robot that is resolving the MCLP as a pilot project for Schiphol in order to put the issue in its proper perspective. From this research, we concluded that there are various methods for solving the MCLP, however, sequencing methods applied to so far are mostly kept simple. Moreover, in practice, baggage items are placed in the order where size and weight are decreasing. The knowledge gap found was that sequencing for the MCLP is not researched. We present two different types of algorithms to provide a well performing sequence. The first algorithm, the genetic algorithm, makes use of multiple generations to optimise the sequence. The other algorithm, the heuristic algorithm, makes use of a set of rules. The resulting packings formed with different sequencing methods show to be different. According to the simulation performed, the heuristic sequencing strategy produces the best results for the main scenario in terms of the layer fill rate. The performance of the packings can with the heuristic sequencer be increased by average with 11 % in comparison with packings generated with a random item sequence. Additionally, the heuristic sequencer's packings are fairly consistent because it consistently manages to place 8 items in one layer of the container. Results in this study point out that sequencing the items with size decreasing performs worse than the packings formed from non-sequenced items, with a mean fill rate of 0.71 for the size decreasing items and 0.87 for the heuristically sequenced items. The alternate scenario produces fewer evident outcomes, but we can point out that simple strategies, such as sorting the items by short side, perform effectively. The results from this applied method: an increase of 7% in comparison with non-sequenced item packings. We assume the simple sorting approach worked well since it grouped all similar shaped objects, resulting in tight packings. The genetic algorithm produced slightly better results, with average outcomes being similar (0.87), but much more consistent packings showing a lower standard deviation of 0.00917 (GA) versus a standard deviation of (0.01747). The results from the heuristic algorithm were the least efficient. This demonstrates that sequencing using a heuristic is effective, but it is not a reliable strategy in all circumstances. The genetic algorithm, in comparison, generates generally worse results, but it can adjust to various situations and configurations. Although the results seem to be convincing, there rise discussion points. One of the major drawbacks is the fact that in this study, simulations are only done in two dimensions. This results in a few assumption regarding stacking behaviour. Thereby, a significant point of future work can be done by introducing the sequencing algorithm to three-dimensional simulations.

Acknowledgements

I have spent the last ten months at Vanderlande as a graduate intern working on this thesis. During my time at this company I have learned a lot. Without this group of people who helped me along the way, I would not have learned the lessons and acquired the guidance that I have. This acknowledgement is given to them as a gesture of gratitude.

The first person I want to thank is my TU Delft supervisor, Prof. dr. ir. Hans Hellendoorn. His guidance and flexibility has really helped me in creating this thesis. Also, despite his busy schedule, he always answered my email promptly, which significantly helped the process. I have discovered that he always placed equal importance on my personal well-being rather than only the thesis itself. This has been tremendously appreciated and has increased my enjoyment of working with you.

Second, I want to express my gratitude to Vanderlande, especially the Innovate team, for welcoming me as an intern. During this thesis, I had the feeling that I was really welcome in the team. Also, everyone in the team valued the input of interns greatly, which I much enjoyed. I am especially grateful to Stijn de Looijer, who was my daily supervisor during this thesis. He was always very involved in the process and was able to give helpful feedback in each of our weekly meetings. Furthermore, I would like to thank him for casual conversations we could which were often off-topic.

Moreover, I would like to thank Wouter Meijer, Alex Keijzer and Berry Vermin for the support in our sparring sessions. Their advice helped me greatly to progress in this thesis. Also, I would like to thank Mitchel Puglia and Mischa de Haan for the reflection sessions and their listening ear. Furtermore, I would like to thank Roselind Vugts for her unconditional support during the whole project, she kept always on track.

Finally, I want to express my gratitude to my family and friends for their help. Without their assistance, I could not have completed my thesis.

Contents

List of Acronyms

Abbreviations

Abbreviation	Definition
MCLP	Multiple Container Loading Problem
NIOSH	National Institute of Occupational Safety and Health
IATA	International Air Transport Association
BHP	Baggage Handling Process
BHS	Baggage Handling System
AV	Autonomous Vehicles
OOD	Out Of Domain
ST	Standard
SP	Special Size
JIT	Just In Time
ACP	Automated Case Picking
CLP	Container Loading Problem
MILP	Mixed Integer Linear Program
MIP	Mixed Integer Program
FF	First Fit
BF	Best Fit
BBA	Block Building Algorithm
DLR	Deep Reinforcement Learning
RGBD	Red Green Blue Depth
KPI	Key Performance Indicator
GA	Genetic Algorithm
GRASP	Greedy Randomized Adaptive Search Procedure

List of Figures

1 Introduction

1.1. Background

It is the 23rd of April 2022, Schiphol calls on travellers not to come to the airport due to heavy traffic and a series of cancellations (NOS, 2022a). The military police had to close off the exit lanes to Schiphol. This chaos was the result of a major strike from all baggage personnel, paired with the high traffic around the airport. According to (FNV, 2022), the strike was organised for two reasons: poor working conditions and enormous work pressure. In this introduction, we will explain that baggage handling personnel have a proven point of concern regarding their work pressure and poor working conditions.

It is undeniable that the world's demographics is fast changing as a consequence of ageing (United Nations, 2019). Moreover, according to (Foot, 2008), the approaching retirement of the baby boom generation raises the risk of future labour shortages. This results in a shortage of labour in some working class sectors. Meanwhile, the aviation industry is growing significantly. A report from the (IATA, 2019) claims that passenger numbers of many markets will double (or more) by 2035. The problem is particularly urgent for large airports, since most of them are frequently found in high-growth developing countries with an even higher shortage of specialized labour (World Maritime University, 2019). This labour shortage also impacts the airport Schiphol, which is resulting in a growth in work pressure.

As Joost van Doesburg claimed in (FNV, 2022), baggage handling personnel also works under poor working conditions. Multiple sources claim that baggage handling leads to pain in their lower back and shoulders. Following research from (Bergsten et al., 2015), of all workers involved in baggage handling, 70% and 60% reported pain in, respectively, the lower back and shoulder. Also, according to (Oxley et al., 2009), 73% of baggage handlers reported trouble with their lower back. In 2000, (U. S. Bureau of Labor Statistics, 2009), even revealed that airline workers lose time due to injuries up to 5 times the national average. Recent sources claim that these problems also apply to Schiphol (Holdert & Meindertsma, 2022). These are worrying statistics that need to be addressed.

(a) Unloading a cart at Stuttgart airport

(b) Loading a ULD at Oslo Airport

Figure 1.1: Baggage Handling Work

Partly as a result of these problems, Schiphol cannot grow anymore and has to cancel flights on a regular basis (NOS, 2022b).

One of the most physical demanding jobs of the baggage handler is baggage loading. Baggage loading (or make-up) (see Figure 1.1b) is the process of efficiently stacking luggage pieces from the baggage transport line in a Unit Loading Device (ULD) (Figure 1.2a) or on a cart (Figure 1.2b). To decrease the work pressure and increase the working conditions for baggage handlers doing this job, Vanderlande Industries is working on a modern solution to one of the problems: a baggage loading robot.

(a) Unit Loading Device (ULD)

(b) Cart

Figure 1.2: Various containers for baggage

The first project introducing robots for baggage loading was started in 2009 by Vanderlande. Over the years, pilot setups were made and installed at airports as Schiphol and Heathrow Airport (see Figure 1.4). Although the few pilot projects involving this robot show major potential, the operational costs of the setup are a bottleneck for further deployment. Although the robot does its work automatically,

the operational costs are high since an operator is still needed. Due to the inefficient fill rates and poor reliability regarding the handling of the bags by the robot, a baggage handler must oversee the robot cell. Figure 1.3 provides a figure with a root cause analysis of the interventions of the operator. With supervision, the robot is working properly, but the deployment of one handler per robot cell increases the operational costs significantly. To cut the operational costs, Vanderlande is now working on a more intelligent and efficient robot cell, which needs significantly less supervision by handlers. This thesis is conducted in light of that strategy.

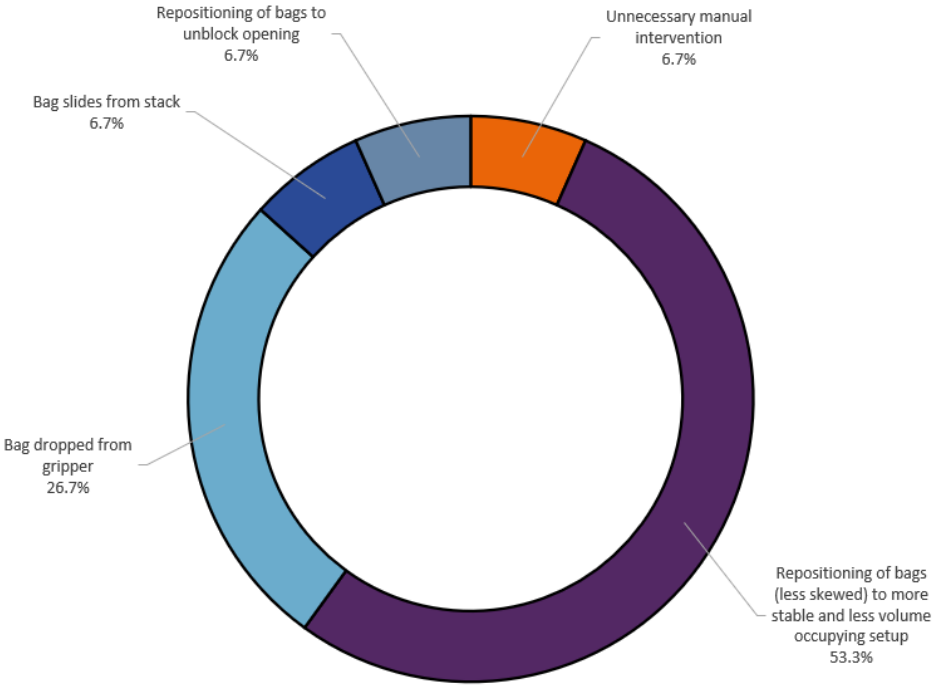


Figure 1.3: Root Cause Analysis for Manual Intervention Baggage Loading Robots by Vanderlande

One of the factors contributing to the efficiency of the robot cell is the stacking robot itself and the software that is determining placement of the items. However, the robot is likely not to be designed and programmed by Vanderlande Industries. The company does not control the stacking strategy of the robot. However, Vanderlande controls the baggage loading process before the bags are given to the robot, which means that they control the sequence. The aim of this thesis is to discover the possibilities of sequencing methods for baggage loading.

1.2. Company

Vanderlande Industries is a market leading system integrator for airport logistics. Their services are used all over the world, including 12 of the top 20 airports (Vanderlande, 2022). The company has an R&D department divided up into an Innovate Department and a Development Department which are constantly innovating the baggage handling systems' business for their customers. They delineate their activity into business units. The business units include airports, warehousing, Amazon, parcel and fashion. The company has good relations with their key customers, especially in the airports business unit (Vanderlande Industries, 2021). Schiphol was one of the first key customers. With the problems at Schiphol and other airports having to schedule less flights, a strong market pull effect is created since the market is asking for technical solutions to their problems. On the other hand, the R&D department of Vanderlande Industries wants to innovate and apply new technologies in solutions and products. This effect is called the technology push effect. When taken together, these effects provide for a convincing business case for automating the baggage loading process.

1.2.1. Current Setup

Baggage loading or make-up is the last step of the Baggage Handling Process (BHP). Firstly, baggage is loaded at the check-in counter or at the self-service baggage drop points. From there, the size of the baggage will be classified as Standard Normal Conveyable baggage (ST), Semi-Conveyable baggage (SC), Out of Gage baggage (OG), Odd Size baggage (OS) or Special Baggage (SP) depending on size and shape. Only ST baggage will be handled by the baggage loading robot. More about these baggage types will be discussed in Chapter 3. After check-in and classification, the bags will be identified with the barcode attached to the bag. Then, the bag will be transported to a screening unit and moved to the Early Bag Store (EBS) or sorted to the correct make-up location. All baggage sorted directly to the make-up locations will be loaded in a cart or ULD manually. Around 1.5 hours before departure, bags that are stored in the EBS are transported to their make-up locations. The baggage loading robot loads a small percentage of the luggage stored in the EBS, while the remainder of the cargo must be loaded by hand. This is done since it will provide predictability over the baggage items.

The current robot cell consist of a simple Vertisorter (Vanderlande Industries, 2020), where the incoming bags from two transport lines come together. After the right bag is chosen, the bag will be conveyed to the end of the conveyor belt. Then, the robot will retrieve the bag from the conveyor belt, and it will place the bag in the cart or ULD according to online load logic (see Figure 1.4). Afterwards, the robot arm with camera on top will scan the cart or ULD and check if the baggage placement is correct. If the placement did not succeed, an error message is given to the operator, and the operator can stop the robot, enter the robot cell and fix the placement. In practice, the operator also intervenes when inefficient stacks are made or baggage is placed skewed (see Figure 1.3). If the bag is placed correctly, the next bag can be picked from the conveyor belt.

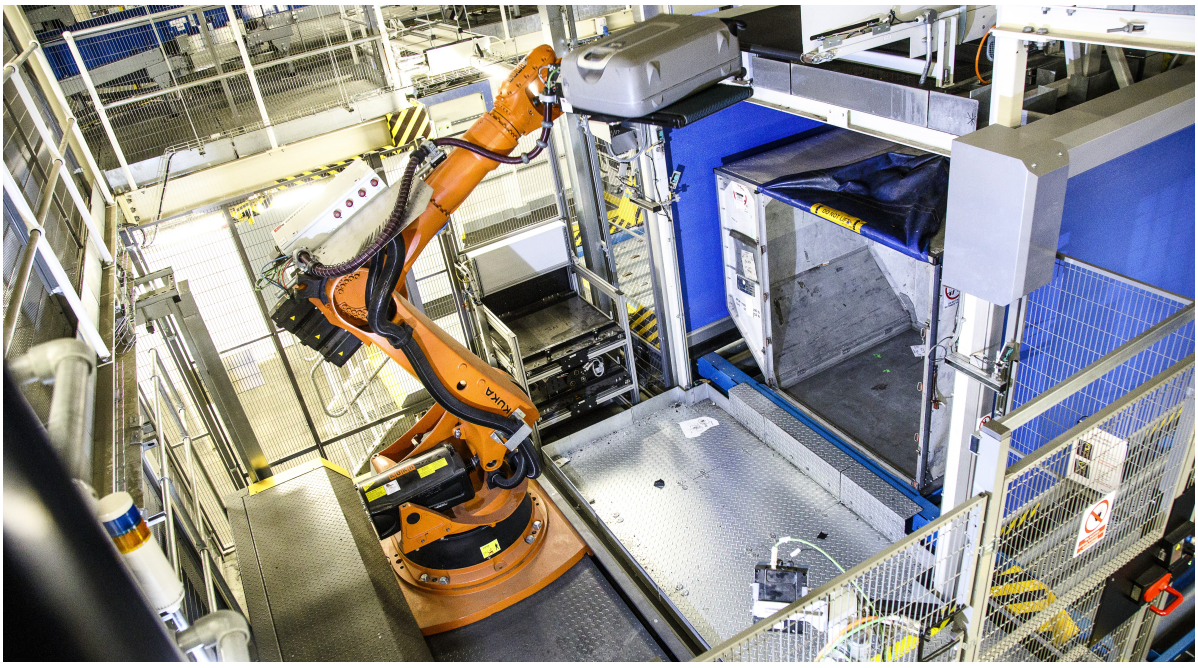


Figure 1.4: Baggage loading Robot at Heathrow Airport

After the cart or ULD is loaded, it will be attached to a vehicle and is transported to the corresponding airplane. ULDs will be directly loaded into the belly of the airplane. The baggage loaded into carts will be unloaded at the airplane and loaded manually into the under deck of the airplane piece by piece. A more detailed description of the process can be found in Chapter 3.

1.3. Research Goal

The robot will make use of a container loading problem algorithm, an algorithm which determines the placement of the items. The goal of a MCLP (Multi Container Loading Problem) algorithm is to fill a minimal set of bins with a set of items. To make sure the fewest number of bins is used, the packing

algorithm must fill each bin as optimally as possible which means is full as possible. In this thesis, we will refer to this fullness with the term 'fill rate'. In practice, such an algorithm is used for the baggage loading robot to fit as many as bags possible in a cart or ULD. In this thesis, the fill rate is considered as the best metric to evaluate packings, however, over multiple runs, the consistency of the fill rate is also key. As stated in Section 1.1, Vanderlande industries is very likely to buy the physical baggage loading robot, and thereby, does not control robot. This means that Vanderlande does not have the influence on how the packings are made. However, Vanderlande controls the Baggage Handling Process before the robot, which means they control the sequence in which the bags are presented to the robot.

In this thesis, we will research the performance difference between various loading algorithms provided with non-sequenced items, a simple sequenced set of items, a heuristically sequenced item and an item set sequenced with a genetic algorithm. Here, the variables fill rate, fill rate standard deviation and computational time are taken into account. The algorithm combinations will be tested in various testcases. The main setup consist of the baggage loading setup explained in Section 1.2.1 and the alternative scenario will consist of a similar setup for parcel loading.

To summarize, the main goal of this thesis is to answer the following research question:

> *How does sequencing items improve fill rate of simple container loading algorithms in a baggage and parcel loading context?*

We believe that sequencing items in most cases will work to improve fill rate of the simple online container loading algorithms. Because with sequencing, one can be a step ahead in packing efficiently and plan for significant better placement for the items. A more efficient placement can result in more items placed in a container, which results in higher fill rates and thus better performance. We expect that sequencing algorithms specifically designed for a scenario with known items and container sizes will perform better than general sequencing rules or algorithms. However, a stricter sequencing plan leads to a less adaptable packing algorithm for errors and less performance in different scenarios.

1.4. Outline

The remainder of this thesis is structured as follows. In Chapter 2, the container loading problem will be explained and variations, simple and state-of-the-art methods will be discussed. Chapter 3 will explain the dataset and the robot cell of the various setups. The design and results of the genetic sequencing algorithm will be discussed in Chapter 4. The design and results of the final algorithm, the heuristic sequencing algorithm, will be explained in Chapter 5. In Chapter 6 all the results will be compared and benchmarked to the regular sequencing strategies. Also, this chapter will elaborate on a new proposal regarding a sequencing setup. Finally, we will summarize and discuss all results in Chapter 7. Also, future work will be elaborated upon in this chapter.

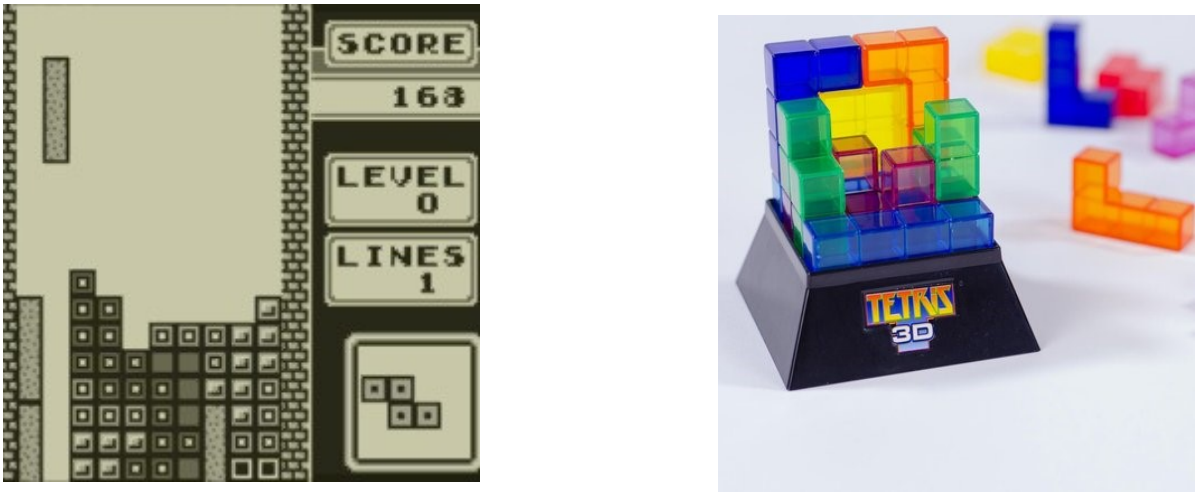
2

Related Work

This chapter is about understanding the Multi Container Loading Problem (MCLP) and the methods for solving the problem. The first section, Section 2.1, will explain the concept in mathematical terms, also variations and constraints of the MCLP will be discussed. Next, classic simple solutions to the problem will be discussed in Section 2.1.3. Afterwards, in the third section, Section 2.2, a set of metaheuristic algorithms will be presented. Finally, we identify a knowledge gap in the final subsection Section 2.3.

