An Optimisation and Forecast Framework for ULD Packing in the Air Cargo Supply Chain

Master Thesis Report

M. R. Koch

A Case Study for Air France KLM Martinair Cargo

TU Delft

An Optimisation and Forecast Framework for ULD Packing in the Air Cargo Supply Chain

Master Thesis Report

by

M. R. Koch

to obtain the degree of Master of Science
at the Delft University of Technology.

Student number: 4232313
Project duration: June 17, 2018 – January 4, 2019
Supervisor: B. F. Santos, TU Delft - Supervisor
A. Bombelli, TU Delft - Daily Supervisor
H. Zwitzer, KLM Cargo - Director Development & Process Support
L. Heling, KLM Cargo - Director Revenue Management

Preface

The final part of the Masters of Aerospace Engineering at Delft University of Technology is the Master's thesis project. Before the thesis, a literature study has been conducted to get to know the state of the art of the thesis topic. The thesis consists of an individual in-depth research in which all the abilities obtained in the Master's are used. This report is the Master's thesis report. The thesis is performed in collaboration with Air France KLM Martinair Cargo and TU Delft. Investigated is how to optimise the ULD packing to increase the revenue considering the unknowns of a booking and uses this to come up with a risk of the optimal loading strategy. The airline delivers the data, while the TU Delft comes up with the model.

I would like to thank Alessandro Bombelli for the weekly meetings and all the coffees. Next I would like to thank Bruno Lopes dos Santos for a critical note on the project when necessary. Many thanks go to Hans Zwitzer for offering this intern position and for its help getting a job within KLM. I would like to thank Luuk Heling, for its enthusiasm about the results. Many thanks go to all other KLM employees who helped when necessary. As last I would like to thank Gijs de Rooij for our study sessions at the TU Delft and all the tea we drank during these sessions.

M. R. Koch
Schiphol-Oost, February 12, 2019

Contents

List of Figures

List of Tables

List of Abbreviations & Symbols

AFKLMP	Air France KLM Martinair Cargo
AF	Air France
AMS	Amsterdam Airport Schiphol
ANN	Artificial Neural Network
AWB	Air Waybill
BPP	Bin Packing Problem
BRUNO	Better Revenue Using New Optimiser
CDG	Paris Charles de Gaulle
CP	Commercial Priority
FPM	Flight Planning & Manifesting
FPO	Flight Palletisation Optimisation
KL(M)	Koninklijke Luchtvaart Maatschappij N. V. (Royal Dutch Airlines)
KP	Knapsack Problem
LDP	Lower Deck Pallet
MAE	Mean Absolute Error
MDP	Main Deck Pallet
MILP	Mixed Integer Linear Programming
MSE	Mean Squared Error
OHG	Overhang
StD	Standard Deviation
ULD	Unit Load Device
RM	Revenue Management
RO	Operations Research
RF	Random Forest
RQ	Research Question
SCb	Shipment Contribution
1DBPP	One-Dimensional Bin Packing Problem
2DBPP	Two-Dimensional Bin Packing Problem
3DBPP	Three-Dimensional Bin packing Problem
744	Boeing 747-400

1

Executive Summary

The air cargo industry is a challenging environment due to the high competition between the stakeholders involved. This demands, as example, high efficiency from cargo airlines. Efficiency can be ensured by designing loading strategies that fully exploit the available cargo volume. Unknowns in the booking dimensions and flight information make this task challenging. This thesis proposes a framework that can improve cargo airlines' decision-making when assessing whether a new booking is received. The model described in the thesis forecasts unknowns of both the bookings and the flights and combines this information with a bin packing heuristic. It is assessed whether the incoming booking should be accepted or not, and a risk associated with the acceptance of the booking is also computed. The research questions (RQs) are defined as follows:

1. How to forecast the unknown characteristics of both the bookings and the flights (RQ1)?
   (a) What are the length, width and height of cargo?
   (b) What is the amount of must-fly cargo per flight?
   (c) What is the volume loss per booking origin?
   (d) What is the accuracy of the flight details?
2. How to palletise the ULDs for a future flight (RQ2)?
3. What is the risk of accepting a booking (RQ3)?

The bin packing problem (BPP) is researched extensively in different forms. However, to the best of our knowledge, the combination with forecasting unknown characteristics of both the flight and the bookings has not been investigated yet. This makes the research both innovative and feasible.

RQ1 defines the different forecasts. For these forecasts a random forest (RF) model is used. This is an ensemble learning method used for both classification and regression. A classification outputs class labels while a regression outputs continuous variables. Features are used as input. The dataset is divided in both a training and testing set. Using a training set, generally based on historical data, decision trees are built, each characterized by different parameters and a different quality of the solution. Decision trees all concur to the computation of the final solution (i.e., weights of the different input features). The solution is then used to evaluate the quality of the model using the testing set. Results applied to the historical dataset used in the thesis are shown in Table 1.1.

Table 1.1: Final parameters for the random forest model.

Parameter:	**Value**:
Test Size:	0.25
Number Decision Trees:	100
Quality Criterion:	MSE
Random State:	42

Table 1.2: Results for the different forecasts.

Model:	**Error**:	**Accuracy**:
Dimensions:	9.89 cm	84.29 %
Must-Fly Cargo:	11.02 m^3	83.30 %
Flight Details:	4.59 %	94.70 %

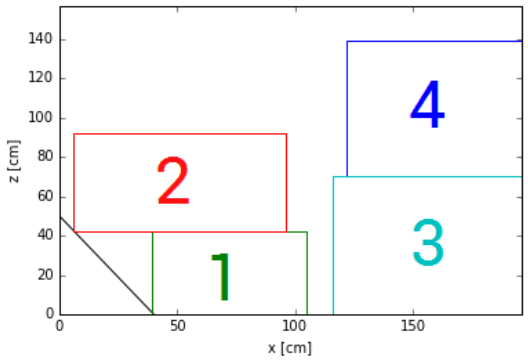


Figure 1.1: An ULD with four loaded pieces.

An efficient loading strategy needs information on dimensions for every piece. This information is generally not available 24 hours before departure. This issue motivates the need for a forecasting model that estimates dimensions of pieces. The RF model with the parameters as in Table 1.2 is applied to five months of historical data. Using the features *AgentURN*, *BookingDestination*, *BookingOrigin*, *CommodityCode*, *PieceVolume* and *ProductCode* the dimensions are forecasted with an error of 9.89 cm per and an accuracy of 84.29 % per dimension. The most important feature is the *BookingOrigin*. This can be exlained by the fact that the same commodities are generally transported from the same origin and they are generally packed in the boxes with the same dimensions. Therefore it is advised to determine the accuracy per *BookingOrigin* and use the inaccuracy (1 - accuracy) as volume loss. Volume loss is used by airlines to increase the volume of a booking, this is done to make sure the booking will fit in the aircraft. The volume loss ranges between 7 and 30 %. Using the test dataset to compare predicted and actual dimensions, it was found that errors (per booking origin) are, within a reasonable approximation margin, normally distributed. Therefore, the mean and standard deviation (StD) per booking origin was determined. This information can be used to provide an estimation of the risk (per ULD) that the actual shipment configuration, with the predicted dimensions, will not be implementable. This is caused by the mismatch between predicted and actual shipment dimensions.

The RF model with the parameters in Table 1.1 is also used to forecast specifications of the flight. First the amount of must-fly cargo that arrives before a flight is forecasted. It is usefull to forecast this, to determine if there is still enough space in the ULDs when accepting a new non-must-fly booking. Must-fly cargo is the cargo that may not be offloaded. Using a year of history data and features *AircaftSubType*, *Arrival*, *CapacityVolume*, *CapacityWeight*, *FlightNumber* and *WeekDay* leads to the results in Table 1.2. Second, the accuracy of the flights capacity forecast is forecasted. Since most cargo is transported in passenger planes, passenger's bags has priority. This limits the capacity for cargo, however the cargo capacity is unknown until the moment of departure. Therefore a cargo capacity forecast is performed by the airline but the accuracy of this forecast is unknown. This accuracy can help the airline decide to accept more or less cargo than forecasted. Using a year of history data and features *AircraftSubType*, *Arrival*, *CapacityVolume*, *CapacityWeight*, *DayBeforeDeparture*, *FlightNumber*, *Month* and *Weekday* leads to the results in Table 1.2. This concludes the forecasting part.

The bin packing problem is tackled in three steps. First a variation of the one-dimensional bin packing problem (1DBPP) is applied since it is the simplest BPP, second the two-dimensional bin packing problem (2DBPP) is investigated since an extra dimension is added which comes closer to reality compared to the 1DBPP and last different heuristics are tested since the 2DBPP turned out to be computationally demanding. The 1DBPP assigns pieces to ULDs considering the volume of both the pieces and the ULDs. Differently than a classic 1DBPP, dimensions of packages are preliminary checked to make sure they can fit in the assigned container. Furthermore, perishables and radioactive pieces are prevented from being loaded in the same ULD, which is a requirement of the air cargo industry. In addition, if a shipment is a car, no other shipment can be assigned to the same ULD. This preliminary step leads to good load factors with a very short computational time. The 2DBPP builds further on the basis of the 1DBPP. Differently than in the 1DBPP, dimensions (length and height) of all pieces are simultaneously considered to make sure geometric constraints are satisfied. The 2DBPP routine stacks pieces on top of each other, it makes sure that all vertical vertices are supported when the piece is not on the ground and It prevents overlap of the pieces. ULDs are not necessarily

rectangular shaped. One or more corners are cut to follow the shape of the aircraft fuselage. These cuts are considered as well. Furthermore both vertical vertices of a piece are supported by either the ground floor, a cut or another piece. The 2DBPP turned out to be computationally very demanding, which would hinder the possibility to use such approach for real-time or quasi real-time application. Therefore three heuristics are investigated. The first heuristic, called *1D2D*, tries to speed up the 2DBPP by assigning each piece to an ULD first, using the 1DBPP. Next, for every ULD a 2DBPP is solved with the pieces assigned to it by the 1DBPP. This leads to a consistent decrease in computational time. However a lot of volume is lost because only two dimensions are considered. The second heuristic is called *Layers* and adds the third dimension to the 2DBPP. The heuristic starts with the first ULD and all pieces. It solves the 2DBPP for this ULD. Some pieces are loaded into this ULD and others are offloaded. When at least one of the offloaded pieces fits behind the loaded pieces, the ULD is split along the dimension initially neglected by the 2DBPP into two layers. The first layer width equals the maximum width of all the loaded pieces while the width of the other layer equals the spare width. The 2DBPP is solved for the second layer considering its width and the offloaded pieces. This process continues until all pieces are loaded or until none of the offloaded pieces fit in the remaining space of the ULD. When this last case happens, the routine moves on to the second ULD and repeats this process for this ULD. This will be repeated for all ULDs, one by one, until no ULDs or pieces are left. This leads to significant better load factors, however the computational time still remains an issue. Therefore it was decided to combine both the *1D2D* heuristic and the *Layers* heuristic. This heuristic is called *1D2D Layers*. This results in good load factors, while keeping the computational time within reasonable bounds.

The final model is called BRUNO, which stands for Better Revenue Using New Optimiser. In the first step, the data is both loaded and cleaned. The first model that is added to BRUNO is the dimension forecast. The model is loaded, and the unknown dimensions are predicted. Some bookings are characterised by a high number of pieces and can be bundeld together. For example, ten boxes of salmon with the same dimensions. Since every piece would be treated separately in the heuristic, this may increase the problem size significantly. To prevent this, one (or more) bigger piece are created out of all these smaller pieces, considering the ULD dimensions. This significantly reduces the model size, while being consistent with operations. This concludes the adaptations to the booking part of the data. Next some adaptation to the flight part of the data are performed. For every flight, the amount of must-fly cargo is forecasted. This forecast outputs a volume in m^3 that is subtracted from the overall available cargo volume. Next, the remaining must-fly volume (must-fly volume forecast minus the must-fly volume from previous bookings), as well as the volume of the bookings are substracted from the capacity. When a positive capacity remains, the heuristic continues. This concludes the data-processing step. A slightly modified version of the heuristic *1D2D Layers* is used. The basis of the heuristic is the same, however when there are no more pre-assigned (by the 1DBPP) pieces to load in the current ULD in the 2DBPP, but there is some space left in the ULD, the heuristic tries to load in the available space the pieces which are offloaded by the 2DBPP (if any) in the previous ULDs. Furthermore, in some cases it might happen that pieces are offloaded, but there are still empty ULDs. In this case the 2DBPP is solved again with the empty ULDs and offloaded pieces only.

BRUNO is run three times with different costs functions. First the shipment contribution (SCb) is maximised, next the volume is maximised and last the commercial priority (CP) is minimised. As expected, the best load factor is achieved when maximising the volume. When the total pieces volume consumes less than 50 % of the flight capacity, the solution does not change much in terms of load factor.

A case-study is performed for a Boeing 747-400 flight with three CTRs and four LDPs. 44 bookings arrived during the booking period for this flight. Every time a booking arrives, BRUNO is run again. BRUNO accepts the booking if the new booking and the previous accepted bookings are loaded. In the end BRUNO accepted 21.24 m^3 while the airline accepted 28.52 m^3. This is a difference of 7.28 m^3. A few recommendations remain to fix the gap between BRUNO and the airline. First of all, it is advised to make a better division between the pieces and the ULDs types. Second it is advised to fill up gaps in the ULDs after the *1D2D Layers* heuristic is solved.

2

Introduction

The air cargo industry is a challenging environment due to the high competition between the stakeholders involved. This demands, as example, high efficiency from cargo airlines. Efficiency can be ensured by designing loading strategies that fully exploit the available cargo volume. Unknowns in the booking dimensions and flight information make this task challenging. For every flight one knows the ULD configuration, but this changes almost every day. Furthermore, each flight has an unknown percentage of must-fly cargo, cargo that may not be offloaded, and an unknown volume loss, which is the unused space in an ULD. An optimisation and forecasting framework is proposed for ULD packing. Goal of the framework is to assist flight analysts with their decision-making, when assessing whether a new booking should be accepted or declined. Unknown booking dimensions and flight details are forecasted using an ensemble learning method. This information is used as input for a bin packing heuristic, which assesses if the booking fits in the aircraft. In addition, we propose a risk index that quantifies what are the chances the proposed loading configuration (per ULD) is not feasible due to a mismatch between predicted and real booking dimensions. This leads to the research question as defined below:

1. How to forecast the unknown characteristics of both the bookings and the flights (RQ1)?
    (a) What are the length, width and height of cargo?
    (b) What is the amount of must-fly cargo per flight?
    (c) What is the volume loss per booking origin?
    (d) What is the accuracy of the flight details?
2. How to palletise the ULDs for a future flight (RQ2)?
3. What is the risk of accepting a booking (RQ3)?

A visual overview of the research question can be found in Figure 2.1. The innovation in this research lies in the combination of (i) forecasting the unknown characteristics of both the pieces and flight and (ii) the definition of a palletisation strategy for the ULDs.

This Master's thesis is part of a case study for Air France KLM Martinair Cargo (AFKLMP) in collaboration with TU Delft, where AFKLMP delivers the data and TU Delft comes up with the model. Although results shown are based on datasets provided by AFKLMP, the methodology can be applied by any airline.

AFKLMP is the specialised air cargo business of the Air France KLM group. It offers a worldwide network with 457 destinations. The network is organised around their two main hubs, Amsterdam Airport Schiphol (AMS) and Paris Charles de Gaulle (CDG). The freight is transported on long-haul flights internationally, and on trucks within Europe. Almost all the freight goes via one of the hubs to its destination, in a classic hub-and-spoke system.

Air cargo can be transported in three types of aircraft. The three types are passenger aircraft, freighters and combis. Each type has its own advantages and disadvantages. Passenger aircraft have the advantage of the passenger network in terms of destinations and frequencies. However, the passenger baggage always has priority. This limits the capacity for freight and makes the available capacity for the cargo uncertain. Freighters

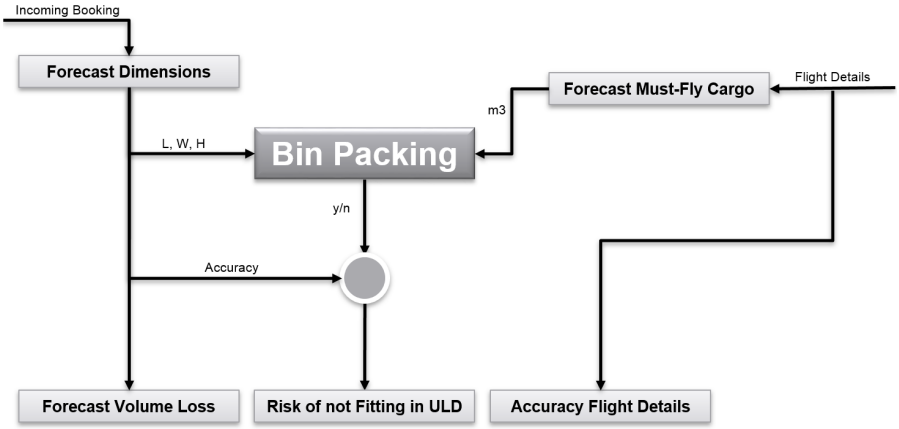


Figure 2.1: A visual overview of the research questions.

