Operational Research and Machine Learning Applied to Transport Systems

Igor Deplano

A thesis submitted in partial fulfilment of the requirements of Liverpool John Moores University for the degree of Doctor of Philosophy

July, 2020

Declaration

The work presented in this thesis was carried out at the Liverpool Logistics Offshore and Marine Research Institute, Liverpool John Moores University. Unless otherwise stated, it is the original work of the author.

While registered as a candidate for the degree of Doctor of Philosophy, for which submission is now made, the author has not been registered as a candidate for any other award. This thesis has not been submitted in whole, or in part, for any other degree.

Igor Deplano
Liverpool Logistics Offshore and Marine Research Institute
Faculty of Engineering and Technology
Liverpool John Moores University
Byrom Street
Liverpool
L3 3AF
UK

July 1, 2020

Abstract

The New Economy, environmental sustainability and global competitiveness drive innovations in supply chain management and transport systems. The New Economy increases the amount and types of products that can be delivered directly to homes, challenging the organisation of last-mile delivery companies. To keep up with the challenges, delivery companies are continuously seeking new innovations to allow them to pack goods faster and more efficiently. Thus, the packing problem has become a crucial factor and solving this problem effectively is essential for the success of good deliveries and logistics.

On land, rail transportation is known to be the most eco-friendly transport system in terms of emissions, energy consumption, land use, noise levels, and quantities of people and goods that can be moved. It is difficult to apply innovations to the rail industry due to a number of reasons: the risk aversion nature, the high level of regulations, the very high cost of infrastructure upgrades, and the natural monopoly of resources in many countries. In the UK, however, in 2018 the Department for Transport published the Joint Rail Data Action Plan, opening some rail industry datasets for researching purposes.

In line with the above developments, this thesis focuses on the research of machine learning and operational research techniques in two main areas: improving packing operations for logistics and improving various operations for passenger rail.

In total, the research in this thesis will make six contributions as detailed below.

The first contribution is a new mathematical model and a new heuristic to solve the Multiple Heterogeneous Knapsack Problem, giving priority to smaller bins and considering some important container loading constraints. This problem is interesting because many companies prefer to deal with smaller bins as they are less expensive. Moreover,

giving priority to filling small bins (rather than large bins) is very important in some industries, e.g. fast-moving consumer goods.

The second contribution is a novel strategy to hybridize operational research with machine learning to estimate if a particular packing solution is feasible in a constant O(1) computational time. Given that traditional feasibility checking for packing solutions is an NP-Hard problem, it is expected that this strategy will significantly save time and computational effort.

The third contribution is an extended mathematical model and an algorithm to apply the packing problem to improving the seat reservation system in passenger rail. The problem is formulated as the Group Seat Reservation Knapsack Problem with Price on Seat. It is an extension of the Offline Group Seat Reservation Knapsack Problem. This extension introduces a profit evaluation dependent on not only the space occupied, but also on the individual profit brought by each reserved seat.

The fourth contribution is a data-driven method to infer the feasible train routing strategies from open data in the United Kingdom rail network. Briefly, most of the UK network is divided into sections called berths, and the transition point from one berth to another is called a berth step. There are sensors at berth steps that can detect the movement when a train passes by. The result of the method is a directed graph, the berth graph, where each node represents a berth and each arc represents a berth-step. The arcs represent the feasible routing strategies, i.e. where a train can move from one berth. A connected path between two berths represents a connected section of the network.

The fifth contribution is a novel method to estimate the amount of time that a train is going to spend on a berth. This chapter compares two different approaches, AutoRegressive Moving Average with Recurrent Neural Networks, and analyse the pros and cons of each choice with statistical analyses. The method is tested on a real-world case study, one berth that represent a busy junction in the Merseyside region.

The sixth contribution is an adaptive method to forecast the running time of a train journey using the Gated Recurrent Units method. The method exploits the TD's berth information and the berth graph. The case-study adopted in the experimental tests is the train network in the Merseyside region.

Acknowledgements

There are many people to thank in my long journey:

Firstly I would like to express my sincere gratitude to my Director of studies Dr Trung Thanh Nguyen, not only for the continuous support of my Ph.D study, his suggestions and his patience, but also for giving me the opportunity to challenge my skills, alongside my Ph.D study, in many research projects. He has been an invaluable source of teachings, knowledge and an example for myself.

I would like to thank my advisor and collegue Dr Charly Lersteau for our discussions and his useful comments; David Stamper, Chris Ellery, and Sheen Mathew from Merseyrail, for their useful inputs and Justin Willett from the RSSB for kindly reviewing the chapters 6, 7 and 8; my supervisors Dr Qian Zhang and Prof Abir Hussain for their useful comments; Dr Danial Yazdany, Yannis Ancele, Dr Ran Wang and Ahmed Makki for the useful discussions done in the Liverpool Logistics Offshore and Marine Research Institute. I would like to thank my examiners Dr Ben Matellini and Prof. Juergen Branke for their useful suggestions. I would like to thanks also Prof Giovanni Squillero and Dr Alberto Tonda, who initiated me into the research profession and convinced me to start a Ph.D.

I would like to give a special thanks to my family: firstly, my parents for their almost infinite patience and support in whatever choice I have made throughout my long path, secondly to my deceased uncles Antonio and Angelo who motivated and taught me as if I were their son.

Last but not the least, I would like to thank my friends and my enemies, who taught me the shades of life and have contributed to make me a better person.

The research done in Chapter 3, Chapter 4, Chapter 5 has been supported by an LJMU PhD scholarship, an NRCP-funded project no NRCP1617-6-125 managed by the Royal Academy of Engineering. The research done in Chapter 6, Chapter 7, Chapter 8 has been supported by an LJMU PhD scholarship, and an RSSB-funded project no COF-INP-05. The data used in Chapter 6, Chapter 7, Chapter 8 were provided by the Merseyrail, the RSSB and Network Rail. Some of the data contains information of Network Rail Infrastructure Limited licensed under the following licence: www.networkrail.co.uk/data-feeds/terms-and-conditions

Declaration of Authorship

I, Igor Deplano, declare that this thesis titled, 'Operational Research and Machine Learning Applied to Transport Systems' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"Success is not final, failure is not fatal: it is the courage to continue that counts."

Winston S. Churchill

Contents

List of Figures

List of Tables

Abbreviations

1DKP	1 **D**imensional **K**napsack **P**roblem
2SP	2 **D**imensional **S**trip **P**acking problem
3DBPP	3 **D**imensional **B**in **P**acking **P**roblem
3DKPP	3 **D**imensional **K**napsack **P**acking **P**roblem
3D-OBP	3 **D**imensional **O**rthogonal **B**in **P**acking problem
ACF	**A**uto **C**orrelation **F**unction
ARMA	**A**uto **R**egressive **M**oving **A**erage
BM	**B**oltzman **M**achine
BPP	**B**in **P**acking **P**roblem
CIS	**C**ustomer **I**nformation **S**ystem
CLP	**C**ontainer **L**oading **P**roblem
CNN	**C**onvolutional **N**eural **N**etwork
CoM	**C**entre **o**f **M**ass
CSP	**C**onstraint **S**atisfation **P**roblem
DfT	**D**epartment **f**or **T**ransport
DT	**D**ecision **T**ree
FFNN	**F**eed **F**orward fully connected **N**eural **N**etwork
GRASP	**G**reedy **R**andomized **A**daptive **S**earch **P**rocedure
GRU	**G**ated **R**ecurrent **U**nit
GSR-KPPS	**G**roup **S**eat **R**eservation - **K**napsack **P**roblem with **P**rofit on **S**eat
GSR-KP	**G**roup **S**eat **R**eservation - **K**napsack **P**roblem
GSR	**G**roup **S**eat **R**eservation
HN	**H**opfield neural **N**etwork
IBPMS	**I**nfrastructure-**B**orne **P**erformance **M**easurement **S**ystem
ISO	**I**nternational **O**rganization for **S**tandardization

LSTM	**L**ong **S**hort-**T**erm **M**emory
MBSBPP	**M**ultiple **B**in **S**ize **B**in **P**acking **P**roblem
MHKP	**M**ulti **H**eterogeneous **K**napsack **P**roblem
MIP	**M**ixed **I**nteger linear **P**roblem
NN	**N**eural **N**etwork
NP-complete	**N**ondeterministic **P**olynomial time complete
OTDR	**O**n **T**rain **D**ata **R**ecorder
PPM	**P**ublic **P**erformance **M**easure
RBM	**R**estricted **B**oltzman **M**achine
RF	**R**andom **F**orest
RNN	**R**ecurrent **N**eural **N**etwork
SPAD	**S**ignal **P**assed **A**t **D**anger
SVM-P	**S**upport **V**ector **M**achine with **P**olinomial kernel
SVM-R	**S**upport **V**ector **M**achine with **R**adial kernel
TDS	**T**train **D**etection **S**ystem
TD	**T**train **D**escriber
TNV	**T**rein-**N**ummer**V**olgsystemen
TOC	**T**rain **O**perating **C**ompany
TOPS	**T**otal **O**perations **P**rocessing **S**ystem
TRUST	**T**train **R**unning **U**nder **S**ystem **TOPS**
TSP	**T**ravelling **S**alesman **P**roblem
UK	**U**nited **K**ingdom
WFBF	**W**eight **F**irst **B**est **F**it

To my parents . . .

Chapter 1

Introduction

1.1 Background

The supply chain management and the transport systems are a pivotal point of every economy. Nowadays, the New Economy has increased the number and types of products that can be delivered directly to the home, challenging the organisation of the last-mile delivery companies. Packing problems are crucial for these companies and the competitiveness in the sector continuously motivates their innovation. In the meantime, on the mainland rail transport is known to be the most eco-friendly transport system in terms of emissions, energy consumption, land use and noise levels to move large quantities of goods and people. The innovation in the rail industry is challenging: it is a risk averse industry, a highly regulated environment, infrastructure upgrades are expensive and in most countries a natural monopoly of the resources. However, in 2018, the Department for Transport of the United Kingdom published the Joint Rail Data Action Plan, opening some rail industry datasets for researching purposes.

In this thesis we will develop techniques to address four main problems found in the transport systems and more generally in the supply chain management.

Section 1.2 introduces a packing problem that nowadays is important, e.g, for last-mile delivery. Section 1.3 introduces the importance of finding a methodology to hybridise some techniques of the operational research field with machine learning methods to produce instantly approximated results. Section 1.4 introduces a problem found in the train seat reservation systems, whilst section 1.5 describes more in detail TD live feed that is going to be used in this thesis to improve the rail transport system.

1.2 Packing problems

The cutting and packing class of problems (Wäscher et al., 2007) has been studied in the field of Operational Research since the 1940s because of its wide range of applications in areas such as logistics, scheduling, and manufacturing.

In chapter 3 we develop a mixed integer linear programming model (MIP) for the Multi-Heterogeneous Knapsack Problem (MHKP), considering the Container Loading Problem (CLP) constraints including stability, weight limits, weight distribution and load bearing constraints (Bortfeldt and Wäscher, 2013) and giving filling priority to smaller bins. We want to extend the state of the art considering items with an arbitrary centre of mass and restricting the acceptable bins' global centre of mass in a pyramidal region. This model can be used as a base model for further research on heuristics for both the knapsack packing and bin packing problems. We study the impact of the CLP constraints on the complexity of the solution space. The pyramidal region of choice is a trade-off between the need for a packing solution and achieving desirable conditions for safe container handling operations such as lifting and truck transport.

The MHKP consists of packing a strongly heterogeneous set of box-shaped items into a weakly heterogeneous set of box-shaped bins such that the wasted space is minimized or equivalently the bins' occupied space is maximized (Bortfeldt and Wäscher, 2013). The item profit is equal to the item volume, which differs from the general Knapsack problem where items with the same volume may have different profits. Items must be placed inside the borders of the bin without overlapping each other and without floating in mid-air. We consider orthogonal rotation of items over the vertical axis.

A key difference between knapsack problems and bin packing problems is that the former considers profit maximization whereas the latter focuses on the minimization of the number of bins used (Martello and Toth, 1990; Kellerer et al., 2004). In our case, we are interested in the minimization of the wasted space and prioritizing smaller bins over bigger ones. This is because companies often prefer to deal with smaller bins as they are less expensive. Giving priority to filling small bins (rather than large bins) is very important in some industries, e.g. fast-moving consumer goods. An example is food delivery companies like Deliveroo and Uber Eats, where it is necessary to prioritise smaller boxes that can be carried by bicycles over larger boxes that have to be carried by cars/vans/trucks. This is so that the food can be delivered to every part of the area in the shortest time. Similarly, for last-mile delivery companies such as Amazon Logistics, Amazon Flex, myHermes, DPDLocal and the like, priority will be given to smaller bins since they can fit into small vans and cars owned by the couriers. Other examples can be found in last-mile delivery for pharmaceutical, consumer electronic, personal care,

household care products, branded and packaged food, spirits and tobacco. The similar prioritisation also underlies the principle of the Physical Internet, which is considered the future of logistics. In the Physical Internet initiative (Montreuil, 2012, 2011), priority is given to packing goods into small, modular boxes if possible. Then thanks to modularisation the smaller boxes can be attached together to be carried in bigger containers if needed. We exploit the work done so far in the Physical Internet by using their containers' dimensions as a proxy for dealing with real-world instances. In Sallez et al. (2016), Physical Internet Containers or π-containers are divided into three main categories: T-container, H-container, and P-container. Table 1.1 illustrates their planned dimensions: T-containers aim to replace 20 feet and 40 feet containers, H-containers are designed to fit in a pallet sized surface, and P-containers have various dimensions aiming to directly contain goods. Containers inside the same category can be combined into groups and locked together (composition) while a container of a smaller category can be encapsulated in a container of a larger category (encapsulation). The locking mechanism is still under development but considering the available options (Landschützer et al., 2015), it is reasonable to assume that items will be placed over a grid (positioning will be restricted to certain points).

TABLE 1.1: Table of dimensions of π-containers (container types marked with * have ten dimension modifiers: 100% | 90% | ... | 20% | 10%)

Type	length	width	height
T-container	1.2m, 2.4m, 3.6m, 4.8m, 6m, 12m	1.2m, 2.4m	2.4m, 1.2m
H-container *	1.2m	1.2m	2.4m
P-container *	1.2m	1.2m	2.4m

It is important to underline that even if the problem instances arise from the Physical Internet Containers, the problem can be adapted to any common containers or box types by restricting the bins to standard dimensions, e.g. 20 feet and 40 feet for ISO-standard containers. From now on, we will adopt the common bin packing vocabulary: T-containers will be called bins and H-containers will be called items.

1.3 Using machine learning to estimate feasibility of packing solutions in constant time - a novel strategy

Hybridising operational research with machine learning algorithms has raised interest in the research communities in the last thirty years. The reason is that exploiting successful machine learning technologies can help solving large scale optimisation problems that describe the real world.

One of the aims of this thesis is to develop a novel strategy to use machine learning to estimate if a particular packing solution is feasible in a constant O(1) computational time. In chapter 4 we combine one of the oldest decomposition techniques in operational research literature with some of the latest results in machine learning literature. The combination will provide an improved feasibility checking in constant time O(1), in comparison to the computational cost inherent in solving an NP-Hard problem. The strategy has been tested using the three dimensional bin packing problem, but can be beneficial also to problems that have the knapsack or bin packing as a subproblem, e.g. the travelling salesman problem. We highlight hereby that the methodology remains valid for any number of dimensions.

The strategy consists of two stages. In the first stage, we exploit the Bender's decomposition to split the problem into a master/slave architecture, where the master allocates the resources, and the slaves certify in parallel the feasibility of the proposed packing solutions. Despite being inefficient as a strategy to directly find an optimal solution, it is efficient to rapidly build a dataset of feasible and unfeasible solutions. In the second stage we exploit the dataset built so far to train a classifier for checking the feasibility of a packing solution.

1.4 The Offline Group Seat Reservation Knapsack Problem with Profit on Seats

In chapter 5 we extend the Offline Group Seat Reservation Knapsack Problem (GSR-KP) presented in Clausen et al. (2010). In the original formulation, a train with W seats stops in H stations. It is required to allocate n reservations. Each reservation i occupies a set of contiguous seats for w_i people from one initial station y_i to a final one h_i. The profit is identified as to maximise the space occupied during the journey. In our extension the value of the profit of the reservation is dependent also on the profits assigned to seats in which the reservation is eventually allocated. Our extension makes the problem more realistic, allowing the modelling of scenarios that were not possible to model with the original formulation. The new scenarios cover all the problems where the ideal position of an item is affected by how long the item must keep the position. We exploit the original naming style and call the new extension Group Seat Reservation Knapsack Problem with Profit on Seat (GSR-KPPS). Moreover, solving realistically sized instances is challenging for a general solver and often having a good solution rapidly may be better than having an optimal solution later, e.g. when there are fixed time constraints. Thus, we suggest a new GRASP procedure that solves GSR-KP and GSR-KPPS. Eventually, we adapt

and improve the original instances considered in Clausen et al. (2010) adding a random profit on seats and proposing five new problems.

The GSR-KP is the problem of maximising the use of seats in a train during its journey. In the offline version, the passengers reserve a seat from a departing station until their arrival station. Each reservation is known in advance and before the train departs. A reservation can occupy one or more seats. Groups of people are considered to be willing to sit on close seats.

GSR-KP belongs to the family of the packing problems in two dimensions. In the packing terminology, the reservations are the items, and the train is the bin. The bin and items are rectangles. Packing rectangles into a rectangle is a strongly NP-Complete class of problems (Leung et al., 1990). Regarding the bin, the dimension of the side parallel to the horizontal axis represents the number of seats, while the dimension of the vertical side represents the journey length of the train. For each item, the dimension of the horizontal side represents the number of people in the reservation, while the dimension of the vertical side represents the journey length of the reservation. The dynamic of a reservation consists of reserving a seat, or a group of seats, from a departing station to an arrival station. This special behaviour is modelled by a special constraint which forces the vertical position of the item.

We propose to extend GSR-KP to create a new model that can distribute the allocation of passengers based on their journey length and the profit of the seats, e.g. allocate reservations for long journeys or groups in the centre of the carriage, and reservations for short journeys or unitary groups near doors, reducing the excess friction during the boarding/alighting phases. Another strategy would be to distribute passengers evenly across carriages such that the number of passengers exiting/entering in a particular station is similar over all carriages. This strategy has been adopted considering the real-time scenario in Yazdani et al. (2019).

Another notable application is in the events industry, e.g different stands may cost differently depending on their location and size. Applications as such can also be modelled using this newly proposed model, considering the lending requests as reservations with time and size, while the price paid to the lender is dependent on the position in which the request will be placed. A similar problem exists also in the tourism industry, for example in the booking system of an hotel, different rooms may have a different profit.

Our work can be especially meaningful for the United Kingdom (UK) rail industry (UK, 2016; Hatano, 2004). The UK rail industry is an open market, Train Operators are private, or a mix of private and public, companies in competition on the main corridors. In longer journeys, i.e. from Liverpool to London, booking a seat in advance is the

common rule of thumb to avoid standing up for the whole journey. Train Operators are interested to reduce delays to improve the Public Performance Measure and gain a competitive advantage over competitors.

1.5 Exploiting data-driven methodologies to improve the performance of the United Kingdom Railway System

Nowadays, railway transport is one of the most eco-friendly transport systems for goods and people. The number of passengers of the UK's railway industry has been more than doubled over the last 25 years, and the industry is facing growing operational performance challenges. The *public performance measure* (PPM) is one of the industry standards to monitor Train Operating Companies' (TOCs) performances. The PPM index, until April 2019, considered the amount of delays measured at destination. The delays considered are higher than a threshold and without any distruption on the planned schedule, i.e. skipping station or cancelling journey. From April 2019, there is a step change from PPM to a basket of new metrics that will provide greater granularity of actual performance on route as well as arrival at destinations, rather than referring back to the recording to the minute. Fig 1.1 shows the PPM values aggregated from all TOCs in Great Britain. The data is since 2011, and each period is 28 days. We highlighted in red the PPM trendline, which shows a consistent decline.

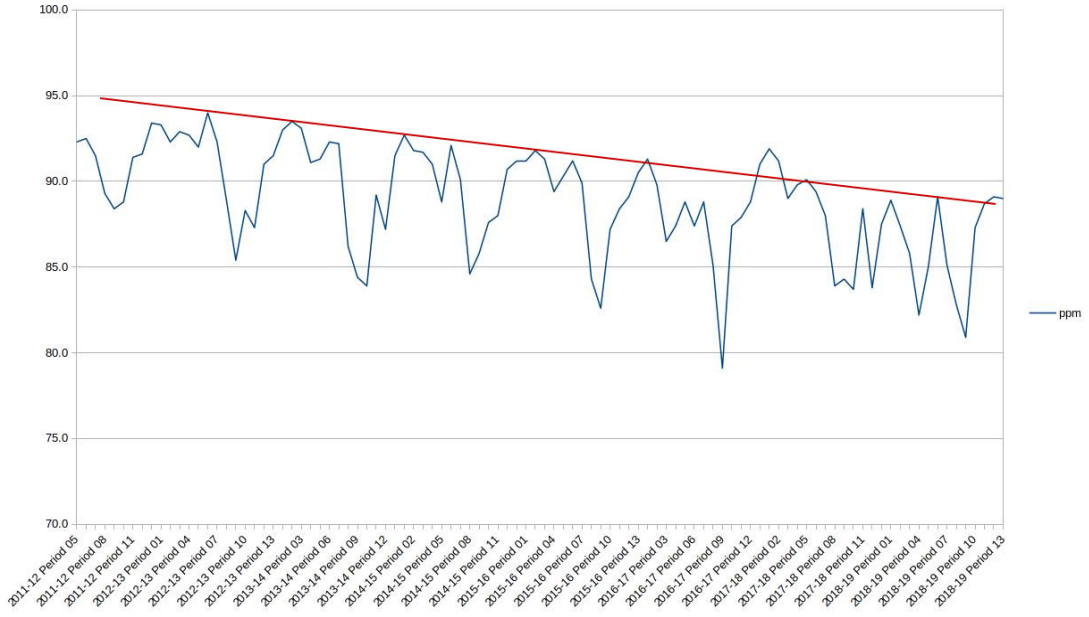


FIGURE 1.1: PPM in the last 8 years, with a trendline highlighted in red. source Office of Rail and Road on data of Network Rail(Rail, b)

As previously introduced, the Department for Transport (DfT) published the Joint Rail Data Action Plan, opening some rail industry datasets for researching purposes. The data sources include the main public and private actors, i.e. Network Rail, National Rail Enquiries, Transport for London and the Rail Delivery Group. Some of these datasets are produced by the Infrastructure-Borne Performance Measurement Systems (IBPMS) that monitor the train movements on physical tracks using Train Detection Systems (TDS). The work in Palmer (2010) describes the TDS technologies in use nowadays. For a broader survey of the train's communication technologies we refer to (Fraga-Lamas et al., 2017). The lowest level of detail achievable using an IBPMS is through the Train Describer (TD) system (Toossi et al., 2017). The TD system is also the basis of the TRUST system, a higher level view of the network traffic that includes information about the schedule. One key note is that TRUST is less precise than TD. Actually, TRUST rounds the train delays to the minute, thus the delay evolution during the journey can become inaccurate considering the new PPM metric.

Most of the railway network is partitioned into geographical *areas*, each one identified by 2-byte area code. An area is divided into track sections called *berths*, each identified by a 4 byte code. However, some locations may be associated to more than one berth code. For example, there could be a code for the signal and another for the berth, both at the same location. Another example is the case of double tracks where the railway has two tracks running in parallel in two directions. In this case, trains passing the same location in two directions will trigger two different berth codes.

The transition point between two berths is called *berth step*. A *berth step* is usually located near interlocking parts of the track (sections, switches and signals). The sensors measure the arrival time at the *berth-step* and this information, along with the train running number (also called *headcode*) is collected in the Train Describers (TD) information system. In TD, the presence of a train at a specific time is identified by a four-character code called the *headcode*.

This thesis will utilise TD data to improve the performance of UK railway systems. Specifically, Chapter 6 exploits TD data to develop a data-driven method to infer useful infrastructure and feasible train routing strategies from open data in UK real network. Chapter 7 exploits the results of Chapter 6 and develops a novel method to estimate the amount of time that a train is going to spend on a berth, whereas Chapter 8 exploits the findings to develop a method to successfully forecast the train running time along a journey.

1.6 Summary of contributions

The contributions of this thesis are summarised as follows:

- In Chapter 3
  - A new mixed-integer linear model that solves the Multiple Heterogeneous Knapsack Problem, for the first time giving priority to smaller bins and considering the following constraints: non overlapping, boundaries and positioning (both constrained and free), rotations (around the vertical axis), orthogonal displacement, weight limit, static stability, weight distribution in a pyramidal region and load bearing considering items' arbitrary centre of mass.
  - A study of the trade-off of adding more constraints to make the problem more realistic and the complexity of finding a solution.
  - New metrics that facilitate the comparison of datasets used in experiments.
  - A new constructive heuristic named Weight First Best Fit to handle large scale instances in a reasonable time.
- In Chapter 4
  - A novel two-stages strategy to exploit machine learning to estimate if a particular packing solution is feasible in a constant O(1) computational time: the first stage exploits the master/slave Bender's decomposition to build a dataset of knapsack solutions, whilst the second stage exploits the dataset to train a classifier for the satisfiability problem.
  - A new dataset of packing solutions and a study on benchmarking different classification algorithms.
  - A comparison of the classification performances in the new dataset for the following algorithms: decision trees (DT), random forest (RF), support vector machine with radial basis function kernel (SVM-R), support vector machine with polynomial kernel (SVM-P), three different architectures of convolutional neural networks (CNN), feed forward fully connected neural networks (FFNN) with one, two and three hidden layers.
- In Chapter 5
  - A mixed-integer linear model for the Group Seat Reservation Knapsack Problem with Price on Seat, an extension of the the Offline Group Seat Reservation Knapsack Problem. We introduce a profit evaluation dependent on not only

the space occupied, but also on the individual profit brought by each reserved seat.

  - A new GRASP based algorithm that solves the original problem and the newly proposed one.
  - New problem instances that represent real world scenarios more realistically.

- In Chapter 6
  - A new algorithm to automatically generate the *berth graph*, a graph that represents the feasible train routing strategies through the network of berths.

- In Chapter 7
  - A new development of two different approaches to estimate the amount of time that a train is going to spend on a berth. The first approach uses an AutoRegressive Moving Average model, the second approach uses the Gated Recursive Unit and the Long short-term memory methods.
  - An analysis on these two approaches, which shows that the best results can be obtained by networks with input sizes that were covering the statistically significant spikes of the AutoCorrelation Function.

- In Chapter 8
  - A new development of two different forecasting systems based on Gated Recurrent Unit models and the *berths graph*. The first one utilises an input of the immediate previous n running times. The second one, instead, uses the previous n running times available 4 hours before the start of the journey.
  - A new sequential training procedure that is a trade-off between accuracy, and training time. Each round improves the model knowledge of the time series. The motivations for building such a procedure are: the berths have different distributions, for some berths, there may be lack of samples or a very low variance distribution, while other berths may have richer variability.
  - An analysis on the performance of the proposed methods, in comparison to the current industry system called Darwin.
  - A new procedure that filters the communication of the new forecasting based on the exceeding of a threshold.

Chapter 2

Literature review

The literature review is organised in four sections, one for each main problem.

2.1 Multiple Heterogeneous Knapsack Problem and Container Loading Problem constraints

This section focuses on mathematical linear models for the three dimensional bin packing problem (3DBPP) or three dimensional knapsack packing problem (3DKPP) that tackle some CLP constraints such as weight distribution, load-bearing, and stability. Readers interested in approximated methods are referred to Zhao et al. (2016a); Coffman Jr et al. (2013); Christensen et al. (2017).

The first application-oriented literature review on cutting and packing problems was published by Sweeney and Paternoster (1992), including papers written since 1940. Coffman Jr et al. (2004) published a bibliography on the bin packing problem. Martello and Toth (1990) and successively Kellerer et al. (2004) made a detailed analysis of the broad class of knapsack problems. Dyckhoff (1990) made the first categorization of the cutting and packing problems, further improved by Wäscher et al. (2007).

According to Bortfeldt and Wäscher (2013), the most frequently used constraints in the CLP can be categorized as follows: weight distribution, loading priorities, orientation, stacking or load-bearing; cargo-related: complete-shipment, allocation, positioning; load-related: stability, complexity patterns. For a state-of-the-art review of the CLP algorithms that also considers heuristics and metaheuristics, readers are referred to the work of Zhao et al. (2016a).