2.1. Multi Container Loading Problem

The MCLP is a mathematical geometric assignment problem, in which three-dimensional items have to be packed into large rectangular containers such that a minimum amount of containers is used and feasibility conditions hold (Bortfeldt & Wäscher, 2013). The general constraints in the MCLP are that the items must not overlap and items must be placed completely inside their respective containers. When just a limited number of containers are available, a subset of the available items must be selected in order to minimise the quantity of containers used. In literature, this problem is often called the 'Bin Packing Problem'. In this thesis, we will refer to it as the Multi Container Loading Problem or MCLP. In essence, the problem with a single container resembles the popular three-dimensional Tetris game from 1984, in which a player must place cubes on a platform efficiently. The game promotes placement of the pieces such that horizontal layers are constructed by removing every completed horizontal layer while increasing the score. The game has a limited amount of item types which results in perfectly fitting stacks or layers. Many of the variations of the MCLP deal with a more diverse item set. Details of item heterogeneity will be discussed in 2.1.2.



(a) Tetris the video game, Source: https://alchetron.com/Tetris-(Game-Boy)

(b) Tetris 3D, Source: https://www.bol.com/

Figure 2.1: Tetris games

The MCLP belongs to the cutting & packing problems and was first researched by (Kantorovich, 1960). By now, according to (Skiena, 1997), the MCLP is the second most requested academic problem, in part because of its widespread application. The MCLP has a wide range of uses in contemporary packaging, transport, and manufacturing (H. Zhao et al., 2021). Furthermore, the use of the *online* variant of the problem extends to other sectors like Bandwidth Allocation, Cloud Computing and Operation Room Scheduling (Perez-Salazar et al., 2021). Moreover, in our increasingly connected society, where speed and efficiency in the delivery of goods are key concerns, logistics and transportation systems play a crucial role. As a result, the problem is intensively researched in the field of operational research during the previous few decades (Duan et al., 2019).

2.1.1. Problem Definition

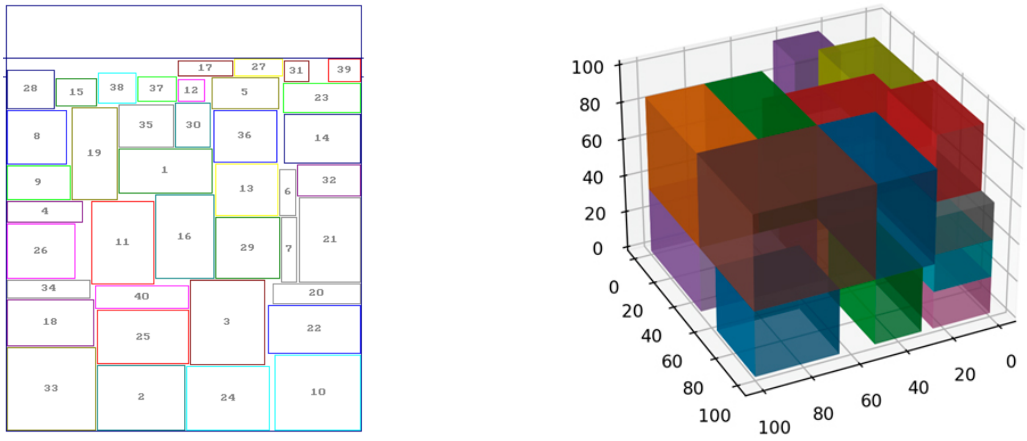
While Tetris is mostly played in 2D, the problem also comes in one and three-dimensional form. The most simple variant is the one dimensional variant. Logically, this problem is called the one-dimensional MCLP. In that problem, there must be more than 1 container in which to load the items in. All items have weights (or volumes) and all containers have a maximum capacity for weight (or volume). Items must be assigned to multiple containers such that the total weight of items in a container is below the container its capacity (H. Zhao et al., 2020) and the amount of containers used is minimised. The mathematical formulation goes as follows. Given a set of $n > 0$ items $N = 1, ...n$ with sizes of each item s_i with $i \in N$ and $0 < s_i < a$. Also, there exists a set of m containers (C_j), all with the same capacity $a > 0$. The one-dimensional MCLP concerns placing items in the container while the sum of the item sizes does not exceed a and the amount of containers used is minimised.

$$\bigcup_{j=1}^{m} C_j = N \tag{2.1}$$

And the following conditions apply:

$$\sum_{i \in C_j} s_i \leq a, \quad 1 \leq j \leq m, \quad i \in N \tag{2.2}$$

The two-dimensional MCLP is slightly different. Here, there exists a set of $n > 0$ items $N = 1, ...n$ with items having a width and a height w_i, h_i and containers having a width and height W_j, H_j where for each item $0 < w_i < W_i \wedge 0 < h_i < H_i$, which means that the item is smaller than the container. When the MCLP allows rotating the items over 90 degrees, $0 < w_i < W_i \wedge 0 < h_i < H_i \vee 0 < w_i < H_i \wedge 0 < h_i < W_i$ applies. Again, the two-dimensional multi container loading problem concerns placing items in the container while the items do not overlap and the stack of items does not exceed the size of the container. An example of this is shown in Figure 2.2a.



(a) 2D Container Loading Problem by darkmakukudo @ Github,

(b) 3D Container Loading Problem by (inpacking)

Figure 2.2: Container Loading Problem

The focus of research on the MCLP will be three-dimensional formulation. This version is the same as the two-dimensional MCLP, but extended with one extra dimension. There exists a set of $n > 0$ items $N = 1, ...n$ with items having a width, a height, and a length w_i, h_i, l_i and containers having a width, height and length W_j, H_j, L_j where for each item $0 < w_i < W_i \wedge 0 < h_i < H_i \wedge 0 < l_i < L_i$, which means that the item is smaller than the container. Compared to the other two versions, the two-dimensional MCLP concerns placing items in the container while the items do not overlap and the stack of items does not exceed the size of the container. An example of this is visible in Figure 2.2b.

The MCLP is a classic *NP Hard* (non-deterministic polynomial-time hardness) type problem (Duan et al., 2019), (Silva et al., 2019), (H. Zhao et al., 2020). In short, following (Rayward-Smith, 1986), "every problem L in NP can be solved in polynomial time by an oracle machine with an oracle for H". In spite of the NP hardness, when the problem is comprehensible enough, MCLP can be solved exactly. This type of methods are called *Exact methods*. Examples of exact methods will be given in 2.1.3.

2.1.2. Problem Variations

As discussed in the previous section, the first variable for the MCLP is the dimensionality of the problem. Nonetheless, there are many problem variants found in the literature. In this section, we will elaborate on the most occurring variations. First, the online and offline variant will be explained. In the online variant, the problem must be solved iteratively. All the items arrive one at the time and the MCLP algorithm has to make a (irreversible) decision where to pack the item before the characteristics of the next item are known. The online MCLP algorithm does not know if there are any more items to pack. The offline variant of the MCLP is the problem where all item characteristics are known in advance. This makes the algorithm able to adapt the sequence of the items before placing the items. Also, offline MCLP algorithms select any of the items from the sequence while a part of the stacking is already done. Most of the time, this type of algorithm outperforms the online variant regarding fill rate. Other variants of the MCLP depend on the items to pack and the containers to be packed. Following (Bortfeldt & Wäscher, 2013), a set of distinctions can be made. Bortfeldt and Wäscher name MCLPs with a weakly heterogeneous set of items cutting stock problems and MCLPs with a strongly heterogeneous set of items.

- Single Stock-Size Cutting Stock Problem (SSSCSP): Packing a weakly heterogeneous set of items into a minimum number of identical containers.
- Multiple Stock-Size Cutting Stock Problem (MSSCSP): The assignment of packing a weakly heterogeneous set of items into a weakly heterogeneous set of containers such that the number of used containers is minimized.
- Residual Cutting Stock Problem (RCSP): The CLP where a weakly heterogeneous set of items needs to be packed into a strongly heterogeneous set of containers such that the number of used containers is minimized.
- Single Bin-Size Bin Packing Problem (SBSBPP): Packing a strongly heterogeneous set of items into a minimum number of identical containers.
- Multiple Bin-Size Bin Packing Problem (MBSBPP): The assignment of packing a strongly heterogeneous set of items into a weakly heterogeneous assortment of containers such that the number of used containers is minimized.
- Residual Bin Packing Problem (RBPP): The MCLP where a strongly heterogeneous set of items needs to be packed into a strongly heterogeneous set of containers such that the number of used containers is minimized.
- Open Dimension Problem (see Figure 2.3) with a weakly heterogeneous set of items: The assignment of packing a weakly heterogeneous set of items into a single container with one or more variable dimensions such that the container volume is minimized
- Open Dimension Problem (see Figure 2.3) with a strongly heterogeneous set of items: The assignment of packing a strongly heterogeneous set of items into a single container with one or more variable dimensions such that the container volume is minimized

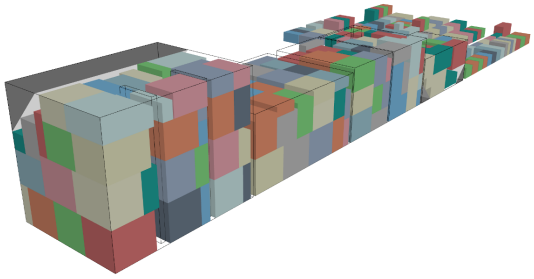


Figure 2.3: Open Dimension Problem by Marius Merschformann @ Github

As stated in the list, three variables need to be specified: the heterogeneity of the items, the container sizes and the open dimension variant.

In literature, the MCLP is solved with a few general constraints. As stated in the first paragraph of this section, general constraints in the MCLP are that the items must not overlap and items must be placed completely inside their respective containers. Although in all literature, the MCLP is constrained with these rules, many additional constraints can also be found. For example, vertical stability is often considered as one of the most important constraints beyond the general constraints. When this constraint is not satisfied, a load can be damaged because items fall from the stack. Many papers do not explicitly mention load stability because authors argue that stability is an immediate result of high fill rates. The stability also depends on the presence of walls. In the classic 'palletizing case', the container is not walled and in some cases, cling wrap is used to prevent stacks from falling (see Figure 2.4b). This is also the case with the automated case picking robot from Vanderlande Industries (Figure 2.4a). According to (Bortfeldt & Wäscher, 2013), load stability is taken into account by demanding that the base of an item must be supported for at least 70% by an item or the container. In some cases, horizontal stability is also taken into account, which is important when moving the container. When vertical stability is taken into account during moving, the inertia of the items must not change the structure of the stack.

(a) Automated Case Picking

(b) Packing wrapped with cling wrap

Figure 2.4: Automated Case Picking by Vanderlande Industries

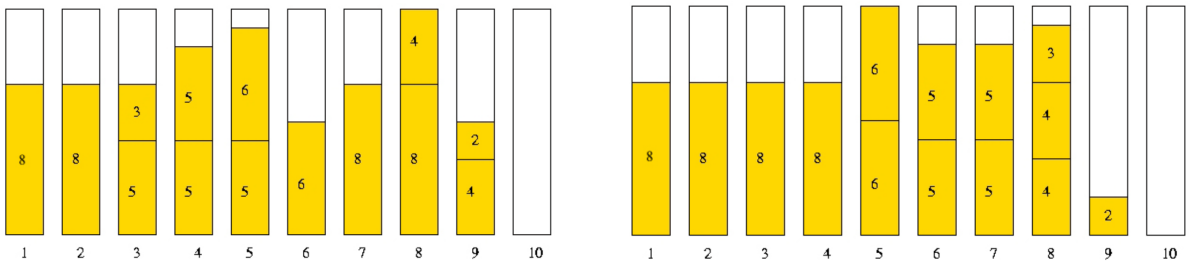
The MCLP algorithm used for the bag loading robot is special. It can control the sequence of the items, however, it treats the items iteratively in an online way. Within Vanderlande Industries, this online MCLP algorithm is called *Ad Hoc Load Logic*, referring to the irreversibility of the placement decision of each item. The items that need to be placed in the containers are typically strongly heterogeneous, since each bag is different in size. The container type for the baggage loading robot depends on what

type of plane the bags needed to be loaded for. In case of a narrow body plane, carts are used as containers. These are rolling containers open on the front and top. One can argue that for this type of loading, the open dimension problem with strongly heterogeneous items is solved. In the case of a wide body plane, ULDs are used. These are containers with a specific shape to fit in an airplane. For this type of container, the single bin-size bin packing problem is solved. More information regarding the container and item characteristics in Chapter 3.

2.1.3. Classic Solutions

In this subsection, we will discuss classic solutions to the MCLP. When considering multiple containers, a MCLP algorithm can choose in which container the item must be placed. This can be done with one of the following heuristics, using the online variant of a MCLP algorithm.

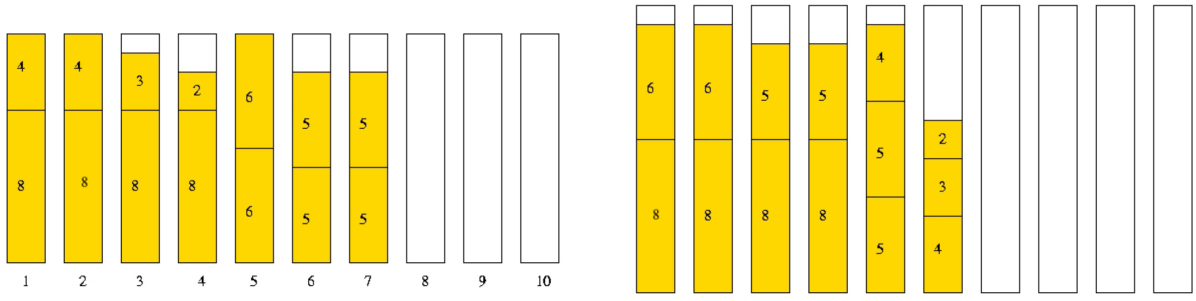
- First-Fit (FF): This algorithm attempts to stack the item in the first container in which it fits.
- Next-Fit (NF): This algorithm only keeps one container open and places the item in that container. If the item does not fit, the current container is closed, the next container is opened and the item is placed in this next container (see Figure 2.5).
- Best-Fit (BF): This algorithm keeps all containers open, when an item arrives. It will attempt to place the item in the container with the smallest unused area after placement of that item.
- Worst-Fit (WF): This algorithm keeps all containers open, when an item arrives. It will attempt to place the item in the container with the largest unused area after placement of that item (see Figure 2.6).



(a) Next Fit Algorithm

(b) Next Fit Decreasing Algorithm

Figure 2.5: Next Fit Algorithms



(a) Worst Fit Algorithm

(b) Worst Fit Decreasing Algorithm

Figure 2.6: Bin Packing Algorithms by (Quarrrb, 2022)

With the offline variant, the sequence of the items can be controlled. Typically, the items will be sorted by size (or weight). The term *decreasing* will be added to the name of the algorithm that first sorts the items from largest to smallest. For example, a first-fit algorithm, will be named first-fit decreasing.

In the next subsections, we will discuss the placement methods for the MCLP. When the container is chosen with one of the heuristics from this subsection, the item will be placed in the container on the location calculated by such a placement method.

Exact Solutions
The first placement algorithm types are the exact solution methods. Because of the complexity of the MCLP, the literature offers few exact solution methods. (Chen et al., 1995) provided the first Mixed Integer Programming for the MCLP based on the relative positions of items. In another paper, a branch-and-bound algorithm is applied to reach an optimal solution for the MCLP by (Martello et al., 2000). When exact methods find a solution, the solution is optimal, the problem is the time needed to calculate such an optimal solution. Exact methods are computationally demanding. Only moderate sized problems are solved in optimality, and they are typically very slow. Far more used methods for the MCLP are placement heuristics.

Placement Heuristics
In this section, we will discuss various placement heuristics. Placement heuristics are efficient methods to solve the MCLP within reasonable time, reaching near optimal solutions. An example of a placement heuristic is the Block Building Approach (BBA). In (Zhu et al., 2012), they came up with a BBA structure where they identify the key decision components for a BBA. As a result, they structured the BBA to 6 elements:

1. How to represent free space in the container
2. How to generate a list of blocks (of items)
3. How to select a free space
4. How to select a block
5. How to place the selected block in the selected space
6. What is the overarching search strategy

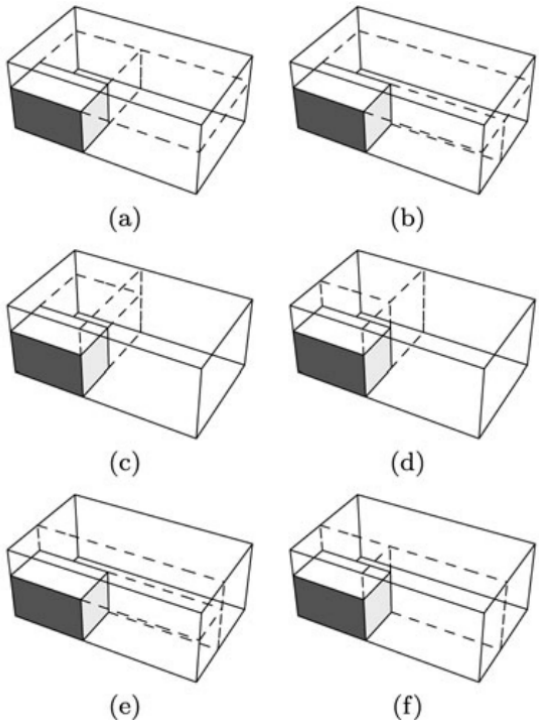
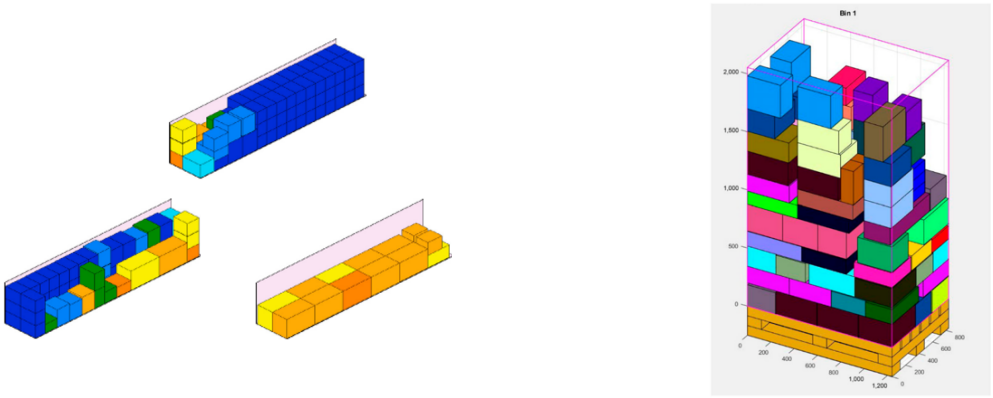


Figure 2.7: Block Building Algorithm by (Zhu et al., 2012)

In essence, BBAs are dividing the container space into blocks and clustered items, afterwards, it matches the clusters with the blocks (see Figure 2.7). Another commonly used placement heuristics is the layer-building heuristic (George & Robinson, 1980). For horizontal layer building (or floor layer building), during the construction of a stack, a layer that is parallel to the ground of the container is created (Moon & Nguyen, 2014). Afterwards, the height of this layer is filled with smaller items to fill

up the space. An example packing, packed with a layer building heuristic, is displayed in Figure 2.8b. For a vertical layer (or wall) building, this process is done perpendicular on the side of the container: a *wall* of items is made parallel to the walls of the container (see Figure 2.8a). Although it is a simple technique, it is frequently used in literature (Saraiva et al., 2015), (Harrath, 2021), (George & Robinson, 1980), (Zhang et al., 2012), (Duarte Alonso et al., 2011). Also, during our research on site at Schiphol, we recognized that the current baggage loading robot uses a horizontal layer building algorithm to work properly (Interview with van Bavel, 2022). In our opinion, these type of algorithms work well with non-rigid items since each layer forms a stable base for the next layer, ensuring maximum overlapping area with items underneath. Another placement heuristic is the special points rule from (Crainic et al., 2008), where the procedure is to scan the free space specified within a container by the shapes of the items that are already placed. When these corner points are scanned, a representative free space is constructed to pack items with simple online heuristics discussed in Section 2.1.3. A last placement heuristic is quite similar to the special point rule: it is called the skyline heuristic and is used by (Wei et al., 2009). The Skyline data structure only keeps a list of the ”skyline” or horizon edges that are created by the highest edges of rectangles that have previously been packed. This list increases linearly in length and is very simple to maintain. The item which is needed to be placed will be placed by default at the bottom left point from the skyline structure (Skyline Bottom Left). However, the position of the item can also for each point in the structure be evaluated for lost area. Then, where a minimal area is lost, the item will be placed. This algorithm is called the Skyline Best Fit algorithm (Jylänki, 2010). Another commonly used algorithm is the Maximal Rectangles algorithm (also known as Maxrects). The Maximal Rectangles algorithm stores a list of free rectangles that represents the free area of the bin after an item is packed.