Table 2.1: The three types of AFKLMP aircraft used to transport freight and their types of pallets.

	MDP:	**LDP**:	**CTR**:
Passenger Aircraft [−]:		X	X
Freighters [−]:	X	X	X
Combis [−]:	X	X	X
Volume [m^3]:	18	10	4
Length [cm]:	318	318	196
Width [cm]:	224	224	147
Height [cm]:	300	160	157

have the advantage of having a very large capacity, since they are fully dedicated to air cargo transportation. Furthermore, their schedule can be adapted to the customers and some goods may only be transported in freighter aircraft. Combi aircraft are, as the name suggests, a combination of a freighter and a passenger aircraft, so they share both the advantages and disadvantages of the two categories. AFKLMP is the only airline to operate combi aircraft, since these aircraft are old, they will phase out in the near future.

The containers and pallets to load both luggage and freight are called ULD's. These allow a large quantity of cargo to be bundled into a single unit. This saves ground crew time and effort since fewer units needs to be loaded. A container (CTR) is a closed box (not necessarily square/rectangular shaped) and a pallet is just a plate. The pallets can be divided into two groups, the main deck pallets (MDP) and the lower deck pallets (LDP). These two differ in the maximum height that is allowed. The maximum height is larger for MDPs than for LDPs. Next to the pallets there are CTRs for both the lower deck and main deck. The ULD types that fits in each aircraft type and the properties of each ULD type can be found in Table 2.1.

The rest of the thesis is organized as follows. Chapter 3 provides a brief literature review addressing (i) the random forest method, which is used to tackle RQ1, and (ii) the bin packing problem framework, which is the starting point to tackle RQ2. RQ1 is described thoroughly in Chapter 4, together with RQ3 which is based on the outcome of the forecasting process. Chapter 5 describes the different models that could be used to tackle RQ2. Chapter 6 describes how the forecasting and bin packing blocks are coupled in the final model that is the core of the thesis. The final model is validated in Chapter 7. In Chapter 8, conclusions and recommendations are discussed.

3

Literature Review

This Chapter provides a brief literature review that encompasses the two main research topics covered in the thesis, i.e., (i) forecasting methods and (ii) the multi-dimensional bin packing problem. A more elaborate literature study can be found in [12]. First, Section 1 discusses forecasting methods and Section 2 discusses the bin packing problem.

1. Forecasting

This Section discusses different methods to analyse data as well as different forecasting methods. To be able to forecast properties of flights and bookings a large dataset with history data is available. A dataset can be summarised both graphically and numerically. This is discussed first. Next to summarising data, it can be used for forecasting. Which is discussed second. These methods can be used to forecast parameters for both flights and bookings.

Summarising a dataset graphically can be obtained by both a histogram and a scatterplot [4]. A histogram divides the data in intervals (bins) and the area represents the fraction of the data. One should take care while choosing the bin size, since a different bin size results in histograms that visually look very different. When the bin size is too large, information is lost, while a too small bin size will make the plot difficult to interpret. The other graphical representation is a scatterplot. A scatter plot is useful for a bivariate dataset and plots each point with a different variable at both axes. From this, a relationship between two variables can be extracted. To statistically represent the data the sample mean, sample median, sample variance and the median of absolute deviations might be determined. The sample mean and sample median are measures for the center location of the data. To measure the spread within the dataset, the sample variance and mean absolute deviation (MAD) may be used. A combination of graphically and numerically representing the data is a boxplot. The boxplot visually represents the minimum and maximum, the upper and lower quartile (25 % and 75 % empirical percentile) and the median. This gives a clear overview of the spread in the dataset. These methods can be used in a first analysis of the forecasting part.

Next to summarising data, data can be used for forecasting. Forecasting can be performed using a least squares method or using machine learning methods. The least squares method is a linear regression model for slope α and intercept β [11]. These can be obtained by minimising the sum of the squared distance between the real value and the regression. Machine learning methods are more advanced methods and might lead to better results. Examples of machine learning methods are artificial neural networks (ANN) and random forests (RF) [17]. ANNs are based on the neural networks within a human brain, where action is taken by processing input stimuli and forwarding the processed information until action is triggered. ANNs contain neurons, and each neuron contains a local computation or function. The activation function determines the output of the neuron and fires when a certain threshold is reached. The information is not stored at a particular location, but it is distributed over the complete network. This makes ANN, at least partially, a black box, where all internal processes and decisions are not easy to track. However, the sensitivity of the model can be tested. A RF builds multiple decision trees and merges them together to get a more accurate and stable prediction. It can be used for both classification and regression. While growing the trees, the RF adds additional randomness to the model. It does not search for the most important feature, but it searches for the best feature in a random subset of features. This results in a wide diversity, which leads to a better model and a

better forecast.

2. Bin Packing Problem

This Section discusses the bin packing problem (BPP). Although BPPs can be studied for a generic number n of dimensions, in practical problems only the one-, two- and three-dimensional BPPs are investigated. In these three cases, objects to be stored and bins are, respectively, one-, two-, three-dimensional. The one-dimensional bin packing problem (1DBPP) is also called the Knapsack problem (KP). A lot of research has been performed on the 1DBPP. In practical problems, the single dimension is generally volume or weight. Problems with a single knapsack [1, 3] or with multiple knapsacks [7, 14, 18, 21] are solved. Focusing on the specific problem we want to tackle, since one aircraft contains more than one ULD (knapsack), the multiple KP is more suitable to the research question. In addition, some models use stochastic values [1, 18, 21] while others [3, 7, 14] use deterministic values. Differently from [1, 3, 7, 18, 21], [14] uses a particle swarm optimisation to solve the KP. The KP can be solved in our context to obtain preliminary insights regarding how to palletise shipments. As example, it can be used to make sure the shipments assigned to each ULD do not exceed, overall, the available volume of the ULD. Switching to an approach with more dimensions, e.g., the two-dimensional bin packing problem (2DBPP), can provide results closer to reality because the geometry of shipments can be better represented. Models [2, 5, 8, 13] maximise the commercial value. [8] goes further and adds a second objective, which is to balance the load within the bin. [9] tries to minimise the delivery costs. All models can be extended with rotation constraints, however the drawback of this addition is that the computational time increases because of more decision variables and constraints. [2] uses guillotine constraints and [8, 9] use a memetic algorithm to solve the 2DBPP. The 2DBPP can also be used in our work. Since two dimensions are used, one approach is to fix the third dimension, and solve the palletisation from the front side of each ULD. The three-dimensional bin packing problem (3DBPP) uses all the three dimensions to solve the bin packing problem. This is the most realistic variant of this typical problem, however this is computational very demanding. The sum of the commercial value is maximised in [10] while the volume loss is minimised in [16, 19]. [6] maximises the density of the bottom layers and [20] optimises the load balance of the bin. Note that according to the specifics, the assumptions and the goals of each model, decision variables might not be the same. The different bin packing models are solved either exactly, mostly with a branch & bound algorithm or with a heuristic. A heuristic is often used to reduce the computational time.

4

Forecasting

This Chapter discusses how a RF model is used to predict shipment dimensions and flight details. The general framework of a RF model is discussed in Section 1, and it is applied in Sections 2 until 4. Section 2 discusses the dimensions forecast including the volume loss prediction and the risk, then Section 3 forecasts the amount of must-fly cargo and Section 4 determines the accuracy of the flight details.

1. Random Forest

To forecast dimensions, must-fly cargo and the accuracy of the flight details a RF algorithm is used. This is an ensemble learning method used for both classification and regression. It builds different decision trees which come up with a solution all together. A random forest model is made in Python using the scikit-learn package. In this section, the parameter choice that brought to the final model will be described and justified in detail. Important drivers in the parameter choice are computational time, which should be minimised, and accuracy, that should be maximised instead.

A RF can be used for both classification and regression. Since for all the three forecasts, integer values are needed, a regression model is chosen. In forecasting models, part of the dataset is used to train the model, while the remaining part of the dataset is later used to test the trained model. To be consistent with common practice in the machine learning community, the test size is set to 0.25. This means that 75% of the data is used to train the RF while the other 25% is used for testing. A random number generator is used to choose the features to grow a decision tree. The random state seeds the random number generator. 42 is used as random state and is fixed to make sure the same results can be obtained again. Two other variables, that need some tweaking, are the number of decision trees that are grown, and the amount of data used to both train and test the model. Five datasets are used to check the influence of the amount of data. These datasets and their sizes can be found in Table 4.1 and contain booking information. The goal is to predict dimensions. For each dataset the updated date of each booking should be between the date in the first row of this table and 2018/05/31. The second row shows the number of rows in each dataset. It was not possible to test the model for more months, since this led to computer memory errors. For every dataset, three tests were executed, for which the number of decision trees equals 1, 10 and 100. For some datasets, tests were performed with 250 and 500 decision trees. The different tests are compared by their mean error, accuracy and computational time. The calculation of the mean error and accuracy can be found in Equations 4.1 and 4.2 respectively. The influence of the number of decision trees on the mean error and accuracy can be found in Figures 4.1 and 4.2 respectively and the influence of the computation time and the number of decision trees and accuracy can be found in Figures 4.3 and 4.4 respectively. Please note that in each of these figures, each colored line represents a different dataset as in Table 4.1 and each data point represents a different number of decision trees.

Table 4.1: Sizes of the different datasets used for testing the variables.

Minimum Update Date Data:	2018/01/01	2018/02/01	2018/03/01	2018/04/01	2018/05/01
Number of Rows:	632,825	508,079	389,464	251,008	124,301

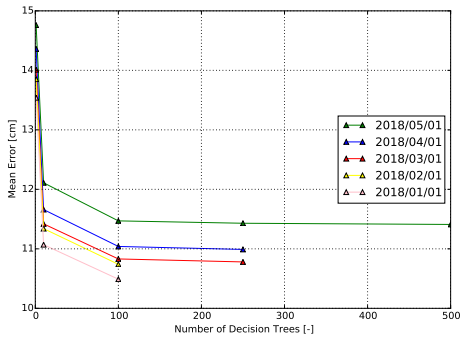


Figure 4.1: Analysis of the number of decision trees vs the mean error.

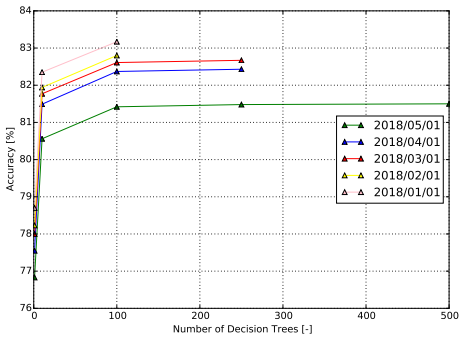


Figure 4.2: Analysis of the number of decision trees vs the accuracy.

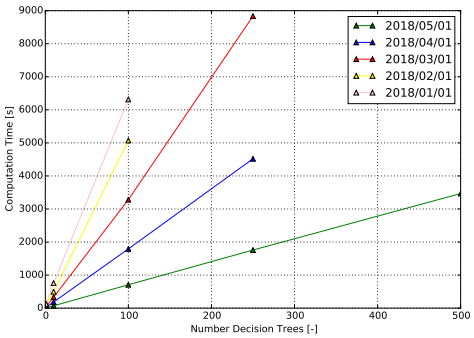


Figure 4.3: Analysis of the number of decision trees vs the computational time.

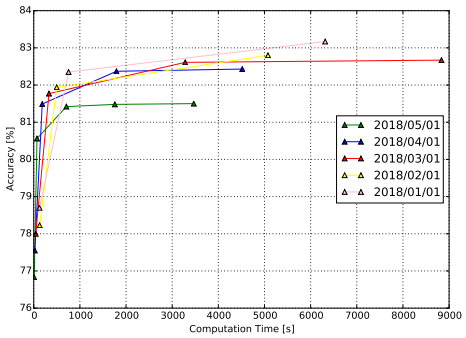


Figure 4.4: Analysis of the number of computational time vs the accuracy.

$$\text{meanError} = \text{abs}(\text{predictedDimension} - \text{realDimension}) \tag{4.1}$$

$$\text{accuracy} = \text{avg}(100 - \text{mean}(100 \cdot \text{meanError}/\text{realValue})) \tag{4.2}$$

As can be concluded from Figures 4.1 and 4.2 a better mean error and accuracy is obtained with more data for the same number of decision trees. Furthermore, the mean error and accuracy keep improving up until 100 decision trees. An increase of decision trees would only cause an increase in the computational time, without an actual accuracy improvement. Figure 4.4 shows that a better accuracy demands more computation time. Furthermore, dataset 2018/01/01 with ten decision trees results in a better accuracy compared to dataset 2018/05/01 with 100 decision trees within almost the same computation time. From Figure 4.3 it can be concluded that more decision trees lead to a longer computation time.

Another variable for the RF is the quality criterion, this can be either mean squared error (MSE) or mean absolute error (MAE). All previous tests were performed with the MSE quality criterion. Next to these tests, dataset 2018/05/01 was tested for the MAE quality criterion with one decision tree, however, this computation did not finish within 30 minutes, while a solution was reached with the MSE within ten seconds. So the MAE computation was stopped and deemed as computationally demanding.

The final parameters used for the RF models in the following Sections are summarised in Table 4.2.

2. Dimensions Forecast

In the air cargo supply chain, dimensions of packages to be loaded onto aircraft are generally not known a priori. This translates into an issue for the flight planning, which motivates the need for a forecasting model that estimates dimension of pieces. While forecasting these, an overall error of 15 cm is seen as acceptable by the airline.

Table 4.2: Final parameters for the RF model.

Parameter:	**Value**:
Test Size:	0.25
Number Decision Trees:	100
Quality Criterion:	MSE
Random State:	42

Table 4.3: The most important columns in the booking data and their description.

Column:	**Description:**
AgentURN	The unique reference number of the customer of the booking
BookingDestination	Destination of the booking.
BookingOrigin	Origin of the booking.
CommodityCode	Code for the type of product.
PieceQuantity	The amount of pieces with this properties.
PieceVolume	The volume of each piece.
ProductCode	The code of the product offered by AFKLMP.
ShipmentPieces	The amount of pieces in the shipment.
ShipmentVolume	The volume of the shipment.
ShipmentWeight	The weight of the shipment.

In this Section the RF model is discussed in Section 2.1. This information is used to find both the volume loss and the risk. These are discussed in Sections 2.2 and 2.3 respectively.

2.1. Forecast

This Section discusses the RF model for predicting the dimensions. First data is discussed, next the features are determined and finally some conclusions are inferred and critically discussed.

The booking data consists of 2,252,037 rows and 37 columns. Every data row represents several pieces with the same properties. The columns represent the properties of the pieces and every column is called a feature. The most important columns can be found in Table 4.3. It may be noted that in this table the column *PieceWeight* is missing. This column is present in the database, however it misses values for most pieces. Therefore, it is left out for this analysis. Furthermore, every booking has different versions, every time something is changed to the booking a new version ("data row") is created. The dataset only contains the last version of the bookings which are updated between 2018/01/01 and 2018/05/31. The data is available from 2016/01/01 onward, however this datafile is too large to be handled in Python. Therefore, it is chosen to use the data from 2018/01/01 onwards. The end date 2018/05/31 was chosen because it is the end of the last month before this project started. Therefore, no unknowns exist of the bookings.

The influence of the different columns used as input for the RF can be checked next. The results of this test with one decision tree and a start date of 2018/05/01 can be found in Table 4.4. The first column in this table present the features used to perform the tests in Section 1, which is considered the baseline for this test. As can be concluded from this table the *BookingOrigin* is by far the most important column. When keeping this column and the *BookingDestination*, the accuracy drops to 32.39 %, so more columns are needed for an accurate solution. The most accurate solution is found in the fifth test, which is an improvement of 1.63 % compared to the baseline. This equals to a reduction in mean error of 1.16 $[cm]$.

The final features which results in the best solution for the RF can be found in Table 4.5. The algorithm is run with these selected variables which resulted in a mean error of 9.89 cm and an accuracy of 84.29 % within 06:22:23. The mean error and accuracy are computed using Equations 4.1 and4.2 respectively. This is an improvement of 0.6 cm or 1.12 % compared to the previous best run. So, combining the best variables leads to a significant better result. Finally a boxplot of the mean error is given in Figure 4.5. From this figure it can be concluded that the lower quartile of the errors is at 0.58 cm and the upper quartile is at 10.34 cm. This means that 75 % of the forecasted dimension has an error lower than 10.34 cm. Since an error of 15 cm is acceptable, the model performs reasonably well. However, 17 % still has an error larger than 15 cm.