The geometrical constraints are common to any packing problem (Crainic et al., 2012b). They are about positioning items inside bins ensuring that they will not overlap and that they will not exceed the bin's dimensions. Positioning can be modelled in absolute terms (in a Cartesian system where the origin is in one vertex of the bin) or in relative terms (the position is relative to other items inside the bin) (Bortfeldt and Wäscher, 2013). Considering only the underlying geometrical constraints (orthogonal packing inside the bins), BPP remains an NP-Hard problem in a strong sense. Delorme et al. (2016) recently published a survey on exact methods focusing on the one-dimensional version of the problem, while for approximation methods to solve the multiple-dimensional version, the most recent surveys are by Christensen et al. (2017, 2016).

Regarding the CLP constraints, weight distribution refers to displacing items inside the bin in order to obtain an overall centre of mass inside a safe region or to minimize the distance from a reference point, which is usually placed in the centre of the container. The centre of mass position is considered important (Bortfeldt and Wäscher, 2013) because it can affect the safety of operations during lifting with cranes, vehicle stability and asymmetric tyre wear.

Trivella and Pisinger (2016) modelled this problem as an MIP model for the multidimensional case, not considering rotations, nor load-bearing and assuming a homogeneous density of rectangular shaped boxes. This implies that a bin's centre of mass falls in the centre of the box. The model is too hard to solve even for small instances. Thus, the authors proposed a multi-level local search heuristic that exploits the Fekete-Schepers interval graphs representation (Fekete et al., 2007; Fekete and Schepers, 2004).

Paquay et al. (2016) tackled a multiple bin size bin packing problem (MBSBPP) considering the following constraints: geometrical (non overlapping, boundaries, rotations and positioning), orthogonal displacement, weight limit, orientation, special shapes of the containers, stability, weight distribution and fragility (this is a special case of load-bearing, where items cannot bear any load). Even in this paper, the mass of items is considered homogeneous.

Load-bearing is the limit of weight that can be stacked over a box. Often it is represented as "do not place more than n boxes" or "do not place box i over box j". Junqueira et al. (2012) tackled the loading of a single bin considering vertical stability (the ability of boxes to not fall/move along the vertical axis) and horizontal stability (the capacity of boxes to not move along the axes that form the horizontal plane). Their load-bearing is constrained inside a 100% vertical stability, which implies that no gaps between items are allowed. Jin et al. (2003) modelled the 3DBPP with stability, heterogeneous bins, and rotations. The large-scale case is solved with an algorithm based on Tabu-Search for the assignment phase and a sub-volume heuristic for the packing phase. Hifi et al. (2010)

modelled a 3DBPP for packing identical bins without considering additional constraints. They proposed a set of lower bound inequalities based on analogies with the parallel-machine scheduling problems. Ceschia and Schaerf (2013) developed two algorithms, one based on Tabu-Search and the other on Simulated Annealing. They considered rotations, load-bearing, multi-drop, weight limits and stability (full-support). De Queiroz and Miyazawa (2013) tackled the oriented two-dimensional strip packing problem (2SP), considering load balancing, load-bearing, and a multi-drop constraint.

The above review shows that currently there is no exact method for solving the MHKP with the following constraints despite these being very common in real scenarios: non overlapping, boundaries and positioning (both constrained and free), rotations (around vertical axis), orthogonal displacement, weight limit, static stability, feasible weight distribution in a pyramidal region and load-bearing considering items' arbitrary centre of mass. Chapter 3 aims to fill this gap by proposing a mixed-integer linear model that solves MHKP giving priority to smaller bins and considering all the aforementioned constraints. Having such a model will help further research on heuristics and approximations models.

2.2 Using machine learning to estimate feasibility of packing solutions in constant time - a novel strategy

Historically, the first approaches to solving combinatorial optimisation problems with neural networks (NN) were done with energy based models like Hopfield Networks (HN) (Hopfield and Tank, 1985), Boltzman machines (BM), and their restricted version (RBM). Some examples can be found in Looi (1992) for the Travelling Salesman Problem (TSP), Ohlsson et al. (1993) for the one-dimensional Knapsack Problem (1DKP), Ramanujam and Sadayappan (1988) for vertex cover and other graph problems. Smith (1999) reports a review of their applications in different combinatorial optimisation problems. Other important classes of applications are the optimisation of nonlinear problems (Kennedy and Chua, 1988), non-convex problems (Zhang et al., 2018) and the solution of partial differential equations (Kumar and Yadav, 2011; Rudd and Ferrari, 2015; Han et al., 2018).

More recently, researchers explored solving TSP with FFNN hybridized with nature inspired algorithms (Masutti and de Castro, 2009), co-adapative NN (Cochrane and Beasley, 2003) and pointer networks (Vinyals et al., 2015). Dual Hopfield NN has been used to solve the mixed-binary quadratically constrained quadratic problem (Travacca and Moura, 2018) and recurrent NN has been applied for the convex optimisation (Qin

et al., 2018). Reiforcement learning algorithms have been exploited in the solution of combinatorial problems, i.e. 1DKP, TSP (Bello et al., 2016) and 3DBP (Hu et al., 2018, 2017; Jin, 2017; Laterre et al., 2018). Deep Convolutional NN achieved good results with the Constraint Satisfation Problem (CSP) (Xu et al., 2018), predicting the satisfiability in a balanced dataset for a small sized problem instance. Lachhwani (2019) surveys the application of NN to the solution of mathematical programming problems.

Between the oldest techniques for decomposing and solving linear programming problems there are Dantzig-Wolfe decomposion (Dantzig and Wolfe, 1960) (columns generation) and Bender's decomposition (Benders, 1962; Geoffrion, 1972) (rows generation). These techniques are deeply connected (Dantzig and Thapa, 2006a,b), in fact it is known that they are the dual of each other.

We exploit Bender's decomposion, which has been used previously in the strip packing problem (Côté et al., 2014), the orthogonal cutting stock problem (Delorme et al., 2017, 2015) and constraint programming (Eremin and Wallace, 2001; Cambazard and Jussien, 2005). A recent review of its applications can be found in (Rahmaniani et al., 2017).

The above review shows that currently the Bender's decomposition has never been exploited to build a classifier, whilst no classifier has been built to test the feasibility of packing instances. Chapter 4 aims to fill this gap by proposing a two-stage method that combines the Bender's decomposition with machine learning. Having such a method will help further research on heuristics and approximations models.

2.3 The Offline Group Seat Reservation Knapsack Problem

To the best of our knowledge, since the original publication of the problem in Clausen et al. (2010), none of the follow-up studies on the Group Seat Reservation Problem has shown to be better than the original work. An online version of the seat allocation problem was first published in Boyar and Larsen (1999), and further analysis was made in Goyal (2018). A real-time algorithm that aims to reduce the boarding/alighting time by maintaining a uniform load on carriages through systematic distribution of passengers with flexible tickets has been recently proposed by the authors in Yazdani et al. (2019).

Many papers have been published in the more general Packing Problems context, some examples of new approximation approaches are genetic algorithms (Gupta et al., 2017; Gonçalves and Resende, 2011b; Jegadeshwari and Jaisree, 2014; Wang and Chen, 2010) and their biased versions (Gonçalves and Resende, 2011a, 2013), divide and conquer algorithms(in which the solution space is partitioned and searched independently) (Wei et al.,

2013), neuro-genetic approaches that mix neural networks and genetic algorithms(Deane and Agarwal, 2013), GRASP algorithms(Resende and Ribeiro, 2019) and GRASP/Path relinking (Alvarez-Valdés et al., 2013), Tabu search (Ceschia and Schaerf, 2013; Crainic et al., 2009) and other greedy randomized heuristics (Perboli et al., 2011; Crainic et al., 2012a).

The GSR is a specialised version of the bin packing problem in the two dimensional case, so every algorithm that has been designed for orthogonal two dimensional rectangular packing will work on a GSR problem. The difference in our contribution is that none of them can exploit the nature of the problem, which is that in a two dimensional problem both dimensions are free. In the original GSR, the allocation toward one dimension is constrained. Chapter 5 aims to fill this gap.

2.4 United Kingdom railway research

Several studies tried to exploit Infrastructure-Borne data to analyse train operations. There are few research studies published on the UK railway systems: (Toossi et al., 2017) introduced Train-Borne and Infrastructure-Borne measurements, (Van Gulijk et al., 2015; Zhao et al., 2016b; Rashidy et al., 2018; Zhao et al., 2018) investigated Signal Passed At Danger (SPAD) events utilising the TD-S-CLASS messages, (Martin, 2016) utilised Bayesian reasoning to predict delays in real-time, (Balfe, 2010) studied the effects of automation in the rail signalling systems.

The low level Infrastructure-Borne measurement system used in the Dutch railway (TNV-system (TNV) (Kecman et al., 2011)) has a logic that resembles the combination of the UK's TD and TRUST. We exploit only TD, but it is useful to make a comparison between the UK and Dutch system to avoid confusion because, despite having a similar logic, the systems are quite different from each other.

The Train Describers, in TNV, "are a unique number (per day) identifying the train line service (characterizing the train type, terminal stations, route, and served stations) and including a discriminating counter for successive trains" (Goverde and Hansen, 2000). These informations, on the UK's system, are available only on TRUST, while in TD there is no information about the schedule and stations. Regarding the size of the Train Describer's data fields TNV uses a field of 9 byte length (Kecman et al., 2011), UK's TD headcode length is 4 bytes. The first one are unique per day, the second one may be unique, but in reality it depends on the frequency of service of the area code.

In the UK system the first byte is devoted to the train class ([0-9]), the second to the train line, and the last two are incremental numbers. There are also exceptions,

i.e. obfuscation for freight or postal service (it is applied on TOC request) and special messages (i.e. speed restrictions). The most updated documentation on the UK's system is held on Open Rail Data Wiki, n.d..

As a result, existing work for the Netherlands system cannot be used for the UK system. This prompts the need for studies specifically made for the UK system.

The first research work on TNV is TNV-Prepare (Goverde and Hansen, 2000), it processed the TNV's logs and converted them into a suitable format. TNV data has been used then in delay extraction and analysis (Goverde and Hansen, 2001; Goverde and Meng, 2011), validation of simulation models (Tromp, 2004), detection of route conflicts and reactionary delays (Goverde et al., 2007; Daamen et al., 2008; Goverde and Meng, 2011; Kecman and Goverde, 2013b).

Train event times are predicted in Hansen et al. (2010), using timed event graphs, and in Kecman and Goverde (2013a) utilising a timed event graph with dynamic arc weights. Regarding dwell time and prediction of running time, Kecman and Goverde (2015) utilised least-trimmed squares robust linear regression, regression trees and random forests. (Becker and Schreckenberg, 2018) utilised the analytical approach. Dwell time prediction, for short stop stations, has also been tackled in Li et al. (2014, 2016, 2018).

Train delays have been predicted using Bayesian networks (Corman and Kecman, 2018; Lessan et al., 2018), a fuzzi Petri net model (Milinković et al., 2013), Berger et al. (2011) presented a stochastic model for delays propagation and forecasts based on directed acyclic graphs. Pongnumkul et al. (2014) used data-driven models, in particular using autoregressive integrated moving average and neighbour models. Oneto et al. (2018) exploited the technology stack of Apache Spark and shallow and deep extreme learning machines. Yaghini et al. (2013) used neural networks, whereas Oneto et al. (2018) used a hybrid decision tree. A different, but related topic, has been shown in Shen et al. (2019), where they compared multiple linear regression, and random forest regression, to predict the recovery from the disruption caused by primary delays.

The above review shows that currently there is no published method to infer from the raw data a graph model that represents the feasible train routing strategies through the network of berths, Chapter 6 aims to fill this gap. Currently there is no published research that exploits the UK TD-C data to forecast the time that a train spends in a berth, nor the forecast of the arrival time at the berth-step of a train journey. Chapter 7 and Chapter 8 aim to fill these gaps.

Chapter 3

A mixed-integer linear model for the Multiple Heterogeneous Knapsack Problem with realistic container loading constraints

The chapter proposes a mixed-integer linear model that solves the Multiple Heterogeneous Knapsack Problem, giving priority to smaller bins and considering the following constraints: non overlapping, boundaries and positioning (both constrained and free), rotations (around the vertical axis), orthogonal displacement, weight limit, static stability, weight distribution in a pyramidal region and load bearing considering items' arbitrary centre of mass.

The first contribution is an MIP Model for the MHKP considering some CLP constraints. The main novelty of the model is the arbitrary items' centre of mass (CoM) and the acceptable global bins' CoM pyramidal shape. This creates a model that, for the first time, covers both the pallet-loading and the container-loading instances with or without interlocking mechanisms.

The second contribution is a study of the impact of the CLP constraints on the complexity of the solution space.

The third contribution is the introduction of new metrics to compare the datasets used for the experimental part. An effort in this direction is important because faster algorithms are often specialised algorithms that efficiently handle only particular cases of the problem (Wolpert and Macready, 1997). Having such metrics will also improve the export of theoretical results in the industry.

The last contribution is the introduction of a simple deterministic heuristic named Weight First Best Fit (WFBF). WFBF is an on-line heuristic that fills the available bins with one item at a time, sequentially, until the available bins or the available items are exhausted. The algorithm is capable of handling any problem size in a reasonable time.

The chapter is organized as follows. Section 3.1 discusses the model in detail. Section 3.2 discusses the heuristic WFBF and its limitations. Section 3.3 describes the dataset used, and section 3.4 shows and discusses the computational experiments used to validate the model. The chapter will end summarizing the results and with a discussion of future research.

3.1 Problem definition

Definition 3.1 (Item)**.** An item $i \in I$ is a parallelepiped object with length l_i, width w_i and height h_i. The possible dimensions are combinations of a predefined set of dimensions (in this chapter, without any loss of generality, we take the dimensions specified in Table 1.1 as an example for experimental purposes). Each item has its load bearing limit Λ_i, which is the maximal weight that the item can bear on its top surface. Moreover, each item has a centre of mass expressed in coordinates κ_i^x, κ_i^y, κ_i^z.

Definition 3.2 (Bin)**.** A bin $j \in J$ is a parallelepiped object with length L_j, width W_j and height H_j. The possible dimensions are combinations of the ones given in Table 1.1. Every bin has weight limit Ω_j, which represents the maximal weight that can be allocated inside.

Definition 3.3 (Bin's Centre of Mass)**.** The feasible global centre of mass of bin j is described by the coordinates of its vertex, ϱ_j, and the ratio of its pyramidal base, ξ_j. The pyramid base is proportional to the bin's size, Fig 3.1 shows an example. The ideal CoM is placed in the centre of the bin's bottom face. This position maximises the safety for crane lifting operations, truck transport and eventually manual handling. The choice of the pyramidal region is a trade-off between the ideal CoM and the need to have feasible solutions. Restricting the base of the pyramid and moving the vertex will result in having a feasible region nearer to the ideal one.

Definition 3.4 (Grid-based positioning)**.** The positions inside the bin along the plane parallel to the ground are restricted by a grid. Each grid cell is a square with β indicating the length of the side. β is the greatest common divisor of all items' sides. This choice ensures item alignment. Items' positions are aligned on the grid. Note that this grid-based positioning can be changed to free positioning by setting $\beta = 1$. Fig. 3.2 illustrates the concept.

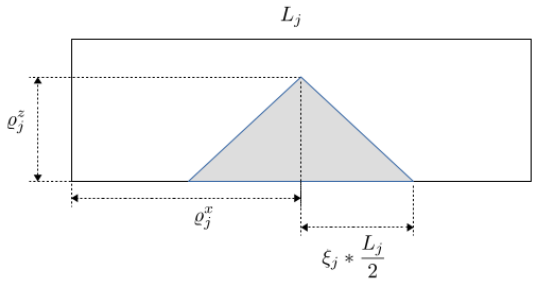


FIGURE 3.1: Description of the feasible centre of mass pyramidal space

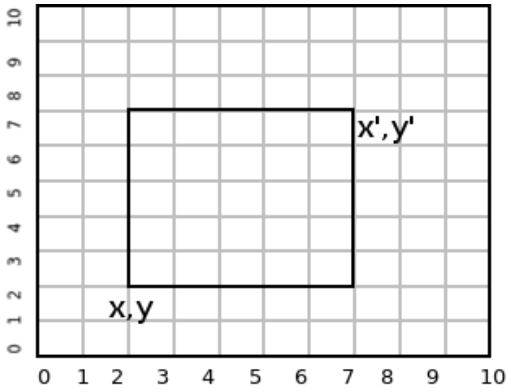


FIGURE 3.2: Description of the feasible position along the x/y plane inside the bin

Definition 3.5 (Rotation of items)**.** Every item can rotate orthogonally around its vertical axis. The number of feasible rotations is two. Table 3.1 describes possible orientations.

TABLE 3.1: Table of orientations

Orientation Type	Parallel to x-axis	Parallel to y-axis	Parallel to z-axis
1	Length	Width	Height
2	Width	Length	Height

Definition 3.6 (Reflection)**.** The reflection operator is a π radiant rotation that affects only the item centre of mass. Fig 3.3 shows an example.

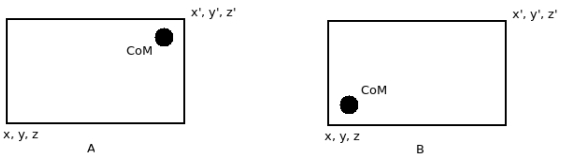


FIGURE 3.3: A shows an item and its CoM, B shows the item's CoM after the reflection is applied. Note that the operator affects only the CoM position.

Definition 3.7 (Coordinate systems)**.** The proposed model uses two coordinate systems. Fig 3.4 shows the description. The first coordinate system has as its origin the bin's front-bottom-left corner, and is used to describe the positions inside the bin. The second

coordinate system has as its origin the item's front-bottom-left corner and is used for defining the position of the CoM in an item.

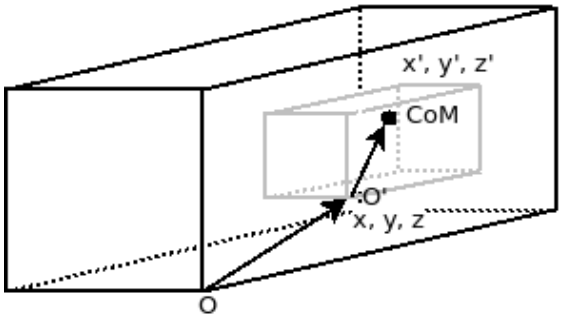


FIGURE 3.4: Description of the coordinate system

TABLE 3.2: Model overview

min	wasted space giving priority to smaller bins
st:	
	each item must be assigned to at most one bin
	assigned items can rotate orthogonally along the vertical axis
	assigned items must be positioned on a grid
	assigned items must be placed inside bins limits
	assigned items must not overlap
	assigned items have to be placed on the ground or one or more items, they cannot float in mid air
	the sum of the items' weight assigned to a bin must be lower than or equal to the bin's supported weight limit
	for each assigned item, the stacked weight over it must be lower than its load bearing limit
	the global centre of mass of every bin must fall inside a pyramidal region

One of the aims of this chapter is to give a mathematical model for the problem in Table 3.2. We developed the model considering items with an arbitrary centre of mass (CoM) and restricting the acceptable bins' global CoM in a pyramidal region. Items can rotate orthogonally around the vertical axis. This implies that there are two feasible rotations for every item. The overview is shown in Table 3.1. The rotation applied to an item affects the position of its CoM. Moreover, as we are considering arbitrary CoM, the model has to recalculate the CoM considering an additional operation, which is a π radiant rotation that we call reflection, that affects only the CoM of the item. Items are placed orthogonally, that is, items' sides are aligned with bin's sides.

In Table 3.3 we list all the model's parameters, Fig 3.5 shows a two-dimensional representation of the parameters and some variables. The variables of the model are presented in Table 3.4.

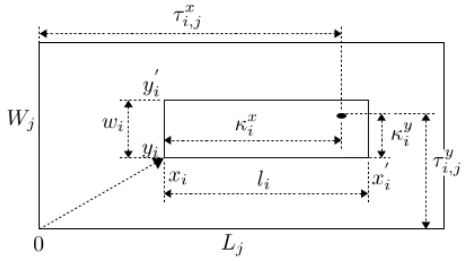


FIGURE 3.5: Parameters and variables representation in two dimensions. The larger parallelepiped is the bin, the internal one is the item. For the sake of simplicity, this figure does not present rotations.

The rotation modelling is inspired by the work of Lin et al. (2014). $\varphi_{o,i} \in \{0, 1\}$ is a binary variable that equals 1 if the orientation of item $i \in I$ follows orientation type $o \in O = \{1, \ldots, r\}$, where r is the number of the feasible rotations for an item (i.e 2). $\varphi_{o,i}$ will be 0 otherwise. The orientation types are defined in Table 3.1. We are considering only 2 orthogonal rotations in the plane x/y because it is a common practice to avoid any rotation of the z-axis. For example, in the MODULUSCHA project (MoDULUSHCA, 2012), an H-container has a male/female mechanism on the top/bottom surface that limits the feasible rotations to only two along the ground plane. Moreover, items and bins are regular parallelepipeds and since we are aiming for optimal configuration, considering non-orthogonal rotations will only increase the complexity of the problem without improving solutions.

Items are aligned in a grid where each cell is a square with β indicating the length of the side. Aligned positioning is a common practice, examples being pallet positioning into ISO-standard containers, boxes inside a pallet or the locking mechanism (Landschützer et al., 2015) in the Physical Internet containers. Furthermore, this practice reduces the feasible search space and can be disabled by assigning $\beta = 1$. It is preferable that the solution includes the items' absolute positioning using one of the bin's corners as the origin of the Cartesian system. We define $\beta * x_i$, $\beta * y_i$, z_i, the coordinates of the front-bottom-left item's corner.

3.1.1 The model

The model's objective is to minimize the wasted space giving filling priority to smaller bins.

Table 3.3: List of parameters

Name	Description
J	Set of bins' indexes
I	Set of items indexes
O	Set of available rotations
n	Cardinality of J
m	Cardinality of I
L_j	Length of bin j
W_j	Width of bin j
H_j	Height of bin j
L	Max length of bins in J
W	Max width of bins in J
H	Max height of bins in J
l_i	Length of item i
w_i	Width of item i
h_i	Height of item i
ω_i	Weight of item i
Ω_j	Weight limit of bin j
Λ_i	Load-bearing limit of item i
ϱ_j	Vertex of the pyramid for bin j, $\varrho_j^{x,y,z}$ are the coordinates using the origin of the bin j as reference system.
κ_i	centre of mass vector for item i
κ_i^s	With $s \in \{x, y, z\}$ coordinate s of the centre of mass for item i
ξ_j	Radius in the base of the pyramid for bin j
β	Dimension of the side of the square cell grid

Table 3.4: List of variables

Name	Description
x_i, y_i	Position of the item i in the grid, $\beta \cdot x_i$, $\beta \cdot y_i$, are the respective coordinates of the front-bottom-left edge
z_i	Coordinate of the front-bottom-left edge of the item i
$x_i^{'}, y_i^{'}, z_i^{'}$	Coordinate of the back-top-right edge of the item i.
$\varphi_{o,i}$	1 if the applied rotation for the item i is $o \in O$, otherwise 0
$S_{i,k}$	1 if item i and item k are assigned to the same bin, otherwise 0
Z_j	1 if bin j is considered in the solution, otherwise 0
$C_{i,j}$	1 if item i is assigned to bin j, otherwise 0
$x_{i,k}^p, y_{i,k}^p, z_{i,k}^p$	1 if item i is placed after item k along the coordinate x, y and z, which means respectively $x_k^{'} \leq x_i$, $y_k^{'} \leq y_i$, $z_k^{'} \leq z_i$, otherwise 0
$\pi_{i,k,p}$	$p \in \{0, 1, 2, 3\}$, 1 if the corner p of the item k is on the top surface of the item i, otherwise 0
$\theta_{i,m}$	1 if item i is placed on the ground of the bin, otherwise 0
$\theta_{i,k}$	1 if item i is placed on the top of item k, otherwise 0
R_i	1 if reflection is applied on item i, otherwise 0
$\upsilon_i^x, \upsilon_i^y$	Coordinate of the centre of mass of the item i after applying the item rotation and reflection, the reference is the origin of item i
$\tau_{i,j}^x, \tau_{i,j}^y, \tau_{i,j}^z$	Coordinate of the centre of mass for the item i using as reference the origin of the bin j
$\lambda_{i,k}$	1 if item k must be considered for the sum of the weights that are placed over the item i, otherwise 0

$$\text{minimize} \qquad \sum_{j \in J} \frac{(L_j \cdot W_j \cdot H_j - \sum_{i \in I}(l_i \cdot w_i \cdot h_i \cdot C_{i,j}))}{(L_j \cdot W_j \cdot H_j)} \tag{3.1}$$

The objective (3.1) is equivalent to maximizing the packing efficiency. The proof for equivalence is trivial and is shown in Appendix A. Note that, using rational representation, the objective satisfies our requirements without having to add other penalties. There are other ways to model the objective to satisfy this requirement. For example, giving unused bins penalty zero would be a good approach if we added a specific constraint to force all items to be allocated.

The model is subject to the following constraints.

$$C_{i,j} \leq Z_j \qquad \forall i \in I, j \in J \tag{3.2a}$$

$$\sum_{j \in J} C_{i,j} \leq 1 \qquad \forall i \in I \tag{3.2b}$$

$$\sum_{i \in I} C_{i,j} \geq Z_j \qquad \forall j \in J \tag{3.2c}$$

$$\sum_{o \in O} \varphi_{o,i} = \sum_{j \in J} C_{i,j} \qquad \forall i \in I \tag{3.2d}$$

Our problem considers selecting a subset of items and bins. The total allocation of items is not mandatory because there may be a set I of items such that, given a set J of bins, there exists no valid solution with total allocation. Constraint (3.2a) forces to 0 every $C_{i,j}$ if the bin j has not been considered in the solution. Constraint (3.2b) ensures that every item is assigned to at most one bin, while constraint (3.2c) states that at least one item must be placed in an opened bin. Bins without at least one item inside are not considered a valid solution. Constraint (3.2d) forces only one rotation if the item is assigned, O are the available rotations, shown in Table 3.1.

$$\sum_{i \in I} C_{i,j} \cdot \omega i \leq \Omega_j \qquad \forall j \in J \tag{3.3}$$

$$\sum_{i \in I} C_{i,j} \cdot l_i \cdot w_i \cdot h_i \leq L_j \cdot W_j \cdot H_j \qquad \forall j \in J \tag{3.4}$$

Constraint (3.3) and (3.4) ensure that the contents of each bin do not exceed a weight limit and a volume limit, respectively.