(a) Wall Building Algorithm by (Saraiva et al., 2015)

(b) Horizontal Layer Building by (Elhedhli et al., 2017)

Figure 2.8: Placement Heuristic

2.2. Meta Heuristics

Metaheuristics are approaches that aim to provide an approximation of a solution to a MCLP in a reasonable time. In most cases, metaheuristics can be applied to other optimization problems since they are methods to search through a solutions space. Metaheuristics generate, find or select a heuristic and thus act on a higher level. In most cases, these types of algorithms work in combination with a placement heuristic. Thus, metaheuristics are approaches to guide the search for a near–optimal solution. The task of preventing these algorithms from becoming stuck in local minima is challenging. The local minima are points in the space of possible solutions that, in comparison to the other possible solutions in the immediate area, seem to be the best option. However, when seen in the context of the whole range of possible solutions, the answer is not ideal.

A classic meta heuristic is the tree search method. Simplified, with tree search, a (binary) *tree* is made for the solution space. Every branch in the tree has a *node* containing a possible solution to the problem as key. While the algorithm is running, every branch is explored consorting a set of logical rules or heuristics. According to (Cormen et al., 2010), each node’s key must be *better* than any solutions in subtrees on the left and *worse* than any keys in subtrees on the right for a tree to act as a search tree. The method is simple and intuitive, which contributes to its wide use in research and industry. (Sheng-

ming et al., 2020) applied this algorithm to baggage loading. The resulting packings are computed fast and have good stability characteristics.
Tabu search is another commonly used metaheuristic. It is a specific search method that allows the algorithm's search on local minima to continue, by acknowledging new directions that do not necessarily improve the solution (X. Zhao et al., 2016) (Pardalos et al., 2013). To further avoid local minima, it also can declare certain solutions as *tabu*, so the algorithm does not consider that solution again.
(Stepán, 2009) presented a metaheuristic called GRASP. GRASP stands for Greedy Randomized Adaptive Search Procedure. The GRASP is made up of iterations that are composed of repeated constructions of a greedy randomized solutions (Feo & Resende, 1995). Afterwards, local search is used to iteratively refine the solution. (Dechow & Douglas J, 2000) presented such a solution applied on the MCLP.
Genetic algorithms are popular algorithms that mimic the process of evolution (Gonçalves & Resende, 2012). Generally, a genetic algorithm (seen in Figure 2.9) starts with a set of random solutions to the problem (1 in Figure 2.9). After that, it will assign a fitness score to every solution. Only the most effective x solutions are chosen (2 in Figure 2.9) after being evaluated using the fitness score. From there, the solutions undergo two different operations: cross-over, and mutation. A cross-over operation is the action where two solutions are combined into one (3 in Figure 2.9). This refers to the creation of a child's genetics from the parent's genetics. The second action, called mutation, is an operation in which the solution is altered slightly (4 in Figure 2.9). This action is done to prevent the algorithm to end up in local minima. When a genetic algorithm is running, new *generations* are made by applying the cross-over and mutation actions on the best solutions (5 in Figure 2.9). In this way, it searches for a near-optimal solution.

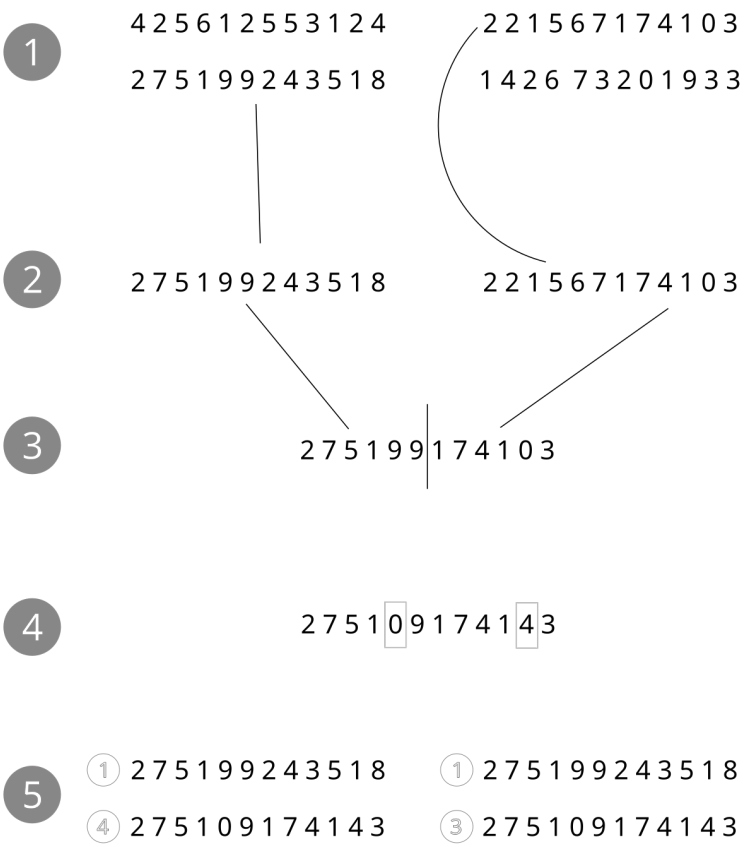


Figure 2.9: ULD filled with Parcel Items

2.3. Knowledge Gap

As illustrated in this literature review, a significant amount of research is done for the MCLP. Following the introduction of this chapter, the MCLP is a highly requested academic problem, in part because of its widespread application. Thus, the question of how items are placed into containers is already answered in literature with a various amount of different approaches. Moreover, the online MCLP algorithms provide less performance than the offline MCLP algorithms, where extending the lookahead of the online algorithms, should improve the performance significantly (H. Zhao et al., 2020). An interesting extension of the state-of-the-art packing algorithms is the optimization of the sequence in which the items arrive in. The sequence of items in literature is mostly captured in sub-problems (Oliveira et al., 2020) (Jiang & Yin, 2012). Also, according to experts, the sequence items arrive in, heavily determines the fill rate of certain online algorithms (Internal Communication, 2022). The purpose of this thesis is to quantify the impact of the arrival sequence on the fill rate of the containers.

3

Dataset and Robotcell

In this chapter, we will discuss the two scenarios where sequencing for the MCLP can take a role for Vanderlande Industries regarding increasing the fill rate of the final packing. At first, the main setup will be discussed in 3.1, where the CLP involves baggage loading. The second section, 3.2, will consist of the setup used for parcel loading. This is a case where the same process is executed and sequencing can also play a role.

3.1. Main Setup

3.1.1. Main Robotcell

As indicated in the introduction of this thesis, the main setup will be the bag loading setup. In this section, we will elaborate on the baggage loading robot cell. Logically, every passenger airplane that takes off has people on board with baggage. This baggage needs to move from the airport itself to the aircraft, either via cart or ULD. This work is one of the activities that has a high workload and a high percentage of all personnel got injured while doing this job. All baggage that is checked in will end up at the make-up station. At Schiphol, only baggage retrieved from the early bag store can be transported to the baggage loading robot.

Firstly, a container is needed to place in the robot cell. This container can be in the bag loading setup, a ULD or a cart (Figure 3.1a). More details about the container types will be explained in 3.1.2. Once the container is at the correct place, the gate can be closed and the packing process can start. To begin, the container has a specific destination and code. Bags matching this destination and code are selected and sent from the EBS to the robot. The order in which the items will be sent is dependent on the weight and size (Figure 3.1b) of the baggage. The largest and heaviest bags will be sent first and the small and light baggage will be sent last. This bag information is already available from the classification action much earlier in the baggage handling process. The bags will be transported to the robot cell (Figure 3.1c) and arrive at one of the two transport lines. From there, a Vertisorter will choose the heavier and larger of the two baggage pieces to transport to the actual robot. The bags will arrive high up in the robot cell above the container (Figure 3.1d). There, the bag is again scanned for dimension, weight, and orientation on the conveyor belt. Then, the robot arm will move up on down to let the camera attached to the robot arm create a three-dimensional image of the container. With the knowledge of the item and the knowledge of the environment (the container), an algorithm will calculate an optimal placement position for the item. This optimal placement position will be calculated using so-called *Ad Hoc Logic*. In practice, it is likely to be a simple horizontal layer building placement heuristic that is used as discussed in Section 2.1.3. Although the use of horizontal layer building algorithm is not confirmed, the approach is clearly visible when watching the robot execute the stacking process in real life (Interview with van Bavel, 2022) and on video footage (Internal Communication, 2022). Afterwards, the robot will calculate the motion the robot arm has to make for placing the item correctly. Then, the arm will move under the conveyor belt, the conveyor belt will rotate, and the bag will be moved on the robot arm's gripper (which is a small conveyor belt itself). Next, the robot arm will execute the

calculated plan and rotate the gripper's conveyor belt, placing the bag in the container (Figure 3.1e). Then, the arm will move to let the camera make a three-dimensional image of the stack to check if placement of the bag is achieved successfully. Now, the next bag can be transported, scanned and placed in the container. This process is looped for ULDs as long as there is room for the gripper to place the bags (the gripper takes 70 cm space). With carts, the robot can stack to full capacity. During this process, a baggage handler oversees the process. This is because the robot makes mistakes and makes inefficient. If that is the case, the operator will stop the robot and rearrange some bags to improve fill rate (Figure 3.1f). If the container is packed to its capacity, the operator will stop the robot and open the gate. Then, it will connect the ULD or container to a vehicle, which will transport it to its designated airplane. In practice, it does not matter in which cart or ULD the bag is placed, as long as the container or cart is transported to the correct airplane.

(a) Container is placed

(b) Items are scanned

(c) Transport to robot

(d) Items arrive on top

(e) Placement of items

(f) Reordering of items by operator

Figure 3.1: Baggage Loading Process

3.1.2. Main Items and Containers

In this main scenario, the *items* that are packed by the robot are baggage pieces. To specify further, the items are ST baggage pieces. ST refers to the category of bags, where each category is handled different at the airport.

- **ST**: Normal Conveyable baggage. Baggage that can fit in a box of 1000 $\times$ 750 $\times$ 650 mm.
- **SC**: Semi-Conveyable baggage. Baggage that can potentially become a problem or *irregular* baggage. This includes baggage pieces which become very sensitive to rolling when packed.
- **OG**: Out Of Gage baggage. This type is larger and mostly heavier than the regular ones (does not fit in box of size: 3000 $\times$ 2000 $\times$ 500 mm and weights between 0.5 and 80 kg).
- **OS**: Odd Size baggage. Baggage items that are either too small or too large to be handled by the system. These items will be delivered manually to their destinations. Examples are: musical instruments, art, walkers, extremely wide or extremely flat baggage.
- **SP**: Special baggage. Every luggage piece that does not meet the conditions of one of the above requirements will be classified as special baggage. Also, OG baggage falls in this category if the system can not handle the form of the piece. Examples are: living animals, chemical products, baggage supported on wheels, open filled bottles, super slippery items. This category of luggage will also be handled manually.

Baggage categories and dimensions

The picture below shows the dimensions of the specified baggage categories:

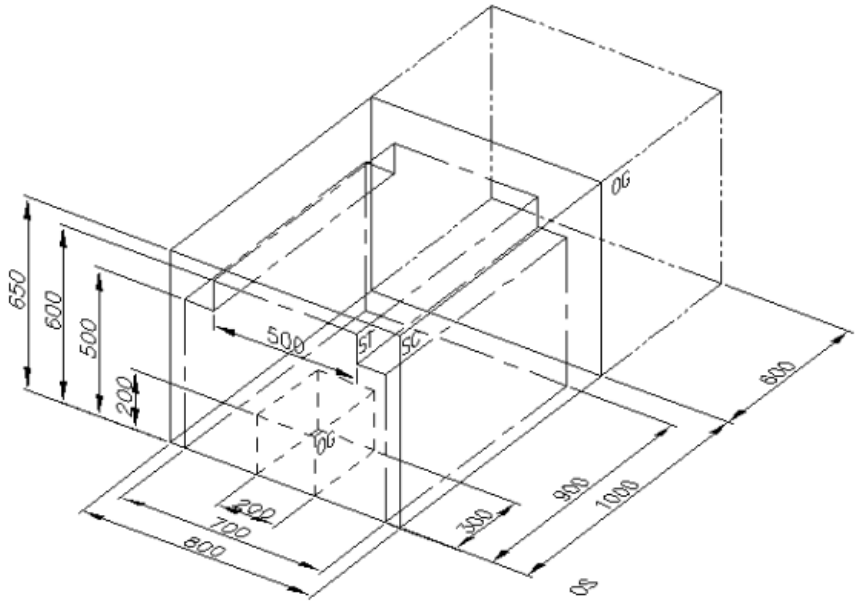


Figure 3.2: Bag Categories

This means that the size of baggage which the robot is stacking is limited to items that fit in a box of 1000 $\times$ 750 $\times$ 650 mm. A full overview of the categories alongside with the dimensions are visible in Figure 3.2.

Every piece of baggage is different, but they can be divided into certain categories provided by the IATA in Figure 3.3. Each item has characteristics that have influence on stacking performance. Here, the material is most important. Some baggage is made from Polycarbonate, which is referred to as *hard case* items. These items provide good stability for items stacked on top of. Other baggage is made of more soft material, Polyester. When placing bags on top of these kinds of baggage pieces, the soft case baggage will partially deform. This deformation will decrease stability characteristics of the stack, which will make it more likely for baggage to fall during the stacking process.

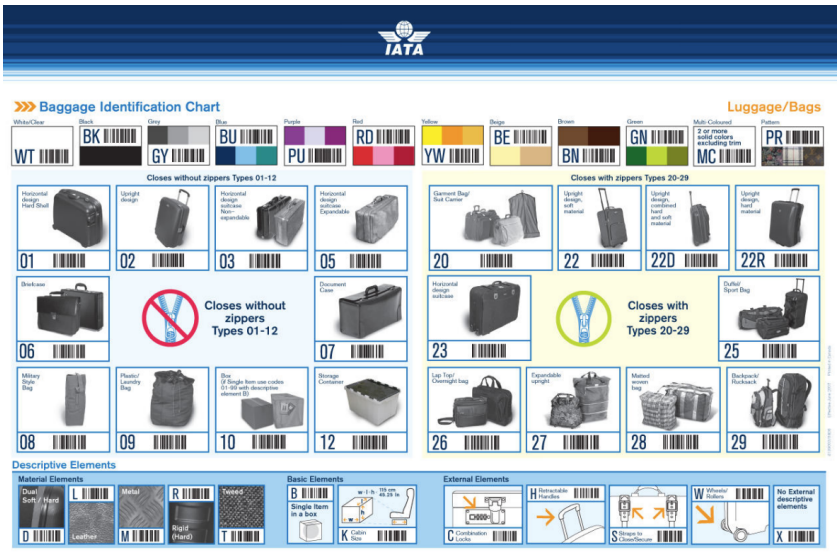


Figure 3.3: IATA Baggage Identification Chart by (IATA, 2016)

In practice for baggage loading, type 02 and type 01 from Figure 3.3 are the most common baggage types.

To limit computation and due to non-available detailed data for this thesis algorithm, some assumptions have to be made about the items. The first one is that the items are rigid cuboids. Here, the item's outer edges are taken, and a rectangular box is drawn fitting these outer edges (see Figure 3.4). Another simplification is the omission of item weight and material.

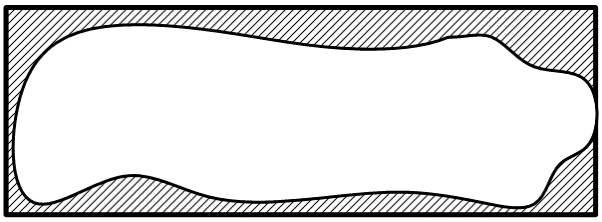
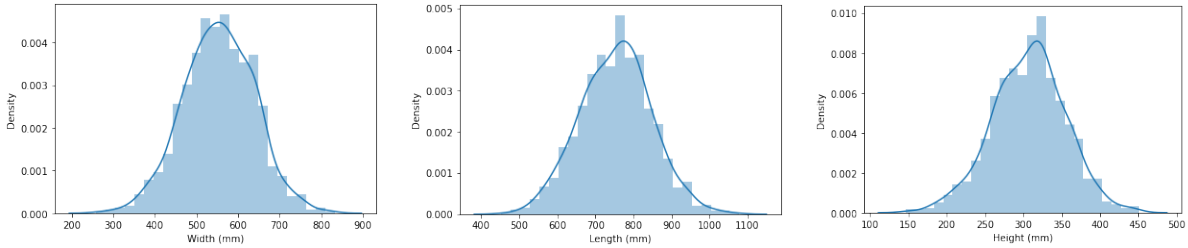


Figure 3.4: Outer Edges of Item

To translate this to the specifications of (Bortfeldt & Wäscher, 2013), the baggage loading case has strongly heterogeneous items, every bag is different in size. In the figure below (Figure 3.5), the distribution is shown of the different dimensions of the items. In the algorithm, samples will be taken from this normal distribution to simulate actual baggage items.



(a) Distribution Baggage Width **(b)** Distribution Baggage Length **(c)** Distribution Baggage Height

Figure 3.5: Distribution Dimensions Baggage

As mentioned earlier in this thesis, the baggage loading case has two different types of containers: ULDs and carts. ULDs are containers that will be placed entirely into the underbelly of wide body airplanes (see Figure 3.6a). When baggage has to be moved to narrow body airplanes, a cart is used to move the baggage to the airplane (see Figure 3.6b). Once arrived, baggage handlers will place the items into the underbelly of the airplane one by one.

(a) Unit Loading Device (ULD) **(b)** Cart

Figure 3.6: Various Containers for Baggage

The containers for each airplane are similar in shape. Table 3.1 below, shows the dimensions of the containers used in this thesis.

	Width [mm]	Height [mm]
Baggage Cart	2831	1370
Baggage ULD	1470	1400
Parcel ULD	3175	2235

Table 3.1: Container Sizes

The ULD used in this case will be the AKE ULD, where only the rectangular dimensions are taken.

Since all containers used for each airplane are similar, according to the terminology used by (Bortfeldt & Wäscher, 2013), the ”Single Bin-Size Bin Packing Problem” will be solved, however, in this thesis, we will refer to it as the Multi Container Loading Problem (MCLP).