Table 4.4: Results for the RF using different columns as input to train the data with a start date of 2018/05/01 and growing one decision tree.

Data Column:	**Importances**:						
	Test 1:	*Test 2*:	*Test 3*:	*Test 4*:	*Test 5*:	*Test 6*:	*Test 7*:
AgentURN	0.0	0.02	0.0	0.0	-	-	0.0
BookingDestination	-	0.03	-	0.05	0.0	0.0	0.04
BookingOrigin	0.69	0.66	0.73	0.68	0.0	0.0	0.68
CommodityCode	0.06	0.03	0.0	0.0	-	0.0	0.0
PieceQuantity	-	0.0	-	-	-	-	-
PieceVolume	0.05	0.01	-	0.0	-	-	0.0
ProductCode	0.06	0.03	0.09	0.0	-	0.0	0.04
ShipmentPieces	-	0.0	-	-	-	-	-
ShipmentVolume	0.0	0.0	0.0	-	-	-	0.0
ShipmentWeight	0.0	0.0	-	-	-	-	-
Mean Error [cm]	14.76	14.53	14.13	13.6	30.63	27.43	14.03
Accuracy [%]	76.83	76.9	77.53	78.46	32.39	43.56	77.7

Table 4.5: Final variables used for the RF.

Test Size:	0.25
Number of Decision Trees:	100
Quality Criterion:	MSE
Minimum Update Date:	2018/01/01
Maximum Update Date:	2018/05/31
Random State:	42
Features:	*AgentURN, BookingDestination, BookingOrigin, CommodityCode, PieceVolume, ProductCode*
Mean Error [cm]:	9.89
Accuracy [%]:	84.29
Computational Time:	06:22:28

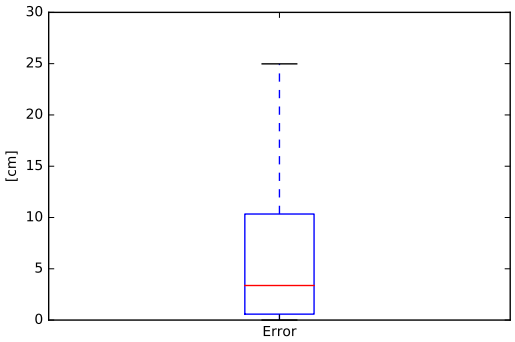


Figure 4.5: Boxplot of the mean error distribution.

2.2. Volume Loss

Volume loss is the unused space in an ULD. Airliners use this factor to increase the volume of pieces with unknown dimensions. This increase in size is a safety factor to estimate the shipment dimensions, to guarantee shipments will fit in the ULD. At the moment 10, 15 or 30 % is used, depending on the flight. The one they pick is based on experience of the flight analyst. It would be useful to have a better estimate of the volume loss. This better estimate is discussed in this Section. Although this is something we do no explicitely consider in the thesis, it could be useful for better decision-making if the strategy of using volume-loss was kept.

Section 2.1 discussed the dimensions forecast model. This model forecasts dimensions with an accuracy of 84.29 %. Since the accuracy of the forecast is known, it would be strange to use something else as volume loss than the accuracy of your dimension prediction. Since volume loss is used to increase the volume of a piece, it is more accurate to increase this using the accuracy of the dimensions forecast. The accuracy of the dimensions model results in a volume loss of $100 - 84.29 = 15.71 \approx 16$ %. So the volume is increased with a factor of 1.16. From Table 4.4 it can be concluded that feature *BookingOrigin* is by far the most important. Therefore, the volume loss can be more accurate when determining it per booking origin. In total there are 292 booking origins, however only the booking origins with more than 100 bookings are considered. This results in 137 different volume losses, 136 for specific booking origins and a general one (16 %) for bookings with a different booking origin. The volume losses per booking origin can be found in Table 4.6. As can be concluded from this table, volume loss ranges from 7 %, for GUA, to 30 %, for ORD.

Table 4.6: The volume loss [%] per booking origin.

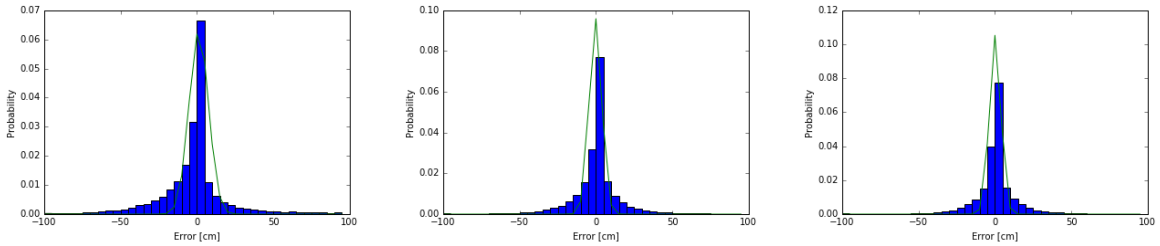
ABZ	25	BSL	9	FLR	25	IST	26	MIA	29	PEK	17	TLV	10
AMS	16	BUD	12	FMO	15	JFK	20	MMX	19	PEN	12	TPE	13
ARN	21	CAI	20	FRA	16	JNB	23	MNL	12	PER	26	TRN	23
ATH	15	CDG	16	GDL	12	KIX	16	MRS	16	PRG	13	TUN	12
ATL	21	CGK	17	GIG	13	KUL	15	MST	11	PTY	18	TXL	12
AUH	25	CGN	18	GLA	17	KWI	19	MTY	18	PVG	15	VCE	21
BAH	21	CMN	13	GOA	17	LAX	18	MUC	17	RTM	16	VCP	27
BCN	16	CPH	13	GOT	18	LCJ	10	MXP	21	RUH	16	VIE	18
BER	15	CPT	25	GRU	17	LHR	19	NBO	18	SCL	18	VRN	28
BEY	27	CTU	17	GRZ	19	LIL	15	NCE	15	SFO	17	WAW	15
BIO	20	DEL	22	GUA	7	LIM	8	NGO	10	SIN	14	YEG	20
BKK	13	DFW	15	GVA	12	LIS	22	NOU	19	SJO	7	YUL	22
BLL	16	DMM	21	HAJ	12	LNZ	17	NRT	19	SJU	16	YVR	21
BLR	20	DUB	16	HAM	18	LOS	17	NTE	17	SOF	18	YYC	23
BOD	17	DUS	14	HEL	12	LYS	17	NUE	16	STR	16	YYZ	22
BOG	9	DWC	20	HOH	13	MAA	18	OPO	25	SVG	21	ZRH	18
BOM	17	DXB	25	HYD	17	MAD	20	ORD	30	SVO	14		
BQY	22	EIN	15	IAD	19	MAN	18	ORY	11	SWK	22		
BRE	16	EZE	17	IAH	25	MCT	18	OSL	19	SXB	18		
BRU	15	FCO	17	ICN	17	MEX	17	OTP	16	TLS	20		

2.3. Risk

Forecasted shipment dimensions, Section 2.1, come with an error. This leads to a certain risk, since packages might be larger than predicted and lead to loading strategies that do not fit the ULD dimensions. This Section discusses an approach to find the risk associated to the forecasted dimensions and the position in an ULD.

To find the risk, a distribution of the dimensions and their probability is needed. The test dataset contains the errors of the prediction for 142,493 pieces. From all these errors, a probability distribution can be found. The errors of the test dataset are plotted as blue bars in Figures 4.6a until 4.6c. From these figures it can be concluded that for all dimensions the distribution of errors can be approximated with a normal distribution. A normal distribution for the corresponding mean and standard deviation (StD) is plotted in the same figures with a green line. As concluded in Section 2.2 booking origin plays a crucial role. Therefore, every booking origin with more than 100 datapoint has its own normal distribution. The mean and StD for every booking origin can be found in Appendix A.

Since the probability distribution is known for every piece's dimension, the next step is to translate this to a risk for a complete ULD. Figure 4.7 shows an ULD with four pieces. Let's assume that the dimensions for all



(a) Length. (b) Height. (c) Width.

Figure 4.6: Distribution for all booking origin per dimension, where the blue bins represents the data and the green line represent the corresponding normal distribution.

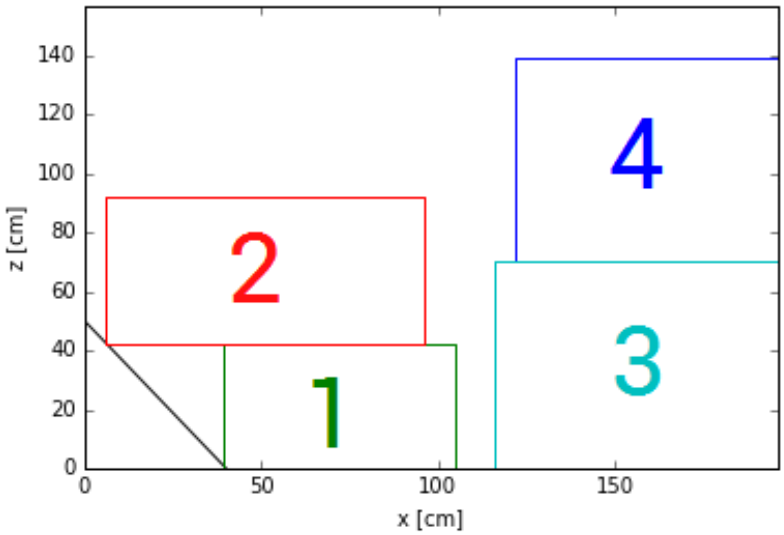


Figure 4.7: An ULD with four loaded pieces.

pieces are predicted and have a normal distribution that is booking origin-specific. In the length (x) direction, pieces one and three are limiting while in the height (z) direction, pieces three and four are limiting. When focusing on the length, the gap between pieces one and three is 10 cm. So the sum of both piece's length might be underestimated by a maximum error of 10 cm before leading to an infeasible loading configuration. To determine what is the risk that such scenario will happen, the normal distributions for both piece's lengths are summed. This results in a new distribution which holds for both pieces when considered simultaneously. The area below this distribution for an error which is greater than 10 cm can be used as an estimation of the risk for this ULD-shipment configuration in length direction. Being the overall area of the probability density function unitary, the area exceeding the limit value (10 cm in this case) multiplied by 100 provides an estimate of the risk the two lengths will exceed the length of the ULD. The same procedure can be repeated for the other two dimensions. Finally, the maximum value of the three risks (one for each dimension) is taken as the risk for a complete ULD. This is an assumption of the model, since the goal is to provide a simple and intuitive estimate of the risk. A more complicated approach might be chosen to provide a better estimate of such a risk.

3. Must-Fly Cargo

Must-fly cargo cannot be offloaded and therefore it is useful for an airline to forecast how much must-fly cargo arrives before every flight. This will be forecasted using a RF. All general information about RFs is given in Section 1. This Section will focus on the RF model for must-fly cargo only.

Table 4.7 shows the importances for different features. The feature combination of test 3, i.e. *Aircaft-*

Table 4.7: Importances of the different columns.

Column:	**Importance**:			
Test Number:	1	2	3	4
AircraftSubType	0.12	0.02	0.44	0.02
Arrival	0.44	0.01	0.18	-
CapacityVolume	-	-	0.01	-
CapacityWeight	-	-	0.0	-
FlightNumber	0.09	0.16	0.09	0.38
MustFlyWeightPreviousFlight	0.01	-	-	-
MustFlyVolumePreviousFlight	0.0	-	-	-
WeekDay	0.06	0.16	0.08	0.17
Mean Error Total Weight:	2905	2830	2742	2876
Mean Error Must-Fly Weight:	2827	2770	2820	2825
Mean Error Total Volume:	12.02	11.61	11.3	11.76
Mean Error Must-Fly Volume:	12.77	12.43	12.54	12.72

Table 4.8: Final variables and errors for the random forest when the parameters are optimised.

Test Size:	0.25
Number of Decision Trees:	100
Quality Criterion:	MSE
Minimum Flight Date:	2017/05/01
Maximum Flight Date:	2018/05/31
Random State:	42
Mean Error Total Weight [kg]:	2391
Mean Error Must-Fly Weight [kg]:	2382
Mean Error Total Volume [m^3]:	10.78
Mean Error Must-Fly Volume [m^3]:	11.02
Computational Time:	0:00:04

SubType, *Arrival*, *CapacityVolume*, *CapacityWeight*, *FlightNumber* and *WeekDay*, lead to the lowest error and highest accuracy. Combining these features with the parameters as found in Section 1 and 13 months of flight data result in the results in Table 4.8. From this table it can be concluded that both the total weight and must-fly weight can be forecasted with a mean error lower than 2400 kg and both volumes can be estimated with a mean error of about 11 m^3. The error range can be found in Figures 4.8 and 4.9. It can be noted that the median is lower than the mean value of the error. This means that the larger part of the dataset has acceptable errors. Furthermore, it can be concluded that the weight error ranges between 0-7800 kg while the volume error ranges between 0-35 m^3. The error is specified per aircraft sub-type in Tables 4.9 and 4.10 for weight and volume respectively. The accuracies in this table are determined with Equation 4.3. From this tables it can be concluded that the accuracies per aircraft type are in the range of 80-90 %.

$$\text{accuracy} = 100 - 100 \cdot \frac{\text{error}}{\text{capacity}} \tag{4.3}$$

4. Flight Details Accuracy

For every flight the cargo capacity, both in weight and volume, is forecasted. This is relevant since capacity is sold. An accurate estimate prevents the following two cases from happening. (i) Capacity is oversold, which results in pieces being offloaded and thus a lower customer satisfaction. (ii) Capacity is undersold, which results in a lower revenue. This forecast is updated every day, when more bookings and information are available. For the specific case of AFKLMP, this process starts seventeen days before departure until one day before departure. Discrepancies exist between the forecasted capacity and the capacity at the moment of departure. To determine if a booking fits in the ULD or not, it could be useful to know the accuracy of the capacity forecast. In this Section a method is shown to determine the accuracy of the capacity forecast for every day before the departure.

To determine the accuracy of the capacity forecast a RF model will be used with the parameters as deter-

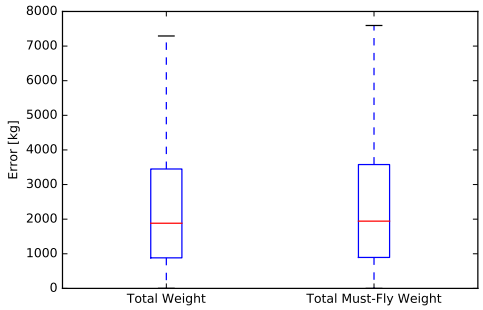


Figure 4.8: Forecasted weight and must-fly weight error distribution.

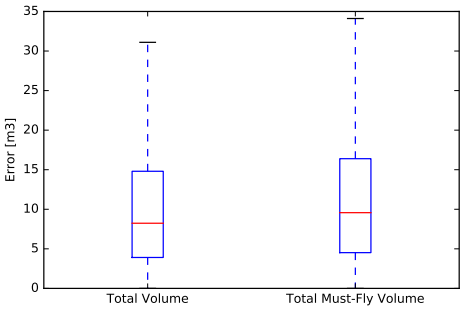


Figure 4.9: Forecasted volume and must-fly volume error distribution.

Table 4.9: The mean errors and accuracy of the random forest weight predictions per aircraft sub-type.

Aircraft	**Total Weight**		**Total Must-Fly Weight**		**Aircraft**
Sub-Type:	**Error** [kg]:	**Accuracy** [%]:	**Error** [kg]:	**Accuracy** [%]:	**Sub-Type**:
332	2150	90	1948	90	332
333	2224	82	2278	82	333
359	3917	81	4357	79	359
744	1759	90	1670	91	744
763	1662	80	1656	80	763
764	2587	84	2550	84	764
772	2547	87	2594	87	772
77W	2901	88	2993	87	77W
788	2380	86	2693	84	788
789	3047	87	3221	87	789

Table 4.10: The mean errors and accuracy of the random forest volume predictions per aircraft sub-type.

Aircraft	**Total Volume**		**Total Must-Fly Volume**		**Aircraft**
Sub-Type:	**Error** [m^3]:	**Accuracy** [%]:	**Error** [m^3]:	**Accuracy** [%]:	**Sub-Type**:
332	9.28	85	8.67	86	332
333	10.97	85	11.94	84	333
359	16.33	82	19.62	79	359
744	7.87	88	9.05	86	744
763	8.79	82	9.16	81	763
764	15.11	78	14.68	78	764
772	9.11	88	10.72	86	772
77W	10.77	90	12.40	89	77W
788	9.92	84	12.56	79	788
789	11.06	88	13.25	85	789

Table 4.11: The influence of a different combination of features on the random forest results.