The positioning and non-overlapping constraints are based on the ones proposed by Paquay et al. (2016) adapted with the rotation style found in Lin et al. (2014). The main difference between them is that Paquay et al. (2016) adopted one boolean variable per side orientation, while Lin et al. (2014) adopted one variable per rotation. In detail, constraints (3.5a)-(3.5c) define the borders of the items along the axes, taking into account the possible rotation.

$$x_i^{'} - \beta \cdot x_i = l_i \cdot \varphi_{0,i} + w_i \cdot \varphi_{1,i} \qquad \forall i \in I \tag{3.5a}$$
$$y_i^{'} - \beta \cdot y_i = l_i \cdot \varphi_{1,i} + w_i \cdot \varphi_{0,i} \qquad \forall i \in I \tag{3.5b}$$
$$z_i^{'} - z_i = h_i \qquad \forall i \in I \tag{3.5c}$$

Constraints (3.6a)-(3.6c) ensure that the shape of each item is contained inside the borders of the selected bin. We precompute $L = \max_{j \in J} L_j$, $W = \max_{j \in J} W_j$, $H = \max_{j \in J} H_j$ as the maximum dimensions available for the considered set of bins. L, W and H are used as big M constraints.

$$x_i^{'} \leq \sum_{j \in J} C_{i,j} \cdot W_j + (1 - Z_j) \cdot W \qquad \forall i \in I \tag{3.6a}$$
$$y_i^{'} \leq \sum_{j \in J} C_{i,j} \cdot L_j + (1 - Z_j) \cdot L \qquad \forall i \in I \tag{3.6b}$$
$$z_i^{'} \leq \sum_{j \in J} C_{i,j} \cdot H_j + (1 - Z_j) \cdot H \qquad \forall i \in I \tag{3.6c}$$

Constraints (3.7a)-(3.7d) deal with the case when two items i,k are allocated to the same bin. The constraints enforce that $S_{i,k}$ is equal to 1 if and only if item i is in the same container as item k. $S_{i,k}$ is useful in other parts of the model.

$$S_{i,k} \leq \sum_{j \in J} C_{i,j} \qquad \forall j \in J, \forall i,k \in I | i \neq k \qquad (3.7a)$$

$$S_{i,k} \leq \sum_{j \in J} C_{k,j} \qquad \forall j \in J, \forall i,k \in I | i \neq k \qquad (3.7b)$$

$$S_{i,k} \geq C_{k,j} + C_{i,j} - 1 \qquad \forall j \in J, \forall i,k \in I | i \neq k \qquad (3.7c)$$

$$S_{i,k} + C_{i,j} + C_{k,l} \leq 2 \qquad \forall j,l \in J | l \neq j, \forall i,k \in I | i \neq k \qquad (3.7d)$$

The non overlapping constraints are basically the same as in Paquay et al. (2016) with minor modifications: we utilized $S_{i,k}$ on the right part of constraint (3.8a) and stated clearly the coordinates considering the position in the grid. Constraint (3.8a) states that two items do not overlap if the boundaries along at least one dimension do not overlap. The overlapping boundaries are characterized by the constraints (3.8b)-(3.8d).

$$x^p_{i,k} + y^p_{i,k} + z^p_{i,k} + x^p_{k,i} + y^p_{k,i} + z^p_{k,i} \geq S_{i,k} \qquad \forall i,k \in I | i \neq k \qquad (3.8a)$$

$$x^{'}_k \leq \beta \cdot x_i + (1 - x^p_{i,k}) \cdot W$$

$$\beta \cdot x_i + 1 \leq x^{'}_k + x^p_{i,k} \cdot W \qquad \forall i,k \in I | i \neq k \qquad (3.8b)$$

$$y^{'}_k \leq \beta \cdot y_i + (1 - y^p_{i,k}) \cdot L$$

$$\beta \cdot y_i + 1 \leq y^{'}_k + y^p_{i,k} \cdot L \qquad \forall i,k \in I | i \neq k \qquad (3.8c)$$

$$z^{'}_k \leq z_i + (1 - z^p_{i,k}) \cdot H \qquad \forall i,k \in I | i \neq k \qquad (3.8d)$$

$$z^p_{i,k} = 0, x^p_{i,k} = 0, y^p_{i,k} = 0 \qquad \forall i,k \in I | i = k \qquad (3.8e)$$

The stability is defined by constraining every item to lie on the bin's floor (3.9a) or on at least one other item (3.9b)-(3.9p). Constraint (3.9q) excludes the possibility of having one item both simultaneously lying over another one $\theta_{i,k} = 1$ and on the ground ($\theta_{i,m} = 1$). $\pi_{i,k,l}, l \in \{0,1,2,3\}$ represents which corner of item k is under item i.

Constraints (3.9b) and (3.9c) define one of the conditions for item i to be stable over an item k is $z_i = z^{'}_k$, i.e. the top surface of item k is at the same height as the bottom surface of item i. The criteria for defining whether an item is on top of another in the x/y plane are defined respectively in (3.9d) - (3.9g) for the x axis and in (3.9h) - (3.9k) for the y axis. The constraints (3.9l) - (3.9o) enforce that if item k is on top of item i, at least two bottom corners of k should be on the top surface of i to maintain

stability. (3.9p) states that two items can be considered stable only if they are in the same container.

$$z_i \leq H \cdot (1 - \theta_{i,m}) \qquad \forall i \in I \quad (3.9a)$$

$$z_i \leq H \cdot (1 - \theta_{i,k}) + z_k^{'} \qquad \forall i,k \in I | i \neq k \quad (3.9b)$$

$$z_i \geq H \cdot (\theta_{i,k} - 1) + z_k^{'} \qquad \forall i,k \in I | i \neq k \quad (3.9c)$$

$$\beta \cdot x_i + W \cdot (\theta_{i,k} - 1) \leq \beta \cdot x_k + (1 - \pi_{i,k,0}) \cdot W \qquad \forall i,k \in I | i \neq k \quad (3.9d)$$

$$\beta \cdot x_k + W \cdot (\pi_{i,k,0} - 1) \leq x_i^{'} + (1 - \theta_{i,k}) \cdot W \qquad \forall i,k \in I | i \neq k \quad (3.9e)$$

$$\beta \cdot x_i + W \cdot (\theta_{i,k} - 1) \leq x_k^{'} + (1 - \pi_{i,k,1}) \cdot W \qquad \forall i,k \in I | i \neq k \quad (3.9f)$$

$$x_k^{'} + W \cdot (\pi_{i,k,1} - 1) \leq x_i^{'} + (1 - \theta_{i,k}) \cdot W \qquad \forall i,k \in I | i \neq k \quad (3.9g)$$

$$\beta \cdot y_i + L \cdot (\theta_{i,k} - 1) \leq \beta \cdot y_k + (1 - \pi_{i,k,2}) \cdot L \qquad \forall i,k \in I | i \neq k \quad (3.9h)$$

$$\beta \cdot y_k + L \cdot (\pi_{i,k,2} - 1) \leq y_i^{'} + (1 - \theta_{i,k}) \cdot L \qquad \forall i,k \in I | i \neq k \quad (3.9i)$$

$$\beta \cdot y_i + L \cdot (\theta_{i,k} - 1) \leq y_k^{'} + (1 - \pi_{i,k,3}) \cdot L \qquad \forall i,k \in I | i \neq k \quad (3.9j)$$

$$y_k^{'} + L \cdot (\pi_{i,k,3} - 1) \leq y_i^{'} + (1 - \theta_{i,k}) \cdot L \qquad \forall i,k \in I | i \neq k \quad (3.9k)$$

$$\pi_{i,k,0} + \pi_{i,k,1} \geq \theta_{i,k} \qquad \forall i,k \in I | i \neq k \quad (3.9l)$$

$$\pi_{i,k,2} + \pi_{i,k,3} \geq \theta_{i,k} \qquad \forall i,k \in I | i \neq k \quad (3.9m)$$

$$\pi_{i,k,0} + \pi_{i,k,1} \leq \theta_{i,k} \cdot 2 \qquad \forall i,k \in I | i \neq k \quad (3.9n)$$

$$\pi_{i,k,2} + \pi_{i,k,3} \leq \theta_{i,k} \cdot 2 \qquad \forall i,k \in I | i \neq k \quad (3.9o)$$

$$\theta_{i,k} \leq S_{i,k} \qquad \forall i,k \in I | i \neq k \quad (3.9p)$$

$$\sum_{k \in I^+} \theta_{i,k} \geq \sum_{j \in J} C_{i,j} \qquad \forall i \in I \quad (3.9q)$$

$$\theta_{i,k} = 0, \pi_{i,k,l} = 0 \qquad \forall i,k \in I | i = k, \forall l \in 0,1,2,3 \quad (3.9r)$$

We are dealing with items with an arbitrary centre of mass and this condition affects both load-bearing and the bins' global CoM. Before proceeding further, we have to model how the centre of mass is affected by the item rotations and reflection. $\kappa^i = (\kappa_i^x, \kappa_i^y, \kappa_i^z)$ represents the actual centre of mass for the item $i \in I$, where the dimensions are relative to the front-bottom-left corner. We use $\upsilon_i^x, \upsilon_i^y$ to represent the CoM point after applying rotations and reflection. Note that CoM along the z-axis is not affected by rotation or reflection. However, if z-axis rotations (i.e. 6-way rotations) are considered, it is trivial to extend the model here to cover such a case.

In the group of constraints (3.10) we state the centre of mass considering reflection and rotation. (3.10a)-(3.10d) define the x coordinate, R_i is a decision variable that

determines which group of equations must be satisfied. R_i is equal to 1 if there is reflection, 0 otherwise. The first group, (3.10a)-(3.10b) represent the case of no reflection, while (3.10c)-(3.10d) represent the case with reflection. The rotations follow the same schema adopted before in the constraints group (3.5).

$$\upsilon_i^x \leq \kappa_i^x \cdot \varphi_{0,i} + \kappa_i^y \cdot \varphi_{1,i} + R_i \cdot W \qquad \forall i \in I \qquad (3.10a)$$

$$\upsilon_i^x \geq \kappa_i^x \cdot \varphi_{0,i} + \kappa_i^y \cdot \varphi_{1,i} - R_i \cdot W \qquad \forall i \in I \qquad (3.10b)$$

$$\upsilon_i^x \leq x_i^{'} - (\kappa_i^x \cdot \varphi_{0,i} + \kappa_i^y \cdot \varphi_{1,i}) + (1 - R_i) \cdot W \qquad \forall i \in I \qquad (3.10c)$$

$$\upsilon_i^x \geq x_i^{'} - (\kappa_i^x \cdot \varphi_{0,i} + \kappa_i^y \cdot \varphi_{1,i}) - (1 - R_i) \cdot W \qquad \forall i \in I \qquad (3.10d)$$

$$\upsilon_i^y \leq \kappa_i^y \cdot \varphi_{0,i} + \kappa_i^x \cdot \varphi_{1,i} + R_i \cdot L \qquad \forall i \in I \qquad (3.10e)$$

$$\upsilon_i^y \geq \kappa_i^y \cdot \varphi_{0,i} + \kappa_i^x \cdot \varphi_{1,i} - R_i \cdot L \qquad \forall i \in I \qquad (3.10f)$$

$$\upsilon_i^y \leq y_i^{'} - (\kappa_i^y \cdot \varphi_{0,i} + \kappa_i^x \cdot \varphi_{1,i}) + (1 - R_i) \cdot L \qquad \forall i \in I \qquad (3.10g)$$

$$\upsilon_i^y \geq y_i^{'} - (\kappa_i^y \cdot \varphi_{0,i} + \kappa_i^x \cdot \varphi_{1,i}) - (1 - R_i) \cdot L \qquad \forall i \in I \qquad (3.10h)$$

Now we are ready to introduce the constraints that restrict the bins' CoM. We define a pyramidal safe region in which the centre of mass has to lie. This region is defined by the vertex ϱ_j, and has a base radius ξ_j. A possible solution could be given by setting the vertex coordinates in the bin's centre, respectively $\varrho_j^z = \frac{1}{2} \cdot H_j$, $\varrho_j^x = \frac{1}{2} \cdot L_j$ and $\varrho_j^y = \frac{1}{2} \cdot W_j$.

$\tau_{i,j}^x$, $\tau_{i,j}^y$, $\tau_{i,j}^z$ represent the coordinates of the CoM of item i considering as coordinate reference system the origin of the bin j. Constraints (3.11a)-(3.11c) represent the reference coordinate for υ_i^x, (3.11d)-(3.11f) for υ_i^y and (3.11g)-(3.11i) remap κ_i^z, because we do not have rotations over the z axis.

The next group of constraints defines the pyramidal region. The linearization exploits the style in Paquay et al. (2016), defining the feasible region for each bin. The principal differences are the region shape and the introduction of the asymmetry in the items' CoM. (3.11j) and (3.11k) define the pyramid limits along the x axis, (3.11l) and (3.11m) for the y axis, while (3.11n) defines the constraint on the actual height of the CoM that must fall under the region previously defined. ξ_j represents the radius in the base of the pyramid for bin j. We pre-compute $\xi_j = \alpha \cdot L_j$ if $W_j > L_j$ or $\xi_j = \alpha \cdot W_j$ otherwise. α is a parameter to define the base of the pyramid, $\alpha \in (0, 1]$. By default, we suggest

$\alpha = 0.8$ which means 80% of the lower dimension.

$$\tau^x_{i,j} \leq W_j \cdot C_{i,j} \qquad \forall i \in I, j \in J \tag{3.11a}$$
$$\tau^x_{i,j} \leq \upsilon^x_i + \beta \cdot x_i \qquad \forall i \in I, j \in J \tag{3.11b}$$
$$\tau^x_{i,j} \geq \upsilon^x_i + \beta \cdot x_i - W_j \cdot (1 - C_{i,j}) \qquad \forall i \in I, j \in J \tag{3.11c}$$
$$\tau^y_{i,j} \leq L_j \cdot C_{i,j} \qquad \forall i \in I, j \in J \tag{3.11d}$$
$$\tau^y_{i,j} \leq \upsilon^y_i + \beta \cdot y_i \qquad \forall i \in I, j \in J \tag{3.11e}$$
$$\tau^y_{i,j} \geq \upsilon^y_i + \beta \cdot y_i - L_j \cdot (1 - C_{i,j}) \qquad \forall i \in I, j \in J \tag{3.11f}$$
$$\tau^z_{i,j} \leq H_j \cdot C_{i,j} \qquad \forall i \in I, j \in J \tag{3.11g}$$
$$\tau^z_{i,j} \leq \kappa^z_i + z_i \qquad \forall i \in I, j \in J \tag{3.11h}$$
$$\tau^z_{i,j} \geq \kappa^z_i + z_i - H_j \cdot (1 - C_{i,j}) \qquad \forall i \in I, j \in J \tag{3.11i}$$

$$\sum_{i \in I} \tau^x_{i,j} \cdot \omega_i \leq \varrho^x_j \cdot (\sum_{p \in I} \omega_p \cdot C_{p,j}) + \frac{\varrho^z_j \cdot (\sum_{p \in I} \omega_p \cdot C_{p,j}) - \sum_{i \in I} \tau^z_{i,j} \cdot \omega_i}{\varrho^z_j} \cdot \xi_j \qquad \forall j \in J \tag{3.11j}$$

$$\sum_{i \in I} \tau^x_{i,j} \cdot \omega_i \geq \varrho^x_j \cdot (\sum_{p \in I} \omega_p \cdot C_{p,j}) - \frac{\varrho^z_j \cdot (\sum_{p \in I} \omega_p \cdot C_{p,j}) - \sum_{i \in I} \tau^z_{i,j} \cdot \omega_i}{\varrho^z_j} \cdot \xi_j \qquad \forall j \in J \tag{3.11k}$$

$$\sum_{i \in I} \tau^y_{i,j} \cdot \omega_i \leq \varrho^y_j \cdot (\sum_{p \in I} \omega_p \cdot C_{p,j}) + \frac{\varrho^z_j \cdot (\sum_{p \in I} \omega_p \cdot C_{p,j}) - \sum_{i \in I} \tau^z_{i,j} \cdot \omega_i}{\varrho^z_j} \cdot \xi_j \qquad \forall j \in J \tag{3.11l}$$

$$\sum_{i \in I} \tau^y_{i,j} \cdot \omega_i \geq \varrho^y_j \cdot (\sum_{p \in I} \omega_p \cdot C_{p,j}) - \frac{\varrho^z_j \cdot (\sum_{p \in I} \omega_p \cdot C_{p,j}) - \sum_{i \in I} \tau^z_{i,j} \cdot \omega_i}{\varrho^z_j} \cdot \xi_j \qquad \forall j \in J \tag{3.11m}$$

$$\sum_{i \in I} \tau^z_{i,j} \cdot \omega_i \leq \varrho^z_j \cdot \sum_{p \in I} \omega_p \cdot C_{p,j} \qquad \forall j \in J \tag{3.11n}$$

The last part of the model concerns load-bearing. We assume that every item i is not perfectly rigid, thus, can resist the pressure of no more than a specific load, Λ_i. We know the centre of mass, κ^i, but how the mass is distributed inside i remains unknown. The weight that an item is bearing is the sum of the items' weights inside the cuboid space over it. Constraints (3.12a) and (3.12b) restrict the z coordinate of items placed

over item i. Constraints (3.12c)-(3.12f) limit the items having their centre of mass over item i, note that those equations follow the same style used for stability, exploiting the previously defined centre of mass. Constraint (3.12g) ensures that the total weight of items placed over item i does not exceed Λ_i.

$$z_i^{'} \leq z_k + (1 - \lambda_{i,k}) \cdot H \qquad \forall i, k \in I | i \neq k \tag{3.12a}$$

$$\lambda_{i,k} \cdot H + z_i^{'} \geq z_k + 1 \qquad \forall i, k \in I | i \neq k \tag{3.12b}$$

$$(\lambda_{i,k} - 1) \cdot W + \beta \cdot x_i \leq \sum_{j \in J} \tau_{k,j}^{x} \qquad \forall i, k \in I | i \neq k \tag{3.12c}$$

$$\sum_{j \in J} \tau_{k,j}^{x} \leq x_i^{'} + (1 - \lambda_{i,k}) \cdot W \qquad \forall i, k \in I | i \neq k \tag{3.12d}$$

$$(\lambda_{i,k} - 1) \cdot L + \beta \cdot y_i \leq \sum_{j \in J} \tau_{k,j}^{y} \qquad \forall i, k \in I | i \neq k \tag{3.12e}$$

$$\sum_{j \in J} \tau_{k,j}^{y} \leq y_i^{'} + (1 - \lambda_{i,k}) \cdot L \qquad \forall i, k \in I | i \neq k \tag{3.12f}$$

$$\sum_{k \in I} \lambda_{i,k} \cdot \omega_k \leq \Lambda_i \qquad \forall i \in I \tag{3.12g}$$

$$x_i, y_i, z_i, x_i^{'}, y_i^{'}, z_i^{'}, \tau_{i,j}^{x}, \tau_{i,j}^{y}, \tau_{i,j}^{z}, \upsilon_i^{x}, \upsilon_i^{y} \in \mathbb{N}, \qquad \forall i \in I, j \in J \tag{3.13a}$$

$$S_{i,k}, Z_j, C_{i,j}, x_{i,k}^{p}, y_{i,k}^{p}, z_{i,k}^{p}, \theta_{i,m}, R_i, \lambda_{i,k} \in \{0, 1\}, \quad \forall i, k \in I, j \in J, m \in I \bigcup \{m + 1\} \tag{3.13b}$$

$$\pi_{i,k,p}, \varphi_{o,i} \in \{0, 1\}, \qquad \forall i, k \in I, p \in \{0, 1, 2, 3\}, o \in O \tag{3.13c}$$

3.1.2 Model splitting and sizes

From the previous model we extract three submodels called V0, V1, V2, with each including a subset of the constraints. Let n be the number of items and m be the number of bins.

The submodel V0 contains constraints (3.1) - (3.9r) and considers only stability. The size of the model is $9n^2+nm+m+9n$ variables and $n^2m^2-nm^2-n^2m+2nm+25n^2+3m-15n$ constraints.

TABLE 3.5: Size of models

	constraints	variables
V0	$n^2m^2 - nm^2 - n^2m + 2nm + 25n^2 + 3m - 15n$	$9n^2 + nm + m + 9n$
V1-V0	$9nm + 5m + 8n$	$3nm + 3n$
V2-V0	$6n^2 + 9nm + 5m + 3n$	$n^2 + 3n + 3nm$
V2-V1	$6n^2 - 6n + n$	n^2

The submodel V1 includes constraints (3.1) - (3.11n) and considers stability and the distribution of the CoM. The size of the model is $9n^2 + 4nm + m + 12n$ variables and $n^2m^2 - nm^2 - n^2m + 11nm + 25n^2 + 8m - 7n$ constraints.

The submodel V2 includes constraints (3.1) - (3.12g) and considers stability, distribution of the CoM and load bearing. The size of the model is $10n^2 + 4nm + m + 12n$ variables and $n^2m^2 - nm^2 - n^2m + 11nm + 31n^2 + 8m - 12n$ constraints.

3.2 Heuristic Weight First Best Fit(WFBF)

In this section we show a simple deterministic heuristic that we name Weight First Best Fit (WFBF). WFBF is an on-line heuristic that fills the available bins with one item at a time, sequentially, until the available bins or the available items are exhausted. WFBF includes a multi-start procedure in case the available bins have the same volume but different dimensions. The procedure randomises the order of the bins in each unique volume set and tests if there are better solutions. WFBF belongs to the category of constructive methods (Zhu et al., 2012) and it is detailed in Algorithms 1 and 2.

WFBF assumes, without any loss of generality, that it is given as an input the list of bins (*bins*), the list of items (*items*) and the maximum repetitions (*maxLimit*). The list of items is sorted by weight in descending order, discarding the items that do not fit in any available bin. The sorting of the list of items may even be internalised in the algorithm as can be easily calculated by a few lines of code.

The list of bins is sorted with the function $randomisedOrderWithPriority(bins)$.
$randomisedOrderWithPriority$ takes as input the list of bins, and returns the list of bins sorted by "randomised volume ascending".
The steps of $randomisedOrderWithPriority$ are the following: 1) partition the list of bins in classes of bins with the same volume 2) for each class, shuffle the list of bins in the class 3) flatten the classes in volume ascending order. 4) return the flattened array.

$randomisedOrderWithPriority$ is necessary for the multi-starting procedure to explore different combinations when there are bins in the same class that have the same volume

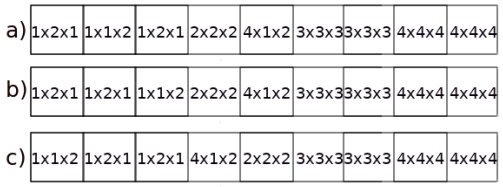


FIGURE 3.6: Example of *randomisedOrderWithPriority* outputs. The bins are sorted by volume ascending, inside each volume class, the order is random. If the bins' shapes are exactly the same, there will not be any difference between every output.

but respectively different shapes. If a class is composed by bins with the same shapes then there is no benefit in shuffling. An example of possible *randomisedOrderWithPriority* outputs is shown in Fig 3.6, given as input any array a,b or c, the output can be any array between a,b and c.

isRandomizable(*bins*) evaluates if the solution may benefit from *randomisedOrderWithPriority*. The procedure is simple, it returns true if for at least one class (in the list of classes of bins with the same volume) there are at least two elements and their shapes are different, false otherwise. This function is used to avoid unnecessary repetitions in the case where the list of bins has every volume class with one element only.

isBetterSolution(*solution*1, *solution*2) returns true if *solution*1 opens fewer bins then a *solution*2. If *solution*2 is empty or not set, then it returns true. The function returns false otherwise.

Let *solution* be the set of final solutions. For each non-empty bin j, let $solution_j$ be the set of items assigned to the bin j with their configuration. The configuration for each item is its position p, its rotation o and a flag r to determine if the reflection is applied.

If the bin list *isRandomizable* then repeat for the *maxLimit* the following filling procedure.

The algorithm fills the next empty bin j if there are still available items. A procedure enumerates all the possible positions in the space of the bin j. This procedure is trivial, and the cardinality of $position_j$ is $\frac{H_j \cdot W_j \cdot L_j}{\beta}$. For each available item i, in every combination of possible positions p, rotations o and reflection r, the algorithm computes a ranking function (Eq: 3.14).

The ranking function ranks the quality of the item configuration, the smaller the rank, the better the configuration. The function promotes configurations that are near the floor of the bin and near the vertical axis placed in the vertex of the bin's feasible centre of mass space. The variables z_i and $\tau_i^{x,y}$ are explained in Table 3.4, respectively. They are the coordinate z of item i and the position of the CoM of item i in the plane

orthogonal to the bin's floor. $\varrho_j^{x,y}$ is listed in Table 3.3 and is the position of the vertex of the feasible CoM region of the bin j.

$$rank \Leftarrow (z_i + \left\| \tau_i^{x,y} - \varrho_j^{x,y} \right\|_2)^2 \tag{3.14}$$

WFBF makes use of three functions, named $isNotOverlapping(solution_j,item_{i,p,o,r})$, $isLoadBearingValid(solution_j,item_{i,p,o,r})$ and $isCDMValid(solution_j,item_{i,p,o,r})$. Their names describe their function, and can be implemented in a straightforward way from the model's constraints.

$isNotOverlapping(solution_j,item_{i,p,o,r})$ tests if the configuration of item i does not overlap any other item already placed inside the bin j

$isLoadBearingValid(solution_j,item_{i,p,o,r})$ tests if adding the item i with the configuration p, o, r breaks some already placed items inside bin j under the CoM of item i.

$isCDMValid(solution_j,item_{i,p,o,r})$ tests if the global CoM in bin j is inside the admissible region.

The *best* rank is initialized to infinite. If the *rank* of a new item is strictly better than the rank of the current *best* (first best fit) and the placement of this new item does not violate the filling constraints, update the best item with the new item.

After all configurations have been tested, if there is no best (i.e. there is no feasible configuration), the item i is placed in a list of discarded items, *discarded*. Items in this list will be considered for the next bin. If a best configuration is found, the $item_{i,p,o,r}$ is appended into $solution_j$, and the variable $feasiblePositions$ is updated by removing the ones that now are overlapping with the $item_{i,p,o,r}$. The last procedure helps to speed up the algorithm.

Since the items are initially sorted by weight in descending order, WFBF may not guarantee a global optimal solution. WFBF computational complexity in the worst case is $O(m \cdot n \cdot q)$ where m is the number of bins, n is the number of items and q is the number of possible positions. The worst case scenario belongs to the class of cases that WFBF cannot handle: consider one bin j with dimensions W_j,L_j,H_j, and at least two items i,k with dimensions $w_k = w_i = W_j$,$l_k = l_i = L_j$,h_k, h_i. If the CoM of the items i,k are outside the feasible CoM region of the bin j, WFBF will not be able to find any feasible packing solution, while, in fact there exists one. The example is shown in

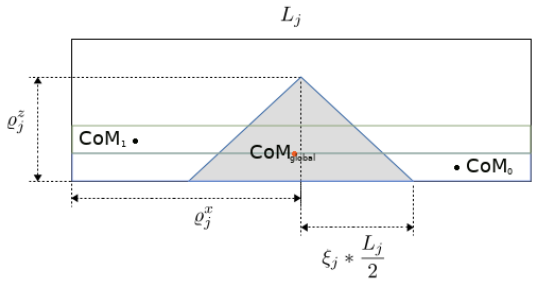


FIGURE 3.7: An example that WFBF cannot solve, despite there being a feasible solution. The feasible solution is represented in the figure, with CoM_{global} in the feasible region.

Fig 3.7, the feasible solution is represented in the figure, with CoM_{global} in the feasible region.

3.3 Dataset

In most studies in the literature, given a fixed dimension for the bin, items dimensions are generated randomly. For example, Junqueira et al. (2012) considered a random dataset made from cubic bins of different sizes and items that were ranging from 10% to 50% or 25% to 75% of the bin size. This procedure, although widely used, could lead to results that do not entirely match with the characteristics of real-world scenarios. A dataset would be ideal if it is useful for both academic and industrial fields. Thus, we decided to exploit the work done by Landschützer et al. (2015) for the Physical Internet challenge (Montreuil, 2012). We built our test instances by randomly picking containers and items from the types defined in the full theoretical set of Physical Internet containers. The set consists of 24 types of bins and 440 types of items. Note that since the PI modular containers are designed to be compatible with existing ISO units, the tested instances should be applicable for current scenarios where ISO loading units (e.g. 40, 20, 10 ft ISO containers) are also used.

The other features, such as load-bearing for items and weight limits for the bins, have been selected to be as realistic as possible. The bin's weight limit is proportional to its volume and the weight limits for actual 40' inter-modal containers (Eq: 3.15), with the exception of the 20' container which follows the actual standard. The same philosophy has been adopted for defining the maximal weight for the items (Eq: 3.16).

$$\Omega_j = \frac{W_j \cdot L_j \cdot H_j}{40' volume} \cdot 40' weight_{Limit} \tag{3.15}$$

Algorithm 1 Heuristic WFBF, part 1

Require: *bins*
Require: *maxLimit*
Require: *items* sorted by weight in descending order, removing the items that do not fit in any available bin

1: $bestSolution \Leftarrow \emptyset$
2: $repeatLimit \Leftarrow 1$
3: $repeat \Leftarrow 0$
4: **if** $isRandomizable(bins)$ **then**
5: $repeatLimit \Leftarrow maxLimit$
6: **end if**
7: **for** $repeat < repeatLimit$ **do**
8: $solution \Leftarrow \emptyset$
9: $itemsleft \Leftarrow items$
10: $bins \Leftarrow randomisedOrderWithPriority(bins)$
11: **while** $j \in bins$ **do**
12: $discarded \Leftarrow \emptyset$
13: **if** $itemsleft \neq \emptyset$ **then**
14: $solution_j \Leftarrow \emptyset$
15: $positions_j \Leftarrow$ enumerate all the possible positions available inside the bin j
16: **for** $i \in itemsleft$ **do**
17: $best \Leftarrow \infty$
18: $winning \Leftarrow \emptyset$
19: **for all** $item_{i,p,o,r}$, $p \in positions_j$, $o \in O$, $r \in \{0,1\}$ reflection **do**
20: $rank \Leftarrow (z_i + \left\|\tau_i^{x,y} - \theta_j^{x,y}\right\|_2)^2$
21: **if** $rank < best$ AND
isNotOverlapping($solution_j$,$item_{i,p,o,r}$) AND
isLoadBearingValid($solution_j$,$item_{i,p,o,r}$) AND
isCDMValid($solution_j$,$item_{i,p,o,r}$) **then**
22: $best \Leftarrow rank$
23: $winning \Leftarrow item_{i,p,o,r}$
24: **end if**
25: **end for**

$$\Omega_i = \frac{w_i \cdot l_i \cdot h_i}{maxitemvolume} \cdot \frac{20' weight_{Limit} \cdot maxitemvolume}{20' volume} \tag{3.16}$$

The items' centre of mass has been generated in a random point under $\frac{3}{5}$ of the height and between 20% and 80% of the other dimensions.