In this thesis, the MCLP will be limited to a 2.5 dimensional problem. Since the main scenario involves a horizontal layer building robot, the problem can be solved in parts for every layer. Each layer solves the two-dimensional CLP. When all layers are solved, these layers can be stacked on top of each other. Thus, all the *slices* are stacked and thereby make a 2.5 dimensional problem. However, one could argue that the height of the items is not incorporated within this approach. This is true, however, in this thesis, we will assume that items will deform vertically when pressure is applied on top. Assuming this, when pressure is applied to the layer, the layer is *flattened* to a certain extent which makes it able to make form a new stable layer on top.

3.2. Alternative Setup

3.2.1. Alternative Robotcell

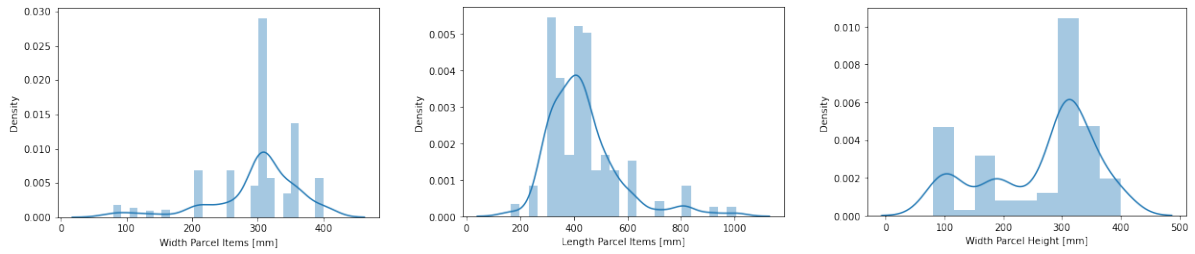
The alternative robot packs parcel instead of baggage. Although the item sizes, variations and containers are different, the process is similar. It must be noted, that this is the likely setup in which parcel loading will take place, since it is not an already existing product.

Firstly, a ULD is needed to place in the robot cell. Once the container is at the correct place, the gate can be closed, the packing process can start. To begin, the items that need to go to the destination where the container is going will be sent to the robot cell in non-specefied order. There the items are scanned for dimension, weight, and orientation on the conveyor belt. Then, the robot arm will scan the container in the same way as the baggage loading robot to create a three-dimensional image of the inside of the container. With the knowledge of the item and the knowledge of the environment (the container), an algorithm will calculate an optimal placement position for the item. Afterwards, the robot will calculate the motion the robot arm has to make for placing the item correctly. Then, the arm will move under the conveyor belt, the conveyor belt will rotate, and the parcel will be moved on the robot arm’s gripper. Next, the robot arm will execute the calculated plan and enable the gripper’s conveyor belt, placing the item in the container. Then, the arm will move to let the camera make a three-dimensional image of the stack to check if placement of the item is achieved successfully. Now, the next item can be transported, scanned and placed in the container. This process is looped for ULDs as long as there is room for the gripper to place the parcel (the gripper takes 70 cm space). During this process, it is likely that a handler oversees the process. If the container is packed to its capacity, the operator will stop the robot, and open the gate. Then, it will connect the ULD to a vehicle, which will transport it to its designated airplane.

Figure 3.7: ULD filled with Parcel Items

3.2.2. Alternative Items and Containers

The items in the parcel case are parcel pieces. These items are typically smaller than baggage pieces. In this alternative scenario, the rigid cuboid shape of the parcel is incorporated into the algorithm, which means that on shape, no simplifications are made. However, the weight of the items is not taken into account for stacking. Therefore, we assume that all items can be placed on top of each other. The items for this case are also *strongly heterogeneous* (Bortfeldt & Wäscher, 2013). In Figure 3.8 below, the distribution of the dimension for parcel items are displayed.



(a) Distribution Parcel Width **(b)** Distribution Parcel Length **(c)** Distribution Parcel Height

Figure 3.8: Distribution Dimensions Parcel

From the images, it is clear that there is less variation of the parcel items. This means that some items are the same size in at least one dimension.

The containers used in the parcel case are ULDs. The ULDs in which parcel is transported are typically larger than baggage ULDs, because the ULDs can be placed on the main deck of an airplane since the airplane is only used for cargo. Below, a figure with some different ULD containers are shown (Figure 3.9).

Figure 3.9: Different Types of Parcel ULDs

3.2.3. Generating Dataset

To create a set of items with representative dimensions, a function is created. This function makes a three-dimensional normal distribution of all dimensions. From this three-dimensional normal distribution, the function samples points, where each point in three-dimensional space refers to an item with the width, length, and height according to its position. This algorithm makes sure that items of dimensions that occur more times, also occur more often as sample. The function can create as many items (n_items) as needed.

4

Genetic Algorithm

This chapter will present the design (Section 4.1), the tuning (Section 4.2) and the results (Section 4.3) of the genetic algorithm used to sequence for the MCLP.

4.1. Design Genetic Algorithm

A genetic algorithm is an algorithm from the family of evolutionary algorithms. It is a technique for solving optimization problems that is based on natural selection, the mechanism that drives biological evolution. The natural selection aspect is the process that determines how the solutions change over time. At each stage of the process, a traditional genetic algorithm chooses members (or solutions) of the existing population to serve as parents and then uses those parents to generate children for the subsequent generation. In Figure 4.1 below, a schematic overview is given.

This method is used for a number of applications such as data mining (Fan & Luo, 2013), image processing (Verma, 2015) and the shortest path problem (Selvanathan & Tee, 2003). According to (Lambora et al., 2019), genetic algorithms come with a great set of advantages. To begin, there is no need for a model or information that can be replicated. In addition, a genetic algorithm seeks the optimal solution and improves over time. Lastly, GAs are effective for huge search spaces. GAs are employed for the single and MCLP in, for example, (Wu et al., 2010), where a genetic algorithm is used altering the relative placements of objects inside a Mixed Integer Program (MIP). This is possible thanks in part to the benefits listed above. Also, (Gonçalves & Resende, 2012), presents a successful GA for the MCLP where they developed a method that combines a new placement operation with a multi-population genetic algorithm based on randomly generated keys. There is a major difference between the genetic algorithms that are discussed in these works and the genetic algorithm that is developed by us for this thesis. The majority of the previously published research make use of a genetic algorithm to establish optimized placement of the items. In the design that we are going to show in this thesis, the genetic algorithm will be used to optimise the sequence in which the items are given to an MCLP algorithm.

In the following paragraph, we will describe how the genetic algorithm is designed for sequencing in order to solve the multi container loading problem. As described in Section 2.2, a genetic algorithm consists of multiple steps: initialisation, evaluation, selection, cross-over, mutation, and termination. In the next subsections, which have been given titles that are appropriately descriptive, a more in-depth discussion of the phases will be provided, as well as an explanation of the many design decisions.

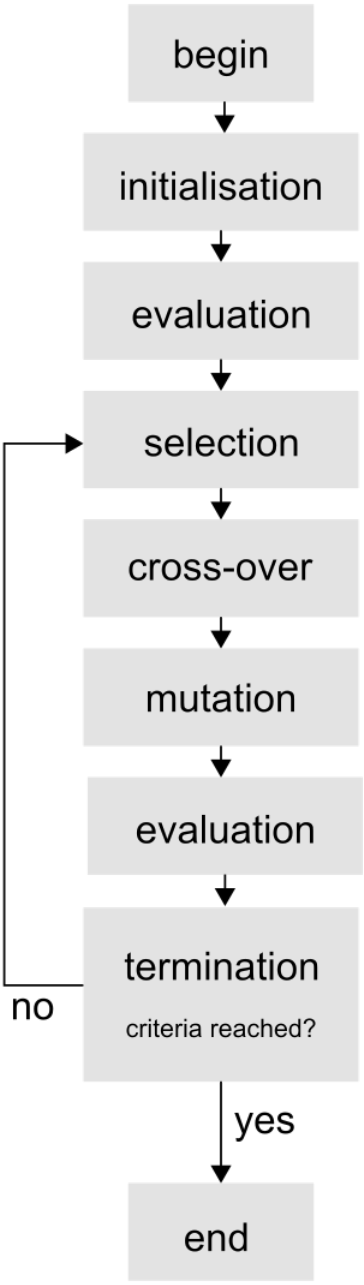
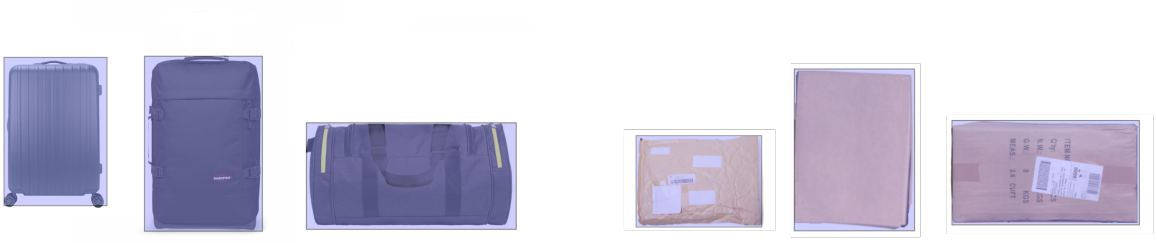


Figure 4.1: Schematic Structure Genetic Algorithm

4.1.1. Initialisation Genetic Algorithm

During the initialization phase, the initial parameters such as container dimensions and items sizes are given their default values. It is necessary to determine the setup before attempting to collect the parameters. Two different scenarios are dealt with, both of which are outlined in Chapter 3. The setup of the first scenario is the cart and ULD loading robot for airports. The second setup is the parcel ULD loading case. When a setup is chosen, the dimensions are set for the items and the containers.

With the container size determined, the items can be generated. In the figure below (Figure 3.4), a subset of actual parcel and baggage items is shown. Next to this example, the mean and standard deviation is displayed, calculated from the whole dataset. A histogram of the whole set can be found in figure Figure 3.5 and Figure 3.8.



(a) Baggage Items **(b)** Parcel Items

Figure 4.2: Items with rectangular boxes around outer edges

First the items are generated according to the method described in Section 3.2.3. At this point, there are sufficient amounts of items. The next step is to form solutions from these items. For that, the items are randomly shuffled $m\_solutions$ times and added to a list. This list forms the first generation: generation 0. Below, such two different solutions are shown in Figure 4.3. The next step is to evaluate the solutions.

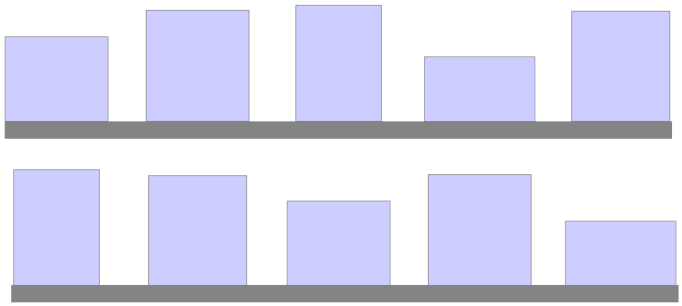


Figure 4.3: Example Solutions: Sequences of Items

4.1.2. Evaluation Genetic Algorithm

A key part of this genetic algorithm is the evaluation process. This is done with a so-called *fitness function*. With this genetic algorithm, the fitness of a solution is determined by the fill rate of the container after the algorithm stacked the items in the order from the solution. The fill rate of the packing is the sum of the area covered by each item, divided by the total area (see Equation 4.1). For example, when a set of items in a specific order stacked in a container cover half of the area of the container, the fitness of this solution will be 0.5. Following (Internal Communication, 2022) and research by (Bortfeldt & Wäscher, 2013), fill rate is the most important metric for the MCLP and, in this case, also for sequencing for the MLCP. The algorithm presented in this thesis will always pack more layers. To incorporate this in the fill rate score, the parameter l is introduced. Also, to incorporate the items from all layers, an extra sum is involved. In Equation 4.2, the mathematical formulation is displayed.

$$f = \frac{\sum_i x_i * y_i}{c_x * c_y} \tag{4.1}$$

$$f = \frac{\sum_l \sum_i x_i * y_i}{c_x * c_y * l} \tag{4.2}$$

As a practical matter, the fill rate is very dependent on the quantity of items that can be forced into a packing, given that the items are generally of comparable dimensions.

The genetic algorithm has a function to evaluate the fitness given a heuristic packing algorithm. In the end of the stage, all the solutions have a fitness associated to a specific sequence.

4.1.3. Selection Genetic Algorithm

When the fitness scores are known, a selection of the solutions must be made. Herein lies the relevance of the parallel to natural selection. By disrupting a bad solution, the principle of *survival of the fittest* is

applied to the selection. Here, the fittest refers directly to the solutions with the highest fitness score. In the genetic algorithm presented in this thesis, the best $p\_selected$ solutions are selected for the next phase of this algorithm: crossover and mutation.

4.1.4. Cross-Over Genetic Algorithm

In most genetic algorithms, a cross-over action will be made during execution of this algorithm. This action refers to combining two solution into one. Here, two solutions can be seen as two chromosomes. The first chromosome will be split at a random place in the solution, while another chromosome (or solution) is split at the same place. Then, in order to create a new solution, the first portion of the first solution and the second half of the second solution are joined together. This process refers to the creation of humans genes when two parents produce a child.

In our algorithm, this action leads most of the time to infeasible solutions. This is the case since the solution is a sequence of items. When cross-over would be applied and for the first solution, one item occurs early, while the same item occurs later in the other solution. Combining these solutions into one will result in the specific item occurring twice in the children solution, which is not possible. The figure below (Figure 4.4) displays this problem. To be clear, this action is not used in the genetic algorithm designed for this thesis.

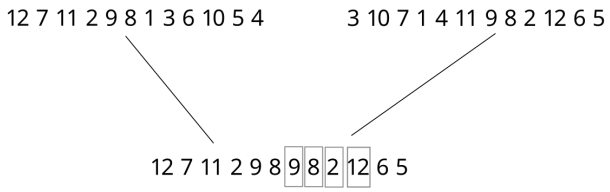


Figure 4.4: Unfeasible Combination, the highlighted number (or items) occur twice in the children solution

4.1.5. Mutation Genetic Algorithm

Mutation is an action where a solution is slightly altered. This process has also similarities to the ecological process of evolution theory, where the DNA of children, compared to the parents' DNA, is slightly altered.

The genetic algorithm presented in this thesis applied this type of action in each generation. An alternation in the solution translates to two random items in the sequence being swapped. The items selected for this action can be actually the same item, in that case, the mutation action does not alter the solution.

4.1.6. Termination Genetic Algorithm

In this stage of the generation, the new solutions do not have a fitness score, and therefore have to be evaluated. When this is done, one generation of the genetic algorithm is finished. The next stage is to check if the best solution from the solutions reaches the criterion. With the algorithm presented in this thesis, the criterion is that $f = 1.0$. This means that the algorithm can stop when the solution produced will result in a packing being filled 100%. However, this is almost impossible due to the nature of the item and container dimensions. Consequently, another criterion is also evaluated: $n\_generation$. The algorithm can run as long as necessary, but since the optimal score is likely never to be reached, a max of generations must be given to stop the algorithm. After each generation, the statement $n\_generation < max\_generations$ is checked to be *True* or *False*. When the equation returns *True*, the loop is continued, and the next generation can be created by selecting the best solutions. When the amount of generations reached the maximum, the algorithm is terminated.

4.1.7. End

When the algorithm is terminated, it will return the best solution found during the whole search over all generations. Also, it will return the fitness score or fill rate.

4.2. Tuning Genetic Algorithm

For the genetic algorithm presented in this thesis, three parameters can be tuned: $m\_solutions$, $max\_generations$ and $p\_selected$. Here $m\_solutions$ refers to the initial set of solutions that is created. $max\_generations$ refers to the max amount of generation that can run before termination of the algorithm. $p\_selected$ is the parameter that sets the amount of solutions selected after evaluation. In Table 4.2, all parameters values are shown.

Parameter	Values
$m\_solutions$	[5, 10, 20, 50, 80, 100]
$max\_generations$	[5, 10, 25, 35, 50, 80, 100]
$p\_selected$	[5, 10, 20, 50]

Table 4.1: Parameters Genetic Algorithm

When comparing performance, it is shown that for each parameter, a greater value corresponds to a longer calculation time. This is due to the fact that each parameter affects the number of solutions to be evaluated. The algorithm takes the most time computing the fitness score of each solution, so high $m\_solutions$, $max\_generations$ or $p\_selected$ result in a longer calculation time.

Because the initial solution population size, also known as $m\_solutions$, is the factor that has the greatest impact on performance in terms of fill rate, a large number has been chosen for this parameter. The $max\_generations$ have more impact on computational time than performance, therefore a moderate amount of maximum generations is chosen for the final algorithm. For the $p\_selected$, a value of 20 is selected since this value limits the computational time, while guaranteeing good performance regarding fill rate.

Parameter	Chosen Values
$m\_solutions$	100
$max\_generations$	35
$p\_selected$	20

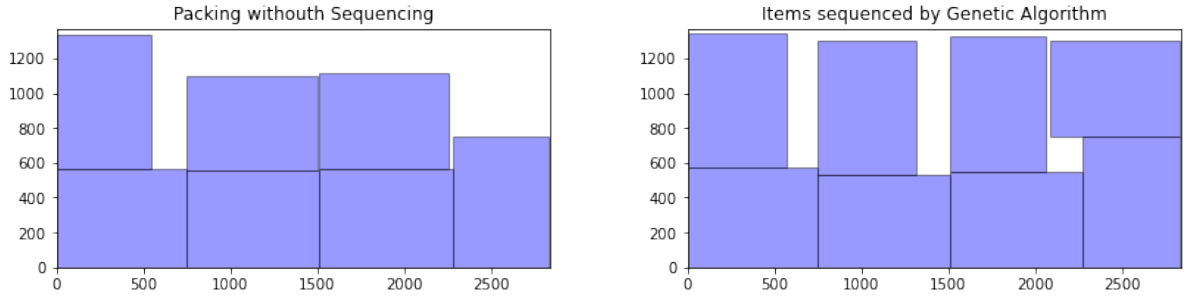
Table 4.2: Chosen Parameters Genetic Algorithm

4.3. Results Genetic Algorithm

When the genetic algorithm is executed, it will normally take roughly two seconds to run through 35 generations, and the algorithm will provide an update after each generation that it goes through. In the next two sections, we will elaborate in detail on the results of the main, and the alternative setup respectively

4.3.1. Results Genetic Algorithm on Main Setup

The first results on single runs look promising. When applying the algorithm to baggage items, the differences between sequenced item sets (Figure 4.5a) and non-sequenced item sets (Figure 4.5a) are directly visible. The difference is also expressed at their respective fill rate, where the fill rate of the sequenced item set packing is 0.75, and the fill rate of the container packed with the sequenced set is 0.87.



(a) Non-sequenced items

(b) Sequenced Items with GA

Figure 4.5: Main Setup, Skyline Wastemap Heuristic at Cart Top View

In the remainder of this section, we will look at results of the sequencing genetic algorithm, varying the heuristic. Also, the algorithm will be run several times to compare the distribution of the different results. In table Table 4.3 below, the different approaches will be shown.

Modes	Heuristic Algorithm	Loading Unit
Sequenced by GA	Skyline Wastemap Reduction	ULD
Random Sequence	Guillotine Best Area Fit Longer Axis Split	Cart
	MaxRects	

Table 4.3: Sequencing Approaches

The first modes compared to each other are already shown in Figure 4.5. Furthermore, in the plot below (Figure 6.6), the normal distribution can of the performance of a sequenced set regarding the top view of a cart loaded with a Skyline Wastemap Reduction heuristic.