Data Column:	**Importances**:				
	Test 1:	*Test 2*:	*Test 3*:	*Test 4*:	*Test 5*:
AircraftSubType	0.39	0.01	0.35	0.08	0.42
Arrival	0.15	0.32	0.13	0.3	0.15
CapacityVolume	0.0	-	0.0	-	-
CapacityWeight	0.0	-	0.04	-	0.0
DayBeforeDeparture	0.14	0.0	0.14	0.0	0.05
FlightNumber	0.09	0.16	0.09	0.14	0.09
Month	-	-	0.0	0.01	0.0
Weekday	0.12	0.16	0.12	0.17	0.13
Mean Error Weight [%]	3.77	6.24	3.26	6.37	3.68
Accuracy Weight [%]	95.26	92.17	95.86	92.03	95.33
Mean Error Volume [%]	2.64	4.77	2.28	4.70	2.96
Accuracy Volume [%]	96.91	94.42	97.33	94.52	96.51

Table 4.12: Final variables used for the random forest.

Test Size:	0.25
Number of Decision Trees:	100
Quality Criterion:	MSE
Minimum Flight Date:	2017/05/01
Maximum Flight Date:	2018/05/31
Random State:	42
Features:	*AircraftSubType*, *Arrival*, *CapacityVolume*, *CapacityWeight*, *DayBeforeDeparture*, *FlightNumber*, *Month*, *Weekday*
Mean Error Weight [%]:	6.28
Accuracy Weight [%]:	92.27
Mean Error Volume [%]:	4.59
Accuracy Volume [%]:	94.70
Computational Time:	00:06:34

mined in Section 1. For every flight the accuracy of both the weight and volume capacity are determined with Equation 4.3. The error in this Equation is the absolute difference between forecasted capacity and the actual capacity at the moment of departure.

Different combinations of features are tested and results can be found in Table 4.11. It can be concluded that test 3 leads to the lowest error and highest accuracy of the RF model. Feature *AircraftSubType* is the most important and features *Arrival*, *DayBeforeDeparture* and *Weekday* are important as well. Although the feature *CapacityVolume* has a low weight, its removal (as seen in test 5) increases the error. Consequently, the feature was not discarded.

The result of the final run with the selected parameters can be found in Table 4.12. The first conclusion that can be inferred from this Table is that volume capacity is predicted more accurately than weight capacity. This is justified by the fact that both have a different spread in historic data. This comes from the fact that both capacities are determined differently. Each flight has a certain number of ULD positions, which results in the capacity volume (ULD volume times number of positions). The capacity weight is forecasted by a tool, which predicts the capacity weight based on historic data. This leads to the difference in accuracies of both. Both have an accuracy higher than 95 %, which makes this model accurate. This can also be concluded from Table 4.13 and Figures 4.10 and 4.11. The table shows the mean errors and accuracies for every day before the departure, while the figures show the spread of errors. It can be concluded that the model has the largest error spread seventeen days before departure for capacity weight and fifteen days before departure for capacity volume. In general, performances increase as the departure date gets closer. This results in the day before departure to be the most accurate, which is consistent with intuition.

Table 4.13: Final variables used for the random forest.

	Weight		**Volume**		
DayBeforeDeparture:	*Mean Error [%]*:	*Accuracy [%]*:	*Mean Error [%]*:	*Accuracy [%]*:	*DayBeforeDeparture*:
1	2.75	96.85	1.15	98.72	1
2	5.52	93.50	3.74	95.84	2
3	5.67	93.25	3.93	95.63	3
4	5.85	92.93	4.09	95.42	4
5	6.12	92.64	4.35	95.09	5
6	6.40	92.16	4.62	94.81	6
7	6.32	92.31	4.74	94.65	7
8	6.38	92.29	4.81	94.45	8
9	6.51	92.12	5.01	94.30	9
10	6.52	91.97	4.91	94.45	10
11	6.59	91.90	5.06	94.32	11
12	6.83	91.63	5.26	93.99	12
13	6.80	91.65	5.28	93.97	13
14	6.97	91.44	5.28	94.00	14
15	7.13	91.17	5.58	93.49	15
16	7.16	90.54	5.39	92.56	16
17	7.56	89.79	5.11	93.66	17

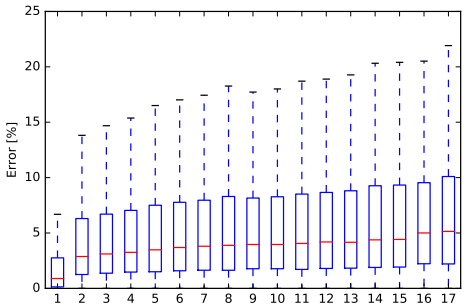


Figure 4.10: Distribution of the error for the accuracy of the weight capacity.

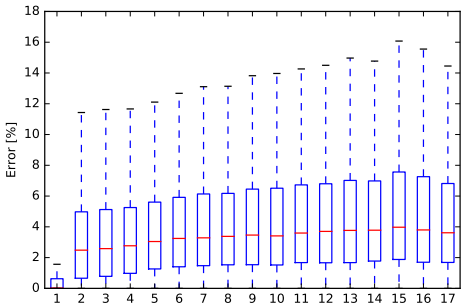


Figure 4.11: Distribution of the error for the accuracy of the volume capacity.

5

Bin Packing Models

This Chapter discusses different models addressing the bin packing problem. First the one-dimensional bin packing problem is discussed in Section 1, next the two-dimensional bin packing problem is discussed in Section 2 and Section 3 discusses different heuristics.

1. One-Dimensional Model

This section discusses the variables and formulation for the one-dimensional bin packing model (1DBPP). Generally speaking, for problems aiming at determining optimal loading configurations, the 1DBPP focuses on volume or weight, rather than a physical dimension (e.g., length) of objects to be loaded. In our context, one-dimensional means that the volume of the pieces is used to assign each piece to an ULD, without determining the exact position in the ULD. A check is added for every piece to make sure its dimensions do not exceed the ULD dimensions. In other words, and additional check is performed to ensure that each piece can physically fit into the assigned ULD when considered singularly.

Let $\mathbf{P} = (p_1, p_2, ..., p_n)$ be the set of n_p pieces considered. The set of pieces is further divided into (i) generic pieces $\mathbf{P}_g$, (ii) perishable pieces $\mathbf{P}_p$, (iii) radioactive pieces $\mathbf{P}_r$ and (iv) cars $\mathbf{P}_c$ such that $\mathbf{P} = \mathbf{P}_g \cup \mathbf{P}_p \cup \mathbf{P}_r \cup \mathbf{P}_c$. While generic pieces do not have special restrictions, a perishable and radioactive piece cannot be loaded into the same ULD, while a car must be the only piece loaded into a ULD. In addition, a subset of $\mathbf{P}$ can be flagged as must-fly. This subset is defined as $\mathbf{P}_s$. Furthermore, each piece is characterised by a commercial priority CP_i. Let $\mathbf{U} = (u_1, u_2, ..., u_{n_u})$ be a set of n_u ULDs. The decision variables are defined as follows:

p_{ij} (binary)	1 if piece i is in ULD j, 0 otherwise.
z_i (binary)	1 if piece i is not loaded at all, 0 otherwise.
r_{iab} (binary)	1 if side a of piece i is in side b of the ULD, 0 otherwise.

Decision variable p_{ij} rewards the 1DBPP for loading pieces while z_i penalises offloaded pieces. r_{iab} is added to give the model the opportunity to rotate a piece, which might result in a piece that fits with a specific orientation only. The objective function is defined as:

$$\min J = \sum_{i \in \mathbf{P}} \sum_{j \in \mathbf{U}} CP_i \cdot p_{ij} + [4 \cdot \max(CP) - CP_i] \cdot z_i \tag{5.1}$$

The main objective is to minimise the sum of CP_i. A lower CP_i results in a higher priority, so the cost function as stated results in pieces with lower CP_i values being weighted more. Since the objective is minimised and offloading pieces is not preferred, an offloaded piece results in a penalty of $4 \cdot \max(CP) - CP_i$. This penalty makes sure that a lower CP_i results in a higher penalty and that the penalty is always higher than the CP_i. The constraints are defined as follow:

$$\sum_{j \in \mathbf{U}} p_{ij} + z_i = 1 \forall\, i \in \mathbf{P} \tag{5.2}$$

$$\sum_{i \in \mathbf{P}} V_{\mathbf{P}} \cdot p_{ij} \leq V_{\mathbf{U}} \forall\, j \in \mathbf{U} \tag{5.3}$$

$$p_{ij} + p_{kj} \leq 1 \forall i \in \mathbf{P}_p, \forall k \in \mathbf{P}_r, \forall j \in \mathbf{U} \tag{5.4}$$

$$p_{ij} + p_{kj} \leq 1 \forall i \in \mathbf{P}_c, \forall k \in \mathbf{P} \backslash i, \forall j \in \mathbf{U} \tag{5.5}$$

$$-M \cdot z_i + l_p \cdot r_{i11} + h_p \cdot r_{i12} + w_p \cdot r_{i13} \leq \sum_{j \in \mathbf{U}} L_j \cdot p_{ij} \forall i \in \mathbf{P} \tag{5.6}$$

$$-M \cdot z_i + l_p \cdot r_{i11} + h_p \cdot r_{i12} + w_p \cdot r_{i13} \leq \sum_{j \in \mathbf{U}} H_j \cdot p_{ij} \forall i \in \mathbf{P} \tag{5.7}$$

$$-M \cdot z_i + l_p \cdot r_{i11} + h_p \cdot r_{i12} + w_p \cdot r_{i13} \leq \sum_{j \in \mathbf{U}} W_j \cdot p_{ij} \forall i \in \mathbf{P} \tag{5.8}$$

$$r_{ik1} + r_{ik2} + r_{ik3} = 1 \forall i \in \mathbf{N}, \forall k \in [1,2,3] \tag{5.9}$$

$$r_{i1k} + r_{i2k} + r_{i3k} = 1 \forall i \in \mathbf{N}, \forall k \in [1,2,3] \tag{5.10}$$

Equations 5.2 until 5.8 present the different constraints. The first constraint, Equation 5.2 makes sure each piece is either loaded or not. Furthermore it makes sure that when a piece is loaded, it is loaded in one ULD only. Equation 5.3 prevents the total volume of the pieces in the ULD exceeding the maximum volume capacity of the ULD. The third constraint, Equation 5.4 prevents perishable and radioactive piece ending up in the same ULD. Equation 5.5, makes sure every car has its own ULD. The three Equations 5.6 until 5.8, make sure the piece fits physically in the ULD, so the dimensions of the piece do not exceed the dimensions of the ULD. Equation 5.6 checks the length of the ULD is not exceeded, Equation 5.7 does the same for the height and Equation 5.8 checks if the width is not exceeded. This also considers the possible rotations of the pieces. Equation 5.9 makes sure each piece side is assigned to one ULD side only and Equation 5.10 makes sure each ULD side is assigned to one piece side only. These two constraints prevent unfeasible rotations. The final results of this model for different flights can be found in Table 5.1.

Table 5.1: Results for the one-dimensional BPP.

Flight Details			**1DBPP**				
					Load Factor		
Nbr:	*ULDs*:	*Pieces*:	*Computational Time*:	*Offloaded*:	*Total*:	*General*:	*Pieces*:
001	14	36	00:00:01	4	62	67	82
002	12	48	00:00:00	0	52	62	100
003	12	46	00:00:00	3	66	72	81
004	9	23	00:00:00	0	72	72	100
005	13	41	00:00:00	15	54	58	46
006	9	60	00:00:01	1	94	94	95
007	10	52	00:00:00	0	85	94	100
008	11	41	00:00:00	17	51	70	46
009	8	26	00:00:00	1	65	74	97
010	13	25	00:00:00	0	46	67	100
011	14	32	00:00:00	3	34	60	71
012	11	67	00:00:00	4	98	98	91
013	13	75	00:00:07	3	76	82	92
014	12	48	00:00:01	0	53	64	100
015	13	106	00:00:00	14	85	85	70
016	14	22	00:00:00	0	36	73	100
017	11	24	00:00:00	0	42	57	100
018	12	74	00:00:00	1	59	59	73
019	12	79	00:00:00	5	88	88	80

Table 5.1 shows the results for the 1DBPP for different flights. The gap optimality for all these solutions is 0 %. This means that the optimal solution for all flights has been found. When looking to the computational

time, for some flights this one equals 00:00:00. The computational time is rounded to seconds, so a computational time of 00:00:00 means that the computational time took less than half a second. Three load factors are displayed in Table 5.1, namely total, general and pieces. The total load factor is the volume of the loaded pieces divided by the volume capacity, the general load factor is the volume of the loaded piece divided by the volume of the used ULDs and the pieces load factor is the volume of the loaded pieces divided by the loaded plus the offloaded pieces. As can be concluded by this table is that the 1DBPP produces quick results which results in good load factors.

2. Two-Dimensional Model

This section discusses the model for the two-dimensional bin packing problem (2DBPP). This builds further on the model discussed in Section 1, which discussed the 1DBPP. Where the 1DBPP only checks if the volume of the ULD is not exceeded, the 2DBPP tries to find a position for every piece in a two-dimensional space. This space consists of the length and height of the ULD, i.e., each ULD is seen from the front side. While finding this position, the pieces should not overlap, and the solution should be vertically stable. This leads to different variables and a substantial increase in the number of constraints. This Section is split into three Subsections. First the different parameters, sets and decision variables are discussed in Section 2.1, next Section 2.3 discusses the constraints and the results are discussed in Section 2.4. The model discussed, is based on the formulation of [15, 16].

2.1. Parameters, Sets and Decision Variables

This Section discusses the parameters, sets and decision variables used in the 2DBPP. The 2DBPP uses the following parameters:

a_c	Horizontal length of ULD cut c.
b_c	Angle of ULD cut c.
CP_i	Commercial priority of piece i.
D_m	Maximum length, height or width of all pieces. Used as *big M*.
f_i	Indicates if piece i is fragile or not.
l_i, h_i, w_i	Length, height and width of piece i.
L_j, H_j, W_j	Length, height and width of ULD j.
N	Total number of pieces.
M	Total number of ULDs.
r_i	Indicates if piece i may be turned upside down or not.
$SCBi$	Shipment contribution of piece i.
V_i	Volume of piece i.
V_j	Volume of ULD j.

The following sets are used:

P	All pieces.
P_c	All cars.
P_g	All generic pieces.
P_p	All perishable pieces.
P_r	All radioactive pieces.
P_s	All must-fly pieces.
U	All ULDs.

Let $\mathbf{P} = (p_1, p_2, ..., p_n)$ be the set of n_p pieces considered. The set of pieces is further divided into (i) generic pieces $\mathbf{P}_g$, (ii) perishable pieces $\mathbf{P}_p$, (iii) radioactive pieces $\mathbf{P}_r$ and (iv) cars $\mathbf{P}_c$ such that $\mathbf{P} = \mathbf{P}_g \cup \mathbf{P}_p \cup \mathbf{P}_r \cup \mathbf{P}_c$. The decision variables can be found below:

q_i (binary)	1 when piece i is not loaded in an ULD, 0 otherwise.
p_{ij} (binary)	1 when piece i is loaded in ULD j, 0 otherwise.
x_i, z_i, y_i (integer)	horizontal, vertical and diagonal position of the lower left corner of piece i.
x_{ik}, z_{ik} (binary)	1 when piece i is to the right of and respectively above piece k.
r_{iab} (binary)	1 when side a of piece i is alongside b of the ULD.
g_i (binary)	1 when piece i lies on the ground.
h_{ik} (binary)	0 when piece k has the suitable height to support box i, 0 otherwise.
o_{ik} (binary)	0 when there's a non-empty intersection between piece i and k, 0 otherwise.
s_{ik} (binary)	1 when piece k supports piece i and are in the same ULD, 0 otherwise.
η^1_{ik} (binary)	0 when $x_k \leq x_i$, 1 otherwise.
η^2_{ik} (binary)	0 when $x'_k \geq x'_i$, 1 otherwise.
β^l_{ik} (binary)	1 when vertex l of piece i is supported by piece k.
γ^c_i (binary)	1 when piece i lays on cut c of the ULD in which it lies.

2.2. Objective Function

This Section discusses the objective function, which can be found in Equation 5.11. The main goal of the model is to minimise the CP. This is done in the first term of the Equation. The second term penalises offloaded pieces. This is the same objective function as in the 1DBPP. Therefore a more elaborate explanation can be found in Section 1.

$$\max J = \sum_{i \in \mathbf{P}} \sum_{j \in \mathbf{U}} CP_i \cdot p_{ij} + [4 \cdot \max(CP) - CP_i] \cdot z_i \tag{5.11}$$

2.3. Constraints

This section discusses the different constraints used for the 2DBPP in the paragraphs below. They are divided into categories to provide a better picture of the problem. Some constraints are model-specific. Some others depend on the shape/requirements of shipments, or on the shape of the ULDs.