The items' load-bearing in real-world boxes varies considerably depending on the material used and construction geometries. Thus, we opt for simplicity and set the load bearing to be 3 times the maximal weight of the item. This is a parameter that can be changed by users if necessary.

Algorithm 2 Heuristic WFBF, part 2

```
26:           if winning ≠ ∅ then
27:             feasiblePositions ⇐ remove the overlapping winning positions from
    feasiblePositions
28:             solution_j ⇐ solution_j ⋃{item_{i,p,o,r}}
29:           else
30:             discarded ⇐ discarded ⋃{i}
31:           end if
32:         end for
33:         itemsleft ⇐ discarded
34:       end if
35:     end while
36:     repeat ⇐ repeat + 1
37:     if isBetterSolution(solution, bestSolution) then
38:       bestSolution ⇐ solution
39:     end if
40: end for
41: return bestSolution
```

$$\Lambda_i = 3 \cdot \Omega_i \tag{3.17}$$

The item weight has been generated randomly with a uniform distribution between 30% and 100% of the items' maximal weight.

Every experiment is composed of n bins and m items. The procedure for generating a test instance is to firstly randomly pick n bins and then randomly pick m items such that, for every item there exists at least one bin that can fit it. There is no loss of generality because this procedure is equivalent to filtering unfitting items before the optimization task. All the instances follow the same item distribution. The set of available types of items is ordered by increasing volume in such a way that every instance is made of 70% of items after the median (higher volume), and 30% before (lower volume).

Moreover, we introduce some new metrics to better evaluate the results of the experiments. The first one is inspired by the eccentricity of conics, which will measure the distance of an item/bin from being a cube. Thus, given l_i, w_i, h_i as the dimensions of the item/bin i, the eccentricity of i is defined by Eq: 3.18.

$$\varepsilon_i = \max\{\left|\frac{l_i - w_i}{l_i + w_i}\right|, \left|\frac{h_i - w_i}{h_i + w_i}\right|, \left|\frac{l_i - h_i}{l_i + h_i}\right|\} \tag{3.18}$$

The second one, α_i in Eq:3.19, measures the asymmetry of the centre of mass. α_i is defined as the Euclidean distance from the actual centre of mass to the centroid (geometric centre) C^i of the item.

$$\alpha_i = \|C^i, CA^i\|_2 \tag{3.19}$$

The third one is the relative volume, Eq: 3.20. This metric is calculated as the volume of item i divided by the volume of the bin j.

$$VolRel_{i,j} = \frac{h_i \cdot w_i \cdot l_i}{H_j \cdot W_j \cdot L_j} \tag{3.20}$$

3.4 Experiments and discussion

The first aim of the experiments is to test the model behaviour by varying the number of items and bins. We also investigate the impact of the constraints on the difficulty of finding a solution. Each experiment terminates with a final script that validates the solution found. We tested the three models listed in Section 3.1.2. Briefly, V0 considers only stability, V1 considers stability and CoM distribution and V2 considers stability, CoM distribution, and load-bearing. The second aim is to have a comparison metric to test the goodness of the proposed heuristic. Having the results from the model we can know how far the heuristic is from finding the optimal solution.

All versions have been implemented in OPL and solved using IBM CPLEX® 12.7. The heuristic has been implemented in Python 3.6. The machine used to perform the experiments was an Intel® Core™ i7-4790 CPU @ 3.60GHz, 16GB RAM, with Windows 7 Enterprise 64-bit.

As reported by many authors, the complexity of the problem depends obviously on the number of items, the number of bins and their dimensions. The first two can be explained by looking at how the number of variables and the number of constraints grows according to the number of items and bins, as shown in Table 3.5. The item/bin dimensions involved affect the number of feasible positions in the solution space. Thus, considering large spaces increases the difficulty.

In each experiment we reported the gap calculated by the solver, the formula is reported in Eq: 3.21. We applied the same formula to calculate the gap in the heuristic results.

$$gap = \frac{|bestbound - bestinteger|}{\frac{1}{10^{10}} + |bestinteger|} \tag{3.21}$$

In the first group of experiments, we focus on packing 10 random items in the smallest bin, with dimensions $120 \times 120 \times 120$. In Table 3.6 we report the average feature values of the solutions found for all the 1,000 instances solved with V0, V1. and V2. The *dataset* column represents the features of the dataset. $\mu_\bullet$ indicates average while $\sigma^2_\bullet$ is the variance. ω indicates the weights, LB is the load bearing limit, $VolRel$ is the relative volume ($\frac{volumeitem}{volumebin}$), ε is the previously defined eccentricity, α is the asymmetry, *objective* value is the result of minimizing the wasted space (Eq. 3.1), *elapsedtime* is the time elapsed from the starting point before the pre-solving phase to the solution or the time limit. V0 is not affected by the weight but it packs fewer items than V1 because it chooses items with bigger volume than V1 (μ_{VolRel}). The degradation of the objective between V0 and V1 is 0.12% and the time to achieve a solution increases by about 30%. V2, instead, is about 15 times more complex to solve than V1. V2 chooses items with higher load-bearing limit (μ_{LB}) which will also have higher volume (μ_{VolRel}) and be slightly less eccentric than V1 but with higher μ_α. We hypothesize that the complexity is due to the combination of the following factors: higher volume items imply a higher centre of mass along the H axis and this brings a lower feasible solution space[1]. Similarly, the higher the eccentricity, the higher the α. Those factors combined with the restricted base will force the solver to explore more nodes before finding a solution.

[1]The higher the height in the pyramid, the less space available.

TABLE 3.6: 10 item 1 bin, average values over 1,000 random instances, time in milliseconds

feature	V0	V1	V2	dataset
available items	10	10	10	10
packed items	9.813	9.817	9.386	-
μ_ω	23.312	23.010	24.500	22.657
σ^2_ω	1167.43	1170.52	1265.41	1111.56
μ_{LB}	107.38	105.69	112.78	103.97
σ^2_{LB}	21698.31	21797.74	23466.60	20633.55
μ_{VolRel}	0.05005	0.04926	0.05259	0.04843
σ^2_{VolRel}	0.00479	0.00481	0.00519	0.00455
μ_ε	0.5587	0.5589	0.5557	0.5604
σ^2_ε	0.0357	0.0358	0.0355	0.0359
μ_α	11.884	11.877	12.120	11.882
σ^2_α	81.871	81.919	82.789	82.305
$\mu_{objective}$	0.5316	0.5328	0.5407	-
$\mu_{elapsedtime}$	330.16	432.86	7019.56	-

TABLE 3.7: Experiments using 1 bin, from 18 to 90 items. Time is measured in seconds (* is when the solver has hit the time limit of 60 minutes).

instance		V 0			V 1			V 2		
item	bin	time	objective	gap%	time	objective	gap%	time	objective	gap%
18	1	9.84	0.3090	0	14.38	0.3090	0	3600*	0.4407	29.87
19	1	2.72	0.5098	0	15.89	0.5098	0	3600*	0.5409	0.62
20	1	1.34	0.6912	0	1.404	0.6912	0	3.681	0.6912	0
21	1	9.06	0.2935	0	22.62	0.2935	0	3600*	0.3754	21.80
22	1	4.74	0.8211	0	5.61	0.8211	0	15.965	0.8211	0
23	1	9.12	0.3325	0	27.22	0.3325	0	3600*	0.5452	39.02
24	1	5.662	0.8965	0	0.936	0.8965	0	65.542	0.8965	0
25	1	3600*	0.0394	100	3600*	0.0394	100	3600*	0.0668	100
26	1	2.574	0.8664	0	6.16	0.8664	0	42.933	0.8664	0
27	1	3600*	0.1172	100	3600*	0.1538	100	3600*	0.3513	99.92
28	1	9.375	0.6917	0	34.42	0.6917	0	3600*	0.7218	4.17
29	1	9.475	0.7164	0	16.03	0.7164	0	256	0.7164	0
30	1	10.202	0.7842	0	11.31	0.7842	0	84.68	0.7842	0
35	1	45.831	0.5476	0	87.42	0.5476	0	3600*	0.6194	11.59
40	1	59.9	0.719	0	111.8	0.719	0	3600*	0.7545	4.70
45	1	3600*	0.2271	1.96	3600*	0.2937	24.20	3600*	0.3412	34.74
50	1	1813.27	0.4771	0	1234.49	0.4771	0	3600*	0.5462	12.65
55	1	3600*	0.0312	100	3600*	0.0430	100	3600*	0.3027	100
60	1	3600*	0.5633	29.75	3600*	0.5682	30.36	3600*	0.6117	35.31
65	1	1142.86	0.5414	0	3236.7	0.5414	0	3600*	0.6615	22.16
70	1	3600*	0.2394	100	3600*	0.2310	100	3600*	0.3841	100
80	1	3600*	0.0622	100	3600*	0.0902	100	3600*	0.3636	100
90	1	3600*	0.325	100	3600*	0.3364	100	3600*	0.4847	100

TABLE 3.8: Experiments using 1 bin, from 100 to 200 items. Time is measured in seconds(* is when the solver has hit the time limit of 60 minutes).

instance		V 0			V 1			V 2		
item	bin	time	objective	gap%	time	objective	gap%	time	objective	gap%
100	1	3600*	0.1833	100	3600*	0.6	100	3600*	0.5666	100
110	1	3600*	0.7291	100	3600*	0.4265	100	3600*	0.5759	100
120	1	3600*	0.0962	100	3600*	0.3555	100	3600*	0.3333	100
130	1	3600*	0.3083	100	3600*	0.3412	100	3600*	0.4458	100
140	1	3600*	0.9111	100	3600*	0.6495	100	3600*	0.5937	100
150	1	21.45	0	0	3600*	0.2305	100	3600*	0.2777	100
160	1	3600*	0.8411	100	3600*	0.5533	100	3600*	0.5650	100
170	1	28.06	0	0	3600*	0.0444	100	3600*	0.1666	100
180	1	3600*	0.3611	100	3600*	0.7777	100	3600*	0.4049	100
190	1	3600*	0.8333	100	3600*	0.8266	100	3600*	0.9333	100
200	1	3600*	0.8819	100	3600*	0.5944	100	3600*	0.6208	100

TABLE 3.9: Experiments using multiple bins. * time limit exceeded. The objective function is minimising ch1:eq:3.1. We report the value of the objective and the number of bins that have been opened(in parentheses). In the last cases (5 bins, from 130 to 200 items), the solver was not able to finish the pre-solving phase before the time limit.

instance		V 0			V 1			V 2		
item	bin	time	objective (used bins)	gap%	time	objective (used bins)	gap%	time	objective (used bins)	gap%
70	2	3600*	1.1022(2)	18.23	3600*	1.3634(1)	30.24	3600*	1.4066(1)	32.39
100	2	3600*	1.8705(2)	31.24	3600*	1.5491(2)	16.97	3600*	1.5696(2)	18.06
130	2	3600*	1.8006(2)	72.33	3600*	1.9791(1)	74.83	3600*	1.9791(1)	74.81
160	2	3600*	1.8888(1)	55.48	3600*	1.9305(1)	56.44	3600*	1.7489(1)	51.91
200	2	3600*	1.5(2)	254.30	3600*	1.2222(2)	289.36	3600*	1.3333(2)	273.58
70	3	3600*	2.2336(3)	9.84	3600*	2.4269(1)	15.19	3600*	2.4486(1)	15.94
100	3	3600*	2.3031(1)	49.45	3600*	2.2376(1)	47.97	3600*	2.2302(1)	47.79
130	3	3600*	2.5999(3)	76.95	3600*	2.6311(2)	70.98	3600*	2.6930(1)	71.64
160	3	3600*	2.6666(1)	97.50	3600*	2.4763(3)	97.31	3600*	2.6969(1)	97.53
200	3	3600*	2.8333(2)	61.09	3600*	2.8353(2)	61.12	3600*	3(0)	91.69
70	5	3600*	3.8309(5)	39.09	3600*	4.4104(2)	47.09	3600*	4.3152(3)	45.92
100	5	3600*	4.5870(2)	28.47	3600*	4.2220(2)	22.28	3600*	4.2713(2)	23.18
130	5	3600*	5(0)	63.45	3600*	4.565(3)	60.06	3600*	4.6055(2)	60.41
160	5	3600*	5(0)	86.98	3600*	5(0)	85.67	3600*	5(0)	85.67
200	5	3600*	5(0)	81.36	3600*	5(0)	77.48	3600*	4.6811(2)	75.84

The second group of experiments has been made to test the behaviour of the model using only one bin. We ran experiments with instances having 18 to 200 items, each experiment had a time limit of 60 minutes. In Table 3.7 and Table 3.8 we report the results, with time and objective value, for V0, V1 and V2. Trying to solve more than 350 items made our computer run out of memory. With 130 items or more, the solver spent most of the time in the pre-solving phase and very few iterations in the solving one. In almost every case, V2 final objective is about 100% worse than V1. In some instances, for example, with 100 items, we can see that V2 obtains a better objective than V1 within the time limit. This may be due to a different choice in the nodes to explore.

The third group of experiments aims to show the behaviour of the model with multiple bins. We tested with 2, 3 and 5 bins and number of items ranging from 70 to 200. Table 3.9 shows the computational time and the objective value for V0, V1, and V2 according to the number of items and the number of bins. Compared to the results for one bin, the overall performance degraded dramatically when the number of bins is two or more. Due to the increased number of bins, for instances with more than 70 items, the solver spent most of its time on the pre-solving phase and very few iterations on the solving phase.

The fourth group of experiments compare the heuristic WFBF with the results of V2. The comparison has been made using the three groups of problem instances used previously. Table 3.10 shows the experiments using 1 bin, from 18 to 90 items. WFBF is capable of finding in some instances the optimal solution, while in general it achieved better results using much less time.

Table 3.11 shows experiments using 1 bin, from 100 to 200 items. As expected, WFBF significantly outperformed the solver, the filling rates are on average greater than 90%, in some cases achieve 100%. This is reasonable since having more items also improves the probability of having items that fit together.

Table 3.12 shows the experiments with multiple bins. The results show that the heuristic clearly outperforms the exact model both in terms of objective value and computational time. An increase in the number of items and bins does increase the computational time and the gap% of the heuristic in certain cases, but this increase appears to be linear. This demonstrates the capability of the heuristic in dealing with larger instances.

The advantage of the heuristic, as demonstrated above, is the fast computational time and better quality than what the exact model can offer within the given time limit. The heuristic does however have its limitations due to its sequential filling procedure (one bin at a time and giving priority to smaller volume bins). According to this procedure,

the first bin will be the one with lowest volume. It is filled having all the items available. The next bin, will have at least the same volume, however can only be filled using the discarded items from the previous bins. These choices have the drawback that every discarded item is going to be placed in a bin which will have at least the same volume as the previous one. Since we are assuming to allocate every item, it may lead to under-filling the last bin, since it is more likely to be also the one with the biggest volume. As an example, considering the experiment with 5 bins and 160 items, the first bin is filled with 4 items and has a 100% filling rate, the second bin is filled with 66 items at a filling rate of 92.44%, the third bin is filled with 66 items at a filling rate of 17% and the fourth one is filled with 24 items at a filling rate of 15%. The volume of the fourth bin is double the volume of the third bin. Other examples of this behaviour are reported in the next group of experiments.

The last group of experiments is to measure the costs/benefits of using a randomised order to discover new configurations. The instances in these experiments consider multiple bins such that for each distinct volume there are at least two bins with different dimensions. We compared the contribution of *randomisedOrderWithPriority* to a simple sorting by volume ascending. The *repeatLimit* is set at 10 iterations for each experiment, in both configurations.

Table 3.13 and Table 3.14 compare the behaviour of the heuristic using simple sorting to that of the heuristic using *randomisedOrderWithPriority*.

In these tables we show also the details for each bin filled. The *opening sequence* is the progression number of the container that has been filled. It is useful to highlight cases where a container has been skipped because there were no fitting items. *items packed* is the number of items that have been allocated inside the bin. *filling rate* is the percentage of the bin's total volume filled by the items. For the majority of the instances CPLEX did not report a lower bound in the time limit of one hour, thus we could not report the gap. In some instances, i.e. Table 3.14 5 bins with 160 to 400 items, WFBF skips a bin because there is no remaining fitting item: the height of the bin is lower than the height of the remaining items. For each instance the objective is highlighted in bold if there is a clear winner between *randomisedOrderWithPriority* and simple sorting, while the cases where the objectives are equal or worse are not highlighted. *randomisedOrderWithPriority* was able to improve the solutions in two cases. Since the simple ordering case is included in the possibles output, we can state that *randomisedOrderWithPriority* is better than having a simple ordering, and the benefit is dependent on the execution of a sufficient number of repetitions.

TABLE 3.10: Heuristic compared to V2. Experiments using 1 bin, from 18 to 90 items. Time is measured in seconds (* is when the solver has hit the time limit of 60 minutes).

instance		Heuristic			V 2		
item	bin	time	objective	gap%	time	objective	gap%
18	1	4.1864	**0.3090**	0.02	3600*	0.4407	29.87
19	1	8.0681	0.7015	23.37	3600*	**0.5409**	0.62
20	1	13.3015	**0.6912**	0	3.681	**0.6912**	0
21	1	4.5546	**0.3269**	10.20	3600*	0.3754	21.80
22	1	27.0961	**0.8211**	0	15.965	**0.8211**	0
23	1	6.2495	**0.4075**	18.41	3600*	0.5452	39.02
24	1	31.3101	**0.8965**	0	65.542	**0.8965**	0
25	1	1.3735	**0.0622**	100	3600*	0.0668	100
26	1	30.9466	**0.8664**	0	42.933	**0.8664**	0
27	1	2.5830	**0.235**	99.88	3600*	0.3513	99.92
28	1	19.0795	**0.69173**	0	3600*	0.7218	4.17
29	1	19.7695	**0.7164**	0	256	**0.7164**	0
30	1	36.5915	**0.7842**	0	84.68	**0.7842**	0
35	1	31.4864	**0.54769**	0.01	3600*	0.6194	11.59
40	1	43.3107	**0.7190**	0.01	3600*	0.7545	4.70
45	1	19.7262	**0.2226**	0.03	3600*	0.3412	34.74
50	1	35.9081	**0.4825**	1.12	3600*	0.5462	12.65
55	1	7.0416	**0.1032**	100	3600*	0.3027	100
60	1	45.3121	**0.3957**	0	3600*	0.6117	35.31
65	1	54.0963	**0.5414**	4.89	3600*	0.6615	22.16
70	1	10.9763	**0.14807**	100	3600*	0.3841	100
80	1	2.3786	**0.0094**	100	3600*	0.3636	100
90	1	15.6290	**0.0474**	100	3600*	0.4847	100

TABLE 3.11: Heuristic compared to V2. Experiments using 1 bin, from 100 to 200 items. Time is measured in seconds(* is when the solver has hit the time limit of 60 minutes). The solver reports a gap of 100% because the best bound found in the time limit is 0.0.

instance		Heuristic			V 2		
item	bin	time	objective	gap%	time	objective	gap%
100	1	8.8989	**0.0707**	100	3600*	0.5666	100
110	1	70.1088	**0.1455**	100	3600*	0.5759	100
120	1	3.3416	**0.0681**	100	3600*	0.3333	100
130	1	10.8469	**0.0263**	100	3600*	0.4458	100
140	1	38.3794	**0.0916**	100	3600*	0.5937	100
150	1	1.3438	**0.0**	100	3600*	0.2777	100
160	1	46.1770	**0.1043**	100	3600*	0.5650	100
170	1	0.3227	**0.0**	100	3600*	0.1666	100
180	1	2.7800	**0.0**	100	3600*	0.4049	100
190	1	12.4953	**0.0619**	100	3600*	0.9333	100
200	1	24.0775	**0.0428**	100	3600*	0.6208	100

TABLE 3.12: Heuristic compared to V2. Experiments using multiple bins. * time limit exceeded

instance		Heuristic			V 2		
item	bin	time	objective (used bins)	gap%	time	objective (used bins)	gap%
70	2	103.2236	**1.0228(2)**	7.02	3600*	1.4066(1)	32.39
100	2	197.5630	**1.2862(2)**	0.01	3600*	1.5696(2)	18.06
130	2	111.9377	**0.6241(2)**	20.12	3600*	1.9791(1)	74.81
160	2	181.0182	**0.9430(2)**	10.81	3600*	1.7489(1)	51.91
200	2	68.4242	**0.0751(2)**	96.76	3600*	1.3333(2)	273.58
70	3	210.6892	**2.1218(3)**	2.99	3600*	2.4486(1)	15.94
100	3	374.9229	**1.9325(3)**	39.75	3600*	2.2302(1)	47.79
130	3	262.8290	**1.6446(3)**	53.56	3600*	2.6930(1)	71.64
160	3	195.6397	**1.0881(3)**	93.88	3600*	2.6969(1)	97.53
200	3	455.4567	**1.6725(2)**	85.09	3600*	3(0)	91.69
70	5	25.8132	**3.4386(2)**	32.13	3600*	4.3152(3)	45.92
100	5	47.9942	**3.3819(2)**	2.98	3600*	4.2713(2)	23.18
130	5	212.4729	**3.0586(3)**	40.39	3600*	4.6055(2)	60.41
160	5	327.8276	**2.7441(4)**	73.89	3600*	5(0)	85.67
200	5	433.9891	**2.7896(3)**	59.46	3600*	4.6811(2)	75.84

TABLE 3.13: Heuristic. Experiments using multiple bins; the bins for each experiment have the same volume and different dimensions. * time limit exceeded.

instance		Heuristic without *randomisedOrderWithPriority*						Heuristic with *randomisedOrderWithPriority*					
item	bin	time	objective (used bins)	gap%	opening sequence	items packed	filling rate%	time	objective (used bins)	gap%	opening sequence	items packed	filling rate%
70	2	48.56	1.3579(2)	34,78				75.13	**1.3097(2)**	13.29			
					1	68	64.20				1	60	52.44
					2	0	0.0				2	10	16.57
100	2	204.68	1.3453(2)	3.29				207.41	1.3453(2)	3.29			
					1	86	30.79				1	86	30.79
					2	13	34.66				2	13	34.66
130	2	108.71	1.0497(2)	76,50				140.88	**0.9792(2)**	0.0			
					1	107	88.04				1	107	76.14
					2	17	6.98				2	23	25.93
160	3	436.29	1.6169(2)	-				496.747	1.6169(2)	-			
					1	18	99.54				1	18	99.54
					2	142	38.75				2	142	38.75
200	3	463.34	**0.3824(1)**	-				528.69	1.3824(2)	-			
					1	200	61.75				1	196	60.03
											2	4	1.72

TABLE 3.14: Heuristic. Experiments using multiple bins; the bins for each experiment have the same volume and different dimensions. * time limit exceeded.

instance		Heuristic without *randomisedOrderWithPriority*						Heuristic with *randomisedOrderWithPriority*					
item	bin	time	objective (used bins)	gap%	opening sequence	items packed	filling rate%	time	objective (used bins)	gap%	opening sequence	items packed	filling rate%
160	5	421.56	2.4695(3)	-				859.03	2.4695(3)	-			
					1	158	52.38				1	135	33.73
					3	2	0.66				3	25	19.31
200	5	506.95	2.4033(3)	-				885.82	2.4033(3)	-			
					1	199	59.33				1	177	39.77
					3	1	0.33				3	23	19.88
400	5	1475.59	**1.8067(3)**	-				1377.57	2.8067(4)	-			
					1	237	90.55				1	323	75.41
					2	134	14.46				2	76	40.58
					3	29	14.30				4	1	3.33
800	5	2721.43	0.61345(3)	-				2756.12	0.61345(3)	-			
					1	142	97.60				1	81	98.24
					2	424	83.50				2	403	87.53
					3	234	57.54				3	316	52.87

3.5 Summary

This chapter proposes a mixed-integer linear model that solves the Multiple Heterogeneous Knapsack Problem while giving priority to smaller bins and considering the following constraints: non overlapping, boundaries and positioning (both constrained and free), rotations (around z-axis), orthogonal displacement, weight limit, static stability, weight distribution in a pyramidal region and load bearing considering items' arbitrary centre of mass. This model can obtain routinely mathematically proved optimal solutions in small sized problems. For medium sized instances it can only achieve sub-optimal solutions. Despite that, it remains a useful tool for studying the problem. It provides a benchmark to compare the quality of the solutions found by heuristics and approximations models.

We studied the trade-off of adding more constraints to make the problem more realistic and the complexity of finding a solution. The model has been derived in submodels V0 (only stability), V1 (stability and centre of mass distribution), V2 (stability, centre of mass distribution and load-bearing). The experiments showed that, even with very small instances, the computational time for finding the optimal solution considering the load bearing constraint was 16.2 times longer than without considering it.

We introduced new metrics that aim to facilitate the comparison and analysis of different knapsack instances. An effort in this direction is important because faster algorithms are often specialized algorithms that only efficiently handle particular cases of the problem. Often decision-makers are not operational research specialists, so having such metrics improves the export of theoretical results to the industry.

We designed a simple deterministic heuristic to deal with large-scale instances in reasonable computational time and memory. Weight First Best Fit is an on-line heuristic that fills the available bins one at a time, sequentially, until the available bins or the available items are exhausted. WFBF significantly outperforms the exact model in almost all instances. We also highlighted some limits of the algorithm.

We have also added a top level search that tests different orderings of the bins. The search is useful when there are bins with the same volume and different dimensions.

Further research is necessary to design new algorithms to overcome the limits of the proposed heuristic.

Chapter 4

Using machine learning to estimate feasibility of packing solutions in constant time - a novel strategy

This chapter shows a novel strategy to use machine learning to estimate if, for a given a set of items, a feasible packing configuration exists in a constant O(1) computational time. The problem is an NP-Hard Knapsack problem, where the objective is to minimise the wasted space and one of the constraints is to allocate all the items. It is expected that this strategy will significantly save time and computational effort. Although the strategy is applied to the knapsack and bin packing problems in this chapter, it can be generalized to deal with a wider range of problems. The strategy consists of two stages. In the first stage, we exploit the master/slave Bender's decomposition to build a dataset of knapsack solutions. In the second stage we exploit the dataset to train a classifier for checking the feasibility of a packing solution. The chapter also proposes a new dataset of packing solutions and benchmarks different classification algorithms. We compared the following algorithms: decision trees (DT), random forest (RF), support vector machine with radial basis function kernel (SVM-R), support vector machine with polynomial kernel (SVM-P), three different architectures of convolutional neural networks (CNN), feed forward fully connected neural networks (FFNN) with one, two and three hidden layers. The best classifier identified the feasibility of the tested instances with an accuracy of 90%.

The first novelty of this chapter is the methodology, we have not found any similar procedure in the literature.

The second novelty is the dataset of knapsack solutions, despite being a famous problem we could not find any comprehensive dataset and we believe that it can be useful to have a new dataset that can be further extended.

The third novelty is the idea of setting up a packing problem as a classification problem, and we benchmark different classification algorithms.

This chapter is organised as follows: Section 4.1 discusses the overall strategy in detail. Section 4.2 describes the dataset used. Section 4.3 presents the classification algorithms used in the comparison and shows and discusses the computational experiments used to validate the methodology. The chapter will end with a summary.