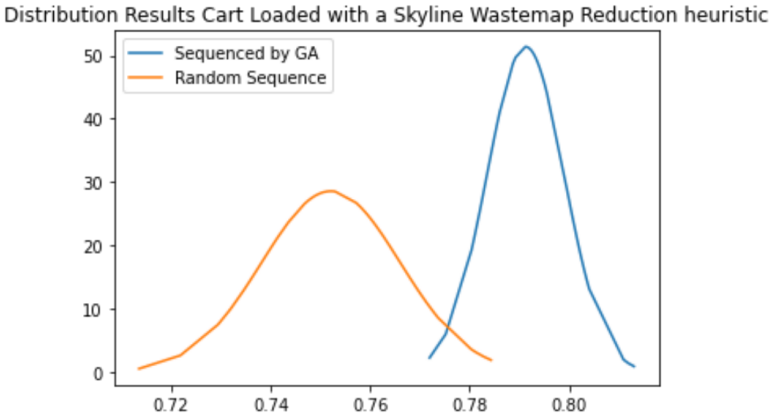


Figure 4.6: Distribution Fill Rate of Packings of Non-Sequenced Items and Items Sequenced by GA (Cart, Skyline Wastemap Reduction Heuristic)

The graph reveals two significant points. The first point is the significant difference in mean fill rate. The randomly sequenced items results point to a mean of 0.75, which means that a large portion of the times when items are generated, the fill rate after packing containers is around 0.75. This is, a respectable score; nevertheless, as the graph makes clear, the standard deviation is rather large (0.01395). The sequence produced by the GA generally leads to significantly higher fill rates. The mean of these result is 0.79, which is almost 5% up from the random sequence performance. Also, the standard deviation in fill rate is remarkably lower than the randomly sequenced item set results, 0.00777 precisely. Overall, it is possible to draw the conclusion that the outcomes of the sequenced item set lead to fill rates that are both greater and more consistent. Because of this consistency, the

algorithm produces results that are more dependable, which is beneficial when it is employed in the real world.

As can be seen in Figure 4.5, the genetic algorithm tends to sequence the items in a way that eight items fit. For that to happen, a *rectangle* of items needs to exactly fit in the right corner, this is not always the case as displayed in Figure 4.7. If it does not exactly fit, only seven items can be packed in the container.

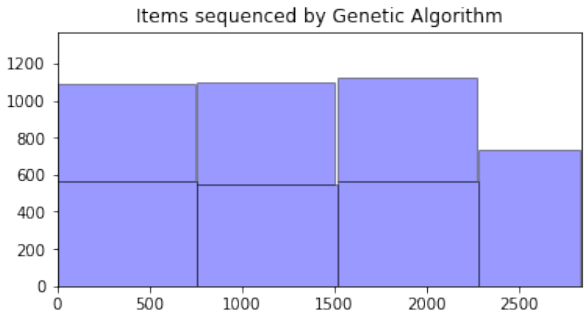


Figure 4.7: Example packing sequenced items by GA, Packing seven items

Moreover, we will look at the performance results in the ULD loading case for bag loading.

Figure 4.8: Example packing ULD case with Genetic Algorithm and with Non-Sequenced Items

Figure 4.9: Main setup, Skyline Wastemap Heuristic at ULD

From Figure 4.9, the same patterns are visible. In the container filled with the sequenced items, the rectangular pattern is shown more often than in the container packed with the non-sequenced items.

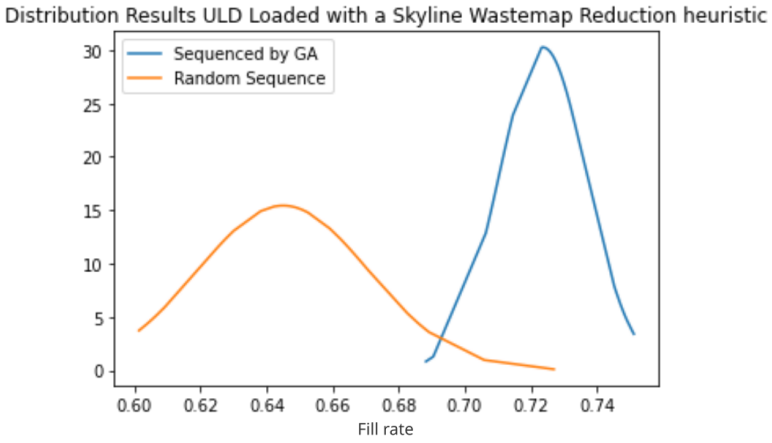


Figure 4.10: Distribution Fill Rate of Packings of Non-Sequenced Items and Items Sequenced by GA (ULD, Skyline Wastemap Reduction Heuristic)

From Figure 4.10, the exact same shapes of the normal distribution can be seen as in the cart loading case. The biggest difference in performance is the mean of the results of the random sequenced items and the sequenced items by the GA, which are respectively 0.64 and 0.72. This refers to the case where there fit three or four items in the layer. With three layers, the container is filled with a score of 0.64 and with four items, the container is filled with a score of 0.74. Again, sequencing the items in a particular proves to be beneficial.

When assuming sequencing is working, the genetic algorithm can work with a various set of heuristics to sequence for. In Figure 4.11 below, the fill rate results are shown for different heuristics.

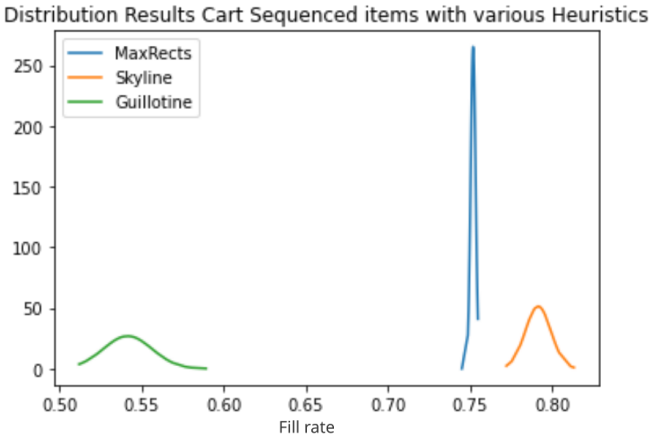
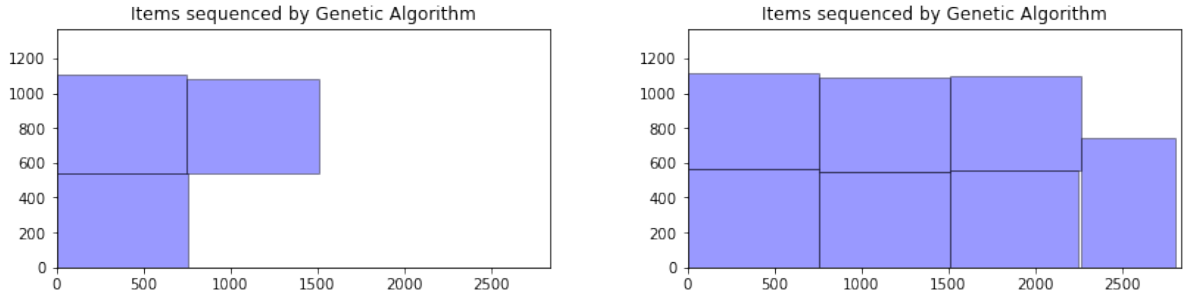


Figure 4.11: Distribution Fill Rate of Packings of Items Sequenced by GA (Cart, Various Heuristics)

Looking at the presented graph, major differences are evident. Firstly, the Guillotine heuristic does not work as an online MCLP algorithm. Moreover, the distribution of the results for that heuristic is very wide, with a standard deviation of 0.0148. The best performer of the heuristic approaches is the Skyline Wastemap Heuristic, showing the results already discussed. The Maxrects heuristic is the option with results in between the Skyline and Guillotine heuristics. The standard deviation of the results are majorly tight (0.0015), almost ten times as less distributed as the results of the guillotine heuristic. The following examples of packings from Figure 4.12 clearly show the variations in fill rates.



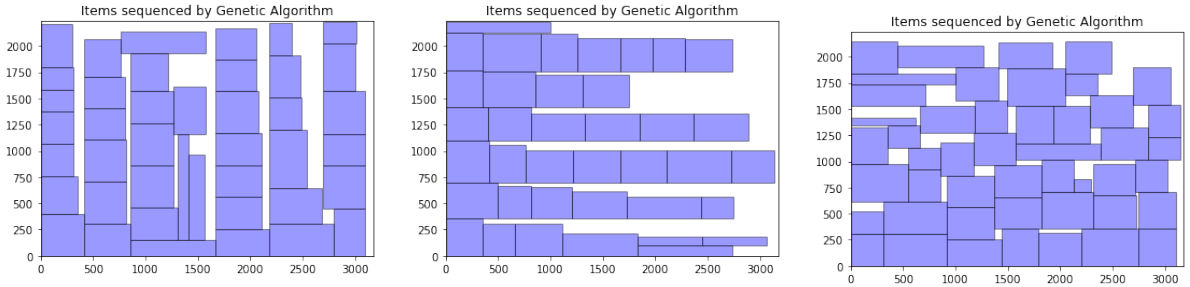
(a) Example Packing, packed with Guillotine Heuristic

(b) Example Packing, packed with Maxrects Heuristic

Figure 4.12: Example Packing Packed with Different Heuristics

4.3.2. Results of Genetic Algorithm on Alternative Setup

In this setup, the appropriate heuristic needs to be determined since the algorithm for alternative setup is unknown. For that we will test the three known heuristics: Skyline Wastemap, Guillotine Best Area Fit Longer Axis Split and Maxrects. First, the three example packing approaches will be shown via example packings in Figure 4.13.



(a) Example Packing Alternative Setup with Skyline Wastemap Heuristic

(b) Example Packing Guillotine Best Area Fit Longer Axis Split Heuristic

(c) Example Packing Maxrects Heuristic

Figure 4.13: Different Example Packing Heuristics for Alternative Setup

It is evident that the first two heuristics reflect the behaviour of layer building heuristics. Where the Skyline heuristic builds vertical layers and the Guillotine heuristic builds the same type of layers but rotated 90 degrees. The Maxrects heuristic uses a different approach, which is explained in Section 2.1.3. Regarding only these example packings, the Maxrects algorithm seems to gain the highest fill rate. To check this fact, the normal distribution of the three heuristics will be plotted against each other in Figure 4.14.

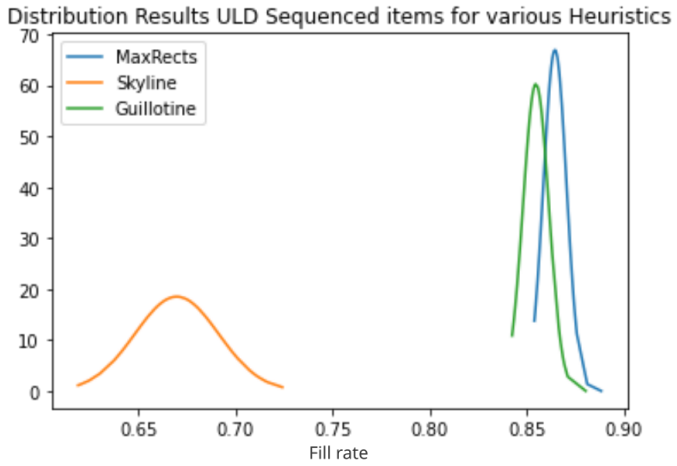


Figure 4.14: Distribution Fill Rate of Packings of Items Sequenced by GA (ULD, Various Heuristics)

The graph confirms that Maxrects reaches the best performance (0.864 mean fill rate score). However, the Guillotine algorithm provides similar results (with 0.854 mean fill rate score). The results also look identical regarding the distribution of the results, both being very consistent, with a standard deviation of 0.00662 for the Guillotine algorithm and 0.00596 for the Maxrects algorithm. Overall, the performance of the Guillotine algorithm is clearly lacking and provides very poor results with a mean fill rate score of 0.669 with a wide standard deviation of 0.02153. In other tests, Maxrects will be the algorithm used.

Next, the sequenced items by genetic algorithm will be tested against the non-sequenced items to check if sequencing has impact on performance of the algorithm.

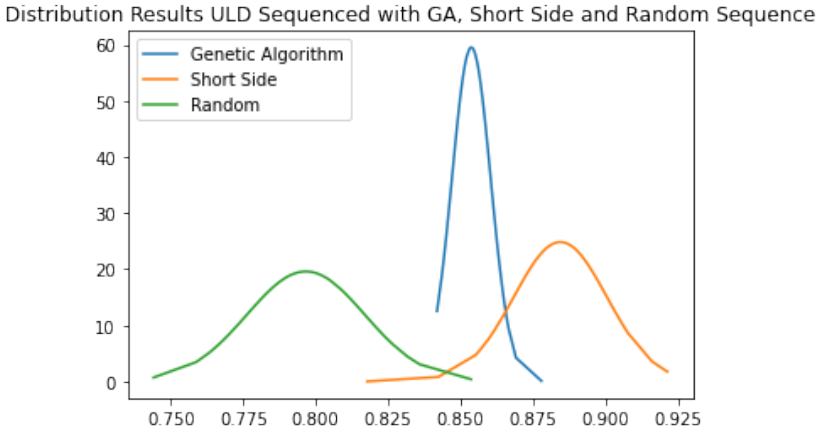


Figure 4.15: Distribution Fill Rate of Packings of Non-Sequenced Items and Items Sequenced by GA (ULD, MaxRects Heuristic)

Figure 4.15 shows that sequencing the items before packing has worked for the scenario with the alternative setup. However, the performance advantages are not nearly as significant as they are in the main setup. The reason for this is the size of the items relative to the container. In the main setup, the items were relatively big. Which resulted in the phenomenon that different sets of items fit differently. Also, if there is an extra item that fits in the main setup, the fill rate significantly improves since this placed item takes a lot of space. In the alternative scenario, the items are small and when more items are placed, it does not mean the fill rate significantly improves.

Another set of very simple sequencing methods are sorting approaches based on characteristics of the items. In Table 4.4, a list of all sorting methods is shown.

Sorting Methods
Area
Difference Width and Height
Short Side
Long Side
Perimeter
Ratio Width / Height

Table 4.4: Simple Sorting Methods

Here, *Area* refers to the items sequenced from largest to smallest area. *Difference* refers to the items sequenced from the largest difference between the length of the sides to the smallest. *Short Side* orders the items based on the smaller side descending and *Long Side* does the same, but with the longer side of the item. *Perimeter* is the length of all sides combined, again, ordered from largest to smallest. The next graph will present an overview of the results of all sorting methods to compare the methods and choose the best performing one.

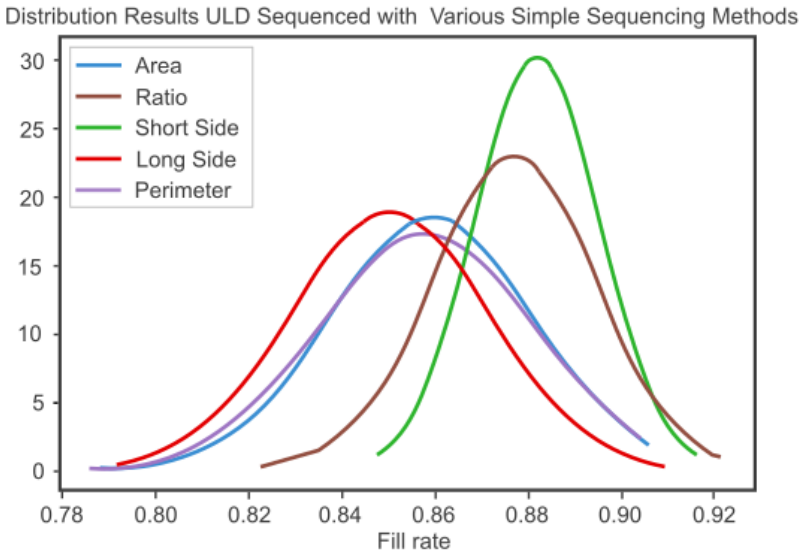
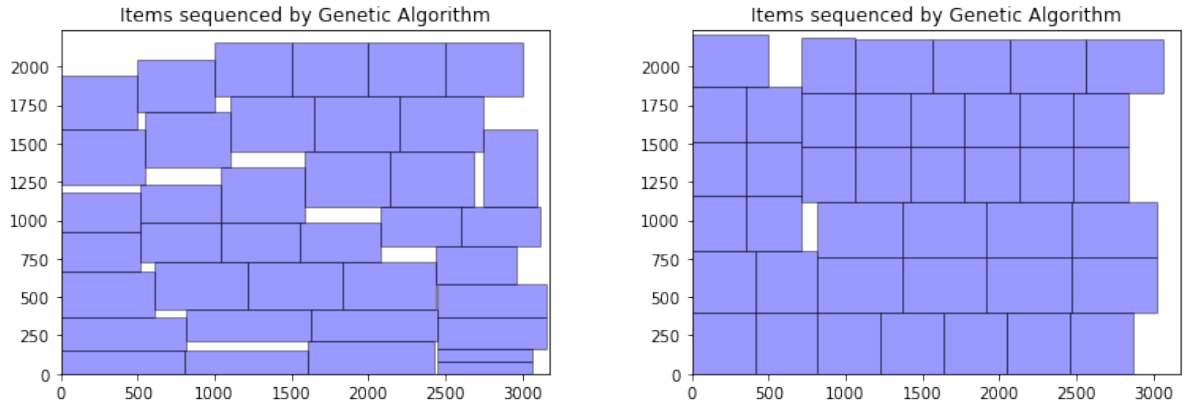


Figure 4.16: Distribution Fill Rate of Packings of Items Sequenced by Simple Sequencing Methods (ULD, Maxrects Heuristic)

In Figure 4.16 above, the results can be seen of all sorting methods presented in Table 4.4. Regarding performance, the *Difference* sorting method is providing the worst results with a mean of 0.85. However, the results of the items ordered in the *Long Side* approach are not much better. As expected, the sorting methods *Area* and *Perimeter* provide very similar performance, since most items with a high perimeter also require relatively more space, therefore the sequences are very similar and the according fill rates are also similar. Sorting the items with descending *Short Side* results in the best packings with a mean fill rate score of 0.88. Some of the stacking behaviour is visible in Figure 4.17, where the *Short Side* sorted items provide much tighter packings than the *Difference* sorted item packings.



(a) Example Packing, Items Sorted by Difference between sides, packed with Maxrects Heuristic

(b) Example Packing, Items Sorted by Short Side, packed with Maxrects Heuristic

Figure 4.17: Example Packings Alternative Scenario

Lastly, to test if simple sorting methods can outperform our GA, we compare the results of the packings of the items sorted randomly, the packing by the items sorted by *Short Side* and the packings of the items sorted by the GA. The results can be seen in Figure 4.18.

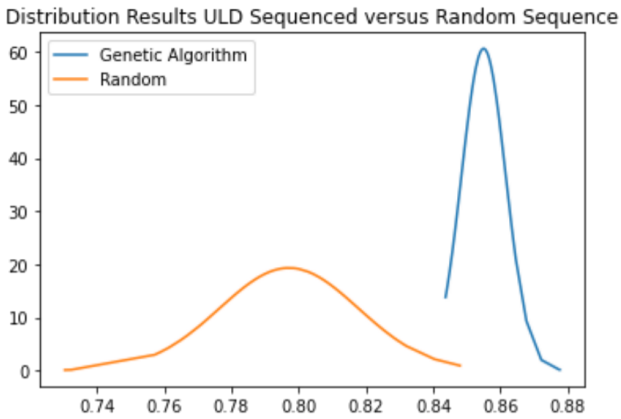


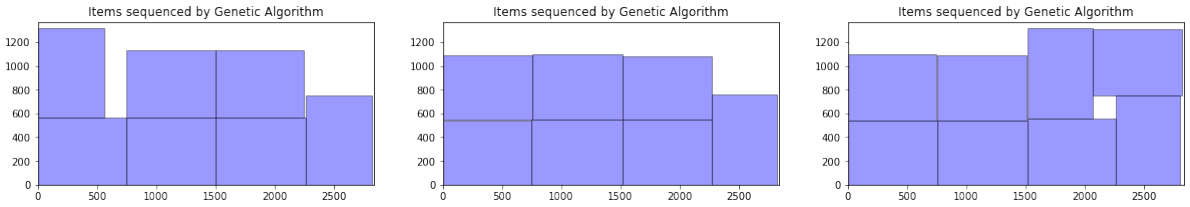
Figure 4.18: Distribution Fill Rate of Packings of Non-Sequenced Items and Items Sequenced by GA (ULD, Skyline Wastemap Reduction Heuristic)

From Figure 4.18, two lessons are learned. The first is that sequencing does improve performance by about 5% till 10%. The sequencing step leading to the highest results is the simple sorting method: sort by short side. Sorting with the genetic algorithm presented in this thesis will also improve fill rate by 5 percent to a mean of 0.85. The second lesson is the consistency regarding the performance of the sequencing methods. Sequencing the items by short side improves the performance the most but is less consistent than the GA, with standard deviations of 0.01606 and 0.00670 respectively. The best sequencing method is thus dependent on the wish of the performance. If the wish is to maximise predictability, the GA is the algorithm to choose. On the other hand, if mean performance is valued the highest, the items can be sorted according to their short side in a descending manner.