General Constraints : Equation 5.12 is the first constraint and this one makes sure that each piece is either loaded or offloaded adn when it is loaded, that it is loaded in one ULD only. Next Equation 5.13 prevents that anything is stacked on top of a fragile piece.

$$q_i + \sum_{j \in \mathbf{U}} p_{ij} = 1 \forall i \in \mathbf{P} \tag{5.12}$$

$$s_{ki} \leq f_i(N-1) \forall i \in \mathbf{P}, \forall k \in \mathbf{P} \tag{5.13}$$

No-Mix Constraints : Equations 5.14 and 5.15 are the no-mix constraint, where the first one prevents perishable and radioactive pieces being loaded in the same ULD and the second one makes sure that a car has its own ULD.

$$p_{ij} + p_{kj} \leq 1 \forall i \in \mathbf{P}_p, \forall k \in \mathbf{P}_r, \forall j \in \mathbf{U} \tag{5.14}$$

$$p_{ij} + p_{kj} \leq 1 \forall i \in \mathbf{P}_c, \forall k \in \mathbf{P} \backslash i, \forall j \in \mathbf{U} \tag{5.15}$$

Rotation Constraints : The constraints in Equations 5.16 until 5.18 make it possible to rotate a piece. The first two constraints assign each piece side to one ULD side only and an ULD side to one piece side only. The third constraint prevents pieces being turned upside down when this is not allowed.

$$r_{ik1} + r_{ik2} + r_{ik3} = 1 \forall i \in \mathbf{N}, \forall k \in [1,2,3] \tag{5.16}$$

$$r_{i1k} + r_{i2k} + r_{i3k} = 1 \forall i \in \mathbf{N}, \forall k \in [1,2,3] \tag{5.17}$$

$$r_{i21} + r_{i23} \leq r_i \forall i \in \mathbf{P} \tag{5.18}$$

Lower Left Corner in ULD : Equations 5.19 and 5.20 make sure the lower left corner of the piece is placed inside the ULD. The first constraint makes sure the piece is placed within the length direction, while the second constraint makes sure the height direction is respected.

$$x_i + l_i \cdot r_{i11} + h_i \cdot r_{i12} + w_i \cdot r_{i13} - [\max(L) - \max(l_i, hi, wi)] \cdot q_i - \sum_{j \in M} L_j \cdot p_{ij} \leq 0 \forall i \in \mathbf{P} \tag{5.19}$$

$$z_i + l_i \cdot r_{i21} + h_i \cdot r_{i22} + + w_i \cdot r_{i13}[h_{j,max} - \max(l_i, hi, wi)] \cdot q_i - \sum_{j \in M} H_j \cdot p_{ij} \leq 0 \forall i \in \mathbf{P} \tag{5.20}$$

No-Overlap Constraints : Equations 5.21 until 5.25 are the no-overlap constraints. Pieces should not overlap at any of the axes, when both pieces are in the same ULD only. This is represented by Equation 5.21. Equations 5.22 and 5.23 prevent that two pieces overlap horizontally while Equations 5.24 and 5.24 prevent them to overlap vertically.

$$-x_{ik} - x_{ki} - z_{ik} - z_{ki} + p_{ij} + p_{ji} \leq 1 \forall i \in \mathbf{P}, \forall k \in \mathbf{P}, \forall j \in \mathbf{U} \tag{5.21}$$

$$x_k + l_k \cdot r_{k11} + h_k \cdot r_{k12} + w_k \cdot r_{k13} - x_i \leq D_m(1 - x_{ik}) \forall i \in \mathbf{P}, \forall k \in \mathbf{P} \tag{5.22}$$

$$x_i - l_k \cdot r_{k11} - h_k \cdot r_{k12} - w_k \cdot r_{k13} - x_k \leq D_m(x_{ik} + 1) \forall i \in \mathbf{N}, \forall k \in \mathbf{N} \tag{5.23}$$

$$z_k + l_k \cdot r_{k21} + h_k \cdot r_{k22} + w_k \cdot r_{k23} - z_i \leq D_m(1 - z_{ik}) \forall i \in \mathbf{N}, \forall k \in \mathbf{N} \tag{5.24}$$

$$z_i - l_k \cdot r_{k21} - h_k \cdot r_{k22} - w_k \cdot r_{k23} - z_k \leq D_m(z_{ik} + 1) \forall i \in \mathbf{P}, \forall k \in \mathbf{P} \tag{5.25}$$

Stacking Constraint : Equations 5.26 until 5.43 are the stacking constraints. They make sure a feasible solution exists that guarantees stacking constraints between shipments are ensured. Equation 5.26 makes sure that each vertical vertex of a piece is supported by the ground, another piece or a cut (if a cut is present). Equation 5.27 defines that a piece is on the ground when g_i equals one. Equations 5.28 until 5.33 define the variables z_i, which is the vertical position of the lower left corner. Equations 5.34 and 5.35 describe if the piece overlap on the horizontal axis. Equations 5.36 and 5.37 make sure that when the bottom face of piece i is supported by the top face of piece k, then $h_{ik} + o_{ik} = 0$. Equations 5.38 and 5.39 state that a piece can only be supported by another piece if both are in the same ULD. Equation 5.40 states that piece k can support a vertex of box i only if this one is supported by piece k. In other words β^l_{ik} can only one when s_{ik} equals one. When the left corner of piece i is supported by piece k, x_k should be less or equal to x_i, otherwise the corner cannot be physically supported by piece k. This is implied by Equation 5.41 and implies that the piece is supported in both height and length direction. The final constraints, Equations 5.42 and 5.43, define the horizontal position of the lower left corner of each piece.

$$\gamma_{i1} + \gamma_{i2} + 2g_i + \sum_{k \in \mathbf{P}} \beta^1_{ik} + \beta^2_{ik} = 2 \forall i \in \mathbf{P} \tag{5.26}$$

$$z_i \leq D_m(1 - g_i) \forall i \in \mathbf{P} \tag{5.27}$$

$$-z_i + z_k + l_k \cdot r_{k21} + h_k \cdot r_{k22} - v_{ik} \leq 0 \forall i \in \mathbf{P}, \forall k \in \mathbf{P} \tag{5.28}$$

$$z_i - z_k - l_k \cdot r_{k21} - h_k \cdot r_{k22} - v_{ik} \leq 0 \forall i \in \mathbf{P}, \forall k \in \mathbf{P} \tag{5.29}$$

$$z_i - z_k - l_k \cdot r_{k21} - h_k \cdot r_{k22} + v_{ik} \leq 2D_m(1 - m_{ik}) \forall i \in \mathbf{P}, \forall k \in \mathbf{P} \tag{5.30}$$

$$-z_i + z_k + l_k \cdot r_{k21} + h_k \cdot r_{k22} + v_{ik} \leq 2D_m \cdot m_{ik} \forall i \in \mathbf{P}, \forall k \in \mathbf{P} \tag{5.31}$$

$$-v_{ik} + h_{ik} \leq 0 \forall i \in \mathbf{P}, \forall k \in \mathbf{P} \tag{5.32}$$

$$v_{ik} - D_m \cdot h_{ik} \leq 0 \forall i \in \mathbf{P}, \forall k \in \mathbf{P} \tag{5.33}$$

$$x_{ik} + x_{ki} \geq o_{ik} \forall i \in \mathbf{P}, \forall k \in \mathbf{P} \tag{5.34}$$

$$x_{ik} + x_{ki} \leq 2o_{ik} \forall i \in \mathbf{P}, \forall k \in \mathbf{P} \tag{5.35}$$

$$-h_{ik} - o_{ik} - s_{ik} \leq -1 \forall i \in \mathbf{P}, \forall k \in \mathbf{P} \tag{5.36}$$

$$h_{ik} + o_{ik} + 2s_{ik} \leq 2 \forall i \in \mathbf{P}, \forall k \in \mathbf{P} \tag{5.37}$$

$$p_{ij} - p_{kj} + s_{ik} \leq 1 \forall i \in \mathbf{P}, \forall k \in \mathbf{P}, \forall j \in \mathbf{U} \tag{5.38}$$

$$-p_{ij} + p_{kj} + s_{ik} \leq 1 \forall i \in \mathbf{P}, \forall k \in \mathbf{P}, \forall j \in \mathbf{U} \tag{5.39}$$

$$\beta^l_{ik} - s_{ik} \leq 0 \forall i \in \mathbf{P}, \forall k \in \mathbf{P}, \forall l \in [1,2] \tag{5.40}$$

$$\beta^l_{ik} + \eta^l_{ik} \leq 1 \forall i \in \mathbf{P}, \forall k \in \mathbf{P}, \forall l \in [1,2] \tag{5.41}$$

$$-x_i + x_k - D_m \cdot \eta^1_{ik} \leq 0 \forall i \in \mathbf{P}, \forall k \in \mathbf{P} \tag{5.42}$$

$$x_i - x_k + l_i \cdot r_{i11} + h_i \cdot r_{i12} + w_i \cdot r_{i13} - l_k \cdot r_{k11} - h_k \cdot rk12 + w_i \cdot r_{i13} - D_m \cdot \eta^2_{ik} \leq 0 \forall i \in \mathbf{P}, \forall k \in \mathbf{P} \tag{5.43}$$

Cut Constraints : The last constraints, Equations 5.45 until 5.48, make sure cuts in ULDs are considered when defining the loading strategy. These equations limit the vertical and horizontal position of the lower left corner of a piece. Equation 5.45 takes into account a cut in the lower left corner, Equation 5.46 a cut in the lower right corner, Equation 5.47 a cut in the upper right corner and Equation 5.48 a cut in the upper left corner. Only the constraint(s) of the corresponding cut(s) in the ULD are added to the model. So when the ULD has a cut in the lower left corner only, only Equation 5.45 is added. As example, AFKLMP only uses ULDs with this specific cut. If the model was to be used by a different airline with ULDs characterized by more cuts, the associated constraints should be activated. These constraints contain the term $\delta \cdot \sqrt{1 + a_c^2}$, this term adds a offset to the cut, which makes sure that integer solution which are δ cm away from the cut are still feasible. This is added to have more feasible solution available, since the positions can only be integers. Lastly C_m represents the so called *big M*, which is determined using Equation 5.44.

$$C_m = \max(a_c) \cdot \max(L_j) + \max(H_j, l_i, h_i, w_i) + \max(b_c) \tag{5.44}$$

$$-a_0 \cdot x_i - z_i + C_m \cdot p_{ij} \leq C_m - b_0 + \delta \cdot \sqrt{1 + a_0^2} \forall i \in \mathbf{P}, \forall j \in \mathbf{U} \tag{5.45}$$

$$a_1 \cdot x_i - z_i + C_m \cdot p_{ij} + a_1 \cdot l_i \cdot r_{i11} + a_1 \cdot h_i \cdot r_{i12} + a_1 \cdot w_i \cdot r_{i13} \leq C_m + b_1 + \delta \cdot \sqrt{1 + a_1^2} \forall i \in \mathbf{P}, \forall j \in \mathbf{U} \tag{5.46}$$

$$a_2 \cdot x_i + z_i + C_m \cdot p_{ij} + a_2 \cdot l_i \cdot r_{i11} + a_2 \cdot h_i \cdot r_{i12} + a_2 \cdot w_i \cdot r_{i13} + l_i \cdot r_{i21} + h_i \cdot r_{i22} + w_i \cdot r_{i23} + \delta \cdot \sqrt{1 + a_2^2} \leq C_m + b_2 \forall i \in \mathbf{P}, \forall j \in \mathbf{U} \tag{5.47}$$

$$a_3 \cdot x_i + z_i + C_m \cdot p_{ij} + l_i \cdot r_{i21} + h_i \cdot r_{i22} + w_i \cdot r_{i23} + \delta \cdot \sqrt{1 + a_3^2} \leq C_m + b_3 \tag{5.48}$$

2.4. Results

This Section discusses the results of the 2DBPP. The model is tested for a flight of which the details can be found in Table 5.2 and the results can be found in Table 5.3. From the last table it can be concluded that the model has a computational time of about five minutes for up to 24 pieces. This is still acceptable for AFKLMP, however 30 pieces already lead to a computational time of fourteen minutes. This is too much. Since this flight has 580 pieces, a heuristic is necessary to speed up the packing process. Therefore different heuristics are discussed in Section 3.

Table 5.2: Flight Details.

AircraftType:	**MDConfig**:	**LDConfig**:	**CTRConfig**:	**Total Pieces**:
789	0	7	3	580

Table 5.3: Results for 2DBPP.

	Test						
	1:	**2**:	**3**:	**4**:	**5**:	**6**:	**7**:
Total Number of Pieces [−]:	10	20	21	22	23	24	30
Total Flying Pieces [−]:	10	20	21	22	23	24	NF
Total Non-Flying Pieces [−]:	0	0	0	0	0	0	NF
Total Must-Fly Pieces [−]:	2	2	2	2	2	2	NF
Total Flying Must-Fly Pieces [−]:	2	2	2	2	2	2	NF
Total Non-Flying Must-Fly Pieces [−]:	0	0	0	0	0	0	NF
Total Decision Variables [−]:	1190	4580	5040	5522	6026	6652	10170
Computation Time [*s*]:	00:00:01	00:00:17	00:00:34	00:01:19	00:01:14	00:05:25	00:14:00
Gap [%]:	0.00	0.00	0.00	0.00	0.00	0.00	113.02

3. Heuristics

The 2DBPP for a full flight is computational demanding and not consistent with operational needs. Different heuristics are proposed to obtain a sub-optimal solution, where computational time is consistent with operational requirements and a good load factor is achieved. This Chapter discusses the different heuristics used to both speed up and improve the 2DBPP model as discussed in Chapter 2. Section 3.1 discusses a heuristic to speed up the 2DBPP using the 1DBPP, next Section 3.2 discusses a concept to transform the 2DBPP into 3DBPP, Section 3.3 combines both heuristics discussed in the previous two Sections and finally Section 3.4 discusses the results of the three heuristics.

3.1. 1D2D

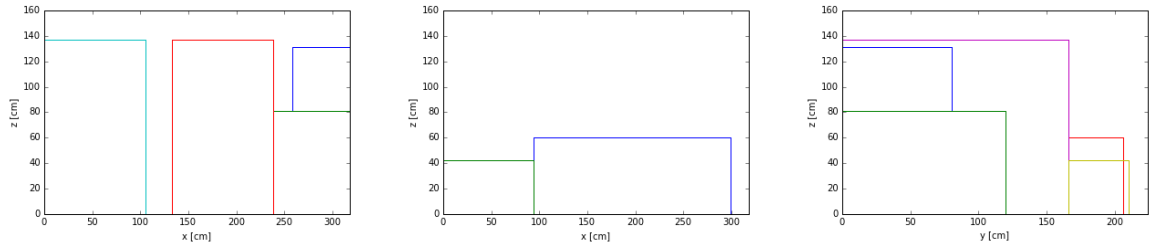
This Section discusses a heuristic to speed up the 2DBPP by using the 1DBPP as a pre-step that split a single 2DBPP into multiple smaller 2DBPPs. In fact, one method to speed up the 2DBPP is by decreasing the number of decision variables. The total number of decision variables highly depends on the number of pieces to be loaded N and the number of ULDs M. So, when these are decreased, the 2DBPP is expected to find a solution faster.

The 1DBPP is used to assign every piece to an ULD first, then M 2DBPP are solved separately, each considering the subset of shipments that the 1DBPP assigned to the current ULD. This leads to significantly fewer decision variables.

3.2. Layers

This section discusses the layers heuristic. The 2DBPP is solved per ULD, as shown in Section 3.1, however the difference is that the pieces are not assigned to a certain ULD, but all unloaded pieces are taken into account and each ULD is divided into layers in the width direction.

The heuristic starts with the first ULD and all pieces. It solves the 2DBPP for the first ULD considering the front side, i.e., the L-H two-dimensional space. Some pieces are loaded into this ULD and some are offloaded. Given the current configuration, the ULD is now considered along the W-direction. Behind the shipment with the maximum width among the ones loaded, the ULD is completely empty. This means that the same ULD with the same L-H configuration, but a reduced width (the original width minus the maximum



(a) Layer 0. (b) Layer 1. (c) Side.

Figure 5.1: Loading strategy for one ULD with six pieces divided over two layers.

width among the shipments loaded), can still be considered to load new shipments. When at least one of the offloaded pieces fits behind the loaded pieces, the ULD is split along the W-dimension into two layers. The first layer width equals the maximum width of all the loaded pieces ($\max(w_{p,loaded})$) while the width of the new layer equals $w_u - \max(w_{p,loaded})$. The 2DBPP is solved for the second layer considering its width and the offloaded pieces. This process continues until all pieces are loaded or until none of the offloaded pieces fits in the remaining space of the ULD. When this is the case, the procedure moves on to the second ULD and the process is repeated. This will be done for all ULDs, one by one, until no ULDs or pieces are left. The sequence of ULDs is sorted from smallest to largest. This heuristic solves 2DBPPs sequentially, but takes all the three dimensions into account since each new layer considers the space left given the width of the previous layers. An example of the results for this heuristic can be found in Figures 5.1a until 5.1c. Figure 5.1a shows the result of the first layer and Figure 5.1b shows the results for the second layer. The second layer is behind the first layer. A side view of the ULD with two layers is shown in Figure 5.1c. This figure shows that pieces belonging to different layers do not overlap as a result of the third dimension (width) being accounted for when generating a new layer.