4.1 Methodology

We will proceed with defining the three-dimensional orthogonal bin packing problem (3D-OBP), exploiting a Bender's decomposition schema. Let $I = \{1, \cdots, n\}$ be a set that indexes n rectangular cuboids (items) and let $J = \{1, \cdots, m\}$ be a set that indexes m rectangular cuboids (bins). For every item $i \in I$ and bin $j \in J$ let us define their dimensions as width w_i (W_j), height h_i (H_j) and length l_i (L_j). 3D-OBP consists in assigning (packing) every item $i \in I$ to a bin $j \in J$ such that the number of bins utilised is minimised. The problem is constrained to avoid any two items i, k assigned to the same bin j to overlap; items must be allocated inside the bins; every item must be assigned to only one bin. The problem can be enriched with various constraints; there is an overview in Bortfeldt and Wäscher (2013).

Two items $i, k \in I$ are equivalent for the relation of equivalence $\cong (i, k)$ defined in (4.1), if and only if their dimensions are equal. Reflexivity, symmetry and transitiveness of (4.1) are easy to prove. (4.1) can be enhanced by preprocessing the items' dimensions with dual-feasible functions (Clautiaux et al., 2010) and conservative scales (Belov et al., 2013). Another enhancement is to modify (4.1) to consider i and k equivalent if there exists a rotation r that overlaps i with k. All the enhancements are out of the scope of this proof of concept, thus we will keep the simple (4.1).

$$i \cong k \Leftrightarrow l_i = l_k \wedge w_i = w_k \wedge h_i = h_k \tag{4.1}$$

Let us define $\mathcal{I} = \frac{I}{\cong}$ as the quotient set of I on $\cong$. It partitions the set of items into disjoint classes of items. Let π be the number of classes. Let $\tau_t, \forall t \in \mathcal{I}$ be the maximal number of items available in the class t. Let l_t be the length, w_t be the width, and h_t

be the height of an arbitrary element in the class t. It is useful to compute the volume $v_t = l_t \cdot w_t \cdot h_t$ of a class element in t, and the volume $V_j = L_j \cdot W_j \cdot H_j$ of the bin j. Moreover, let $q_{t,j}$ be the quantity of items of class t that are assigned to the bin j. Let $Z_j \in \{0, 1\}$ be 1 if the bin j is opened (has at least one item inside), 0 otherwise. $\beta_{t,c}$ is the quantity of the class t in the infeasible partition c.

Now we describe the 3D-OBP, exploiting a Bender's decomposition schema. The master problem (4.2)-(4.6) selects how to partition the items in the available bins.

$$\text{minimise} \sum_{j \in J} Z_j \tag{4.2}$$

$$\text{subject to} \sum_{j \in J} q_{t,j} \leq \tau_t, \ \forall t \in \mathcal{I} \tag{4.3}$$

$$\sum_{t \in \mathcal{I}} \sum_{j \in J} q_{t,j} \geq \sum_{t \in \mathcal{I}} \tau_t, \tag{4.4}$$

$$\sum_{t \in \mathcal{I}} q_{t,j} * v_t \leq V_j * Z_j, \ \forall j \in J \tag{4.5}$$

$$\sum_{t \in \mathcal{I}} (\beta_{t,c} - q_{t,j})^2 \geq Z_j, \ \forall j \in J, c \in \mathcal{C} \tag{4.6}$$

Briefly, the objective is to minimise the opened bins (4.2), subject to: for each class t, the overall quantity of items allocated must not exceed the items' class maximum number τ_t (4.3); all the items must be allocated (4.4); the total volume of the items in bin j must be lower than the volume capacity of bin j, if it has been opened, otherwise must be lower or equal to 0, i.e. the bin must contain items only if the bin is opened, and the overall items' volume is lower than the bin's limit (4.5); all the previously tested unfeasible partitions are excluded (4.6).

For every bin j, the slave problem checks if the partition $\mathcal{P}_j = \{q_{t,j}, t \in \mathcal{I}\}$ satisfies the constraints for the orthogonal packing. We will now skip further description of the slave problem, which can be found in Appendix B. Let $\mathcal{C}_j$ be the set of partitions that have been proved unfeasible, and $\mathcal{F}_j$ the set of feasible partitions.

Constraint (4.6) binds the master problem with the slave problem. It avoids any partition $c \in \mathcal{C}_j$ being evaluated again. $\beta_{t,c}$ is the quantity of the class $t \in \mathcal{I}$ in the unfeasible partition c.

This master/slave approach is not effective for finding optimal solutions, because it finds an optimal solution by exclusion, but the oscillations between master and slave produce rapidly a dataset $\mathcal{D}_j = \mathcal{C}_j \bigcup \mathcal{F}_j$ of unfeasible and feasible packings. Suppose that we have enough points in the dataset, the Pareto dominating partitions $\mathcal{P}_j^d \subset \mathcal{F}_j$ will be the facets of the integer polytope that separate the feasible solutions from the unfeasible ones. It should be noted that $\mathcal{C}_j$ is a set of covers for $\mathcal{F}_j$: for every $p \in \mathcal{F}_j$ there

exists $d \in \mathcal{C}_j$ such that d dominates p. Note that, in the slave problem, $\mathcal{P}_j^d \subset \mathcal{F}_j$ is a 0/1 knapsack polytope. The relations between these polytopes can be found e.g. in Atamtürk (2005) and Pochet and Wolsey (1995); Pochet and Weismantel (1998).

Let us suppose now that we have a function $\theta(\mathcal{P}_j) := \mathcal{D}_j \mapsto \{0, 1\}$ that maps a partition $\mathcal{P}_j$ to the solution of the feasibility of the slave problem. The problem will become:

$$\text{minimise} \sum_{j \in J} Z_j \tag{4.7}$$

$$\text{subject to} \sum_{j \in J} q_{t,j} \leq \tau_t, \ \forall t \in \mathcal{I} \tag{4.8}$$

$$\sum_{t \in \mathcal{I}} \sum_{j \in J} q_{t,j} \geq \sum_{t \in \mathcal{I}} \tau_t, \tag{4.9}$$

$$\theta(q_{0,j}, \ldots, q_{\pi,j}) \leq Z_j, \ \forall j \in J \tag{4.10}$$

We will show now that the supposition is correct, and $\theta(\mathcal{P}_j)$ is a classifier. Moreover, if we extend eq. (4.1) to the set of bins, the number of needed classifiers is equal to the number of bins' classes. Eventually, a $\theta(\mathcal{P}_j)$ that classifies correctly all the partitions can represent $\mathcal{P}_j^d$ and can be linearly approximated, but this is out of the scope of this chapter.

We would like to emphasize a core difference between the two formulations. θ solves the following decision problem: given a set S of items and a bin j, does a configuration exist of all the items in S such that they completely fit inside the bin j? The aim of θ is to give an answer without defining the positions of the items inside j. It is hard because giving an answer implies knowing that a solution to the knapsack packing problem exists just by looking at the problem instance.

4.2 Dataset

Without any loss of generality, we consider the task of classification for only one bin class k. The dataset $\mathcal{D}_k$ has been built considering that in the slave problem, items can rotate orthogonally along the vertical axis (the updated relation of equivalence is trivial and can be found in Appendix C). We collected a total of 9,826 partitions where 5,809 are unfeasible and 4,017 are feasible. The number π of items' classes $\mathcal{I}$ is 112.

TABLE 4.1: A dataset snapshot, aggregating knapsack solutions by filling rate and feasibility. The dataset is partitioned in aggregation buckets of 10 on the filling rate. (A) is the average filling rate in the bucket, (B) is the number of elements in the bucket. The filling rate is expressed in percentage.

filling rate	feasible		unfeasible	
bucket	A	B	A	B
(0, 10]	5.80	114	-	0
(10, 20]	14.86	126	-	0
(20, 30]	24.64	92	-	0
(30, 40]	34.64	54	-	0
(40, 50]	46.07	87	-	0
(50, 60]	56.00	208	56.39	2
(60, 70]	65.50	494	67.30	16
(70, 80]	75.14	771	77.09	84
(80, 90]	85.03	1533	87.74	1531
(90, 100]	93.31	538	94.81	4176

Fig 4.1 shows the procedure, whilst Fig 4.2 shows the legend: the dashed block represents the process that is done in parallel, one process for each slave; squared blocks are processes; the dataset is represented as the block of data that contains the feasible and the unfeasible solutions.

The procedure in Fig 4.1 has two nested loops. The outer loop creates random problems, whilst the inner loop solves a problem. The condition to solve the master problem is that for every slave problem, the solution is feasible. In the process of finding the correct feasible partioning, the dataset of solutions is updated with non dominated solutions. The Pareto dominance of the feasible solutions is the standard one: let $x^{'}, x \in X \subseteq R^n$ be two points, $x^{'}$ dominates x if and only if $\exists j \in \{1 \cdots n\}$, such that $\forall i \in \{1 \cdots n\} \setminus \{j\}, x_i^{'} \geq x_i \wedge x_j^{'} > x_j$. The definition of the Pareto dominance of the unfeasible solutions is adapted by substituting $>$ with $<$ and $\geq$ with $\leq$. In the Fig 4.1 we show a parallelisation of a part of the inner loop, it is easy to parallelise the outer loop too.

Table 4.1 shows a summary of the dataset aggregating the knapsack solutions by their filling rates and feasibility. The dataset has been partitioned in intervals of 10% filling rate. The table is built as follows. A partition c, column *bucket*, is delimited by a minimal value a, and a maximal value b. A knapsack solution d is aggregated in c if its filling rate $f(d)$ is $a < f(d) \leq b$. It is reasonable to assume unfeasible partitions are mostly concentrated in higher filling rates and absent in the lowest.

Fig 4.3 shows the frequency of the features, of which each feature is an item class. The dataset is unbalanced: some features are more frequent than others. This imbalance is caused by the aggregation preferences of the solver. Testing techniques to re-balance,

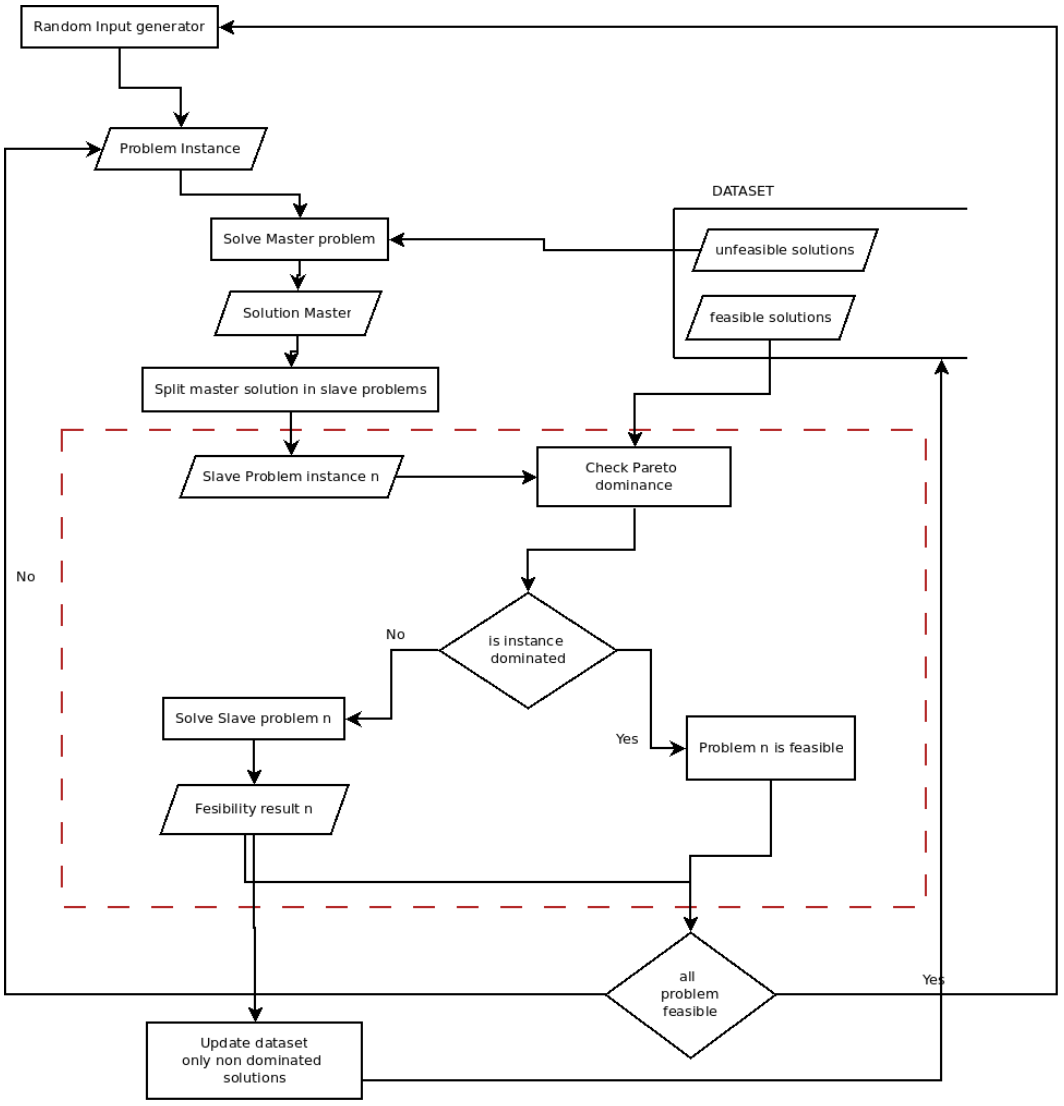


FIGURE 4.1: Flow chart of the procedure. The legend for the block components is shown in Fig 4.2.

or sampling the polytope evenly, is not within the scope of this chapter. However some strategies will be outlined in the conclusions section.

4.3 Classification, results and discussion

The nature of the dataset prevents its dimensionality from being reduced: every feature is meaningful. In fact, if the dataset contains all the possible points, and there exists an algorithm $\theta(\mathcal{P}_j)$ that is able to successfully classify as feasible all the feasible points, then, $\theta(\mathcal{P}_j)$ would represent the facial structure of the convex hull of the integer knapsack set.

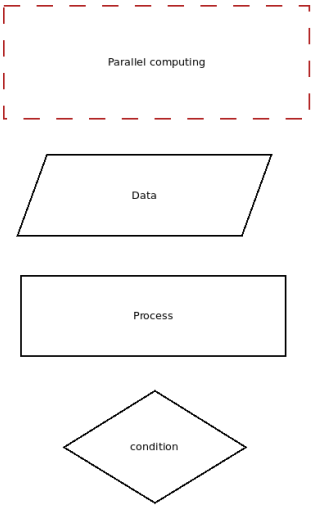


FIGURE 4.2: Legend of the flow chart in Fig 4.1.

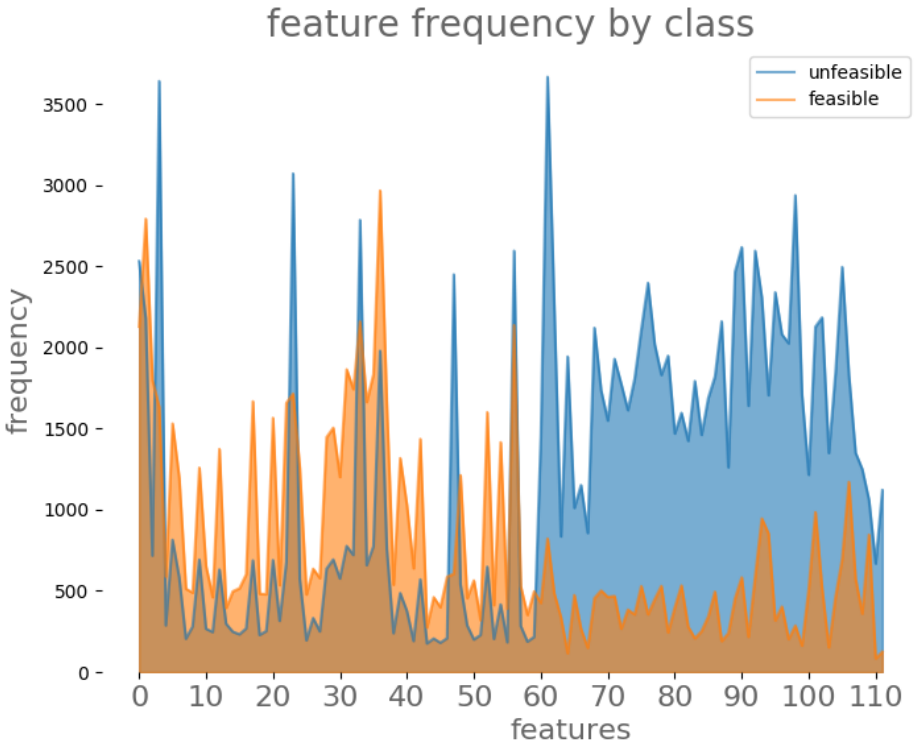


FIGURE 4.3: Dataset frequencies for each item class. The dataset is aggregated by item class and feasibility. The blue area is coloured by the unfeasible solutions, while the yellow by the feasible solutions.

The classification task is as follows. Given a partition $p \in D_j$, p is made of π ordered features, of which each feature $q_{t,p}$ is the quantity of elements of the class $t \in \mathcal{I}$ in the partition p. The classification task is to find $\widetilde{\theta}(\mathcal{P}_j)$ such that $\widetilde{\theta}(\mathcal{P}_j) \approx \theta(\mathcal{P}_j)$.

The programming language adopted in the experimental part is Python 3.6 and Ubuntu 18.04 as operating system. The models for the optimisation task have been implemented

Table 4.2: CNN architectures.

CNN-1	CNN-2	CNN-3	CNN-4
11X11	11X11	11X11	11X11
20@5x5/relu	20@5x5/relu	30@3x3/relu	30@4x4/selu
max(2x2)	max(2x2)	max(2x2)	60@5x5/selu
50@5x5/relu	50@5x5/relu	60@5x5/relu	max(2x2)
max(2x2)	max(2x2)	max(2x2)	flatten
flatten	50@5x5/relu	flatten	dense(150)/selu
dense(500)/relu	max(2x2)	dense(150)/relu	dense(150)/selu
dense(2)/softmax	flatten	dense(2)/softmax	dense(2)/softmax
	dense(200)/relu		
	dense(2)/softmax		

using OPL and solved with IBM CPLEX 12.6.

In our experiments, the algorithms DT, RF, SVM-R and SVM-P exploit the implementations of scikit-learn (Pedregosa et al., 2011) in their standard configuration. DT uses CART for extracting the decision rules, RF in the standard configuration uses 30 decision trees, SVM-R and SVM-P have a kernel cache size of 200MB, the class weight is balanced, the decision function shape is set as one-vs-rest, and the max iteration is unbounded. SVM-P uses a polynome of 6^{th} grade.

All the neural network architectures have been implemented using Keras (Chollet et al., 2015) with Tensorflow (Abadi et al., 2015).

Inspired by the promising performances reported in (Xu et al., 2018), we adapted to our problem the following CNN architectures: LeNet-5 (Lecun et al., 1998), AlexNet (Krizhevsky et al., 2012), and GoogleNet (Szegedy et al., 2015). We chose these architectures because LeNet-5 has historical importance, AlexNet suggested the use of parallel convolutions, and GoogleNet showed impressive performances with its inception modules. In all the configurations, unless specified differently, the pooling method is *maxpooling* with 2x2 strides and *channel first* data format, and convolutions have *same padding*.

The architectures layers are reported in Table 4.2 and Table 4.4. Table 4.3 shows the building blocks. The convolution layer uses the format *filterNumber@kernelSize/activationFunction*, the pooling layer uses *poolingFunction(size)*, the *flatten* layer flattens the input in one dimension, and *dense(number of neurons)/activationFunction* is a layer densely connected with an activation function *activationFunction*. The first layer is the input layer.

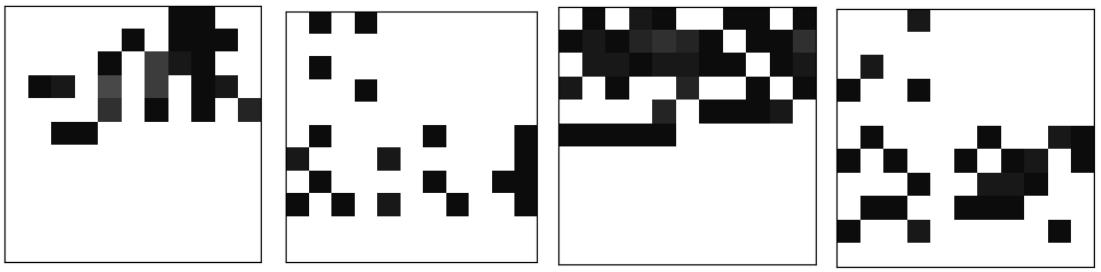
Because these architectures are created for image classification, we thought that the most natural way to exploit them was to transform the input vector into an image. We

TABLE 4.3: CNN architectures building blocks. In I3, the max pooling strides is 1x1. | is concatenation.

symbol	Layer 0	Layer 1
T1	30@2x2/selu	max(2x2)
T2	30@3x3/selu	max(2x2)
T3	30@4x4/selu	max(2x2)
T4	30@5x5/selu	max(2x2)
I0	64@1x1/relu	
I1	64@1x1/relu	64@3x3/relu
I2	64@1x1/relu	64@5x5/relu
I3	max(3x3)	64@1x1/relu
INC	I0 | I1 | I2 | I3	

TABLE 4.4: CNN architectures. | is concatenation.

CNN-5	CNN-6	CNN-7
11X11	11X11	11X11
T1 | T2 | T3 | T4	T1 | T2 | T3 | T4	INC | INC | INC
30@4x4/selu	flatten	INC | INC | INC
60@5x5/selu	dense(500)/selu	flatten
max(2x2)	dense(250)/selu	dense(500)/selu
flatten	dense(2)/softmax	dense(250)/selu
dense(150)/selu		dense(2)/softmax
dense(150)/selu		
dense(2)/softmax		



(A) (B) (C) (D)

FIGURE 4.4: Imagification outputs. Each example is an 11x11 pixel image. To have a clear image, and for visualisation purpose only, we rescaled the colours as follows: a white, 255, if the input feature for the pixel is 0, and a grey scale, minmax rescaled in the interval [0, 180], if the input value is greater than 0.

named the preprocessing procedure *imagification*: 1) given an input of n elements, find $h \in \mathcal{N}$ such that $\sqrt{n+h} = c \in \mathcal{N}$, 2) let m be the concatenation of the input n with an array of h zeros, 3) reshape m in a squared matrix, 4) rescale the input with a minmax scaler. The result is a single channel image. We show some examples of such outputs in Fig. 4.4.

Our classification problem has fixed input dimension and fixed output dimension. Another natural way to address the problem is using the universal approximation of FFNNs

(Trenn, 2008). All the FFNN configurations explored have an input layer of 112 nodes and an output layer with a 2 nodes soft-max (one-hot). The hidden layers are densely connected and the activation functions are Leaky ReLu (Maas et al., 2013) with the parameter $\alpha = 0.3$. We tested the architectures with one, two, and three hidden layers, with the number of nodes for each layer ranging from 10 to 100. The bias and initial weights for each layer have been initialised with the Xavier's method (Glorot and Bengio, 2010). The FFNN with three hidden layers have a dropout filter (Srivastava et al., 2014) with probability of 0.1.

The trainings in the CNN and FFNN experiments used a Stochastic Gradient Descendent optimiser with Nesterov momentum (Nesterov, 1983) and Learning rate decay of 1e-6. The loss function is a categorical cross entropy and the metric is accuracy.

The classification methods have been cross-validated by splitting the dataset in 10 folds, each one built with stratified sampling with shuffling before splitting into batches.

Due to our hardware limited capacity, we firstly made an exploratory training to find which architecture was most promising, then we tuned the most promising. The CNN experiments have been trained for 80 epochs, with a training batch size of 64, testing batch size of 8, and learning rate of 0.01. The FFNN have been trained for 30 epochs, with a training batch size of 32, testing batch size of 8, and learning rate of 0.01. Table 4.5 and Table 4.6 report some of the best results of the exploratory phase.

Table 4.5 shows the confusion matrix. FF represents the number of unfeasible instances that have been classified as unfeasible, FT is the number of unfeasible instances that have been classified as feasible, TF is the number of feasible instances classified as unfeasible, TT is the number of feasible instances classified as feasible. The columns report the average and the standard deviation (in parenthesis) among the 10 folds.

Table 4.6 shows precision, accuracy and F1 index calculated from the confusion matrix.

The metric that we are most interested in is the accuracy, since we need to maximise the sum of FF and TT. CNN and SVC-RBF are quite close to the best, but FFNN outperforms all other methods. In addition, FFNNs are faster to train and have fewer parameters than CNN.

We chose to further tune the configuration with one hidden layer with 100 neurons, because it has the best combination of accuracy and precision. Learning rates (LR) ranged from 0.0001 to 0.001, momentum from 0.6 to 0.9, the training and testing batch size from 256 to 4, the training epochs from 20 to 250.

The best configuration after tuning (Leaky-ReLu $\alpha = 0.3$, LR=0.001, decay=1E-06, training batch size=16, training epochs=120, test batch size=16, momentum=0.9) achieved

Table 4.5: Confusion matrix for the best results. FF means an unfeasible solution is correctly predicted as unfeasible, FT means an unfeasible solution is mistakenly predicted as feasible, TF means a feasible solution is mistakenly predicted as unfeasible, and TT means a feasible solution is correctly predicted as feasible. The values reported are the average over 10 buckets, while in parenthesis we report the standard deviations.

name	FF	FT	TF	TT
FFNN-90	539.7(10.8)	41.2(10.6)	59.4(10.5)	342.3(10.5)
FFNN-100	536.2(9.7)	44.7(9.5)	56.3(10.5)	345.4(10.6)
FFNN-60	534.6(9.8)	46.3(9.7)	58.2(6.9)	343.5(7.2)
FFNN-50	534.1(8.7)	46.8(8.7)	58.2(9.3)	343.5(9.3)
FFNN-50-30-40	526.7(8.0)	54.2(8.0)	51.3(8.9)	350.4(8.9)
FFNN-30-30-60	530.8(9.6)	50.1(9.7)	56.3(17.1)	345.4(17.0)
FFNN-30-30	526.8(13.6)	54.1(13.5)	56.4(12.1)	345.3(11.9)
FFNN-80	521.8(19.4)	59.1(19.3)	51.8(12.4)	349.9(12.2)
CNN-1	527.5(13.3)	53.4(13.2)	69.1(15.1)	332.6(15.1)
CNN-7	511.7(28.8)	69.2(28.8)	54.2(14.3)	347.5(14.2)
CNN-6	523.4(13.9)	57.5(13.7)	68.5(13.3)	333.2(13.2)
CNN-4	527.9(13.7)	53.0(13.7)	73.3(15.9)	328.4(16.1)
SVC-RBF	487.0(0)	93.0(0)	34.0(0)	367.0(0)
CNN-2	515.8(24.2)	65.1(24.2)	66.7(20.1)	335.0(20.1)
CNN-3	518.1(20.3)	62.8(20.3)	71.7(20.0)	330.0(20.0)
CNN-5	514.3(13.1)	66.6(13.0)	70.7(15.6)	331.0(15.6)
RF-30	498.0(0)	82.0(0)	69.0(0)	332.0(0)
DT	475.0(0)	105.0(0)	92.0(0)	309.0(0)
SVC-POLI	565.0(0)	15.0(0)	362.0(0)	39.0(0)

Table 4.6: metrics: precision, accuracy, F1

name	precision	accuracy	F1
FFNN-90	0.8521	0.8976	0.8719
FFNN-100	0.8598	0.8972	0.8724
FFNN-60	0.8551	0.8936	0.8680
FFNN-50	0.8551	0.8931	0.8674
FFNN-50-30-40	0.8723	0.8926	0.8692
FFNN-30-30-60	0.8598	0.8917	0.8665
FFNN-30-30	0.8596	0.8875	0.8621
FFNN-80	0.8710	0.8871	0.8632
CNN-1	0.8280	0.8753	0.8445
CNN-7	0.8651	0.8744	0.8492
CNN-6	0.8295	0.8718	0.8410
CNN-4	0.8175	0.8715	0.8387
SVC-RBF	0.9152	0.8705	0.8525
CNN-2	0.8340	0.8659	0.8356
CNN-3	0.8215	0.8631	0.8307
CNN-5	0.8240	0.8603	0.8282
RF-30	0.8279	0.8461	0.8147
DT	0.7706	0.7992	0.7583
SVC-POLI	0.0973	0.6157	0.1714

an accuracy of 0.9033 with standard deviation of 1.35. The second best (Leaky-ReLu $\alpha = 0,2$, LR=0.001, decay=1E-06, training batch size=16, training epochs=100, test batch size=16, momentum=0.7) achieved 0.9006 with standard deviation of 1.46.