5

Heuristic Algorithm

Some interesting patterns become clear when approaching the MCLP with a genetic algorithm in the main scenario. To begin, it is clear to see the difference between high-percentage fill rate packing and low-percentage fill rate packings. Also, most packings are similar to one of the next three packings (Figure 5.1).



(a) Example Packing, with Sequenced Items by Genetic Algorithm

(b) Example Packing, with Sequenced Items by Genetic Algorithm

(c) Example Packing, with Sequenced Items by Genetic Algorithm

Figure 5.1: Example Packings, with Sequenced Items by Genetic Algorithm

In Figure 5.1 we see that packing c works well, being able to place eight items. However, packing a and b are worse performing packings and also occur often. This pattern of packing is fairly similar since the objects are relatively comparable in size. In addition to this, a crucial factor is the comparatively big size of the objects in comparison to the size of the container. Because the quantity of objects put in the packings has such a significant influence on the fill rate, the last factor also has the effect of making the performance similar to a discrete value. In this chapter, we will present a heuristic algorithm that will strive to construct consistent high fill rate packings. In other words, the algorithm will aim to maximise efficiency. The focus of this algorithm will be on the main cart loading scenario, since that is the focus of this study, however, we will elaborate on the design for the alternative scenario in Section 5.2.

5.1. Design Heuristic Algorithm

In the succeeding packing from Figure 5.1, one of the layers consists of eight items. To begin, we shall determine whether or not it is possible to coerce an algorithm into always packing in such a subsequent packing structure. However, this did not work out as planned, since the rectangle structure from Figure 5.1c only works for certain item sizes. To make the algorithm useful, different items sizes must be fitted in a layer.

The next step is to think creatively about packaging that can handle eight items on a cart. Considering the average dimensions of the objects and the size of the container, there seems to be space for precisely five upright items at the bottom of the container. When items are placed this way, there is 645 mm room above the first row of items on the bottom. This is where items can fit with length down. The structure can be seen in Figure 5.2 below.

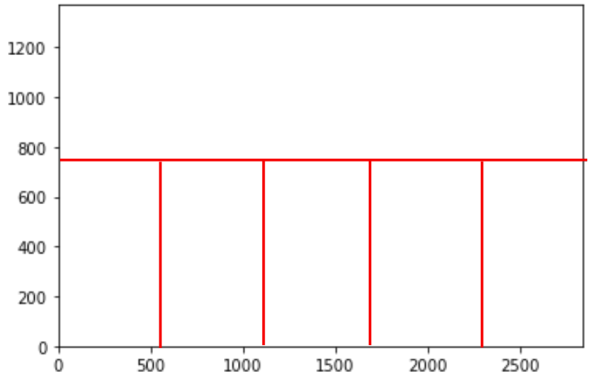


Figure 5.2: Packing plan

The next stage was to determine whether or not such packaging consistently fit in the container. Below some examples that show the packings fit (see Figure 5.3).

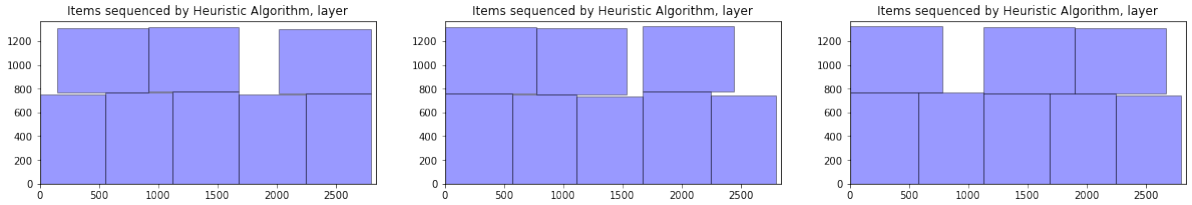


Figure 5.3: Example Packing in Preferred Packing Structure

At this point, a heuristic can be made to sequence and rotate the items correctly for the Skyline Wastemap Reduction Heuristic algorithm to pack the correct structure. The steps done by this algorithm will be explained in the next subsections.

5.1.1. Initialisation Heuristic Algorithm

First, the items must be created with a similar algorithm used in the genetic algorithm from Section 4.1.1. Also, within this step, the container dimensions are set. From there, two types of subsolutions are created. As indicated in the section above, the first row of items needed to be packed is the five items placed on their short side. So for this step, solution are created of random sets of five items, rotated 90 degrees. Meanwhile, there are also three items selected from the item list which will not be rotated. These solutions will be evaluated just like a genetic algorithm.

5.1.2. Evaluation Heuristic Algorithm

The evaluation step is similar to the genetic algorithm. However, the score does not consist of the fill rate of the packing. The score for the first row is majorly determined by the container width minus the sum of all items in the x direction. The width of the red part in Figure 5.4 indicates the lost space. However, the inverse is taken, since the highest score will be considered *good*.

$$e_{solution} = 1 - \frac{w_{container} - \sum_{n} w_n}{w_{container}} \tag{5.1}$$

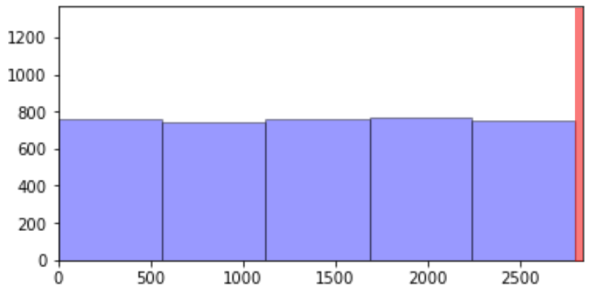


Figure 5.4: Score Calculation of Down Layer, the width of the container minus the width of the sum of the items, indicated in red

For the second row, the longest three items will be selected to be placed on top. Therefore, a similar evaluation score is made but with only three items in the length direction instead (see Figure 5.5).

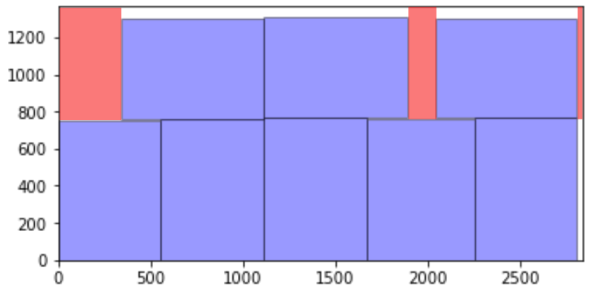


Figure 5.5: Score Calculation of Upper Layer, the width of the container minus the width of the sum of the items, indicated in red

5.1.3. Selection Heuristic Algorithm

In this step, solutions with the highest score will be selected. The first five items that are rotated will in that shape be removed from the items list. The three items selected for the upper row are also removed from the item list.

5.1.4. Sequencing Heuristic Algorithm

The selected items are then given one by one to a heuristic Skyline Wastemap Reduction algorithm, which packs the items according to the approach mentioned at the beginning of this section. It is important to point out that the packing algorithm cannot rotate the items to its preference. This is because the rotation of the pieces has been predetermined by the sequencing algorithm. The sequencing algorithm described in pseudocode is presented in Algorithm 1.

5.2. Design Heuristic Algorithm for Alternative Scenario

The heuristic algorithm is completely tailored for the main baggage loading setup, since this scenario is the main use of this thesis. However, the algorithm can be slightly modified to sequence for the alternative scenario.

In this case, the mean of width of the items is 285 mm, which means that probably at least nine items fit in one row. The other difference is that there are five rows of items made instead of of five or three.

5.3. Results Heuristic Algorithm

When the heuristic algorithm runs, it will typically take less than one second to run and come up with a feasible solution. The algorithm does not work in generations as the GA did, so the first solution is the final solution

In the next two sections, we will elaborate in detail on the results of the main, and the alternative setup respectively.

Algorithm 1 Heuristic Sequencing Algorithm

while $n \leq n_{layers}$ **do**
  while $m \leq m_{runs}$ **do**
    Sample 5 items $(i_n), n = 1, .., 5$ ▷ For the down layer
    $i_n(width) = i_n(height), i_n(height) = i_n(width)$
    $e_m = 1 - \frac{w_{container} - \sum_n w_n}{w_{container}}$
    $Score = [i_n, e_m]$
  $i_n \leftarrow \max Score$
  Output: i_n^{down}
  while $m \leq m_{runs}$ **do**
    Sample 3 items $(i_n), n = 1, .., 5$ ▷ For the upper layer
    $e_m = 1 - \frac{w_{container} - \sum_n w_n}{w_{container}}$
    $Score = [i_n, e_m]$
  $i_n \leftarrow \max Score$
  Output: i_n^{up}

5.3.1. Results Heuristic Algorithm Main Setup

It was anticipated that the results of the heuristic algorithm would provide highly consistent packings regarding the fill rate. This was due to the fact that the same packing structure could be packed in each container. Below in Figure 5.6 some examples show that the identical packing structure works for different items.

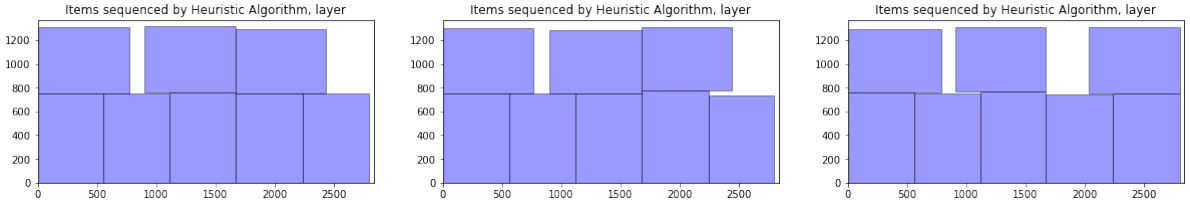
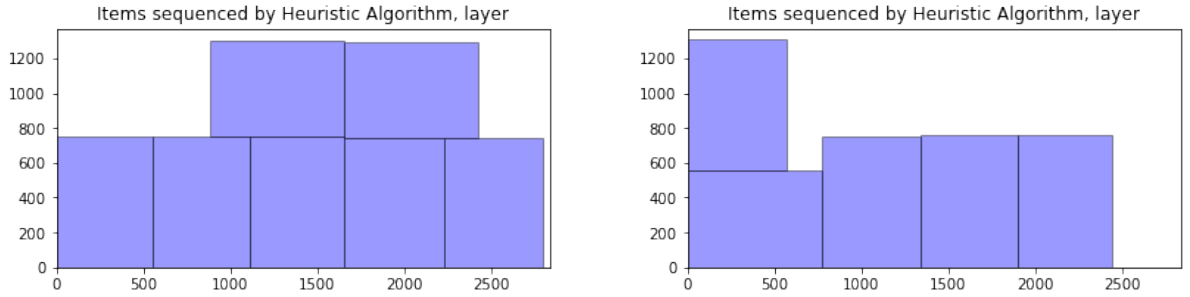


Figure 5.6: Example Packing in Preferred Packing Structure

The expectation is that this packing structure can always be made. However, when a container is not packed with the eight items, problems accumulate to other layers. A packing error can lead to one of the items not being packed according to plan. For instance, when only seven items are packed and one of the top items is missing, as in Figure 5.7. This results in the fact that the non-rotated item selected for the top row of the packing will be packed first at the next layer. This means that the plan does not match with the packing, since the plan is to begin the packing of the layer with five rotated items. Instead, the first item is now a non-rotated item which means that the packing structure cannot be maintained in this layer, leading to less than eight items being packed. The error accumulates, because now there will be more non-rotated items not packed in the according layer. They have to be packed in the new layer, again making it impossible to stack according to the packing plan.



(a) First layer, error occurs

(b) Second layer, error accumulates

Figure 5.7: Accumulating Error

The significantly lacking robustness is one of the drawbacks of the specifically designed heuristic algorithm.

Although the expectation was that the algorithm could find great packings consistently, to check this hypothesis we will need to look into the distribution of the fill rate results (Figure 5.8), running the algorithm various times.

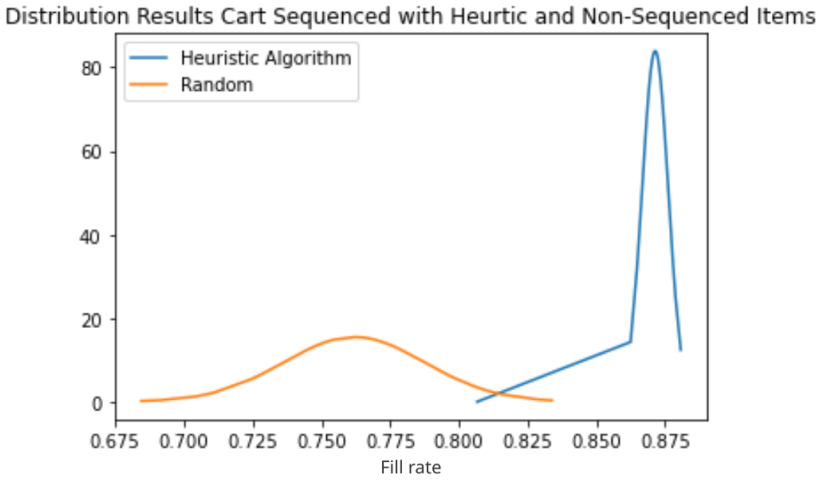


Figure 5.8: Distribution Fill Rate Sequenced with Heuristic and Non-Sequenced Items Methods

The distribution of the result regarding fill rate show that sequencing in this scenario in most cases does help very consistently improving the fill rate. Here, the mean fill rate with the sequenced item packings is 0.87, where the non-sequenced item packings only reach a mean fill rate of 0.75. This indicates that the heuristic for sequencing has the potential to enhance the fill rate in this particular case by an average of 12%, which is a substantial gain. Also, the consistency proves to be as expected. Apart from specific outliers, the consistency is high. Outliers included, the standard deviation is 0.01113. As already explained in Section 4.3, the non-sequenced item packings have a much larger standard deviation, in this case it has the value of 0.02545. The outliers in the results of the heuristically sequenced items are likely the result of the accumulating problems described above.

The heuristic sequencing algorithm presented in this thesis is made for the Skyline Wastemap Reduction algorithm, other algorithms lead to worse packings like Figure 5.9.

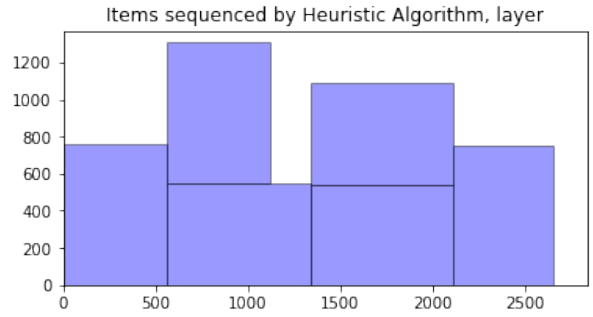


Figure 5.9: Example Packing Heuristic Sequenced Items provided to Maxrects algorithm

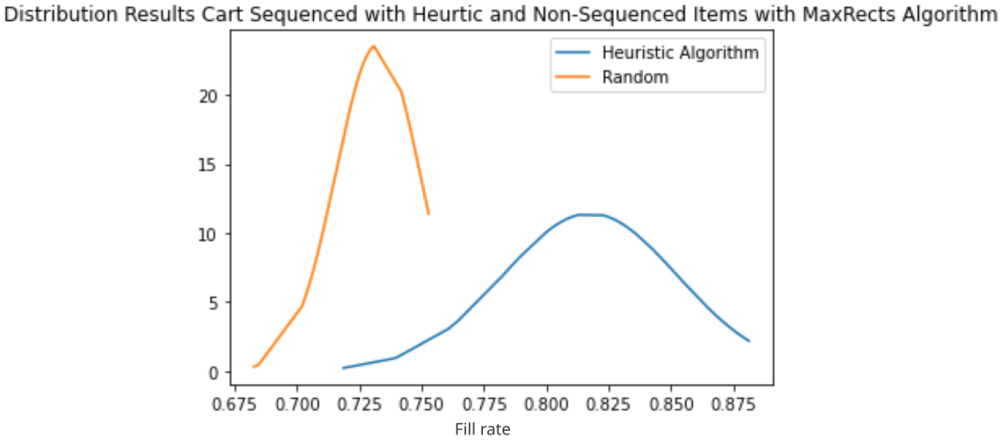


Figure 5.10: Distribution Fill Rate of Packings of Non-Sequenced Items and Items Sequenced by Heuristic (Cart, Skyline Wastemap Reduction Heuristic)

The distribution shown in Figure 5.10 also confirm that other heuristics such as Maxrects paired with the heuristic sequencer do not acquire the same results.

In conclusion, this heuristic sequencing algorithm that is presented in this thesis is specifically adjusted to one case. In this case, most of the time, the sequencer works very well, improving the performance of the packing algorithm with 12% extra fill rate. The drawback of the algorithm is that it is tailor-made for one case and that it is not robust to errors since the error accumulates over other containers, resulting in outliers in the results. However, the non-robust character is not a problem when the algorithm is used in the baggage loading context, where the arriving items' dimensions are varying within a domain, and the container sizes are the same.

5.3.2. Results Heuristic Algorithm Alternative Setup

As indicated earlier in this chapter, the algorithm is made for the main scenario. This is also expected to see in the results when using the heuristic as sequencer in the alternative scenario. Below, a few examples are shown to see what packings by the heuristic look like (see Figure 5.11).

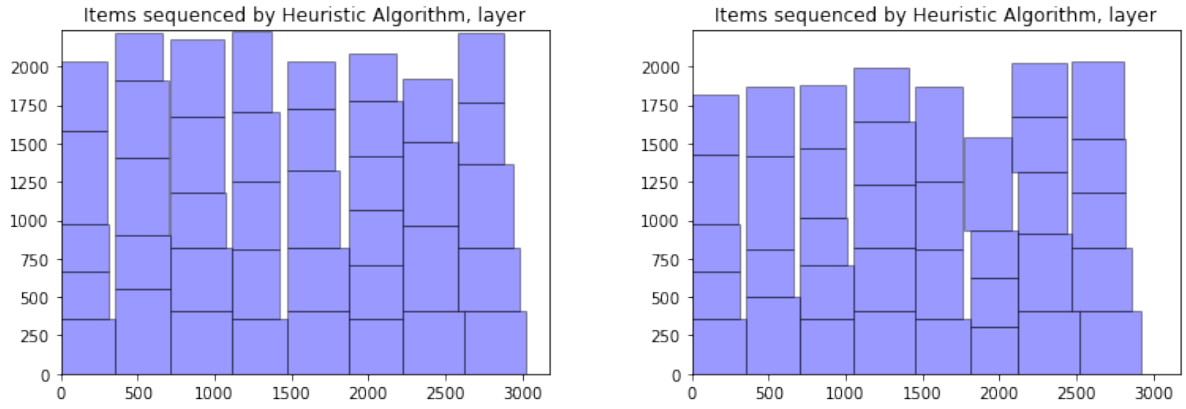


Figure 5.11: Example Packing Alternative Setup

It is plain to see that the method of layer building does not function as well as it does in the main setup. This is due to the fact that the dimensions of the items are spread out more evenly. The poor performance is also made apparent in the Figure 5.12, which shows that the sequencing component only modestly improves fill rate. It is essential to note the results demonstrate the outcome of packings utilising the Skyline Wastemap Reduction packing method.