3.3. 1D2D Layers

This Section combines the heuristics of Sections 3.1 and 3.2. This heuristic first assigns shipments to ULDs, as in Section 3.1. Then, a 2DBPP is solved per ULD. If all shipments are loaded, the heuristic moves to the next ULD. Otherwise, a second layer is created as described in Section 3.2 and a 2DBPP is solved for that layer with the offloaded shipments. The process, for the current ULD, is repeated until either all shipments are assigned, or if none of the remaining pieces fit behind the loaded pieces. This heuristic better uses the available space in an ULD without the complexity of a full 3DBPP while also reducing the problem size and thus the computational time by adding the initial 1DBPP step.

3.4. Results

This section discusses the results of the different heuristic. These can be found in Table 5.4. From this Table it can be concluded that the layers heuristic provides the best results in terms of offloaded pieces, however it has the largest computational time. The 1D2D heuristic provides a quicker results but has significant more offloaded pieces. Combining both heuristics leads to an improved result than the 1D2D in terms of offloaded pieces, and also a faster result compared to both other heuristics. Therefore, it can be concluded that the 1D2D Layers heuristic performs the best out of the three heuristics, being a trade-off between computational time and performance in terms of loaded pieces.

Table 5.4: Results for the one-dimensional BPP.

Flight Details			**Offloaded Pieces**			**Computational Time**		
Nbr:	*ULDs*:	*Pieces*:	*1D2D*:	*Layers*:	*1D2D Layers*:	*1D2D*:	*Layers*:	*1D2D Layers*:
001	14	18	0	0	0	00:00:20	00:01:16	00:00:03
002	12	42	6	1	1	00:03:24	00:05:19	00:03:06
003	12	31	2	2	2	00:03:19	00:04:23	00:01:12
004	9	16	0	0	2	00:01:02	00:01:03	00:00:02
005	13	28	2	1	1	00:01:17	00:02:10	00:00:07
006	9	41	3	3	3	00:04:08	00:06:19	00:03:21
007	10	53	11	0	0	00:05:15	00:07:47	00:01:57
008	11	8	0	0	0	00:00:12	00:00:01	00:00:01
009	8	26	10	1	1	00:00:02	00:05:07	00:00:04
010	13	21	0	0	0	00:01:03	00:01:03	00:00:11
011	14	28	4	0	8	00:01:05	00:02:35	00:01:04
012	11	52	7	0	0	00:05:25	00:10:03	00:02:10
013	13	69	23	2	13	00:04:22	00:15:37	00:01:38
014	12	47	3	0	0	00:03:59	00:06:59	00:03:05
015	14	33	9	0	0	00:01:04	00:09:50	00:00:05
016	11	19	1	0	0	00:01:06	00:01:10	00:01:03
017	12	66	1	1	1	00:06:14	00:14:32	00:07:43
018	12	56	8	6	6	00:06:35	00:09:33	00:05:05

6

The Final Model

In the previous chapters, different models are discussed for both forecasting and bin packing. The final model is a combination of these different models and is called Better Revenue Using a New Optimiser (BRUNO). This chapter discusses the structure on BRUNO, and how the forecasting and optimisation blocks are merged to lead to the final ULD loading strategy. BRUNO is visually represented by the flowchart in Figure 6.1.

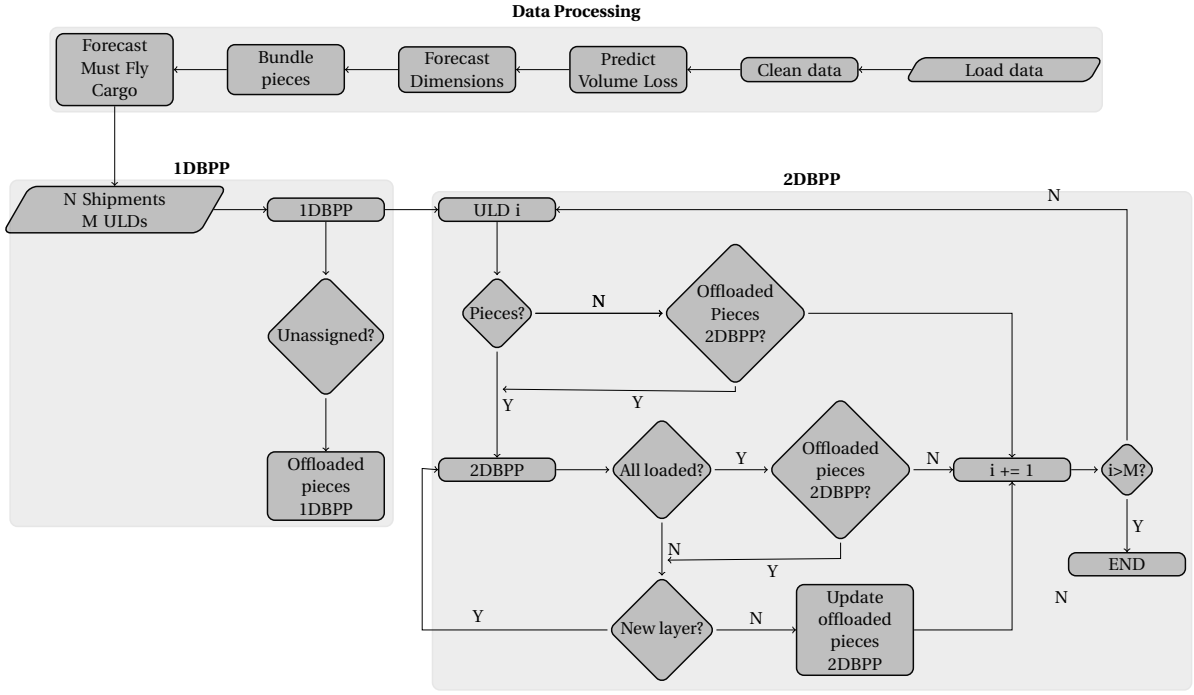


Figure 6.1: Flowchart for BRUNO.

In the first step, data is both loaded and cleaned. In the dataset, every row represents specifications of a booking. This contains both the properties of the booking as well as information about the flight. The booking information consists of information like booking origin, booking destination, agent number, commodity, volume and weight while the flight data consists of information like flight number, departure date, origin, destination, configuration and capacity.

The first block that is added to BRUNO is the volume loss prediction, as determined in Section 2.2 of Chapter 4. Volume loss is not used further, but it can be stored and used by flight analysts for different needs. The volume loss is determined per booking origin, so every booking origin is characterized by a specific volume loss. The dimension forecast is carried out for those bookings where shippers did not specify a dimension,

as described in Section 2.1 of Chapter 4. This concludes the first forecasting block. An intermediate step is taken before running the ULD loading heuristic block. Some bookings are characterised by a high number of identical pieces (e.g., high-tech devices) and can be bundled together. Since every piece would be treated separately in the heuristic, this may increase the problem size significantly. To prevent this, one (or more) bigger pieces are created out of all these smaller pieces, considering the ULD dimensions. This significantly reduces the model size, while being consistent with real operations. Algorithm 1 explains this procedure in more detail. This concludes the adaptations to the booking part of the data.

Algorithm 1 Combine smaller pieces to one larger piece.

initialise numberPiecesPacked to zero
initialise numberLeftPieces to the number of pieces that need to be packed
initialise maxPiecesInDimension to the number of pieces that fit in the length, width and height of the ULD
if at least one piece fits in the ULD **then**
  while numberLeftPieces > 0 **do**
    if enough pieces are available to fill a complete ULD **then**
      set the three dimensions of the new piece to maxPiecesInDimension · Dimension.
      add number of packed pieces to the numberPackedPieces
      remove number of packed pieces from the leftPieces
    else if enough pieces are available to fill the length and height dimension of the ULD **then**
      set the length and height of the new piece to maxPiecesInDimension · Dimension
      set the width to the width of one piece
      add number of packed pieces to the numberPackedPieces
      remove number of packed pieces from the leftPieces
    else
      if enough pieces are available to fill the length dimension of the ULD **then**
        set the length of the new piece to maxPiecesInDimension · Dimension
        set the width and height to the width and height of one piece
        add the number of packed pieces to the numberPackedPieces
        substract the number of packed pieces from the leftPieces
      else
        set the length of the new piece to numberLeftPieces · Dimension
        set the width and height to the width and height of one piece
        add the number of packed pieces from the numberPackedPieces
        substract the number of packed pieces from the leftPieces
      end if
    end if
  end while
end if

Next, flight details are accounted for before running the loading heuristic. For every flight, the amount of must-fly cargo is forecasted using the model as discussed in Section 3 of Chapter 4. This forecast outputs the amount of must-fly cargo in m^3 and the volume of the bookings labeled as must-fly is subtracted from the forecast, this is called the remaining must-fly volume. Next, the remaining must-fly volume as well as the volume of the bookings are subtracted from the capacity. When a positive capacity remains, the heuristic continues.

This concludes the data-processing step, which is the intermediate step between the forecasting block and the ULD loading heuristics block. A slightly modified version of the heuristic *1D2D Layers*, as discussed in Section 3.3, is used. The basis of the heuristic is the same, however when there are no more pieces to pack in an ULD in the 2DBPP, but there is some space left in the ULD, BRUNO attempts to add the offloaded pieces generated by the 2DBPP from previous ULDs (if any). Furthermore, in some cases the heuristic might output a solution where some pieces are offloaded and some ULDs are left empty. In this case, a reduced-size 2DBPP is solved again with the empty ULDs and offloaded pieces only. This framework provides results with good performances, while keeping the computational time within boundaries compatible with a real-time implementation. Results are discussed in Chapter 7.

7

Validation

This Chapter discusses the validation of BRUNO and the applicability of the model to a real case. First, a sensitivity analysis is performed in Section 1. Next, a case study is discussed in Section 2.

1. Sensitivity Analysis

This Section performs a sensitivity analysis for BRUNO. For the sensitivity analysis, BRUNO is run three times for every flight. In the first run the shipment contribution (SCb) is maximised, in the next run the volume is maximised and in the third run the commercial priority (CP) is minimised. The corresponding objective functions can be found in Equations 7.1 until 7.3 and results can be found in Table 7.1. This table shows the load factor of the pieces, this is the percentage of the total pieces weight that is loaded.

From the table, it can be concluded that when the volume is maximised the best load factor is achieved, as intuition would suggest. This also makes sense since volume and weight are dependent while weight is independent from both SCb and CP. SCb can be CP dependent. For some bookings, customers pay more (higher SCb) in order to increases the CP, however on the other side perishables, for example, have a high CP due to their perishability but the SCb is generally low for these products. Therefore, the relation between the SCb and CP is different for different products.

The next take-away that can be inferred from Table 7.1 is that when the total shipment volume consumes less than 50 % of the flight capacity, the objective function does not really change the solution in terms of load factor. This is the case for flights 003, 006 and 009. In general, all pieces are loaded. In other words, for cases where capacity is severely under-utilized, all cost functions guarantee all shipments will be loaded, although configurations might change depending on the cost function. Pieces may have a different position in the same ULD, they might be loaded in different ULDs or may be offloaded. Depending on the objective function a piece with a large volume and a high CP is loaded first when maximising volume while it is loaded last when the CP is minimised.

On the other hand, it is concluded that the objective function influences the solution more significantly when the overall volume gets closer to maximum capacity. Therefore, it is very important to choose the right objective function depending on the specific needs of the airline.

$$\max J = \sum_{i \in \mathbf{P}} \sum_{j \in \mathbf{U}} SCb_i \cdot p_{ij} - 10 \cdot SCb_i \cdot z_i \tag{7.1}$$

$$\max J = \sum_{i \in \mathbf{P}} \sum_{j \in \mathbf{U}} V_i \cdot p_{ij} - 10 \cdot V_i \cdot z_i \tag{7.2}$$

$$\min J = \sum_{i \in \mathbf{P}} \sum_{j \in \mathbf{U}} CP_i \cdot p_{ij} + [4 \cdot \max(CP) - CP_i] \cdot z_i \tag{7.3}$$

Table 7.1: Load factor of the pieces for different costs functions for different flights.

			Costs Function		
Nbr:	*ULDs*:	*Pieces*:	*max SCb*:	*max Volume*:	*min CP*:
001	14	27	74	82	70
002	12	42	59	93	63
003	9	21	100	100	93
004	13	33	62	87	73
005	9	57	51	76	60
006	10	52	100	100	96
007	11	29	61	53	61
008	8	26	84	72	84
009	13	24	100	100	100
010	14	28	46	72	60
011	11	62	86	68	78
012	13	74	59	73	45
013	12	49	82	85	73
014	13	101	53	56	56
015	14	22	99	99	99
016	11	23	88	88	88
017	12	73	73	74	57
018	12	78	49	70	71

2. Case Study

This Section discusses the case study. For this case study, we simulate a real process where anytime a new booking is received, flight analysts need to make a decision and accept or decline the incoming booking. Using the chronologically ordered sequence of booking for a real flight, every time a booking is received, BRUNO is run to check if the booking fits or not. If a new booking and all the previously accepted pieces are loaded, the new booking is accepted. When one of the accepted bookings or the new booking is offloaded by BRUNO, the new booking is declined. The underlying assumption is that the introduction of a new booking should guarantee that previous bookings are not canceled.

All details about the specific flight can be found in Table 7.2. The flight considered is performed with a Boeing 747-400 aircraft equipped with three CTRs and four LDPs. The must-fly forecast outputs that 26.40 m^3 of must-fly cargo will be loaded. On the real flight, 24.03 m^3 of must-fly cargo was loaded, so the forecast provides an accurate estimate.

All the bookings can be found in Table 7.3. 44 bookings were received for this flight. The first column, *Nbr*, shows the number of the booking in order of arrival. So, booking 00 was booked first, next booking 01, etc. Sometimes a booking consists of more pieces. See, as example, bookings 00, 09, 13, 14 and 43. In these cases a suffix is added to differentiate the different pieces for the same bookings. Bookings 02 and 41 consist of two pieces with the same properties. This is indicated in the column *pieces*. The column *Must-Fly* indicates if the piece is flagged must-fly or not. Columns *Length*, *Height* and *Width* indicate respectively the length, height and width of the piece(s) and column *Forecasted Dims* indicates if these are forecasted or not.

Table 7.2: Flight details for the case study.

				Must-Fly Forecast	
Aircraft Type [−]:	*CTR [−]*:	*LDP [−]*:	*MDP [−]*:	*Volume [m^3]*:	*Weight [kg]*:
744	3	4	0	26.4	3632

Table 7.3: Booking details for the case study. L, H and W represent the dimensions of the piece (length, height and width respectively).

Nbr [−]:	*Pieces [−]*:	*Origin [−]*:	*Must-Fly [−]*:	*Forecasted Dims [−]*:	*L [cm]*:	*H [cm]*:	*W [cm]*:
00.1	1	AMS	True	True	305	47	56
00.2	1	AMS	True	True	61	47	56
01	1	AMS	True	True	311	196	209
02	2	AMS	True	False	300	120	228
03	1	SYD	False	False	155	58	156
04	1	AMS	True	True	179	191	127
05	1	BOG	True	False	113	29	120
06	1	AMS	True	True	182	177	152
07	1	AMS	True	True	193	56	71
08	1	AMS	True	False	195	40	50
09.1	1	BOM	False	False	252	37	28
09.2	1	BOM	False	False	222	27	37
10	1	BKK	False	True	66	38	47
11	1	AMS	True	True	133	108	64
12	1	AMS	True	True	67	78	62
13.1	1	AMS	True	True	270	134	177
13.2	1	AMS	True	True	270	67	59
13.3	1	AMS	True	True	90	67	59
14.1	1	AMS	True	True	270	134	177
14.2	1	AMS	True	True	270	67	59
14.3	1	AMS	True	True	90	67	59
15	1	AMS	False	True	234	16	14
16	1	VCE	False	False	53	16	40
17	1	AMS	False	False	60	25	40
18	1	MUC	False	False	228	90	120
19	1	LHR	False	False	120	122	80
20	1	LBA	False	False	120	82	80
21	1	LOS	False	True	110	180	74
22	1	LNZ	False	True	228	25	33
23	1	BUD	False	False	119	114	90
24	1	HEL	False	False	120	95	80
25	1	MXP	True	False	45	140	45
26	1	BOM	True	False	66	28	15
27	1	YUL	False	True	207	141	132
28	1	AMS	True	True	286	87	198
29	1	AMS	True	True	290	176	186
30	1	AMS	True	True	322	193	222
31	1	BRU	False	False	42	43	41
32	1	VCE	False	False	77	34	53
33	1	IST	True	False	52	49	56
34	1	AMS	True	True	147	27	28
35	1	EIN	False	False	82	66	80
36	1	DXB	False	False	157	38	45
37	1	BCN	True	False	123	26	31
38	1	EIN	False	False	112	76	110
39	1	AMS	True	True	69	50	52
40	1	AMS	True	True	160	122	123
41	2	AMS	True	True	219	116	131
42	1	AMS	True	True	173	134	126
43.1	1	DXB	True	False	276	34	54
43.2	1	DXB	True	False	184	34	54

Table 7.4: Results for the case study.