4.4 Summary

This chapter shows the effectiveness of a novel strategy to determine if a particular packing solution is feasible in a constant O(1) computational time. We show a concrete application using a bin packing problem. The strategy consists of two stages. In the first stage, we exploit the master/slave Bender's decomposition to build a dataset of knapsack solutions. In the second stage we exploit the dataset to train a classifier to test the feasibility of a packing solution. The second novelty is a new dataset of packing solutions, which is exploited to benchmark different classification algorithms: decision trees, random forest, support vector machine with radial basis function and polynomial kernel, three architectures of convolutional neural networks, and feed forward fully connected neural networks with one, two and three hidden layers. The most promising classifier that has been tuned reports an accuracy of about 90%. The setup of a packing problem as a classification problem is the third novelty.

Chapter 5

The Offline Group Seat Reservation Knapsack Problem with Profit on Seats

In this chapter we present the Group Seat Reservation Knapsack Problem with Profit on Seats. This is an extension of the Offline Group Seat Reservation Knapsack Problem. In this extension we introduce a profit evaluation dependent on not only the space occupied, but also on the individual profit brought by each reserved seat. An application of the new features introduced in the proposed extension is to influence the distribution of passengers, such as assigning seats near the carriage centre for long journeys, and close to the door for short journeys. Such distribution helps to reduce the excess of dwelling time on platform. We introduce a new GRASP based algorithm that solves the original problem and the newly proposed one. In the experimental section we show that such algorithm can be useful to provide a good feasible solution very rapidly, a desiderable condition in many of the real world booking systems, such as the online booking. Another application could be to use the algorithm solution as a startup for a successive branch and bound procedure, when optimality is a must. We also add a new class of problem with five test instances that represent the worse case real world scenarios more realistically and we evaluate both the existing model, the newly proposed model, and the limitations of the proposed algorithm facing more realistic scenarios.

The first novelty of this chapter is the introduction of a new problem extension, the GSR-KPPS, that binds seat-based profit with the length of the journey reservation in a mixed integer programming (MIP) model.

The second novelty is a GRASP based algorithm that is suitable for both GSR-KP and GSR-KPPS. Such an algorithm is useful when the time to achieve a solution is fixed.

The third novelty is the adaptation and extension of the instances used in Clausen et al. (2010). We introduced a new group of problems that better represents worst case real world scenarios than those previously suggested in the original paper.

The chapter is organised as follow. Definitions and terminology follows in Section 5.1 along with the MIP model in detail, Section 5.2 outlines the proposed algorithm. The new instances are explained in Section 5.3, Section 5.4 shows computational results and the chapter ends with a summary in Section 5.5.

5.1 Definitions, terminology and MIP model

Using the usual terminology of the Packing Problems and utilising as much as possible the terminology used in Clausen et al. (2010), a train contains W seats and stops at H stations. Let $N = \{1, \ldots, n\}$ be the set of reservations. Each reservation asks to reserve w_i seats from station y_i to station $y_i + h_i$. Without any loss of generality we can assume that $w_i \leq W$.

First we briefly describe the original GSR-KP as shown in Clausen et al. (2010). The active stations are represented as $Y := \{y_i, |i \in N\} \bigcup \{y_i + h_i, i \in N\}$,

and N_y is the set of requests using a seat at station $y \in Y$

$N_y := \{i \in N\} | y_i \leq y < y_i + h_i$

Associated with each station $y \in Y$ there is a "height" H_y that represents the distance from station y to the next active station in Y. Let $\delta_i = 1$ if request i is selected. Let x_i be the first seat (left coordinate) of request i. Let $E = \{(i, j)\}$ be the set of rectangle pairs which in some way share a station (y-coordinate). Finally, let $l_{ij} = 1$ if and only if request i is located left of request j. The original model is shown in Eq: 5.1-5.8, the item profit is identified with the item area $(w_i \cdot h_i)$. We refer the reader to the original paper for a complete explanation.

$$\text{maximize} \quad \sum_{i \in N} h_i \cdot w_i \cdot \delta_i \tag{5.1}$$

$$\text{s.t.} \quad \sum_{i \in N_y} w_i \cdot \delta_i \leq W, y \in Y, \tag{5.2}$$

$$\delta_i + \delta_j - l_{i,j} - l_{j,i} \leq 1, (i,j) \in E, \tag{5.3}$$

$$x_i - x_j + W \cdot l_{i,j} \leq W - w_i, (i,j) \in E, \tag{5.4}$$

$$0 \leq x_i \leq W - w_i, i \in N, \tag{5.5}$$

$$l_{i,j} \in \{0,1\}, (i,j) \in E, \tag{5.6}$$

$$\delta_i \in \{0,1\}, i \in N \tag{5.7}$$

$$x_i \geq 0, i \in N. \tag{5.8}$$

Our model extends the original one by considering also a profit value associated to the seat. From the modelling perspective, it translates in a two-dimensional knapsack problem where the item profit is dependent on a combination of its area and its position inside the bin. The distribution of the passengers among and along the carriages can be modelled by assigning profits on the seats, i.e. considering the seats of each carriage, the central seats have higher profit than the seats near to the doors (this profits assignment will allocate reservations with longer journeys or more people in the centre of the carriage).

Let $Q := \{1, \ldots, W\}$ be the set of seats, and $p_k, k \in Q$ be the profit p_k associated to the seat k. Let $\gamma_{i,k}, i \in N, k \in Q$ be 1 if and only if reservation i occupies seat k.

The new formulation is shown in Eq: 5.9-5.19.

$$\text{maximize} \quad \sum_{i \in N} \sum_{k \in Q} h_i \cdot \gamma_{i,k} \cdot p_k \tag{5.9}$$

$$\text{s.t.} \quad \sum_{i \in N_y} w_i \cdot \delta_i \leq W, y \in Y, \tag{5.10}$$

$$\delta_i + \delta_j - l_{i,j} - l_{j,i} \leq 1, (i,j) \in E, \tag{5.11}$$

$$x_i - x_j + W \cdot l_{i,j} \leq W - w_i, (i,j) \in E, \tag{5.12}$$

$$w_i \cdot \delta_i \leq \sum_{k \in Q} \gamma_{i,k} \leq w_i \cdot \delta_i, \forall i \in N \tag{5.13}$$

$$-(1 - \gamma_{i,k}) \cdot 2W + x_i \leq \gamma_{i,k} \cdot k \leq x_i + w_i \cdot \delta_i, \forall i \in N, k \in Q \tag{5.14}$$

$$\gamma_{i,k} \in \{0,1\}, i \in N, k \in Q \tag{5.15}$$

$$0 \leq x_i \leq W - w_i, i \in N, \tag{5.16}$$

$$l_{i,j} \in \{0,1\}, (i,j) \in E, \tag{5.17}$$

$$\delta_i \in \{0,1\}, i \in N \tag{5.18}$$

$$x_i \geq 0, i \in N. \tag{5.19}$$

The differences between the models are on the objective 5.9 which now includes the profit associated on the seat, and in three additional constraints 5.13-5.15. Considering a unitary profit, we will produce the same results of the original model, thus, the proposed model can be considered as an extension of the original problem. Constraint 5.13 represents the total allocation of a reservation. If the reservation i is booked, then w_i seats must be allocated, otherwise none has to be assigned. Constraint 5.14 enforces the contiguity of the allocated seats k, for the reservation i. Constraint 5.10 represents the knapsack constraint, which enforces allocation inside the train. Constraint 5.11-5.12 represents the fact that two items i,j cannot overlap.

One limitation of the objective function of the model is that it does not distinguish between one reservation for n stations and n reservations of one station. For example, a long-distance traveller travelling from station A to H will have the same profit of other travellers travelling from A to C, C to D, D to H.

5.2 Proposed algorithm

In this section we describe the proposed algorithm. The algorithm is a GRASP procedure (Feo and Resende, 1989, 1995; Resende and Ribeiro, 2016, 2019) that exploits a

percentage of the best bound found by the continuous relaxation of the problem (relaxing integer and boolean variables to real variables) for enforcing a stopping condition. The rationale to use a GRASP based method is to produce a simple algorithm that produces good solutions in a very limited time. Such algorithm can be used as a start-up for a successive branch and bound procedure, or can be used directly when achieving a solution in the timelimit is more important than the optimality of the solution.

The main procedure, Algorithm 3 (Algorithm will be abbreviated as Alg from now on), is composed of the following steps: create a random candidate solution, evaluate the candidate and update the best solution if the candidate improves the current best solution. If the solution is not improved then pick a uniformly random number $c \in [0, 1]$ and if $c \leq 0.5$ try to improve the current candidate, otherwise try to improve the best solution found so far.

The stopping criteria of the main procedure are met when at least one of the following conditions is met. First, the maximal number of iterations *max_iterations* has been achieved. Second, a time threshold *timelimit* has been reached. Third, a threshold has been reached on the best candidate evaluation c_{best}. The last threshold is calculated as the fraction *bratio* of the objective value *relaxed_bound* of the continuous relaxation of the problem. The combination of these three stopping criteria has been chosen to keep the running time of the algorithm balanced in borderline conditions.

The return values of the main procedure are *best*, *limit* and c_{best}. *best* is the sequence of indexes that represents the best solution, *limit* is the position of the last fitting reservation index in *best*, c_{best} is the evaluation of the profit totalised in the feasible part of the best solution.

The evaluation procedure, Alg 4, requires as input the *candidate* list. We remind the reader that a *candidate* solution is a permutation of the n indices that represent the reservations, the evaluation procedure "cuts" the candidate up to the last feasible element *limit*. There are two ideas behind the evaluation: firstly to exploit the corner point concept presented in (Martello and Toth, 1990) and secondly to exploit the problem structure, and reduce the positions to evaluate along the horizontal axis only. The evaluation procedure keeps a list of candidate positions in *corner*. The algorithm tries to place the items in the first feasible candidate position. *corner* is firstly initialised with the position 0. After an item i fits in a position $x \in corner$, the candidate positions list is updated with the corner of the item i, $corner := corner \bigcup \{w_i + x\}$. The evaluation procedure is a constructive first fit heuristic (Zhu et al., 2012).

Algorithm 3 $main\_procedure(relaxed\_bound, bratio, timelimit, max\_iterations)$

$c_{best} \Leftarrow epoch \Leftarrow 0$, $start \Leftarrow current\_time()$, $best \Leftarrow \emptyset$
while $current\_time() - start \leq timelimit$ **and** $relaxed\_bound \cdot bratio > c_{best}$ **and** $epoch < max\_iterations$ **do**
  $candidate, limit, bound \Leftarrow generate\_candidate()$
  if $bound > c_{best}$ **then**
    $c_{best}, limit_{best}, best \Leftarrow bound, limit, candidate$
  else
    if $uniform(0, 1) \leq 0.5$ **then**
      $candidate, limit, bound \Leftarrow local\_search(candidate, limit)$
      if $bound > c_{best}$ **then**
        $c_{best}, limit_{best}, best \Leftarrow bound, limit, candidate$
      end if
    else
      $candidate, limit, bound \Leftarrow local\_search(best, limit_{best})$
      if $bound > c_{best}$ **then**
        $c_{best}, limit_{best}, best \Leftarrow bound, limit, candidate$
      end if
    end if
  end if
  $epoch \Leftarrow epoch + 1$
end while
return $best, limit, c_{best}$

The candidate generation, Alg 5, makes use of a $shuffle(x)$ function, where x is the set to shuffle. $shuffle(x)$ returns a random permutation of x. After that, the candidate is evaluated with the procedure in Alg 4.

The local search procedure, Alg 6, swaps half of the positions of the feasible region with positions picked randomly. The method exploits the solution structure: items that belong to feasible regions are located in the initial part of the solution array. So swapping half of the items forces the method to evaluate new solutions while keeping parts of the solution. An example is shown in Fig 5.1. The local search procedure is in fact reducible to the 2-opt local search (Croes, 1958), with the identification of the sets of the candidates to swap with the feasible and unfeasible region.

The Alg 3-6 are designed to be a very fast procedure that can be used to determine lower bounds for a branch and bound framework. The component with varying computational cost is the evaluation procedure Alg 4, since Alg 5 and Alg 6 have a constant number of operations.

Let n be the number of items and W be the maximal number of seats. The worst case scenario for Alg 4 is having groups of one element, the procedure will make $\frac{n \cdot (n+1)}{2}$ $isNotOverlapping(x, i, positioned)$ operations. $isNotOverlapping(x, i, positioned)$ can be implemented as a loop with a check if the new item overlaps with the others already

Algorithm 4 *evaluate_candidate*(*candidate*)

Require: *candidate* ordered list of indexes
Require: n number of items
Require: $N \Leftarrow$ set of items
  $corner \Leftarrow \{0\}$
  $positioned \Leftarrow \emptyset$
  $end = n$
  $bound \Leftarrow 0$
  for $i \in candidate$ **do**
    $test \Leftarrow$ **False**
    if $h_i + y_i \leq H$ **then**
      for $x \in corner$ **do**
        if $x + w_i \leq W$ **then**
          if $isNotOverlapping(x, i, positioned)$ **then**
            $positioned \Leftarrow positioned \bigcup \{(x, i)\}$
            $corner \Leftarrow corner \bigcup \{x + w_i\}$
            $test \Leftarrow$ **True**
          end if
        end if
      end for
    end if
    if $test =$ **True then**
      $bound \Leftarrow bound + \sum_{j \in Q} p_j * h_i$
    else
      $end \Leftarrow$ position of i in candidate
  return*candidate*, *end*, *bound*
    end if
  end for
  return*candidate*, *end*, *bound*

Algorithm 5 *generate_candidate*()

Require: n number of items
  $candidate \Leftarrow shuffle([1, \ldots, n])$
  return*evaluate_candidate*(*candidate*)

placed. In the worst case it has to compare N items. Summarising, the evaluation procedure Alg 4 has a worst case scenario with a time complexity of $O(N^3)$.

This complexity can be reduced to $O(log_2(n) \cdot N^2)$ by using a balanced binary tree to represent the already placed objects, and a dichotomy search for the $isNotOverlapping(x, i, positioned)$ procedure.

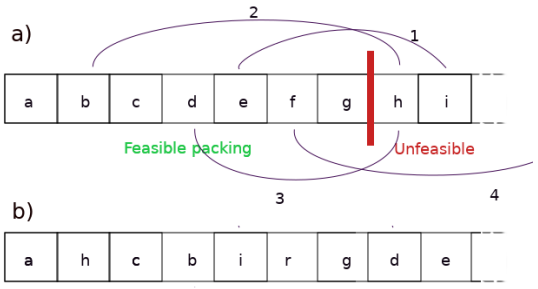


FIGURE 5.1: An example of a local search procedure. a) shows the swap sequence between the feasible and unfeasible region, b) reports the array consequence of the swapping sequence, the limit of the feasible region is removed because the new array requires a new evaluation. The reader should note that since the swapping sequence is random, the combination of multiple swaps may result in a swap of elements in the feasible region.

Algorithm 6 $local\_search(candidate, limit)$

Require: $n, number of available items$
Require: $candidate$ ordered list of indexes from 1 to n
Require: $limit$ index of the candidate list that indicate the first element that does not fit in the bin
  $s \Leftarrow round(limit/2)$
  if $s = 0$ **then**
    $s \Leftarrow 1$
  end if
  $counter \Leftarrow 0$
  while $counter \leq s$ **do**
    $source \Leftarrow uniform(1, limit)$
    $target \Leftarrow uniform(1, n)$
    if $source \neq target$ **then**
      $swap(candidate_{source}, candidate_{target})$
    end if
  end while
  return $evaluate\_candidate(candidate)$

5.3 Class instances

The original paper (Clausen et al., 2010) considers problem instances used in the literature of the two-dimensional packing, in a total of 190 experiments in five main classes, namely CGCUT (Christofides and Whitlock, 1977), WANG (Wang, 1983), GCUT (Beasley, 1985), OKP (Fekete et al., 2007), GXON and GXOU (Clausen et al., 2010). The instances can be downloaded from the author's repository [1].

[1] http://hjemmesider.diku.dk/~pisinger/codes.html

TABLE 5.1: Main features of the original instances compared with the proposed one

class	experiments	stations		seats		journey		reservations		groups	
name	number	min	max	min	max	min	max	min	max	min	max
CGCUT	15	15	40	10	70	2	33	16	62	1	43
GXON	60	100	100	100	100	1	45	20	50	1	45
GXOU	20	100	100	100	100	1	35	20	50	6	37
GCUT	65	250	3000	250	3000	62	970	10	50	63	1890
OKP	25	100	100	100	100	1	100	30	97	1	99
WANG	5	70	70	40	40	11	43	42	42	9	33
DEPL	5	6	50	400	750	1	48	2000	2500	1	9

Table 5.1 shows a comparison between the main features. For each feature we report the minimum and maximum values for each parameter to show a broad picture of the problem class.

Experiments number shows the number of instances available in the class, *stations* reports the journey length measured as number of stations, *seats* is the number of seats available in the train, *journey* represents the journey length for the reservations, *reservation* stands for the number of reservations for each instance and *groups* are the group dimension in the reservations.

DEPL is the new class of instances that we propose, considering also the recent work of (Smith-Miles and Lopes, 2012). The idea behind this is to provide a worst case class inspired by a real-world scenario that can be hard to solve. DEPL has a high number of reservations to represent a busy connection between cities, and a range of limits for the other features compatible with a broad range of railway journeys. The reason for DEPL to consider up to 50 stations only is that a journey lasting 50 stations is unlikely to happen bar exceptional cases. An example of an exceptional case could be the Trans-Siberian Railway (the longest in the world), which stops in 157 stations during the journey from Moscow to Vladivostok.

5.4 Experimental results

The experiments are divided into two main groups, group one considers unitary profits while group two considers random profit. For each group we evaluated all the instances of the classes in Table 5.1: a total of 195 problem instances per group.

In each experiment we reported the gap, defined as the best bound of the continuous relaxation of the model. The formula is reported in Eq: 5.20. We applied the same formula to calculate the gap in the heuristic results as well.

$$gap = \frac{|bestbound - bestinteger|}{\frac{1}{10^{10}} + |bestinteger|} \tag{5.20}$$

The model in Eq:5.9-5.19 has been implemented in OPL and solved using IBM CPLEX® 12.7. The proposed Algorithms 3-6 have been implemented in Python 3.6.7. The machine used for running the experiments is an Intel® Core™ i7-7700HQ @ 2.80GHz, 16GB DDR4 RAM. The operating system is Ubuntu™ 18.04. The time limit for solving each instance is 20 minutes: this choice is reasonable if we consider 1) a booking system that accepts prenotations up to one hour before the train departure and 2) the seats must be communicated before people arrive at the platform, e.g. by email, by printing the seat number at the station gate, or by a smartphone application. The proposed algorithm has been run three times for each instance to avoid eventual bias due to a lucky initialisation. The stopping criteria for Algorithm 3-6 are: $bratio = 0.95$ (95% of the result of the continuous relaxation), $max\_iterations = 15000$, and $timelimit = 1200$ seconds. For most of the experiments the $max\_iterations$ and the $bratio$ are the triggering stopping criteria, while in the DEPL class, because of the massive number of items, the evaluation procedure is considerably slower and the $timelimit$ becomes the only stopping criterion.

Table 5.2 and Table 5.3 report the experiments of group one. Table 5.4 and Table 5.5 report the experiments of group two. For CPLEX, the tables report average and standard deviation of the gap, calculated using as the baseline the continuous relaxation, and time (reported in seconds). For the algorithm, the tables report the average and standard deviation of computational time (seconds), and average, standard deviation, minimum and maximum. The last column is the difference between the mean gap achieved by CPLEX and the mean gap achieved by the algorithm. This column highlights the degradation of the objective. We highlight any gap degradation lower than 10% in bold font.

In all the instances solved in Table 5.2-5.3-5.4-5.5, our algorithm achieved a maximum running time of 6.35 seconds and a minimum of 5.67E-05 seconds. The average running time in the first group is 1.95 seconds, while in the second group it is 2.32 seconds.

The experiments of group one, apart from G40U20, G50U20 and GCUT13, are relatively easy to solve for CPLEX: 50% of the experiment classes in the first group have an average gap difference lower than 10%.

The second group is more difficult to solve for CPLEX: 57.89% of the classes ran an average of more than 16 minutes, while the objective degradation was averagely less than 10% in 26.31% of the experiment classes.

TABLE 5.2: Experiment group one, comparison with unitary profit, first part

instance	CPLEX				algorithm						
	gap		time		gap				time		
name	mean	std	mean	std	mean	std	min	max	mean	std	difference
CGCUT01	1.38	3.08	0.04	0.01	28.52	11.61	18.18	42.11	1.92	0.09	27.14
CGCUT02	1.52	1.77	0.73	0.27	9.06	6.83	1.23	22.22	1.69	1.34	**7.54**
CGCUT03	4.66	4.49	1.09	0.3	20.38	16.02	1.83	47.87	1.92	0.83	15.72
G20N10	1.42	1.28	2.73	2.33	6.43	3.17	2.85	11.90	1.64	1.21	**5.01**
G20N20	0.18	0.24	1.54	1.41	4.88	3.39	0.43	12.37	1.26	1.52	**4.70**
G20N30	0	0	0.54	0.05	6.25	5.77	0.90	16.51	0.86	1.86	**6.25**
G20U20	1.73	2.63	3.14	2.85	34.61	18.87	10.39	62.57	2.27	0.48	32.88
G30N10	4.32	2.96	7.81	3.94	11.41	2.19	6.68	17.57	3.15	0.16	**7.09**
G30N20	0.79	0.71	5.52	2.91	11.23	4.19	2.40	18.37	3.05	0.63	10.44
G30N30	0	0	1.97	1.3	3.69	0.89	0.78	5.17	0.32	0.36	**3.69**
G30U20	0.9	0.83	40.3	43.59	42.21	5.52	33.67	55.40	2.55	0.38	41.31
G40N10	4.59	4.18	9.36	7.9	23.94	3.41	13.81	36.68	3.21	0.12	19.35
G40N20	0	0	13.9	10	10.03	5.26	3.09	23.81	2.90	1.11	10.03
G40N30	0	0	5.26	2.37	8.99	6.21	2.99	20.00	2.55	2.22	**8.99**
G40U20	2.01	0.95	70.19	71.48	53.11	11.08	28.18	73.76	2.72	0.15	51.10
G50N10	3.36	1.87	22.96	27.98	29.19	3.74	21.77	35.92	3.48	0.29	25.83
G50N20	1.69	2.36	12.1	3.96	19.45	10.18	8.46	38.86	3.49	0.11	17.76
G50N30	0	0	6.03	3.55	8.95	4.96	4.40	22.22	5.28	0.83	**8.95**
G50U20	0.99	1.11	195.36	303.41	64.08	12.06	43.03	84.65	2.66	0.20	63.09

DEPL experiments with random profit are reported in Table 5.6. CPLEX ran out of memory in all the experiments carried out. We were not able to provide the solution of the continuous relaxation, thus we were not able to calculate the gap between the relaxation and the best solution found. Consequently, we decided to run the instances with the proposed algorithm only at different time limits, one minute, three minutes and one hour. Table 5.6 reports the average number of iterations made, maximum, minimum and average objective with standard deviation value found with three different time limits for the heuristic. The result shows that considering a much larger number of items, the algorithm's chances to improve an already good solution by remixing part of the best solution or part of the actual solution are less. The evaluation process becomes much slower. For example, with the instance DEPL_0, tripling the time from 60 to 180 seconds only produced a gain in the average objective of 1.64%, and when we increase the time limit from 1 minute to 60 minutes, the gain in the objective was only 5.07%. A similar pattern can be seen in the other cases, where the best gain after an hour of computation was 12.17% in the objective value. To sum it up, the experimental results have shown that the proposed heuristic is a useful tool to provide good, feasible and quick solutions for the challenging instances in which CPLEX fails. However, letting the heuristic run for an extended period will not improve performance significantly.

Table 5.3: Experiment group one, comparison with unitary profit, second part

instance	CPLEX gap		CPLEX time		algorithm gap				algorithm time		
name	mean	std	mean	std	mean	std	min	max	mean	std	difference
GCUT01	27.62	5.23	0.19	0.02	76.96	75.95	23.32	257.14	0.87	0.79	49.34
GCUT02	11.45	8.7	0.84	0.1	74.73	42.07	17.65	185.71	0.34	0.75	63.28
GCUT03	13.17	11.46	1.78	0.96	69.64	77.38	1.42	259.12	0.84	0.90	65.47
GCUT04	1.71	1.59	4.75	2.76	12.78	7.60	2.64	31.25	1.54	0.90	11.07
GCUT05	5.16	7.77	1.11	0.5	5.29	7.67	0.45	18.46	0.69	0.93	**0.13**
GCUT06	10.37	12.67	2.3	1.27	11.49	11.67	1.27	24.53	0.70	0.90	**1.12**
GCUT07	8.49	5.63	3.94	1.45	8.84	5.26	4.60	18.06	1.43	0.71	**0.35**
GCUT08	1.06	1.09	14.05	3.46	4.82	1.53	0.20	9.41	1.02	0.84	**3.76**
GCUT09	10.62	10.64	1.82	0.8	11.10	10.11	0.35	24.53	0.97	0.88	**0.48**
GCUT10	1.99	1.6	3.63	0.78	2.62	1.21	0.81	4.44	0.06	0.08	**0.63**
GCUT11	7.31	7.75	14.04	10.55	11.56	6.29	0.60	17.79	1.46	0.68	**4.25**
GCUT12	1.36	0.7	18.67	6.94	5.77	3.31	0.70	18.98	0.86	0.65	**4.41**
GCUT13	21.3	22.02	864.27	0.16	14.98	2.59	9.64	21.63	3.60	0.11	**-6.32**
OKP01	6.13	2.48	1.13	0.41	38.46	9.86	17.33	68.38	2.07	0.03	32.33
OKP02	12.55	1.51	0.67	0.16	19.27	7.95	12.52	35.08	1.86	0.06	**6.72**
OKP03	6.26	1.3	0.39	0.06	7.95	2.26	5.41	22.64	1.83	0.05	**1.69**
OKP04	8.7	2.73	1.83	0.35	44.10	9.66	25.51	74.96	2.26	0.04	35.40
OKP05	17.57	2.95	6.99	2.13	75.39	18.04	36.19	123.60	2.90	0.03	57.82
WANG20	11.22	5.75	0.43	0.11	34.56	9.42	13.93	56.44	1.91	0.03	23.34

5.5 Summary

In this chapter we have developed a mixed integer programming model for the Group Seat Reservation Knapsack Problem with Profit on Seat. It is an extension of the Offline Group Seat Reservation Knapsack Problem that introduces a profit evaluation dependent on the reservation profit, proportional to the journey length and group size, and on the seat profit in which the reservation is allocated. The proposed extension covers situations where the ideal position of an item is affected by how long the item must keep that position.

We have developed a new GRASP based algorithm that solves the original problem version and the newly proposed one.

We have improved the instances considered in the original paper with five new problems that better represent worst cases of real world scenarios and we have evaluated the limitations of the proposed algorithm.

In the experimental section we have shown that the proposed algorithm can be useful to provide a first lower bound very rapidly, which can be used as a startup for a successive

Table 5.4: Experiment group two, comparison with random profit, part one

instance	CPLEX gap		CPLEX time		algorithm gap				algorithm time		
name	mean	std	mean	std	mean	std	min	max	mean	std	difference
CGCUT01	46.80	13.36	0.47	0.25	118.66	17.27	117.93	120.10	1.70	0.09	71.86
CGCUT02	58.84	12.23	1200.39	0.84	82.06	15.02	80.62	84.49	2.47	0.27	23.22
CGCUT03	64.91	14.20	580.02	409.21	106.61	27.69	101.06	114.82	1.98	0.08	41.70
G20N10	60.69	9.27	827.67	477.01	78.81	7.14	77.53	80.71	2.70	0.24	18.11
G20N20	60.46	9.04	946.67	347.33	88.37	12.94	86.22	90.43	2.65	0.30	27.91
G20N30	53.84	13.73	1141.04	131.85	91.99	11.63	91.68	92.19	4.57	0.78	38.15
G20U20	67.83	11.86	1042.28	216.89	133.10	28.98	127.70	139.59	1.97	0.45	65.28
G30N10	70.59	10.41	1029.90	252.47	93.82	6.27	87.39	99.70	2.80	0.16	23.23
G30N20	64.13	7.36	1008.56	337.22	95.74	12.98	90.78	100.44	2.82	0.17	31.61
G30N30	54.73	14.48	1200.01	0.00	82.83	9.72	81.74	83.61	4.87	1.16	28.10
G30U20	81.32	3.34	665.11	312.73	151.91	11.82	144.30	160.55	2.25	0.37	70.59
G40N10	68.56	8.22	1177.32	52.17	110.12	14.65	104.02	115.68	2.85	0.17	41.56
G40N20	75.04	10.48	1099.03	226.54	101.28	7.28	96.85	105.63	3.01	0.19	26.25
G40N30	57.90	13.72	1200.26	0.54	97.77	21.76	93.40	101.14	4.88	0.78	39.88
G40U20	157.29	97.61	924.14	275.48	182.74	25.11	168.71	193.63	2.32	0.15	25.45
G50N10	78.02	8.25	1121.18	177.59	128.50	18.86	119.13	140.83	3.03	0.31	50.48
G50N20	75.50	7.91	1149.88	112.31	120.58	17.31	113.41	128.85	3.11	0.10	45.08
G50N30	57.06	8.72	1092.70	239.96	99.18	11.33	95.14	103.16	4.81	0.55	42.12
G50U20	172.74	79.85	937.84	263.62	199.80	12.50	190.17	209.26	2.34	0.16	27.05

branch and bound procedure. However, it may be used in the cases where achieving a solution in the timelimit is more important than the optimality of the solution.