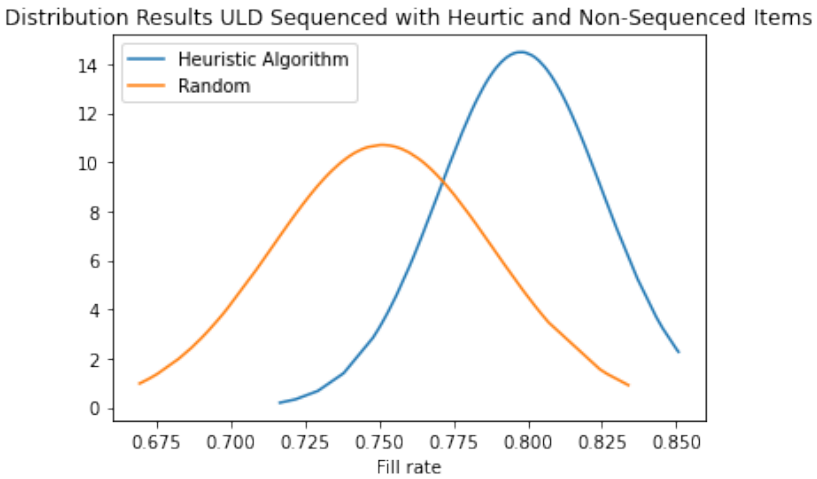


Figure 5.12: Distribution Fill Rate of Packings of Non-Sequenced Items and Items Sequenced by Heuristic (ULD, Skyline Wastemap Reduction Heuristic)

However, as discovered in Section 4.3, the Maxrects algorithm does outperform the Skyline Wastemap Reduction significantly. Therefore, the results are also benchmarked to the results of non-sequenced items using a Maxrects packing algorithm in Figure 5.13.

Figure 5.13 shows that, the sequenced items perform similarly to the random items packed by the Maxrects algorithm. Therefore, one can conclude that the heuristic sequencing algorithm does not work in the alternative scenario.

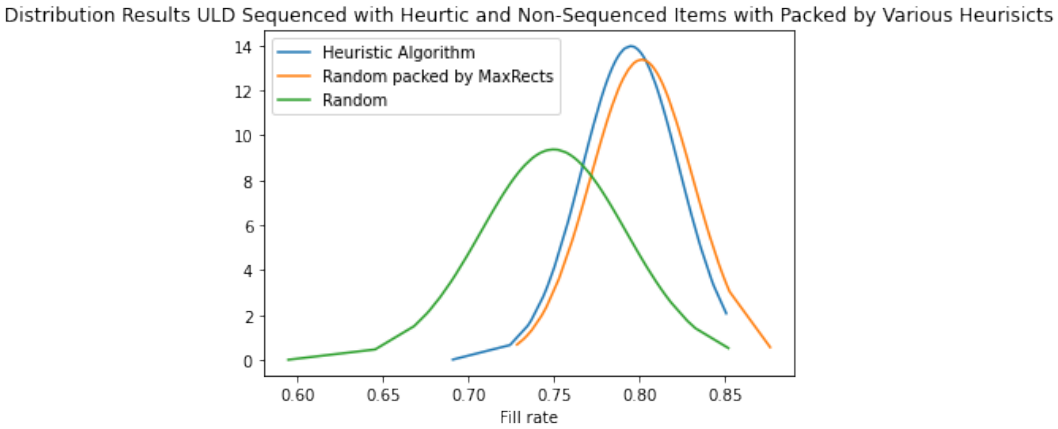


Figure 5.13: Distribution Fill Rate of Packings of Non-Sequenced Items and Items Sequenced by Heuristic and Items Packed with MaxRects Packing Algorithm (ULD, Skyline Wastemap Reduction Heuristic and Maxrects)

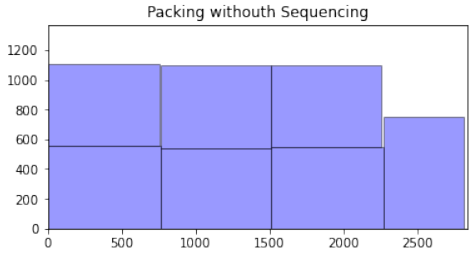
6

Comparing Results & Setup Proposal

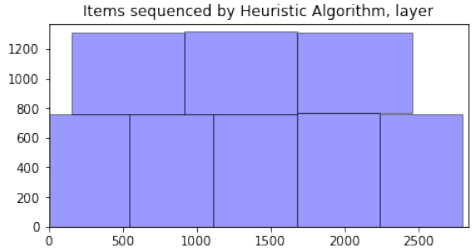
This chapter will consist of two sections. The first section, Section 6.1, will compare all the packing results regarding the main scenario and setup. In the latter section, Section 6.2, the alternative scenario's results are compared.

6.1. Comparing Results of Main Scenario

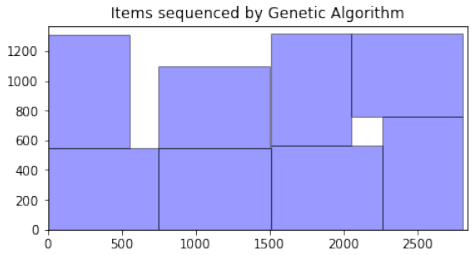
The primary scenario is elaborately described in Section 3.1.2. Here, we showed that in the main scenario, the items are strongly heterogeneous. In this section, we will discuss and compare all sequencing methods for this scenario. First, several packing examples are provided below in Figure 6.1 to illustrate the packing features after sequencing using different approaches.



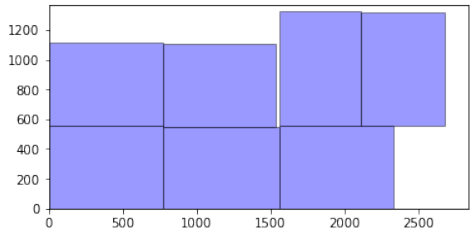
(a) Example Random Packing



(b) Example Packing Sequenced Items by Heuristic Algorithm



(c) Example Packing Sequenced Items by Genetic Algorithm



(d) Example Packing Sequenced Items by Descending Area, Used by Vanderlande Industries

Figure 6.1: Example Packings with Various Sequencing Methods in Main Scenario

It is obvious from the above packing example that the majority of sequencing techniques combined with the Skyline Wastemap Heuristic packing algorithm starts packing with flat-positioned items. Only the heuristic sequencing algorithm does otherwise, and forcefully rotates the first 5 items to pack. Additionally, the heuristically sequenced items and the genetically sequenced items are the only sample packings that place eight items, which results in improved performance in these layers. This is be-

cause the genetic algorithm and the heuristic both sequence the items in order to make more efficient packings.

Lastly, we notice that in contrast to the randomly sequenced item packings and packings with items sequenced by the genetic algorithm, the items sequenced by descending area, only pack the largest items. This results in the fact that the space next to the three first flat positioned items, is too small for another rotated item. Also, the items that must be placed next, in this case, are also relatively large, resulting in a even lower chance of the item being able to be packed in the right corner. The results of the algorithms running 300 times are shown in Figure 6.2, in order to provide a complete overview of the findings and to demonstrate how the algorithms function across many runs. The accompanying statistics are summarized in Table 6.1.

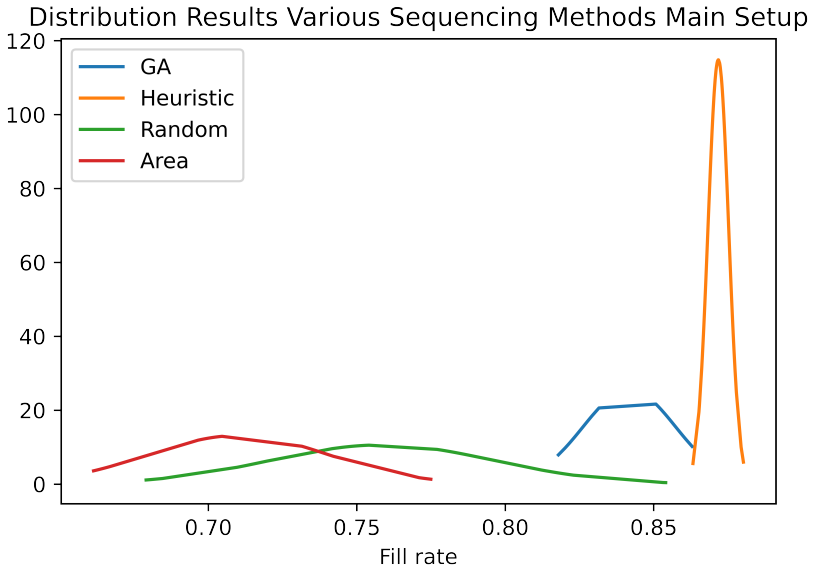


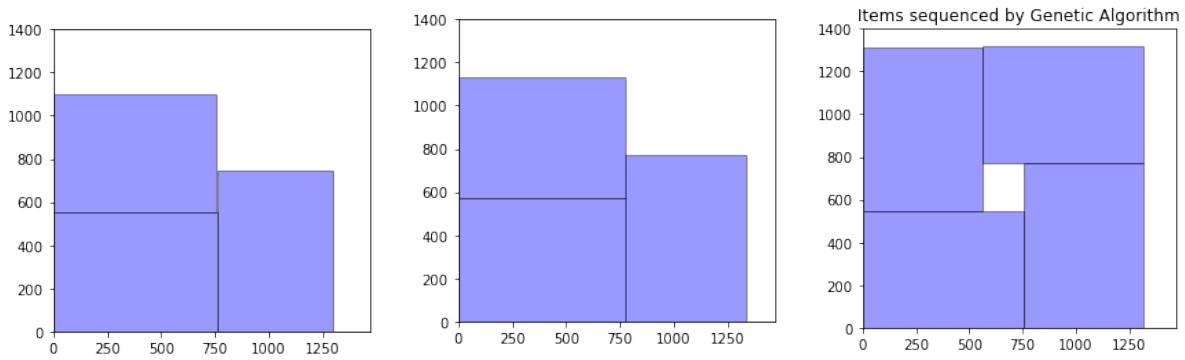
Figure 6.2: Results Main Scenario with Various Sequencing Methods

	Mean	Stdd
GA	0.84	0.0156
Non-Sequenced	0.76	0.0375
Heuristic	0.87	0.0034
Area	0.71	0.0303

Table 6.1: Main Results

Major variability can be seen between the sequencing methods and are visible in Figure 6.2 and Table 6.1. As explained in Chapter 3, Vanderlande Industries currently applies the *Descending Area* method, where all items are sequenced according to area in descending order. The performance in comparison with the fill rate of packings with items in a random sequence, according to the data presented in this thesis, is worse with a mean fill rate of 0.71. Also, the standard deviation of 0.0303 shows how inconsistently this sequencing algorithm manages to reach this fill rate. In earlier sections, Section 4.3 and Section 5.3, we already showed the performance of the designed sequencing methods. Figure 6.2 confirms the usefulness of both methods, where the heuristic algorithm works the best with a mean of 0.87. Also, the consistency could improve majorly by adding a heuristic sequencer to the MCLP pipeline, since the standard deviation can improve from 0.0303 (descending area) to 0.00347.

For the other container type, the ULD, we did not run our heuristic algorithm, but we will compare the results of the genetic algorithm, with the items sorted by area and random packings. Below, some example packings for running the different sequencing strategies are shown (Figure 6.3).



(a) Example Packing Random Sequence

(b) Example Packing Sequenced by Descending Area

(c) Example Packing Sequenced by Genetic Algorithm

Figure 6.3: Example Packings Baggage ULD case

From the images of Figure 6.3, it is evident that the genetic algorithm reaches a much higher fill rate score, since it packs 4 items. The first two packings are very similar, the only difference is the overall size of the items being packed. Here, the example packing with the items sequenced by area packs larger items, resulting in a higher fill rate. To further look into the performance of these strategies, a distribution of the results will be shown in Figure 6.4.

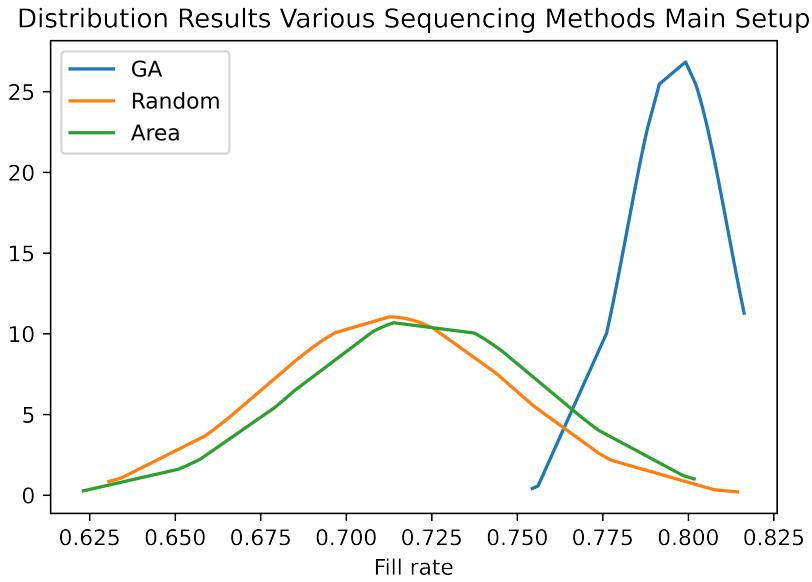


Figure 6.4: Distribution Results for Various Sequencing Methods in Baggage Loading Case with ULD container

	Mean	Stdd
GA	0.80	0.0147
Non-Sequenced	0.71	0.0361
Area	0.72	0.0364

The graph above, confirms the characteristics visible in Figure 6.3. The GA outperforms the other simple methods significantly, with a mean fill rate score of 0.80 versus a mean fill rate score of 0.71 with randomly sequenced items. Also, the consistency is better, with a standard deviation of 0.0147 for the GA, and 0.0364 for the non-sequenced items. The items sequenced by descending area create overall slightly better packings with 1% performance difference versus the randomly sequenced items.

This is the case since the same packing structures are made, but the items are slightly larger, since the largest items are packed.

6.2. Comparing Results of Alternative Scenario

Chapter 3 also described the alternative scenario involving the alternative setup where parcel is stacked in ULDs. All of the several sequencing techniques will be explored and compared in this section. Several packing examples are provided below (Figure 6.5) to illustrate the features of the packings that emerge from various sequencing methods.

Figure 6.5: Example Packings with Various Sequencing Methods in Alternative Scenario

From the packing examples, a few lessons can be learned. When packing the items sequenced by area, one can see a variation in shapes, but similar in area packed after each other. This means that the different shapes do not match and do not stack well together. On the contrary, sorting the items along one side descending leads to a sequence where similar items are sequenced next to each other. The result is a much more filled container, as can be seen in Figure 6.5. The heuristic sequencer places the items row per row. It seems that the items after the first row do not fit properly, resulting in a packing similar to the random packing.

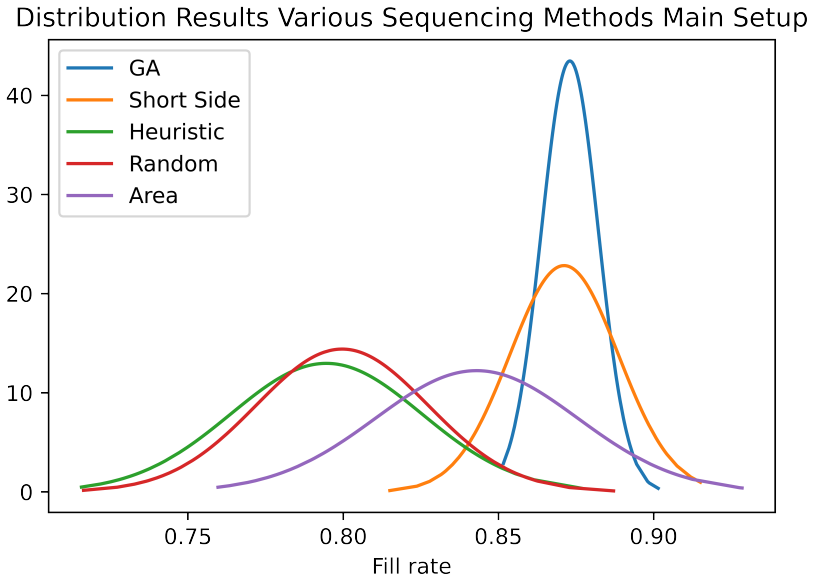


Figure 6.6: Example Packing from Genetic Algorithm

	Mean	Stdd
GA	0.87	0.0092
Non-Sequenced	0.80	0.0276
Heuristic	0.80	0.0308
Short Side	0.87	0.0175
Area	0.84	0.0326

The distribution results show some interesting insights. As already hinted in the example packings, the random packing actually provided better performance than the heuristically sequenced items. Ordering the items in descending area, seems to improve the results, however, only a 4% increase of fill rate is made on average. The best sequencing methods are the sequencing with descending short side. As previously said, one explanation for this might be because comparable items pack together closely, producing excellent packings. The best performing sequencer is the genetic algorithm, which proves to be robust to this alternative scenario, with a mean fill rate of 0.87 and a small standard deviation of 0.0092.

6.3. New Setup Proposal

In this section, we propose a new approach to the loading of bags or parcel into containers. Again, the section is split between the main scenario (Section 6.3.1) and the alternative scenario (Section 6.3.2), since they acquire different design decisions.

6.3.1. New Setup for Main Scenario

In this scenario, we would like to propose two option for Vanderlande Industries to improve their bag loading setup's performance.

Main: Option 1

As described in Chapter 3, the bag load setup requires to *pull* items from the EBS. In the current scenario, the robot can not load items that come directly from check-in. In this way, the robot as able to take more time loading the containers, since the robot takes more time than manual loading. Making use of this system, as option 1, we propose to let the heuristic sequencer calculate a packing plan, per layer, before the items are retrieved from the EBS. If the plan succeeds and over 83% of the container

can be filled with the 8 proposed items in each layer, the process will proceed to the next step. If less than the required fill rate is reached, the genetic algorithm will be used to try to plan for better packings which reach at least the 83% requirement. Then, the items will be sent to the robot in the correct sequence. Also, the information must be sent to the robot telling that the first 5 items must be placed rotated 90 degrees. This would be to make sure that the packing plan succeeds. An advantage of this setup is that it does not need extra hardware, while this proposed setup can gain up to 12% fill rate. A downside is that the items often do not arrive at the robot in the order they were sent from the EBS. This occurs since the transport lines from the EBS to the robot are highly redundant, and multiple routes can be taken. The Vertisorter in the end of the transport lines to the robot can be of help for this problem, since it allows switching up different items when the arrival. Option 2 eliminates this problem.

Main: Option 2
With this option, we would like to propose an extra component in the logistical system for bag loading: the exact sequencer. This component will retrieve the items the same way as option 1, but it can reorder the items when the arrival sequence does not match the planned sequence. A major downside of this option, is that it is costly, since a new component has to be made for resequencing the items. On the other hand, it allows for impurities in the arrival sequence of the items. Also, it allows items to be sent directly from the check-in lines. It remains for further investigation if the option is cost worthy.

6.3.2. New Setup for Alternative Scenario

Since the current alternative setup does not have an EBS where all the parcel can be stored, for improvement of the performance of the packings, we would like to propose the use of a sequencing element as well. The same as in option 2 above, it will be an element that can buffer certain items. Within this buffer, the genetic sequencing algorithm designed in this thesis will be used to propose a sequence in which the items have to arrive at the robot. The element then will send the items in the correct sequence for the robot to be packed. However, before this system will be used, the algorithm needs to be tested more extensively. Also, the packing approach of the robot packer must be known in order to work properly.

7

Conclusion, Discussion and Future Work

In this chapter, firstly, we will conclude the research done in this thesis in Section 7.1. Afterwards, we are going to discuss the drawbacks of the methods used in Section 7.2. Finally, there will be elaborated on future work in Section 7.3 that can be done to further develop sequencing strategies for the MCLP.