Nbr:	00.1	00.2	01	02	03	04	05	06	07	08	09.1	09.2	10
AFKLMP:	True	True	True	True	True	True	True	False	False	True	True	True	True
BRUNO:	True	True	False	False	True	True	True	True	False	False	False	False	True
Risk:	2.29	2.29	-	-	2.29	2.29	2.29	50.61	-	-	-	-	2.29
Nbr:	11	12	13.1	13.2	13.3	14.1	14.2	14.3	15	16	17	18	19
AFKLMP:	False	False	False	False	False	False	False	False	False	False	True	False	False
BRUNO:	True	True	False	False	False	False	False	False	False	True	True	False	True
Risk:	2.29	46.64	-	-	-	-	-	-	-	74.33	46.67	-	74.33
Nbr:	20	21	22	23	24	25	26	27	28	29	30	31	32
AFKLMP:	True	False	True	False	False	False	True	False	True	False	False	True	False
BRUNO:	True	True	False	True	False	False	False	True	False	False	False	False	True
Risk:	74.33	54.69	-	2.29	-	-	-	52.22	-	-	-	-	30.43
Nbr:	33	34	35	36	37	38	39	40	41	42	43.1	43.2	
AFKLMP:	True	True	True	True	False	True	False	False	False	False	True	True	
BRUNO:	False	False	False	False	False	False	False	False	False	False	False	False	
Risk:	-	-	-	-	-	-	-	-	-	-	-	-	

Table 7.4 shows the results of the case-study. Label *Nbr* represents the number of the booking and this one corresponds with the numbers used in Table 7.3. *AFKLMP* shows if the booking was loaded on the flight by AFKLMP or not. When it was offloaded, it means that a booking was delayed, did not fit, was rebooked, or was not loaded as initially agreed for a different reason. However, for this case study it is assumed that the booking is declined because it did not fit. Note that, in real operations, many hidden causes might prevent a shipment from being loaded (e.g., lack of personnel) that are not modeled in the framework we hereby present. The *BRUNO* row indicates if the booking is accepted by BRUNO or not. The last row, *Risk*, shows the risk when a booking is accepted by BRUNO, as discussed in 4. The final solution is visually represented in Figures 7.1 until 7.12.

When analysing Table 7.4 it can be concluded that BRUNO loaded 21.24 m^3 on the flight and AFKLMP loaded 28.52 m^3. This is a difference of 7.28 m^3. When analysing the bookings, bookings 00, 03, 04, 05, 06, 10, 17, and 20 are loaded by both AFKLMP and BRUNO, bookings 01, 02, 08, 22, 26, 28, 31, 33, 34, 35, 36, 38 and 43 are loaded by AFKLMP and are offloaded by BRUNO, while bookings 11, 12, 16, 19, 21, 23, 27 and 32 are offloaded by AFKLMP and are loaded by BRUNO. All other bookings are offloaded by both AFKLMP and BRUNO. A more in-depth analysis will be performed on bookings that are loaded by AFKLMP and are offloaded by BRUNO. Bookings 01 and 02 are offloaded because their dimensions exceed the dimensions of the largest ULD. In real operations, shipments will be loaded anyway if dimensions only slightly exceed the ULD dimensions. As example, shipments can be slightly tilted as long as this does not prevent stability, while in BRUNO shipments have edges always aligned with the ULD sides. Furthermore, some space exist between ULD positions. LDPs and MDPs are just a plate and do not have vertical walls. Therefore, in operations pieces can slightly overlap LDPs and MDPs. These features are not implemented in BRUNO, therefore the pieces are offloaded. Booking 01 has forecasted dimensions, however its real dimensions are not available in the data, therefore it is unknown whether the forecast is highly inaccurate or not. Booking 08 is offloaded in BRUNO because it is loaded in a CTR by the 1DBPP without any other bookings. Its length equals 195 cm and this dimensions only fits in the length direction (196 cm) of the CTR. However due to the cut and the lack of another piece to support the second vertex, it is offloaded. In real operations, tilting the shipment might suffice to guarantee the correct loading of the shipment. Booking 22 is loaded in the 1DBPP in the same ULD as booking 06. Due to their dimensions they need to be stacked on top of each other, however also due to their dimensions only one vertical vertex can be supported. BRUNO only loads a booking when both vertical vertices are supported by the ground, a cut or another shipment, therefore booking 06 is offloaded. Since booking 06 is already accepted, booking 22 is declined. When trying to load booking 26, it is loaded in a CTR. This results in another booking loaded in a CTR first, now being loaded on a LDP which results in no more space for booking 11 on a LDP. Therefore booking 11 gets offloaded by BRUNO, but since it was already accepted, booking 26 is declined. Booking 28, 31, 33, 34, 35, 36, 38 and 43 are offloaded for the same reasons. They are either offloaded by BRUNO, or another already accepted piece gets offloaded. This last one might happen since CP is minimised. When the new piece has a lower CP than some other pieces, it might happen that an already accepted piece with higher CP gets offloaded.

Figures 7.1 until 7.12 show the final results of BRUNO. From those figures it can be concluded that ULDs 0, 2, 3 and 5 are almost full, while ULDs 1 and 4 have lots of space for other bookings, for example bookings 26, 31, 33, 35 and 39. These are offloaded due to the assignment strategy of pieces to ULDs by the 1DBPP. They get offloaded by the 2DBPP and the heuristic does not find an opportunity to load them somewhere else. A recommendation would be to improve the quality of the 1DBPP, by preferring small pieces to be loaded in CTRs for example, or improve the 2DBPP block, where empty gaps within ULDs are filled with offloaded pieces, if possible.

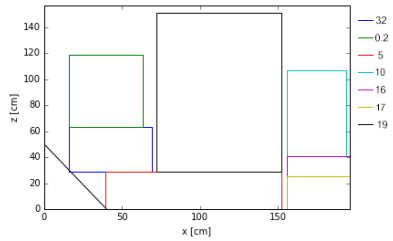


Figure 7.1: ULD 0 - Layer 0.

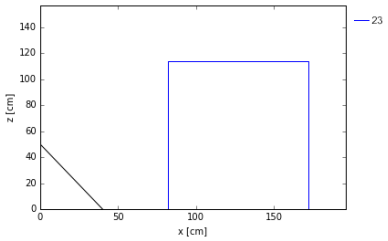


Figure 7.2: ULD 1 - Layer 0.

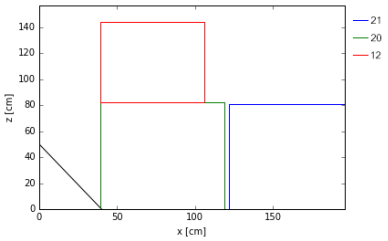


Figure 7.3: ULD 2 - Layer 0.

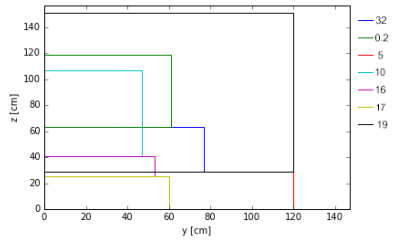


Figure 7.4: ULD 0 - Side.

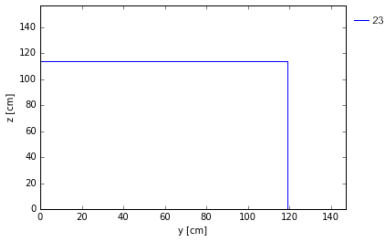


Figure 7.5: ULD 1 - Side.

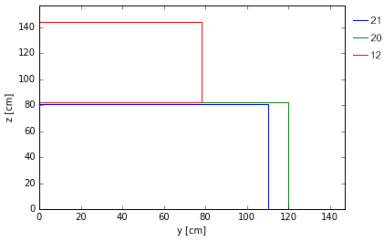


Figure 7.6: ULD 2 - Side.

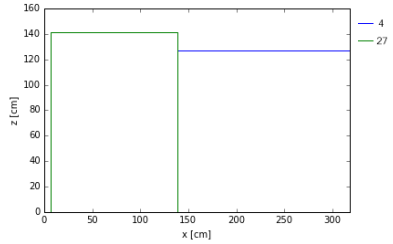


Figure 7.7: ULD 3 - Layer 0.

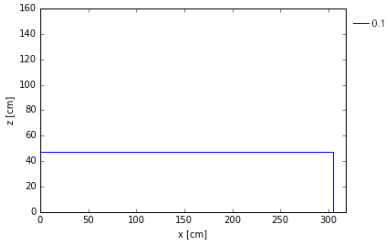


Figure 7.8: ULD 4 - Layer 0.

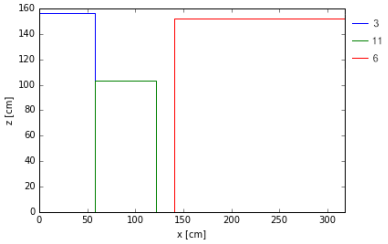


Figure 7.9: ULD 5 - Layer 0.

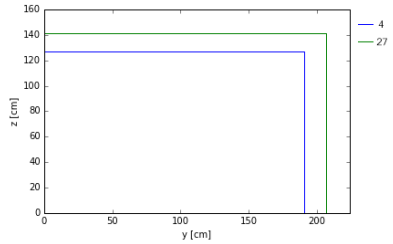


Figure 7.10: ULD 3 - Side.

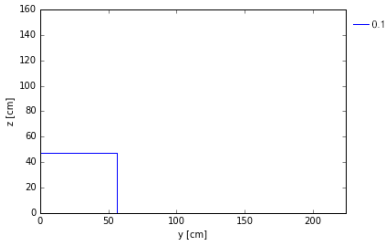


Figure 7.11: ULD 4 - Side.

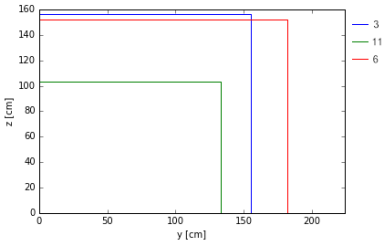


Figure 7.12: ULD 5 - Side.

8

Conclusions & Recommendations

This thesis discusses a framework whose goal is to assist cargo airlines in their cargo acceptance and loading strategy. The model forecasts unknowns of both each new incoming booking and of flight details and combines this information with a bin packing heuristic. Given the current list of accepted bookings, for every new request it is assessed whether the incoming booking can be loaded in the aircraft without preventing the already accepted bookings from being offloaded. In addition, given dimensions of most bookings are forecasted and are not known a priori, a risk index is provided which estimates what are the chances the loading strategy will not be implementable because of an underestimation of booking dimensions.

For the forecasting part a random forest (RF) is used. This is an ensemble learning method used for both classification and regression. Since, in this context, dimensions are predicted, a regression model is chosen. A particular focus was put on the choice of the dataset size and the number of decision trees for the model. To determine the more appropriate sizes, a trade-off between computational time and accuracy of the results was chosen. A better accuracy is obtained with more data and the same number of decision trees. Nevertheless, processing a too large dataset will lead the RF model to encounter computer memory issues. As a consequence, only five months of data are used to train the model. Since more data led to computer memory issues, it was not possible to train the RF with more data. It is expected that the results improve when at least one year of data is used to train the model, however the results obtained with the current dataset provided results considered acceptable by the airline. On the other hand, accuracy keeps improving until 100 decision trees. An increase in the number of decision trees would only cause an increase in the computational time without a tangible increase in the performance. As a consequence, the model was trained setting the number of decision trees to 100.

This RF model is initially used to forecast dimensions of bookings. The dimensions are predicted with an mean error of 9.89 cm which results in an accuracy of 84.29 %. With an importance of 0.68, *BookingOrigin* is the most important feature. This means that cargoes from different origins will generally be characterized by considerably different dimensions, probably due to the different commodity type. This also suggests a strong orientation of the air cargo industry towards specialized trade flows according to the specific airport origin-destination pairs (e.g., flower trade from Kenya to the Netherlands). Other features that influence the results are *AgentURN*, *BookingDestination*, *CommodityCode*, *PieceVolume* and *ProductCode*. The next forecast is the volume loss. Volume loss is the unused space in an ULD. Airlines use this factor to increase the volume of pieces with unknown dimensions. This increase in size is a safety factor to estimate the shipment dimensions, to increase the chances shipments will fit in the ULD. Since the accuracy of the dimensions forecast is known, it would be strange to use something else as volume loss than the accuracy of the dimension prediction. Since volume loss is used to increase the volume of a piece, it is more accurate to increase this using the accuracy of the dimensions forecast. The accuracy of the dimensions model results in a volume loss of $100 - 84.29 = 15.71 \approx 16$ %. So, the volume is increased with a factor of 1.16. Since *BookingOrigin* is the most important feature, the volume loss is determined per booking origin. This gives the airline a more accurate estimate for determining the volume loss. Forecasted shipment dimensions, being the outcome of a prediction, come with an error. This leads to a certain risk, since packages might be larger than predicted and this might lead to loading strategies that do not comply with the ULD dimensions. To find the risk, a distribution of the dimensions and their probability is needed. The test dataset contains the errors of the prediction for 142,493 pieces. From all these errors, a probability distribution can be found. The errors follow a distribution

that strongly resembles a normal distribution. This made it possible to define an easy, yet intuitive way to compute a risk per dimension (per ULD) that the designed loading strategy would not fit the ULD along the specified dimension. The method identifies the critical shipments, sums their distributions and computes the area of the distribution that exceeds the ULD dimension along that direction. The area is then translated into a risk. This risk is very useful for airliners, since it provides a tangible indicator to decide whether to accept or decline a booking.

The second forecast forecasts the amount of must-fly cargo that is expected to arrive for a specific flight. Must-fly cargo cannot be offloaded and therefore it is useful for an airline to forecast how much must-fly cargo is expected before every flight. Using the RF model discussed before, the must-fly cargo can be forecasted with an error of 11 m^3. The most important features are *AircraftSubType* and *FlightNumber*. This can be explained by the fact that different aircraft types have different cargo capacities, which in turn are reflected in different must-fly capacities. In addition, the same flight number is generally associated with the same origin-destination airport pair. Consistently with what described for dimensions prediction, flights serving the same airport pair at different times might have higher chances to display similar characteristics. Features *Arrival*, *CapacityVolume*, *CapacityWeight* and *WeekDay* also influence the final result. The accuracies are determined per aircraft sub-type as well. An outcome of this analysis is that must-fly cargo on aircraft used by AFKLMP can be predicted more accurately, while the must-fly cargo on the partner aircraft is less accurate. This is due to a higher accuracy of the dataset provided by AFKLMP. For future research it is recommended to add *Month* as a feature since must-fly cargo is very seasonal. Furthermore, it is advised to create a risk profile, like for the dimensions forecast, to have a better overview of the risk of your forecast.

The last forecast is the accuracy of the flight details. This is relevant since capacity is sold. An accurate estimate prevents the following two cases from happening. (i) Capacity is oversold, which results in pieces being offloaded and thus a lower customer satisfaction. (ii) Capacity is undersold, which results in a lower revenue. The accuracy is forecasted with an accuracy of 93.49 %. This is very accurate and can be used by the airline to decide to sell more or less capacity.

To determine if a booking fits into the aircraft, a bin packing problem (BPP) is solved. First a 1DBPP is researched. Generally speaking, for problems aiming at determining optimal loading configurations, the 1DBPP focuses on volume or weight, rather than a physical dimension (e.g., length) of objects to be loaded. In our context, one-dimensional means that the volume of the pieces is used to assign each piece to an ULD, without determining the exact position in the ULD. It turns out that the 1DBPP produces quick results which results in good load factors. However, the real problem is 3D, therefore the 2DBPP is investigated as well. The 2DBPP considers the length and height of the pieces and determines the exact position of each piece in a 2D-space. A full 2DBPP turned out to be computationally demanding, therefore a full 3DBPP is not investigated and instead heuristics are investigated. A heuristic called 1D2D Layers, performed the best. This heuristic first solves the 1DBPP. Every piece is now assigned to a ULD or offloaded. It solves the 2DBPP for the first ULD considering the front side, i.e., the L-H two-dimensional space. Some pieces are loaded into this ULD and some are offloaded. Given the current configuration, the ULD is now considered along the W-direction. Behind the shipment with the maximum width among the ones loaded, the ULD is completely empty. This means that the same ULD with the same L-H configuration, but a reduced width (the original width minus the maximum width among the shipments loaded), can still be considered to load new shipments. When at least one of the offloaded pieces fits behind the loaded pieces, the ULD is split along the W-dimension into two layers. The first layer width equals the maximum width of all the loaded pieces ($\max(w_{p,loaded})$) while the width of the new layer equals $w_u - \max(w_{p,loaded})$. The 2DBPP is solved for the second layer considering its width and the offloaded pieces. This process continues until all pieces are loaded or until none of the offloaded pieces fits in the remaining space of the ULD. When this is the case, the procedure moves on to the second ULD and the process is repeated. This will be done for all ULDs, one by one, until no ULDs or pieces are left. This heuristic produces good load factor within an acceptable computational time.