Table 5.5: Experiment group two, comparison with random profit, part two

instance	CPLEX gap		CPLEX time		algorithm gap				algorithm time		
name	mean	std	mean	std	mean	std	min	max	mean	std	difference
GCUT01	98.60	19.95	18.74	7.33	223.77	186.94	201.86	252.10	0.76	0.69	125.17
GCUT02	76.81	7.21	318.76	460.77	105.62	31.45	97.83	112.42	0.59	0.80	28.81
GCUT03	80.59	11.51	684.56	547.00	162.21	96.29	145.38	176.55	0.94	0.85	81.62
GCUT04	74.59	7.79	1200.02	0.00	99.73	9.33	85.59	109.15	1.86	0.09	25.13
GCUT05	75.91	8.78	1077.14	121.75	80.96	7.82	80.96	80.96	1.45	0.07	**5.05**
GCUT06	75.38	3.87	934.90	429.82	78.44	5.58	78.44	78.44	1.46	0.01	**3.06**
GCUT07	88.10	4.04	1200.01	0.00	92.48	8.26	92.27	92.59	1.53	0.02	**4.38**
GCUT08	81.50	6.64	1168.73	70.00	83.73	6.66	80.18	85.66	1.85	0.03	**2.23**
GCUT09	76.22	11.59	1200.02	0.01	80.34	11.83	80.34	80.34	1.37	0.04	**4.12**
GCUT10	67.50	7.57	1001.04	273.37	69.89	9.37	69.89	69.89	1.43	0.02	**2.39**
GCUT11	83.63	11.37	1200.08	0.05	89.59	7.40	86.36	94.30	1.63	0.03	**5.96**
GCUT12	77.25	6.42	1200.13	0.16	82.05	6.29	78.14	89.62	1.81	0.02	**4.80**
GCUT13	111.15	19.27	1200.21	0.21	103.62	6.89	98.23	110.13	3.15	0.07	**-7.53**
OKP01	41.72	8.33	53.60	25.01	89.48	6.21	80.30	99.89	1.79	0.02	47.77
OKP02	56.61	11.81	17.76	4.98	74.03	15.99	70.20	79.53	1.62	0.08	17.42
OKP03	49.32	12.73	6.15	2.92	57.78	7.12	52.34	63.65	1.60	0.03	**8.46**
OKP04	46.60	7.83	51.68	70.13	100.59	24.30	82.61	119.71	1.94	0.04	53.99
OKP05	69.01	8.52	1135.56	144.74	153.13	10.19	144.65	162.06	2.52	0.06	84.13
WANG20	61.46	14.80	6.54	4.80	119.14	25.29	109.05	130.81	1.71	0.07	57.67

Table 5.6: Experiment group two, DEPL, with different time limits

instance	CPLEX iterations	algorithm obj				time	gain
name		#	mean	std	min	max	limit	%
DEPL_0	oom	7	10289	242.396	10055	10539	60	0
DEPL_0	oom	16	10458.33	160.051	10331	10638	180	1.64
DEPL_0	oom	284	10810.66	127.021	10685	10939	3600	5.07
DEPL_1	oom	20	9706.33	105.547	9599	9810	60	0
DEPL_1	oom	55	9895	152.302	9771	10065	180	1.94
DEPL_1	oom	1020	10654.66	219.62	10449	10886	3600	9.77
DEPL_2	oom	8	18436.67	303.216	18173	18768	60	0
DEPL_2	oom	19	19467	502.012	18967	19971	180	5.58
DEPL_2	oom	334	20342	172.6	20233	20541	3600	10.33
DEPL_3	oom	35	13006.67	114.988	12913	13135	60	0
DEPL_3	oom	103	13342	137.328	13195	13467	180	2.57
DEPL_3	oom	1858	14589.66	465.6	14271	15124	3600	12.17
DEPL_4	oom	64	20474.67	380.245	20060	20807	60	0
DEPL_4	oom	179	20837.67	447.474	20321	21101	180	1.77
DEPL_4	oom	3930	22723.66	582.85	22189	23345	3600	10.98

Chapter 6

A data-driven methodology to infer the graph of the feasible train routing strategies from open data in UK rail network

In this chapter we present a data-driven method to infer useful infrastructure and feasible train routing strategies from open data in UK rail network. The majority of the network is divided into sections called berths, and the transition point from one berth to another is called a berth step. There are sensors at berth steps that can detect the movement when a train passes by. The result of the method is a directed graph, where each node represents a berth and each arc represents a berth-step. The arcs represent the feasible routing strategies, i.e. where a train can move from one berth. A connected path between two berths represents a connected section of the network.

Our work is important for the following reasons. Firstly, the *berth graphs* are not publicly available in an easy machine-readable format. All existing copies are only for a local network, and were created manually as hard copies. Secondly, a method to infer the graph directly from the stream of data produced allows automatic re-configuration after special maintenance, i.e. if a new line is built, a junction is introduced, and so on. Thirdly, after the graph is established and validated, it is easy to modify the method for detecting in real-time events such as: failure of the train detection system, i.e. trains that cannot be detected easily due to railhead contamination problems or on-board systems not working properly. Such an event detection can help improve service recovery and network maintenance. Fourthly, the *berth graph* is the model that represents the feasible train routing strategies in network. The model has been exploited to develop

a reactionary delay simulator and an optimiser of mitigation strategies in the RSSB funded COF-INP-05: "Anticipating and mitigating reactionary delays: A case study on Merseyrail's Northern Line".

The chapter is organised as follows. Section 6.1 provides an overview about the data we will use; Section 6.2 shows the algorithm; and Section 6.3 reports the results. The chapter finishes with the conclusions in Section 6.4.

6.1 Overview about the C-class of Train Describer data

The Train Describer (TD) data reports seven types of messages, divided into two classes: *C-class* and *S-class*. The first one, TD-C-class, provides information about the transition of trains between berths. The second one, TD-S-class, provides information about the state of the signalling hardware placed on the network. This study exploits the TD-C-class. Scholars or practitioners who are interested in the TD-S-CLASS should consider to have access to software such as the Network Rail's VISION software or, eventually, the configuration map between the signal addresses and the TD-S-CLASS bitmap (Open Rail Data Wiki, n.d.).

The C-class message data source will be abbreviated as TD-C in the next parts. TD-C messages have 7 fields (Open Rail Data Wiki, n.d.). The field *timestamp* is the time that the message has been received in the mainframe, `area_id` identifies the geographical *area*. `msg_type` is the message type, Table 6.1 shows the possible message types for TD-C. `headcode` represents the train code, the first character is the train class, the second is the destination area and the last two are a numeric counter. `from` and `to` are berth id, in combination with `area_id` they represent the berth step. `report_time` represents the hour, minute and second of the message when it was sent. Table 6.2 summarizes the fields and in which message type they appear. The field *descr* represents the *headcode* in the official data-feed documentation. From now on, for the sake of clarity, we are going to use *headcode*.

TABLE 6.1: TD-C message types

Event type	Message description
CA	Transition from a berth to another
CB	Cancels the description of the previous (from) berth
CC	Overwrites the description of the current (to) berth
CT	Heartbeat, periodically sent

A typical TD-C message inside an area code is shown in Table 6.3. Table 6.4 shows a sequence of messages produced during a transition between an area code and another one.

TABLE 6.2: TD-C message fields according to message types.

Field	Size	CA	CB	CC	CT	Description
timestamp	10	•	•	•	•	UNIX timestamp in milliseconds
area_id	2	•	•	•	•	Area code
msg_type	2	•	•	•	•	CA, CB, CC, CT
from	4	•	•			From berth
to	4	•		•		To berth
descr	4	•	•	•		Headcode
report_time	6				•	Reporting time

TABLE 6.3: Typical TD-C message sequence (Rail, a) (comments added by the authors)

Timestamp	Type	TD area	Headcode	from	to	Time	Comment
1514795790048	CA	SS	2U10	0054	0052	083629	Departure at Sandhills
1514795844213	CA	SS	2U10	0052	0036	083723	
1514795925445	CA	SS	2U10	0036	0034	083844	Arrival at Moorfields
1514796018739	CA	SS	2U10	0034	0030	084018	Departure at Moorfields
1514796114024	CA	SS	2U10	0030	0026	084153	

The TD system is also used by controllers to map exceptional events, i.e. tracks occupied with engineering works, temporary speed restrictions, and so on. Using TD-C class could be challenging because there are duplicated and missing messages. In very busy networks, the same headcode may be used more than once during the same day. Moreover, even if it should be rare, two trains having the same headcode might run in the same area.

Another very important task, to controllers, planners and end-users, is to forecast the train running time, from one point of the network to another. Controllers and planners can exploit a precise forecast to better plan and control the railway traffic, during normal operating conditions and during disruption management. Commuters use the forecast every time that they look at any Customer Information Screen (screens that display the arrival and departure times at station/platform).

6.2 Methodology

In this section we show how to infer the berth graph that represents the feasible train routing strategies from the TD messages flow. The output is a directed graph, where each node represents a berth and each arc represents a berth-step. The arcs represent

TABLE 6.4: A TD-C message in a transition between the XL and the SS areas (Rail, a)

Timestamp	Type	TD area	Head-code	Berth from	Berth to	Time	Comment
151...182	CC	XL	2S01		A087	055911	
151...853	CA	XL	2S01	A087	0091	060538	
151...854	CA	XL	2S01	STIN	L091	060538	Departure Hunts Cross
151...146	CA	XL	2S01	0091	0099	060659	Arrival at Liverpool South Parkway
151...148	CA	XL	2S01	STIN	L099	060659	
151...149	CA	XL	2S01	L091	COUT	060659	
151...821	CA	XL	2S01	0099	0101	061000	Departure Liverpool South Parkway
151...829	CA	XL	2S01	L099	COUT	061001	
151...436	CA	XL	2S01	0101	0103	061230	Arrival Aigburth
151...437	CC	SS	2S01		H103	061230	
151...443	CC	SS	2S01		HC2A	061230	
151...950	CA	SS	2S01	H103	H105	061440	
151...950	CA	XL	2S01	0103	0105	061440	Arrival at St Michaels
151...554	CA	SS	2S01	H105	H107	061716	
151...554	CA	XL	2S01	0105	0107	061716	Arrival at Brunswick
151...556	CB	SS	2S01	HC2A		061716	
151...142	CA	SS	2S01	H107	0021	061944	
151...142	CA	XL	2S01	0107	M021	061944	
151...143	CC	SS	2S01		M021	061945	
151...320	CA	SS	2S01	0021	0023	062027	
151...323	CB	SS	2S01	M021		062027	
151...435	CA	XL	2S01	M021	COUT	062041	
151...788	CA	SS	2S01	0023	0025	062154	
151...889	CA	SS	2S01	0025	0027	062220	Arrival at Liverpool Central

the feasible routing strategies, i.e. where a train can move from one berth. A connected path between two berths represents a connected section of the network.

The first step is to parse the XML data feed, and convert it into a more convenient format. Specifically, we convert the messages into a matrix, in which the columns are the message fields and the rows are the single messages.

We preprocess the matrix and remove all messages that have an empty headcode, or are duplicated (all fields equal with a timestamp difference of less than one second). We also remove all the messages with their headcode having non-alphanumeric characters

(such as `*`, `+`, `-`, `.`, `_`). These are messages with an obfuscated or malformed headcode. The graph of a specific region can be extracted by filtering the `area_id`.

After the preprocessing phase, we can split the stream into journeys. The task is done with the following operations:

1. Sort *stream* by `area_id`, `headcode` and `timestamp`
2. For each day
   (a) Group the daily stream by `area_id` and `headcode`
   (b) Sort each group by `timestamp` ascending
   (c) For each group, split the sequence when a gap greater than α minutes is detected between two successive timestamps. We choose $\alpha = 180$ minutes, but a smaller value may be used.
3. For each couple of sequences `a`,`b`
   (a) If they have the same `headcode` and the distance between the last `timestamp` of `a` and the first `timestamp` of `b` is less than a $\beta = 35$ minutes, merge them.

Algorithm (7-8) uses some external functions. Let s be a string, the function *transitionsChars*(s), returns *true* if s matches the regular expression `^(H|M)[0-9]+`, *false* otherwise.
onlyCharacters(s) returns *true* if s matches with the regular expression `^[a-zA-Z][a-zA-Z][a-zA-Z][a-zA-Z]`, *false* otherwise. isFirst(s, c) returns *true* if the string s starts with the character c, *false* otherwise.

Algorithm (7-8) parses the sequence of messages for each journey. It checks that for each message i, the value in $from_i$ is equal to the value in to_{i-1}. This constraint is useful to maintain coherence in the chain of messages, avoiding false introduction of a new edge just because of a missing message. If the algorithm finds a broken sequence, where the *to* of the previous message does not match with the $from$ of the current message, it registers the exception and increments a counter.

Note that there may be multiple causes for the existence of an exception, i.e. a lost message, a broken sensor, low-adhesion event, and a configuration setting. To handle each exception e, we have adopted a simple method: if the counter of e is greater than a threshold, then insert e into the graph. Otherwise, e is discarded.

Once the graph is built, it may be important to analyse the causes of the exceptions. To do so, we need to maintain in the exception list also the `timestamp` of the event

Algorithm 7 Generate graph of berths(1)

Require: *journeys* list of journeys
Require: *maxLimit*
1: set: nodes ← {∅}, links ← {∅}
2: **for** *journey* ∈ *journeys* **do**
3: set: pl ← {}, switch ← False , last ← none, lastrow ← none, exceptions ← {}
4: **for** $m \in journey$ **do**
5: skip ← false
6: **if** m_{type} = CA **then**
7: **if** (onlyCharacters(m_{from}) OR onlyCharacters(m_{to})) OR (transitionsChars(m_{to}) AND transitionsChars(m_{from})) OR isFirst(m_{from},'H') **then**
8: skip ← true
9: **end if**
10: **if** skip = false AND switch=true **then**
11: **if** $\{m_{areaid}, m_{from}\} \notin nodes$ **then**
12: $nodes_{\{m_{areaid},m_{from}\}} \leftarrow 1$
13: **end if**
14: **if** $pl_{to} = m_{from}$ **then**
15: **if** $\{\{pl_{area}, pl_{from}\}, \{m_{areaid}, pl_{to}\}\} \notin links$ **then**
16: $link_{\{\{pl_{area},pl_{from}\},\{m_{areaid},pl_{to}\}\}} \leftarrow 1$
17: **else**
18: $link_{\{\{pl_{area},pl_{from}\},\{m_{areaid},pl_{to}\}\}} \leftarrow link_{\{\{pl_{area},pl_{from}\},\{m_{areaid},pl_{to}\}\}} + 1$
19: **end if**
20: **end if**
21: set: switch ← false, pl ← {}, last ← m_{from}
22: **end if**
23: **if** skip = false AND transitionsChars(m_{to}) AND NOT transitionsChars(m_{from}) **then**
24: set: switch ← true, pl ← {area:m_{areaid},from:m_{from},to:m_{to}}
25: **if** $(m_{areaid}, m_{from}) \notin nodes$ **then**
26: $nodes_{(m_{areaid},m_{from})} \leftarrow 1$
27: **end if**
28: **end if**

(change line 40 of Algorithm 8). Once this has been done, the cause of exception can be identified by checking the following cases: 1) for each link `d` inferred by an exception, let `a,b,c` be a chain of berths. If there exists a link `d` from `a` to `c`, then it is likely that the sensor in `b` is broken, or there may be a maintenance event in the railway that involves `b`; 2) if there is no maintenance in `b`, then `b` is broken; 3) if the `d` events that involve `b` have a burst pattern in the data stream, then we may be in the presence of a low-adhesion event.

Algorithm 8 Generate graph of berths(2)

if skip=false AND m_{from} AND m_{to} **then**
  if $(m_{areaid}, m_{from}) \notin nodes$ **then**
    $nodes_{(m_{areaid},m_{from})} \leftarrow 1$
  end if
  if switch=false AND $(m_{areaid}, m_{to}) \notin nodes$ **then**
    $nodes_{(m_{areaid},m_{to})} \leftarrow 1$
  end if
  if last AND m_{from}=last **then**
    calculate the timestep between last_row and row
    $nodes_{(m_{areaid},m_{from})} \leftarrow nodes_{(m_{areaid},m_{from})} + 1$
  else
    push into exceptions $(last\_row_{areaid}, last, m_{areaid}, m_{from})$, if already exists increment counter
  end if
  set: last $\leftarrow m_{to}$, last_row $\leftarrow$ row
  if switch=false **then**
    if $((m_{areaid}, m_{from}), (m_{areaid}, m_{to})) \notin links$ **then**
      $link_{((m_{areaid},m_{from}),(m_{areaid},m_{to}))} \leftarrow 1$
    else
      $link_{((m_{areaid},m_{from}),(m_{areaid},m_{to}))} \leftarrow link_{((m_{areaid},m_{from}),(m_{areaid},m_{to}))} + 1$
    end if
  end if
 end if
 end if
 end for
end for
return $Graph(nodes, links)$

6.3 Results and discussion

The dataset used to test Algorithm (7-8) is the TD-C-CLASS stream in the UK from January, 1^{st} 2018 to April, 30^{th} 2018. We used the algorithm to build the berth graph for the network of Merseyrail. The number of journeys available is 74212.

Since this is a data-driven method, we need enough data to infer all the berths and connections that represent feasible train routing strategies. We can link only berths that have been used in the dataset. If a transition between two berths is rare (i.e. is used only during major disruptions or in case of network maintenance) then the link, despite its existence in the berth diagram, will not be registered.

Fig 6.1 shows the result for the network managed by Merseyrail without managing the exceptions. The graph produced is disconnected, with nine subgraphs. Each subgraph is a coherent sequence of transitions, and this coherency automatically ensures that the graph is validated.

FIGURE 6.1: The Merseyrail network's graph of berths, without managing the exceptions. The graph is disconnected, there are 9 subgraphs.

The next step is to analyse the exceptions. Table 6.5 shows the first 10 exceptions sorted by their counter in descending order. We assume that if an exception really happens, i.e. its frequency of occurrence is less than a small threshold, it can be ignored, otherwise the exception must be handled by adding an edge from the previous *to* to the current *from*.

Since the threshold depends on the number of journeys available, we have tried different thresholds. Fig 6.2 shows the histogram of sub-graph numbers when varing the threshold. Fig 6.3 shows the number of edges in the graph when varying the threshold. The analysis reveals that only with a threshold of 0 (Fig 6.4), and 1 (Fig 6.5), can we create a connected graph for the whole Merseyrail network. The main difference between them is the number of edges, in the first case we have 1,118 edges, while in the second one

TABLE 6.5: The first 10 exceptions sorted by their counter in descending order.

area_id_from	from	area_id_to	to	counter
XZ	E299	XZ	E045	882
XZ	E299	XZ	E053	255
XZ	0AP2	SS	0740	148
SS	0737	XZ	E053	144
XZ	0AP1	SS	0740	143
SS	0715	XZ	E295	111
XZ	E297	SS	0715	109
SS	0741	XZ	E053	101
SS	0741	XZ	0006	92
XZ	0006	SS	0737	91

882. Increasing the threshold produces graphs with fewer edges, and more disconnected. Fig 6.6a shows the graph computed with a threshold of 20, with 7 subgraphs. Fig 6.6b has been made using a threshold of 5, and number of sub-graphs is six.

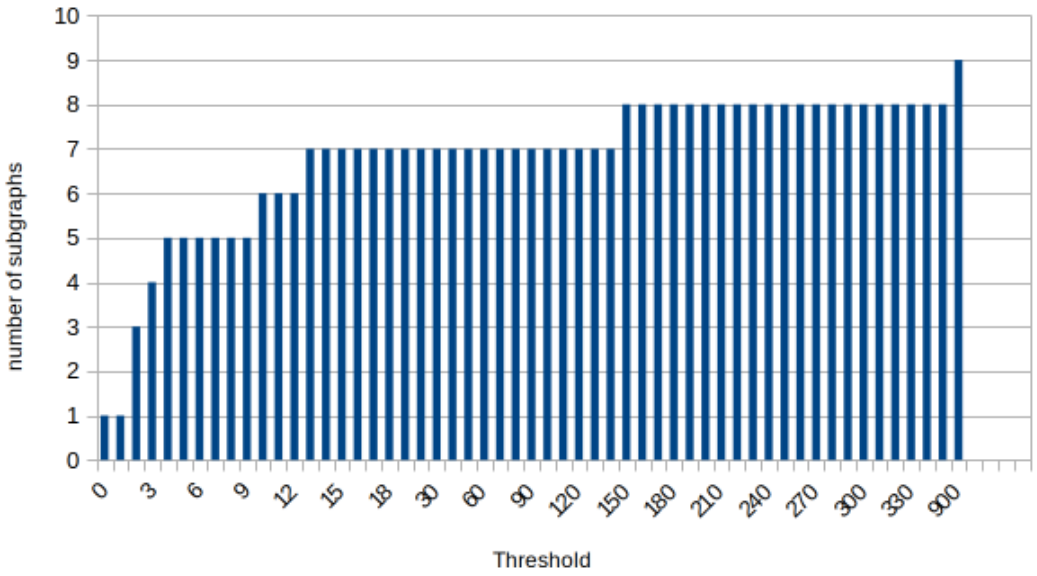


FIGURE 6.2: Histogram of threshold compared to the number of subgraphs. Note that, the threshold axis does not follow a linear scale: from 0 to 20, the step is 1; from 20 to 350, the step is 10; over 350 there is only one value, 900, equivalent to the infinite threshold

6.4 Summary

In this chapter we presented a data-driven method to infer the *berth graph*. The method does not need to access any additional input except for access to the raw data stream.

The algorithm's output is a directed graph, where a node is a berth and an arc is a berth step. The graph represents the feasible train routing strategies on the network

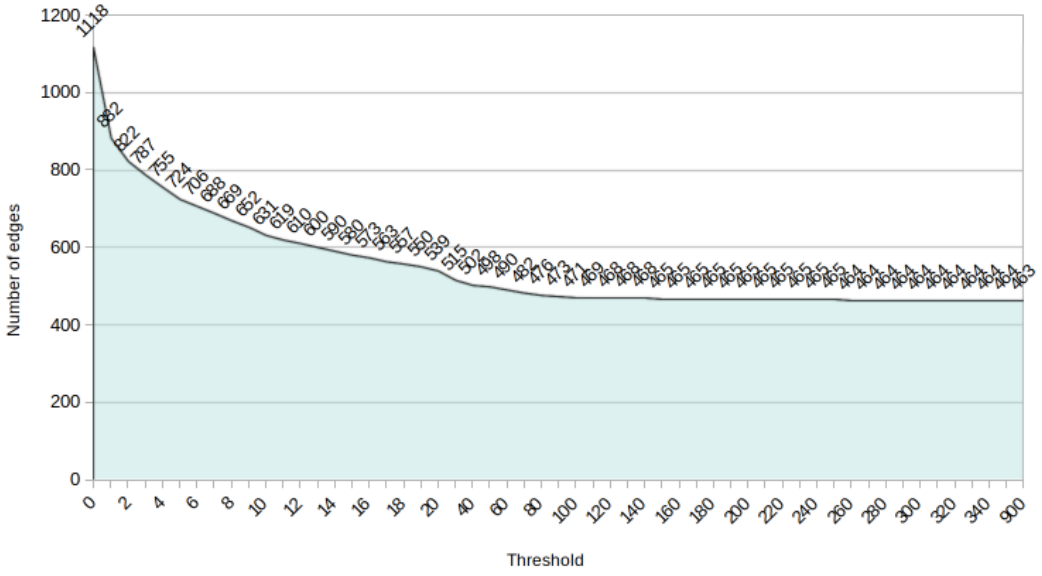


FIGURE 6.3: The number of edges compared to the threshold. Note that, the threshold axis does not follow a linear scale: from 0 to 20, the step is 1; from 20 to 350, the step is 10; over 350 there is only one value, 900, equivalent to the infinite threshold

of berths. This graph is useful for the following reasons. Firstly, this will be the first algorithm to generate berth graphs from TD data automatically. Secondly, the algorithm automatically adapts the graph in case of major network maintenance, i.e. if a new line is built, a junction is introduced, and so on. Thirdly, the validated graph can be exploited to detect some network problems in real-time. Fourthly, the graph has been exploited to develop a reactionary delay simulator and an optimiser of mitigation strategies.

The algorithm has been tested with the TD C-class stream in the UK from January, 1^{st} 2018 to the April, 30^{th} 2018. We built the network of a UK Train company (Merseyrail), testing the algorithm utilizing different parameterizations.

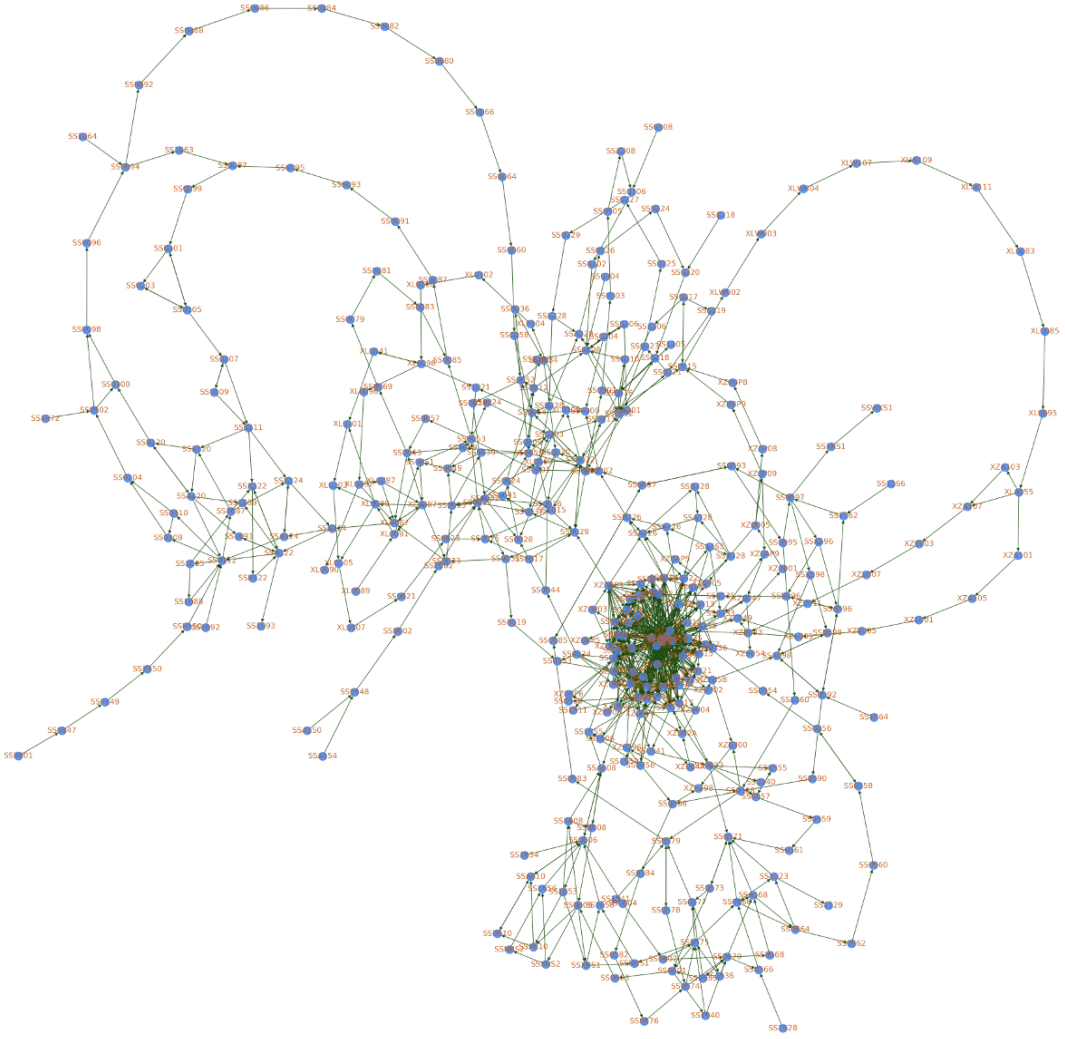


FIGURE 6.4: The Merseyrail network's graph of berths, after managing the exceptions with a threshold strictly greater than 0.