7.1. Conclusion

In this thesis, we conducted research into sequencing items in various ways for the MLCP in a bag and parcel load context. In this report, we attempted to find an answer to the following research question:

> *How does sequencing items improve fill rate of simple container loading algorithms in a baggage and parcel loading context?*

Firstly, we looked into the literature to get a grip of the academic landscape regarding the MCLP. Also, there has been looked into state-of-the-art methods. To see the problem in its context, a visit was made to Schiphol to see the physical robot which is solving the MCLP as a pilot project for Schiphol. From this research, we concluded that there are various methods for solving the MCLP, however, sequencing methods are mostly kept simple. Also, in practice, baggage items are placed in the order where size and weight are decreasing. The knowledge gap found was that sequencing for the MCLP is not done, especially not for in an industrial context. This knowledge suits the desire for Vanderlande Industries to research the use of sequencing methods on their product in the baggage and parcel case.

In the chapters Chapter 4 and Chapter 5 we presented two kinds of algorithms to provide a calculated sequence. Here, the heuristic algorithm makes use of a set of rules and the genetic algorithm makes use of multiple generations to optimise the sequence. The resulting packings show to be different, varying the sequencing approach. Some of the example packings for the main baggage scenario are shown in Figure 6.1. For the main scenario, according to the simulation made, the heuristic sequencing method proves to generate that best results regarding the fill rate of the layers. On average, the performance of the packings can with the heuristic sequencer be increased by average with 11 % in comparison with packings generated with a random item sequence. Also, the consistency is very high since it always manages to position eight items in one layer of the container. As mentioned earlier in this section, Vanderlande Industries sequences the items by descending size and weight. However, results in this study point out that these methods performs worse than random sequenced items, with a mean fill rate of 0.71. Although the heuristic worked well in this scenario, it is lacking robustness, since it performs the worst in the alternative scenario.

The alternative scenario shows less obvious results, however, simple techniques like sorting the items from descending short side show to work well. The results using this method are: an increase of 7% in comparison to random item sequence packings. We assume the simple sorting approach worked well since it grouped all similar shaped objects, resulting in tight packings. Slightly better results were generated by the genetic algorithm, producing on average similar results (0.87) but significantly more consistent packings indicated with a lower standard deviation of 0.00917 (GA) versus a standard deviation of (0.01747). As stated earlier, the heuristic produced the worst results. This proves that sequencing with a heuristic works, but it is not a robust method for any scenario. On the contrary, the

genetic algorithm produces a bit less performing results, but it can adapt to different scenarios and setups.

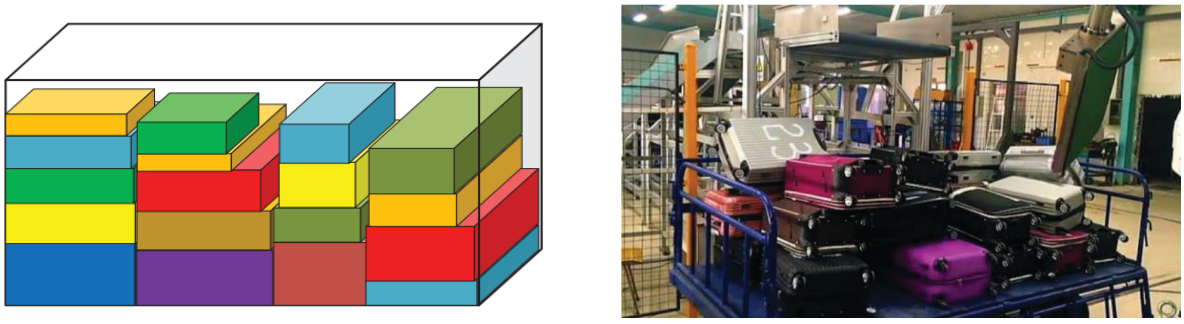
To finalise, sequencing helps improve fill rate for simple container loading problems. In the main setup, the baggage case, the fill rate can be improved with 12% and in the alternative scenario or parcel case, the fill rate can be improved with 7%. This improved sequence is made with a heuristic algorithm for the baggage case and a genetic algorithm in the parcel case. We think sequencing is working the best in cases where the item dimensions are relatively large compared to the container that needs to be packed. This results in the fact that great sequences can lead to more items to be packed, resulting in significantly higher fill rates.

Regarding the robustness of the algorithms, the hypothesis stated in Chapter 1 is met, since the more general approach of the genetic algorithm results in a flexible algorithm which can be applied to different cases. On the other hand, we expected a tailored algorithm to work only in a specific case. This is also reinforced in this thesis with the great results of the heuristic algorithm in one case.

With this conclusion, we filled the knowledge gap regarding the sequencing part of simple packing methods, given light into what sequencing is able to do in two industrial scenarios.

7.2. Discussion

Although hard conclusions may seem to be taken from the results, there were some critical assumptions made in the design of the algorithm and for testing the algorithm in simulation. Firstly, the assumption was made that all items were cuboid. This was done to make it able for existing heuristics to pack the items. As a result, the packing became more efficient and fitting. In real life packings of the main scenario, the items are not cuboids like illustrated in Figure 4.2. As explained in Chapter 3, the outer edges of typical baggage pieces are taken to simulate the baggage items. This assumption results in a mismatch of the fill rate of actual packings and simulated packings. This must be taken into account when looking into the results. If the actual shape is taken account into the simulation, it will also have influence on stability characteristics of the stack. Since the items in the layer on top of the other layer are less supported in real life than when simulated as rigid cuboids. An example is visible in research by (Shengming et al., 2020). Figure 7.1 shows the difference between simulated and real life packings. Also, in Chapter 3 explains that baggage pieces deform depending on the material of the luggage. This is also something that is not incorporated in the stacking algorithm since we assumed that for each layer, the layer on top will be supported with a flat layer underneath since the layer underneath deforms until it is a flat layer. We are not sure till what extent this would happen, it was assumed to make the calculations and stacking easier. This ensured that we could only simulate with two-dimensional MCLPs, which made the overview and structure easier. Although the baggage case seemed to work in layers, for the parcel case this assumption is not supported.



(a) Simulated Packing

(b) Real Life Packing

Figure 7.1: Simulated Packing and Real Life Packing by (Shengming et al., 2020)

A second discussion point is that the item data was very limited. For the baggage case, we sampled the dimensions of possible baggage items from a three-dimensional normal distribution taken from actual data. However, in practice, the baggage did look very similar, even more similar than in real life. This limits representativeness of the actual packings, since items can be very different. In the parcel case, we sampled from existing parcel items. Although, the amount of items was very limited and in the future, more items needed to be added.

Adding up, sequencing must be done given a certain packing algorithm. In this thesis, we made the assumption that in most cases, the packing algorithm is a Skyline Wastemap Reduction algorithm. This was assumed since it seemed to behave like a horizontal layer building heuristic, which was likely to be used in the Schiphol pilot project. For the alternative scenario, the packing algorithm was also compared to a Maxrects algorithm, showing better results. Before using the sequencing algorithms in other scenarios than the bag loading robot, one would have to make sure to know the actual packing algorithm. If the algorithm is unknown, one must pick an algorithm which makes packings a similar approach.

Moreover, in this thesis, we assumed that in sequencing of items is deterministic. This means that when a specific order of items is sent to the robot, the items will arrive in this order. In practice, the system is not deterministic and some items get delayed in the transportation system since there are multiple routes that can be taken in the maze of conveyors at for example Schiphol. This phenomenon will result in the fact that planned packings do not always match with the actual packings, they are likely to be less efficient.

A last, but major discussion point is the genetic algorithm. Since there is no crossover action used in the genetic algorithm presented in this thesis, it does not actually converge to an optimal solution. This affects the result significantly and only when such a step is incorporated, the true potential can be seen of this algorithm. However, it would also extend the running time of the algorithm significantly.

7.3. Future Work

An opportunity to develop the sequencing method further is to add an action in one of the algorithms. As already indicated in Section 7.2, the crossover action is not incorporated in the current genetic algorithm. In (Wu et al., 2010), the problem of non-feasible children solutions from parents also occurs. However, they use two type of repairing schemes to *fix* the children solution. So-called *Sequential Repair* aims to repair the solutions to take the missing items, and replacing the items which occur twice in the sequence with it. The other repair method they present is the random repair action. Here, a random doubled item is taken and replaced by a random item which is not in the sequence, repairing the solution. The crossover action and this repairing action can provide the algorithm with a better solution, since the combination of solution can also be used in that case.

Also, more research can be done on the predictability of the system. As described in Section 7.2, the planning does most of the time not match up with actual packing. Since in the baggage case, the fill rate is highly dependent on the sequence, we would like to point out that more research needs to be done on this aspect. Furthermore, when packings are made, sometimes errors are made by the robot. In this case, the operator has to step in and has to fix the issues. By doing this, he or she is likely to adjust the packing. This also has influence on the performance. In future work, one could design a sequencer that is robust in these kind of situations.

A third point can be to add stability to the simulation. In that case, a top layer can only lean on a bottom layer on the places the bottom layer is *filled*. This would limit the placement of all items not on the first layer of the container.

As a final point of future work, we would like to test the algorithms in three-dimensional simulations to know how sequencing affects the results of three-dimensional packings.

References

Bergsten, E. L., Mathiassen, S. E., & Vingård, E. (2015). Psychosocial work factors and musculoskeletal pain: A cross-sectional study among swedish flight baggage handlers. *BioMed Research International*, *2015*. https://doi.org/10.1155/2015/798042

Bortfeldt, A., & Wäscher, G. (2013). Constraints in container loading-A state-of-the-art review. *European Journal of Operational Research*, *229*(1), 1–20. https://doi.org/10.1016/j.ejor.2012.12.006

Chen, C. S., Lee, S. M., & Shen, Q. S. (1995). An analytical model for the container loading problem. *European Journal of Operational Research*, *80*(1), 68–76. https://doi.org/10.1016/0377-2217(94)00002-T

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2010). *Introduction to Algorithms* (Vol. 7). https://doi.org/10.1017/CCO9781139424783.002

Crainic, T. G., Perboli, G., & Tadei, R. (2008). Extreme point-based heuristics for three-dimensional bin packing. *INFORMS Journal on Computing*, *20*(3), 368–384. https://doi.org/10.1287/ijoc.1070.0250

Dechow, P. M., & Douglas J, S. (2000). A Maximal-Space Algorithm for the container loading problem. *Journal of Allergy and Clinical Immunology*, *130*(2), 556. http://dx.doi.org/10.1016/j.jaci.2012.05.050

Duan, L., Hu, H., Qian, Y., Gong, Y., Zhang, X., Wei, J., & Xu, Y. (2019). A multi-task selected learning approach for solving 3D flexible bin packing problem. *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, *3*, 1386–1394.

Duarte Alonso, A., Kok, SK, & O Brien, S. (2011). An online packing heuristic for the three-dimensional container loading problem in dynamic environments and the Physical Internet. *Lecture Notes in Computer Science*, *1*, 140–155. http://researchonline.ljmu.ac.uk/id/eprint/8705/

Elhedhli, S., Gzara, F., & Yan, Y. F. (2017). A MIP-based slicing heuristic for three-dimensional bin packing. *Optimization Letters*, *11*(8), 1547–1563. https://doi.org/10.1007/s11590-017-1154-5

Fan, B., & Luo, J. (2013). Spatially enabled emergency event analysis using a multi-level association rule mining method. *Natural Hazards*, *67*(2), 239–260. https://doi.org/10.1007/s11069-013-0556-7

Feo, T. A., & Resende, M. G. (1995). Greedy randomized adaptive search procedures. *Journal of global optimization*, *6*(2), 109–133.

FNV. (2022). Nu is het moment om werk op Schiphol aantrekkelijk te maken [[Online; accessed 30. Aug. 2022]]. https://www.fnv.nl/nieuwsbericht/algemeen-nieuws/2022/05/nu-het-moment-om-werk-op-schiphol-aantrekkelijk-te

Foot, D. K. (2008). Some economic and social consequences of population aging. *PROGRAMME DE PRIORITÉS POUR LE CANADA*.

George, J. A., & Robinson, D. F. (1980). A heuristic for packing boxes into a container. *Computers and Operations Research*, *7*(3), 147–156. https://doi.org/10.1016/0305-0548(80)90001-5

Gonçalves, J. F., & Resende, M. G. (2012). A parallel multi-population biased random-key genetic algorithm for a container loading problem. *Computers and Operations Research*, *39*(2), 179–190. https://doi.org/10.1016/j.cor.2011.03.009

Harrath, Y. (2021). A three-stage layer-based heuristic to solve the 3D bin-packing problem under balancing constraint. *Journal of King Saud University - Computer and Information Sciences*, 1319–1578. https://doi.org/10.1016/j.jksuci.2021.07.007

Holdert, M., & Meindertsma, B. (2022). Arbonormen bagagepersoneel Schiphol jarenlang overschreden, deel heeft gezondheidsklachten. *NOS Nieuws*. https://nos.nl/collectie/13911/artikel/2443508-arbonormen-bagagepersoneel-schiphol-jarenlang-overschreden-deel-heeft-gezondheidsklachten

IATA. (2016). Baggage ID [[Online; accessed 29. Sep. 2022]]. https://www.iata.org/en/publications/store/baggage-id

IATA. (2019). Iata report.

Interview with van Bavel. (2022). *Schiphol viewing report*.

Jiang, J., & Yin, S. (2012). A self-adaptive hybrid genetic algorithm for 3D packing problem. *Proceedings - 2012 3rd Global Congress on Intelligent Systems, GCIS 2012*, 76–79. https://doi.org/10.1109/GCIS.2012.34

Jylänki, J. (2010). A Thousand Ways to Pack the Bin - A Practical Approach to Two-Dimensional Rectangle, 1–50.

Kantorovich, L. V. (1960). Mathematical Methods of Organizing and Planning Production. *Management Science*, *6*(4), 366–422. https://doi.org/10.1287/mnsc.6.4.366

Lambora, A., Gupta, K., & Chopra, K. (2019). Genetic Algorithm- A Literature Review. (1998).

Martello, S., Pisinger, D., & Vigo, D. (2000). Three-dimensional bin packing problem. *Operations Research*, *48*(2), 256–267. https://doi.org/10.1287/opre.48.2.256.12386

Moon, I., & Nguyen, T. V. L. (2014). Container packing problem with balance constraints. *OR Spectrum*, *36*(4), 837–878. https://doi.org/10.1007/s00291-013-0356-1

NOS. (2022a). Chaos op Schiphol door KLM-staking: reizigers opgeroepen niet meer te komen [[Online; accessed 29. Aug. 2022]]. https://nos.nl/artikel/2426169-chaos-op-schiphol-door-klm-staking-reizigers-opgeroepen-niet-meer-te-komen

NOS. (2022b). KLM schrapt maandag 42 vluchten vanwege drukte Schiphol [[Online; accessed 22. Sep. 2022]]. https://nos.nl/artikel/2445080-klm-schrapt-maandag-42-vluchten-vanwege-drukte-schiphol

Oliveira, Ó., Matos, T., & Gamboa, D. (2020). Adaptive Sequence-Based Heuristic for the Three-Dimensional Bin Packing Problem. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11968 LNCS*, 69–76. https://doi.org/10.1007/978-3-030-38629-0_6

Oxley, L., Tapley, S., & Hill, H. (2009). Musculoskeletal ill-health risks for airport baggage handlers report on a stakeholder project at east midlands airport. *HSE Books*, 1–9. http://www.hse.gov.uk/research/rrpdf/rr675.pdf

Pardalos, P. M., Du, D. Z., & Graham, R. L. (2013). *Handbook of Combinatorial Optimization* (Vol. 1-5). https://doi.org/10.1007/978-1-4419-7997-1

Perez-Salazar, S., Singh, M., & Toriello, A. (2021). Adaptive Bin Packing with Overflow. *ACGO*, 1–47.

Quarrrb, M. C. (2022). Excursions in Math [[Online; accessed 30. Sep. 2022]]. http://personal.morris.umn.edu/~mcquarrb/teachingarchive/M1001/Resources/BinPacking.html

Rayward-Smith, V. J. (1986). *A first course in computability*. Blackwell Scientific.

Saraiva, R. D., Nepomuceno, N., & Pinheiro, P. R. (2015). A layer-building algorithm for the three-dimensional multiple bin packing problem: A case study in an automotive company. *IFAC-PapersOnLine*, *28*(3), 490–495. https://doi.org/10.1016/j.ifacol.2015.06.129

Selvanathan, N., & Tee, W. J. (2003). A genetic algorithm solution to solve the shortest path problem in OSPF and MPLS. *Malaysian Journal of Computer Science*, *16*(1), 58–67.

Shengming, C., Wei, Z., Zhaobin, D., & Liwen, W. (2020). A tree search loadind algorithm for airport checked baggage. *IET Conference Publications*, *2020*(CP776), 923–927. https://doi.org/10.1049/icp.2021.0332

Silva, E. F., Toffolo, T. A. M., & Wauters, T. (2019). Exact methods for three-dimensional cutting and packing: A comparative study concerning single container problems. *Computers and Operations Research*, *109*, 12–27. https://doi.org/10.1016/j.cor.2019.04.020

Skiena, S. S. (1997). *The Algorithm Design Manual* (3rd). https://algorist.com/algorist.html

Stepán, J. (2009). GRASP: A Search Algorithm for Propositional Satisfiability. *Vnitrni lekarstvi*, *55*(5), 448–54. http://www.ncbi.nlm.nih.gov/pubmed/19514609

U. S. Bureau of Labor Statistics. (2009). Occupational injuries and illnesses by selected characteristics news release. https://www.bls.gov/news.release/archives/osh2_12042009.htm#

United Nations. (2019). *World population prospects 2019*. http://www.ncbi.nlm.nih.gov/pubmed/12283219

Vanderlande. (2022). Vanderlande industries: Airports business unit. https://www.vanderlande.com/airports/

Vanderlande Industries. (2020). Vanderlande - Parcel Innovative systems: VERTISORTER. *Vanderlande*. https://www.vanderlande.com/systems/sorting/vertisorter

Vanderlande Industries. (2021). *Commercial Report Commercial Report FY2021* (tech. rep.).

Verma, A. A. (2015). A Survey on Image Contrast Enhancement Using Genetic Algorithm. *International Journal of Science and Research (IJSR)*, *4*(3), 740–744. https://www.ijsr.net/archive/v4i3/SUB152087.pdf

Wei, L., Zhang, D., & Chen, Q. (2009). A least wasted first heuristic algorithm for the rectangular packing problem [[Online; accessed 1. Sep. 2022]]. *Computers & Operations Research*, *36*(5), 1608–1614. https://doi.org/10.1016/j.cor.2008.03.004

World Maritime University. (2019). *Transport 2040: Automation, Technology, Employment - The Future of Work* (tech. rep.). https://commons.wmu.se/lib_reports/58

Wu, Y., Li, W., Goh, M., & de Souza, R. (2010). Three-dimensional bin packing problem with variable bin height. *European Journal of Operational Research*, *202*(2), 347–355. https://doi.org/10.1016/j.ejor.2009.05.040

Zhang, D., Peng, Y., & Leung, S. C. (2012). A heuristic block-loading algorithm based on multi-layer search for the container loading problem. *Computers and Operations Research*, *39*(10), 2267–2276. https://doi.org/10.1016/j.cor.2011.10.019

Zhao, H., She, Q., Zhu, C., Yang, Y., & Xu, K. (2020). Online 3D Bin Packing with Constrained Deep Reinforcement Learning. (Skiena 1997). http://arxiv.org/abs/2006.14978

Zhao, H., Zhu, C., Xu, X., Huang, H., & Xu, K. (2021). Learning Practically Feasible Policies for Online 3D Bin Packing. *Sci China Inf Sci*, 1–18. http://arxiv.org/abs/2108.13680

Zhao, X., Bennell, J. A., Bektaş, T., & Dowsland, K. (2016). A comparative review of 3D container loading algorithms. *International Transactions in Operational Research*, *23*(1-2), 287–320. https://doi.org/10.1111/itor.12094

Grote paper met alle methodes voor bin packing

Zhu, W., Oon, W. C., Lim, A., & Weng, Y. (2012). The six elements to block-building approaches for the single container loading problem. *Applied Intelligence*, *37*(3), 431–445. https://doi.org/10.1007/s10489-012-0337-0