The final model is called BRUNO, which stands for Better Revenue Using New Optimiser. BRUNO forecasts dimensions, predicts the amount of must-fly cargo, solves the 1D2D Layers heuristic and calculates the risk of the loading strategy. A comparison between results (expressed as loaded volume in m^3) of BRUNO and results provided by AFKLMP showed that BRUNO is generally comparable in terms of loaded volume with what happened in reality. In some cases the suggested loaded volume was higher than the one actually loaded, while in some others BRUNO underperformed.

Along with the fact that BRUNO never solves a fully 3DBPP, which would be the natural environment for such a problem, some additional causes were identified that make real case operations different with respect to what BRUNO models. Sometimes one large piece is loaded in a container (CTR), and due to the cut in

the CTR it cannot be placed on the ground. For this reason, it gets offloaded in the 2DBPP, since the 2DBPP model wants both vertical vertices to be supported. This can be prevented by making a small adaptation to the 1DBPP, where these pieces are not assigned to CTRs. As an alternative, the 2DBPP needs to be adapted in a way that only one vertical vertex needs support or just the center of gravity of the piece needs support (or the piece can be slightly tilted without having the sides perfectly aligned with the ULD sides). Furthermore, it is recommended to fill up empty spaces in the ULDs, after the heuristic is solved, with a post-processing step that checks if offloaded pieces might fit in the gaps left between layers.

Airliners can use BRUNO starting from the booking phase until the moment of departure. This model provides them with insights into the loading strategy of the aircraft, solves the unknowns of both the booking and the flight and provides a risk index that can help them decide whether to accept or decline bookings. Every time something is added to the flight, which might be a new booking or a booking which was offloaded from a previous flight, BRUNO can be run to help the airline decide to accept or decline the booking. In its current form, the heuristic approach is fast enough for real-time or quasi real-time applications.

A

Table A.1 contains the parameters (both mean and StD) per booking origin. It contains only booking origins with at least 100 datapoints available, other booking origins should use booking origin *ALL*. The parameters for *ALL* represent the normal distribution for the complete dataset.

Table A.1: Parameters for the normal distributions per booking origin.

Booking	Length		Height		Width		Booking	Length		Height		Width	
Origin	Mean	StD	Mean	StD	Mean	StD	Origin	Mean	StD	Mean	StD	Mean	StD
ALL	1.21	6.34	-0.29	4.15	0.17	3.79	LCJ	5.19	5.63	-4.36	4.63	1.01	3.97
ABZ	9.64	7.97	-4.85	4.35	2.1	4.12	LHR	1.17	7	0.98	4.49	-0.31	3.87
AMS	0.72	6.15	0.07	4.05	0.29	3.73	LIL	-4.5	6.11	0.86	4.11	3.56	5.76
ARN	-6.67	7.93	0.68	4.1	0.23	3.79	LIM	-1.62	4.1	0.54	3.09	0.09	2.59
ATH	3.71	5.42	-1.48	3.63	-0.38	3.34	LIS	0.57	7.51	-0.87	4	0.77	4.27
ATL	-0.84	6.09	-1.5	4.38	2.24	4.3	LNZ	1.04	7.12	-0.24	4.23	-0.38	3.56
AUH	9.18	7.14	-2.39	3.91	-0.78	3.73	LOS	-2.97	4.48	0.1	3.7	0.58	3.13
BAH	2.6	5.78	-3.67	4.64	2.4	4.47	LYS	1.96	6.5	0.01	4.16	-0.81	3.71
BCN	2.14	6.31	-0.11	4.23	-0.03	3.61	MAA	-4.35	4.82	-1.47	3.53	2.27	3.25
BER	-2.69	6.3	-0.54	4.06	-1.54	3.59	MAD	-0.77	6.72	1.02	4.59	-0.61	3.91
BEY	4.54	6.63	-0.34	4.13	0.49	4.21	MAN	-2.9	6.72	0.98	4.26	0.66	3.57
BIO	9.03	8.7	-2.84	4.77	-2.17	4.3	MCT	-2.62	5.31	-0.33	3.53	0.96	3.24
BKK	-2.51	4.67	-0.03	3.47	2.57	3.94	MEX	9.75	8.79	-1.9	5.45	-1.34	4.65
BLL	-0.12	6.2	0.71	4.1	-0.02	3.53	MIA	2.73	7.27	-1.27	4.5	-0.98	4.24
BLR	-1.9	5.06	-0.85	3.68	1.35	3.33	MMX	-1.46	7.2	0.08	3.78	-0.07	3.47
BOD	4.62	6.62	-1.51	4.03	0.19	3.64	MNL	-3.78	5.63	1.97	2.96	-1.01	2.82
BOG	6.42	6.77	-1.93	3.41	-0.71	3.39	MRS	0.16	5.81	-0.26	4.05	0.05	3.54
BOM	-0.85	5.19	-0.73	3.69	0.41	3.28	MST	-0.95	5.07	1.42	3.81	-0.54	3.55
BQY	0.56	7.03	0.37	4.12	-0.09	3.84	MTY	8.05	6.85	-5.24	4.15	1.27	4.5
BRE	-6.66	6.17	1.44	3.9	0.2	3.52	MUC	2.97	6.83	-0.52	4.19	-0.24	3.63
BRU	1.99	6.41	-0.36	4.17	-0.26	3.61	MXP	2.97	6.15	-0.15	3.89	-0.43	3.83
BSL	-0.21	5.11	0.43	3.35	-0.24	2.89	NCE	-1.65	5.33	0.97	3.77	0.18	3.42
BUD	-2.83	5.28	1.5	3.83	0.05	3.09	NGO	1.62	4.25	-0.65	3.49	0.25	3.26
CAI	-4.21	7.64	1.37	4.24	-0.55	3.44	NOU	5.31	5.94	0.39	3.22	-1.07	3.15
CDG	1.56	5.83	-0.33	4.08	0.35	3.6	NRT	1.34	6.04	-1.03	4.18	1.21	3.92
CGK	1.21	6	-0.08	4.17	0.8	3.98	NTE	-6.88	5.74	5.55	4.84	-1.3	3.48
CGN	2.72	6.43	-0.3	4.23	-0.43	3.71	NUE	1.58	5.87	0.72	4.4	-1.08	3.63
CMN	0.21	4.67	-1.74	3.36	0.67	3.18	OPO	4.77	7.74	-1.06	4.78	-1.73	4.09
CPH	-1.42	4.97	0.85	3.66	-0.44	3.02	ORD	3.37	7.94	1.35	5.52	1.81	5.06
CPT	12.96	7.51	-1.37	4.35	-1.64	3.97	ORY	0.83	4.87	-0.34	3.04	-0.73	2.72
CTU	0.45	4.54	-3.16	3.61	4.58	3.81	OSL	4.46	6.7	-0.37	3.84	-1.03	3.3

Continued on next page

Table A.1 – continued from previous page

Booking	Length		Height		Width		Booking	Length		Height		Width	
Origin	Mean	StD	Mean	StD	Mean	StD	Origin	Mean	StD	Mean	StD	Mean	StD
DEL	1.72	6.44	-0.22	3.37	0.07	3.36	OTP	1.31	6.11	-0.87	4.35	-0.72	3.68
DFW	-2.84	6	0.76	3.66	1.64	3.66	PEK	-1.06	5.42	-0.56	4.15	1.44	3.67
DMM	8.06	7.45	-1.68	4.21	-1.63	3.74	PEN	1.05	5.3	1.32	3.97	-1.23	3.25
DUB	-1.7	5.53	3.55	5.17	-1.04	3.29	PER	-1.68	7.05	0.17	3.83	3.5	4.39
DUS	1.78	6.1	-0.13	3.85	-0.26	3.34	PRG	1.75	6.23	0.1	4.21	-1.16	3.61
DWC	5.65	6.2	-2.36	4.02	0.25	3.91	PTY	-14.28	4.81	3.84	3.73	7.61	4.02
DXB	-0.13	6.05	0.21	4.21	0.45	4	PVG	-3.04	4.8	-2.63	4	4.92	3.98
EIN	0.83	6.28	-1.42	4.28	0.46	3.43	RTM	-1.87	5.91	0.87	3.77	-0.08	3.39
EZE	-0.77	6.97	0.58	4.31	1.45	4.01	RUH	-3.63	4.94	3.03	3.7	-1.43	3.47
FCO	-0.48	5.43	1.19	3.29	-0.43	3.52	SCL	-1.79	4.83	-2.18	3.96	3.88	4.12
FLR	7.66	7.24	-0.38	4.3	-0.7	4.6	SFO	-2.36	5.23	0.36	4.41	0.85	3.87
FMO	4.51	6.31	-1.3	4.06	-0.9	3.48	SIN	0.21	4.98	-0.01	3.71	0.19	3.23
FRA	0.18	5.9	0.16	4.09	-0.21	3.45	SJO	-2.15	4.04	0.62	3.21	0.65	3.07
GDL	-4.88	5.09	0.69	3.94	3.44	3.44	SJU	14.08	4.13	2.61	4.11	4.64	3.21
GIG	0.08	4.71	1.28	3.5	0.2	3.2	SOF	1.28	6.11	0.62	3.99	-1.49	3.15
GLA	2.91	6.53	-0.55	4.34	-0.21	3.55	STR	1.88	5.86	0.13	4.13	-0.55	3.59
GOA	-1.26	5.44	1.32	3.63	-1.43	3.3	SVG	6.85	6.28	-4.09	3.67	1.49	3.44
GOT	3.55	5.89	-0.65	4.27	-0.18	3.31	SVO	4.09	5.26	-2.16	3.43	0.38	3.22
GRU	2.21	5.2	-1.97	4.04	1.07	3.63	SWK	3.09	7.38	-0.29	4.48	0.67	4.24
GRZ	2.47	6.83	-0.58	4.23	-1.06	3.9	SXB	-8.95	6.7	0.83	4.23	0.62	3.47
GUA	-3.88	2.84	0.58	2.81	1.09	2.32	TLS	0.71	6.73	-0.74	4	0.76	3.8
GVA	-0.26	4.56	0.45	3.41	-0.63	2.79	TLV	-0.97	4.08	-0.74	3.08	2.01	3.28
HAJ	2.05	5.78	-0.57	3.52	-0.26	3.03	TPE	-1.01	4.42	-0.91	3.63	1.78	3.67
HAM	1.62	6.15	0.12	3.79	-0.39	3.45	TRN	0.79	5.61	2.95	4.4	-2.2	3.87
HEL	4.4	6.17	-1.43	3.56	-0.33	3.22	TUN	-1.4	4.78	-0.07	3.11	-0.57	3.02
HOH	-0.38	5.55	1.61	3.59	-1.57	2.94	TXL	-0.1	5.53	2.17	4.34	-1.55	3.5
HYD	-6.67	3.25	0.85	2.96	3.45	2.96	VCE	2.37	6.33	-0.06	4.21	-1.29	3.71
IAD	2.08	6.09	-0.05	3.83	1.36	4.33	VCP	12.27	7.27	-2.32	4.62	-1.66	4.81
IAH	-2.33	6.88	-3.4	4.47	-1.21	4.35	VIE	-2.81	5.96	0.2	4.21	0.25	3.74
ICN	0.51	5.32	-0.17	3.66	0.62	3.67	VRN	3.27	8.36	1.78	5.01	-1.49	4.27
IST	2.88	7.24	-2.31	4.73	5.34	5.26	WAW	3.71	6.79	-0.36	4.18	0.18	3.43
JFK	-0.03	6.04	-0.01	4.34	-0.65	3.91	YEG	8.34	6.46	-5.44	4.03	1.28	3.92
JNB	3.02	7.05	-1.15	4.12	-0.19	4.05	YUL	2.97	6.86	-1.79	4.52	0.27	4.1
KIX	3.35	5.91	-2.65	3.82	2.08	3.98	YVR	-0.65	5.49	-1.24	4	1.64	3.78
KUL	0.02	5.01	1.04	4.07	-0.56	3.6	YYC	-3.21	6.59	-1.29	4.1	3.44	4.27
KWI	8.61	7.02	-2.29	4.26	-0.85	3.86	YYZ	0.89	6.45	-0.21	4.58	0.92	3.81
LAX	2.85	6.84	-0.76	4.23	0.95	3.95	ZRH	7.47	6.84	-2.87	4.59	0.47	3.77

Bibliography

[1] Lisser Abdel and Lopez Rafael. Stochastic quadratic knapsack with resource. *Electronic Notes in Discrete Mathematics*, 36:97–104, 2010.

[2] Nicos Christofides and Eleni Hadjiconstantinou. An exact algorithm for orthogonal 2-d cutting problems using guillotine cuts. *European Journal of Operational Research*, 83:21–38, 1995.

[3] Claudia D'Ambrosio, Fabio Furini, Michele Monaci, and Emiliano Traversi. On the product knapsack problem. *Springer-Verlag GmbH Germany, part of Springer Nature*, 2017.

[4] dr. G. Becheri, dr. W. Ruszel, and dr. M. Joosten. Probability and statistics for aerospace engineering (lr), September 2015.

[5] Jens Egeblad and David Pisinger. Heuristic approaches for the two- and three-dimensional knapsack packing problem. *Computers & Operations Research*, 36:1026–1049, 2007.

[6] Samir Elhedli, Fatma Gzara, and Yi Feng Yan. A mip-based slicing heuristic for three-dimensional bin packing. *Springer-Verlag Berlin Heidelberg*, pages 1547–1563, 2017.

[7] D. Fayard and G. Plateau. An algorithm for the solution of the 0-1 knapsack problem. *Computing*, 28: 269–287, 1982.

[8] Antonio Fernández, Consolación Gil, Raúl Baños, and María G. Montoya. A parallel multi-objective algorithm for two-dimensional bin packing with rotations and load balancing. *Expert Systems with Applications*, 40:5169–5180, 2013.

[9] Qian Hu, Lijun Wei, and Andrw Lim. The two-dimensional vector packing problem with general costs. *Omega*, 74:59–69, 2018.

[10] Leonardo Junqueira, Reinaldo Morabito, and Denise Sato Yamashita. Three-dimensional container loading models with cargo stability and load bearing constraints. *Elsevier Computers & Operations Research*, 39:74–85, 2010.

[11] R. Klees and R. P. Dwight. Applied numerical analysis, June 2012.

[12] M. R. Koch. Optimising the uld packing for the air crago industry - literature study, 2018.

[13] Andrea Lodi and Michele Monaci. Integer linear programming models for 2-staged two-dimensional knapsack problems. *Math. Program*, 94:257–278, 2003.

[14] Luis Fernando Mingo López, Nuria Gómez Blas, and Alberto Arteta Albert. Multidimensional knapsack problem optimization using a binary particle swarm model with genetic operations. *Soft Comput*, 22: 2567–2582, 2017.

[15] Célia Paquay. *The Three-Dimensional Rectangular Multiple Bin Size Bin Packing Problem with Transportation Constraints*. PhD thesis, HEC Liège, 2014.

[16] Célia Paquay, Sabine Limbourg, and Michaël Schyns. A mixed integer programming formulation for the three-dimensional bin packing problem deriving from an air cargo application. *International Transactions Inoperational Research*, 23:187–213, 2014.

[17] Judith Redi. Ti2736-a computational intelligence neural networks, September 2014.

[18] Bingyu Song, Yanling Li, Yuning Chen, Feng Yao, and Yingwu Chen. A repair-based approach for stochastic quadratic multiple knapsack problem. *Knowledge-Based Systems*, 145:145–155, 2018.

[19] Gregory S. Taylor, Yupo Chan, and Ghulam Rasool. A three-dimensional bin-packing model: Exact multicriteria solution and computational complexity. *Springer Science+Business Media New York*, pages 397–427, 2015.

[20] Alessio Trivella and David Pisinger. The load-balanced multi-dimensional bin-packing problem. *Elsevier Computers & Operations Research*, 74:152–164, 2016.

[21] D. D. Tönissen, J. M. van den Akker, and J. A. Hoogeveen. Column generation strategies and decomposition approaches for the two-stage stochastic multiple knapsack problem. *Computers on Operations Research*, 83:125–139, 2017.