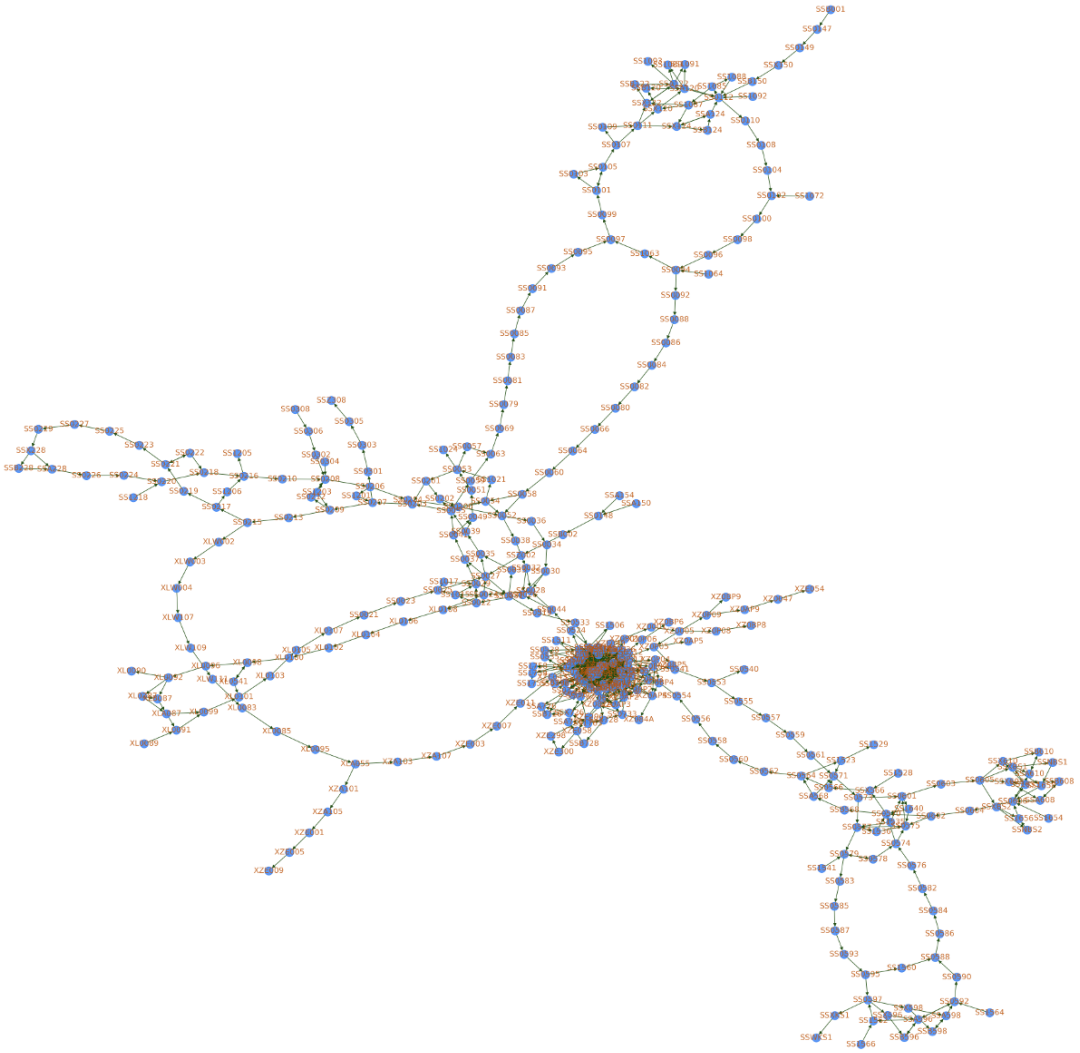


FIGURE 6.5: The Merseyrail network's graph of berths: threshold strictly greater than one. The graph is connected.

(A) Threshold strictly greater than 20 exceptions

(B) Threshold strictly greater than 5 exceptions

FIGURE 6.6: The Merseyrail network's graph of berths, after managing the exceptions with different thresholds.

Chapter 7

A comparison between the ARMA, GRU and LSTM: forecast the time that a train spends to run a track section.

In this chapter, we present a novel method to estimate the amount of time that a train is going to spend on a section of the track. We exploit a real-world case study, one berth that represents a busy junction in the Merseyside region. We describe the dataset in detail, and we statistically analyse each method of choice in detail. We compared two different approaches, the AutoRegressive Moving Average (ARMA) and two Recurrent Neural Netwoks (RNN) models: the Gated Recurrent Unit and the Long short-term memory.

Experimental results on historical data showed that the RNN models outperformed the ARMA model. We analyze both approaches and we show that the best results are obtained by networks with input sizes that were covering the statistically significant spikes of the AutoCorrelation Function.

The novelties are the following: we make a detailed statistical analysis of the berth data exploiting a case study from the Merseyside region; we compare two different approaches, the AutoRegressive Moving Average model with two Recurrent Neural Networks models; we analyze the two approaches and we found that adapting the Box-Jerkins method to defining a range of the sizes of the input of the Recurrent Neural Network may be beneficial to save processing power.

Our method is the building block to estimate expected delays. The TD's berth information delivers the highest accuracy possibly achievable, without changing the TDS technology stack.

The chapter is organised as follows. Section 7.1 introduces the case study utilised. Section 7.2 describes the methodology and Section 7.3 shows the results. The chapter will end with the conclusions in Section 7.4.

7.1 Case study

We have restricted the case study on one berth in the Merseyside region. We have choosen the berth *SS 0028*, because, being just outside Liverpool Central station, it is a very busy junction in the network. The period that we have considered for our experiments ranges from 2018-01-01 08:01:24 to 2018-04-30 22:56:59. There are 6,315 messages with the berth *SS 0028* in the *from* field or *to* field. The possible paths that include the berth *SS 0028* are shown in Table 7.1 with their cumulative frequency.

TABLE 7.1: Paths of berth *SS 0028*. There are three valid paths, the most frequent goes from berth *SS 0030* to berth *SS 0024*, passing through berth *SS 0028*.

from	throught	arrive	frequency
SS 0030	SS 0028	SS 0024	5731
SS 0032	SS 0028	SS 0024	5
SS 0044	SS 0028	SS 0024	579

Fig. 7.1 shows how the measured times in berth are distributed by frequency. The figure represents the three different paths using three different colours. The less frequent one, from *SS 0032* to *SS 0024*, is not visible because of its scarce frequency, but it is positioned between the distributions of the other two paths. We are not going to further analyse it, since it is not frequent enough to draw any conclusions. For the sake of simplicity, we will omit *SS* from now on, and we will represent the path from *XZ* to *YW* using the notation *Z-W*.

Figure 7.2 compares the time distributions during the week. The median values remain stable in both paths, the first and third quartile vary consistently during the days.

We have restricted the further analysis on the path *0030-0024*. This is because this path has more values, and thus can draw more statistically meaningful conclusions. We have done an Augmented Dickey-Fuller test (ADF) to verify the hypothesis that our time series is stationary. The null hypothesis (H0) is that our data can be represented by a unit root, which means that it is not stationary and thus has some time-dependent structure. The alternate hypothesis (H1) is that the time series is stationary. Table 7.2

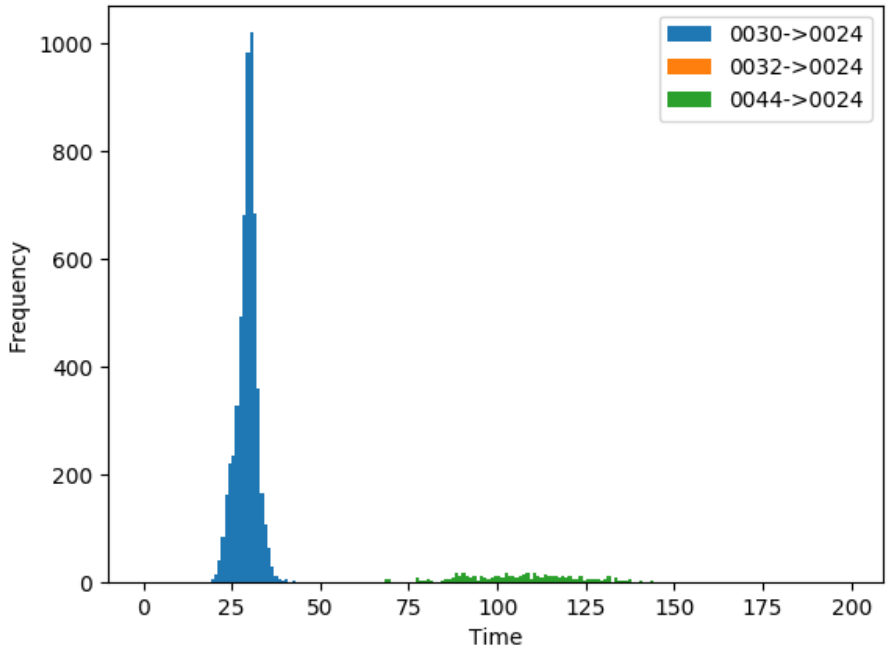
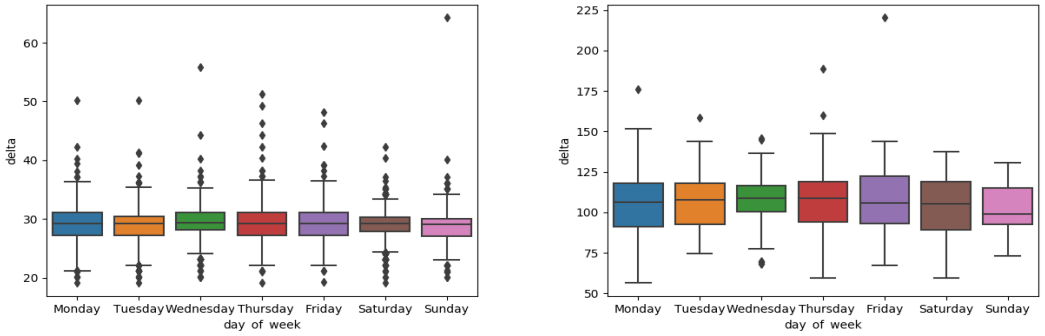


FIGURE 7.1: Berth *SS 0028*, distribution of times differentiated by path.



(A) Path from 0030 to 0024

(B) Path from berth 0044 to 0024

FIGURE 7.2: Weekly boxplot, vertical axis is the time delta, horizontal axis is the week day.

shows the results. The ADF rejects the null hypothesis with 99% confidence threshold in every grouping condition considered. We have tested the hypothesis on the data grouped by: weekly, working days, non-working days, peak time and non-peak time.

Figure 7.3 shows the distributions of the times during the day. Figure 7.4 compares the difference between peak time, and non-peak time. In the UK, peak time (Fig 7.4a) is defined as the interval from Monday to Friday, from 6:30 to 9:30, and from 16:00 to 19:00. Non-peak time (Fig 7.4b) is the other time on the day, weekend and festivities.

TABLE 7.2: Stationary tests on the data grouped by: weekly, working days, non working days, peak time and non-peak time.

#	ADF	cv-1%	cv-5%	cv-10%	mean	std	p-value	description	length
0	-21.833	-3.431	-2.862	-2.567	29.018	3.067	0.000	weekly	5731
1	-19.378	-3.432	-2.862	-2.567	29.076	3.093	0.000	working days	4405
2	-10.711	-3.435	-2.864	-2.568	28.824	2.970	3.327206e-19	non working days	1326
3	-17.932	-3.434	-2.863	-2.568	29.164	3.039	2.881641e-30	peak time	1865
4	-17.149	-3.433	-2.863	-2.567	28.849	3.501	6.966646e-30	off-peak time	2395

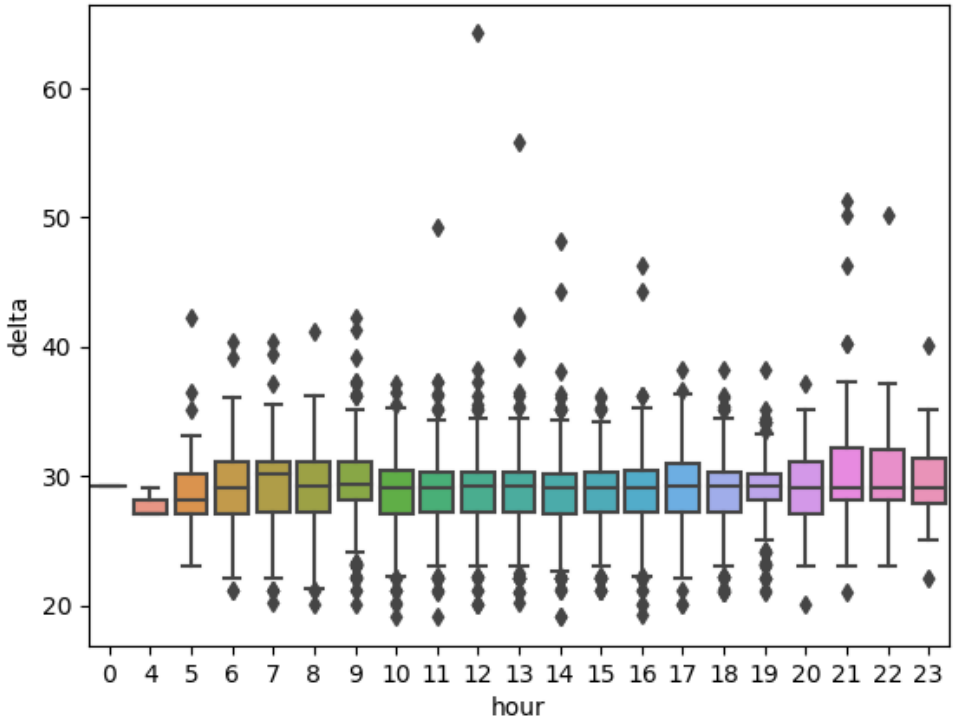
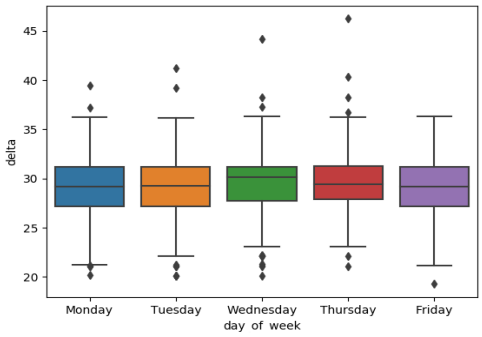
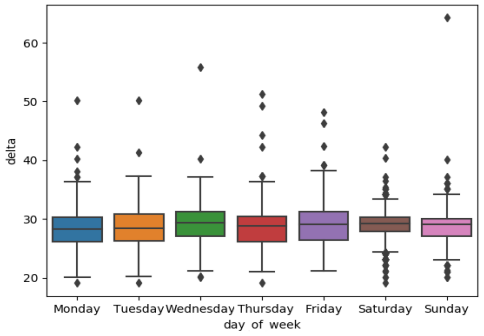


FIGURE 7.3: Hourly boxplot. Vertical axis is the time spent in berth, horizontal axis is the hour of the day.

To present more detailed statistics, the data has been grouped in hourly (Table 7.3), non-peak time (Table 7.4), peak time (Table 7.5) and weekly (Table 7.6). We report mean value, standard deviation, median, skew, minimum value, maximum value, first quartile and third quartile. In the time between 1:00am and 4:00am, there is no running train in the considered path.



(A) Path *0030-0024*: peak time (from Monday to Friday, hours from 6:30 to 9:30 in the morning and 16:00 to 19:00 in the evening)



(B) Path *0030-0024*: non-peak time

FIGURE 7.4: Comparison between the peak time data, and the non-peak time data. In the boxplots: the vertical axis is the time spent in the berth, and the horizontal axis is the day of the week.

7.2 Methodology

The task is to forecast the future value of a time series. Figure 7.5 shows the full dataset, without removing outliers and without preprocessing. The time series has no visible trend, as confirmed by the ADF test, and is stationary.

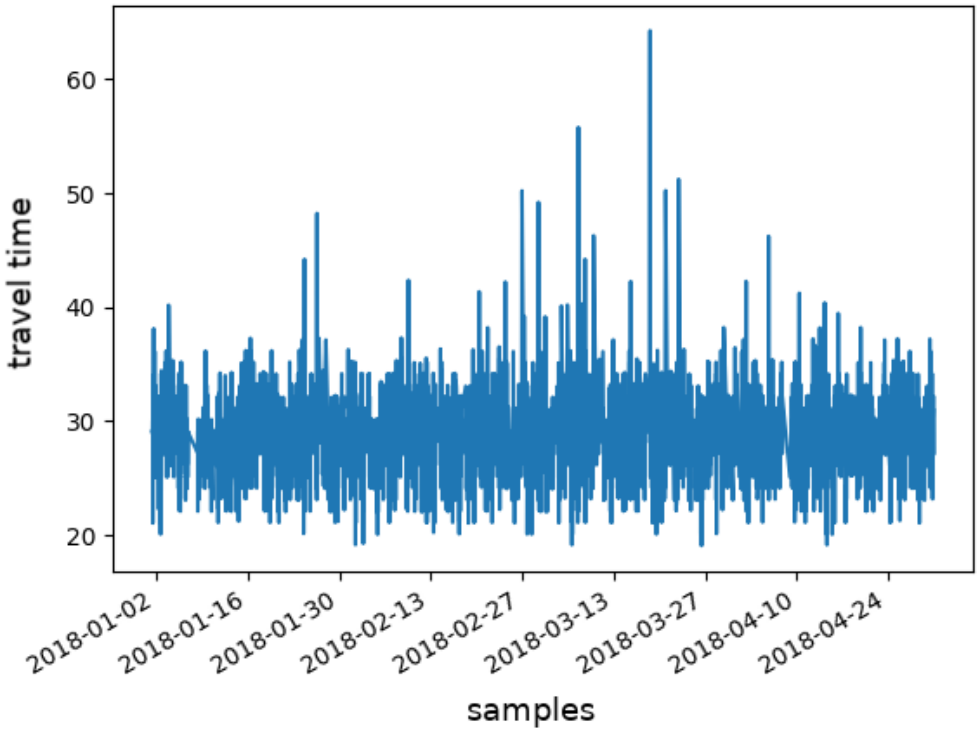


FIGURE 7.5: The full time series for the path *0030-0024*

Table 7.3: Path from 0030 to 0024: hourly statistics. Data from 2018-01-01 08:01:24 to 2018-04-30 22:56:59.

hour	mean	std	median	skew	min	max	first quartile	third quartile
0	29.160	N/A	29.160	N/A	29.160	29.160	29.160	29.160
4	27.766	1.164	27.095	1.732	27.092	29.110	27.092	27.092
5	28.670	3.008	28.117	1.668	23.078	42.275	23.251	23.596
6	28.973	2.833	29.126	0.235	21.115	40.395	21.120	21.818
7	29.430	2.909	30.138	-0.343	20.162	40.343	20.958	21.119
8	29.191	2.945	29.207	-0.273	20.121	41.232	21.116	21.391
9	29.452	3.000	29.417	0.130	20.075	42.220	21.120	22.090
10	28.796	2.772	29.154	-0.474	19.090	37.134	20.096	21.099
11	28.819	3.017	29.150	0.401	19.174	49.205	21.067	21.285
12	28.913	3.236	29.174	2.760	20.096	64.258	20.101	21.174
13	29.043	3.333	29.177	1.220	20.156	55.786	21.082	21.150
14	28.651	3.220	29.145	0.405	19.161	48.233	19.270	21.127
15	28.787	2.787	29.148	-0.534	21.103	36.195	21.112	21.328
16	28.902	3.135	29.143	0.263	19.278	46.232	20.096	21.130
17	28.967	2.920	29.183	-0.333	20.102	38.207	20.207	22.112
18	28.891	2.873	29.160	-0.357	21.065	38.220	21.072	21.103
19	29.039	2.644	29.165	-0.389	21.076	38.220	21.155	21.917
20	29.182	3.303	29.107	-0.157	20.072	37.170	20.454	21.218
21	30.807	5.046	29.131	1.919	21.078	51.237	21.482	22.289
22	29.996	4.499	29.093	2.046	23.076	50.216	23.088	23.113
23	29.716	3.784	29.078	0.775	22.067	40.124	22.240	22.587

Table 7.4: Path from 0030 to 0024: non peak time statistics. Data from 2018-01-01 08:01:24 to 2018-04-30 22:56:59.

day of week	mean	std	median	skew	min	max	first quartile	third quartile
Friday	29.153	4.103	29.145	0.983	21.105	48.233	21.615	22.074
Monday	28.449	3.848	28.333	0.916	19.090	50.216	19.690	20.137
Saturday	29.048	2.710	29.159	-0.287	19.161	42.220	20.271	22.097
Sunday	28.403	3.370	29.085	2.473	20.072	64.258	20.226	21.077
Thursday	28.757	4.396	28.770	1.282	19.174	51.237	20.219	21.101
Tuesday	28.757	4.039	28.415	0.875	19.164	50.231	19.708	20.675
Wednesday	29.371	3.901	29.336	1.413	20.088	55.786	20.121	20.614

Fig 7.6 splits the distibution of times by *train class*. The *train class* is the first letter of the *headcode*, and represents the train typology. It is evident that considering the time series for each train class separately is beneficial. This is because the underlying distribution may be different due to different train classes may imply different speed limits (i.e. freight, special, and rail head treatment trains may run more slowly than

TABLE 7.5: Path from 0030 to 0024: peak time statistics. Data from 2018-01-01 08:01:24 to 2018-04-30 22:56:59.

day of week	mean	std	median	skew	min	max	first quartile	third quartile
Friday	28.855	2.979	29.172	-0.355	19.278	36.318	20.934	22.114
Monday	29.061	2.996	29.197	-0.235	20.162	39.458	21.052	21.117
Thursday	29.552	3.043	29.402	0.383	21.110	46.232	22.062	23.117
Tuesday	28.991	3.146	29.231	-0.281	20.096	41.232	20.100	20.961
Wednesday	29.351	2.990	30.146	-0.207	20.144	44.215	21.030	21.253

TABLE 7.6: Path from 0030 to 0024: weekly statistics. Data from 2018-01-01 08:01:24 to 2018-04-30 22:56:59.

day of week	mean	std	median	skew	min	max	first quartile	third quartile
Friday	29.024	3.088	29.168	0.366	19.278	48.233	21.230	22.109
Monday	28.947	3.060	29.168	0.153	19.090	50.216	20.150	21.103
Saturday	29.048	2.710	29.159	-0.287	19.161	42.220	20.271	22.097
Sunday	28.403	3.370	29.085	2.473	20.072	64.258	20.226	21.077
Thursday	29.181	3.215	29.177	0.828	19.174	51.237	21.108	22.111
Tuesday	28.906	3.117	29.166	0.165	19.164	50.231	20.101	21.106
Wednesday	29.330	2.966	29.341	0.558	20.088	55.786	20.314	21.686

other passenger trains).

To deal with this type of time series, a suitable forecasting method should be able to adapt automatically to the service disruptions, i.e. temporary change of speed. The below analysis will be done to identify such a method.

We have compared two different approaches. The first one is based on the consolidated theory of the time series analysis: since the time series is stationary with a constant variance (there is no need for the difference operator), one could choose a stationary method such as an AutoRegressive Moving Average (ARMA) model. The second approach exploits two Recurrent Neural Networks (RNN) architectures, that have been very successful in sequence learning: the Long-Short Term Memory (LSTM)(Hochreiter and Schmidhuber, 1997) and the Gated Recurrent Units (GRU)(Cho et al., 2014).

The ARMA(s,q) model is the composition of s autoregressive terms θ_j, and q moving average terms γ_k. (7.1) shows the model: m is a constant, ϵ_t represents the white noise (independent identically distributed random variable sampled from a normal distribution with zero mean), θ_j are the parameters of the autoregressive model, γ_k are the parameters

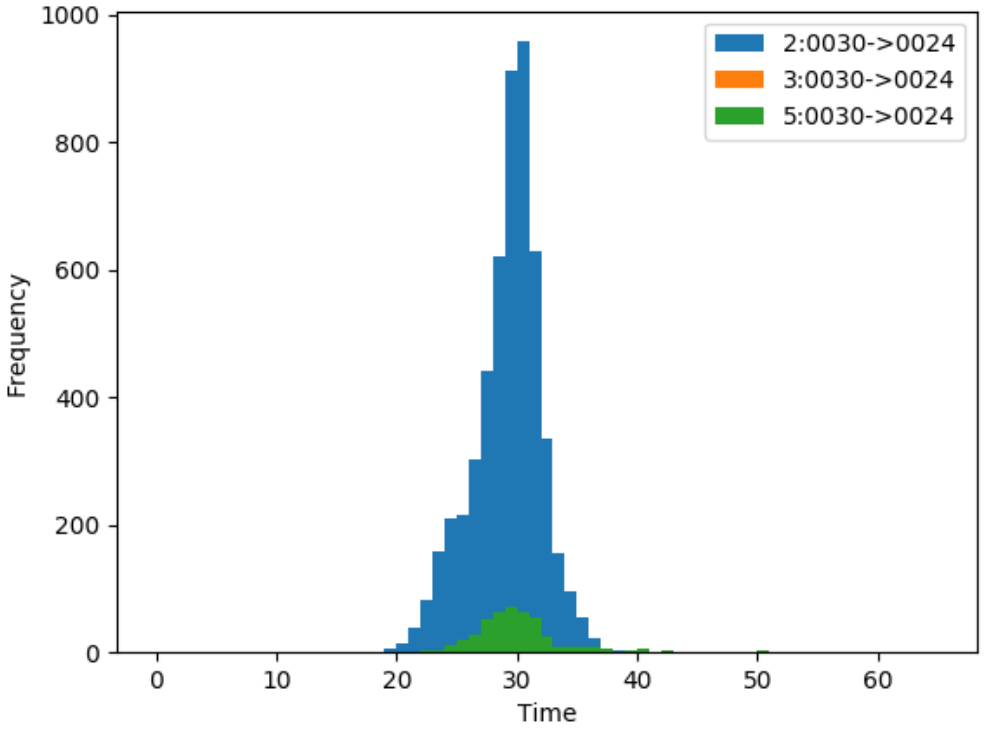


FIGURE 7.6: Berth path *0030-0024*, distribution of times differentiated by train class.

of the moving average models. X_t is the value of the time series to forecast, whilst X_{t-j} is the previous j value.

$$X_t = m + \epsilon_t + \sum_{j=1}^{s} \theta_j X_{t-j} + \sum_{k=1}^{q} \gamma_k \epsilon_{t-k} \tag{7.1}$$

Let d be the number of input features, p be the number of hidden units, and t the time step. Let $\circ$ define the element-wise product (Hadamard product). The LSTM model is described in (7.2), where: $x_t \in \mathbb{R}^d$ is the input vector, $f_t \in \mathbb{R}^p$ is the activation vector of the forget gate, $i_t \in \mathbb{R}^p$ is the activation vector of the input gate, $o_t \in \mathbb{R}^p$ is the activation vector of the output gate, $h_t \in \mathbb{R}^p$ is the hidden state vector (output vector), $c_t \in \mathbb{R}^p$ is the cell state vector, $W^\pi \in \mathbb{R}^{p \times d}$, $U^\pi \in \mathbb{R}^{p \times p}$ are the weight matrices, and $b^\pi \in \mathbb{R}^p$ is the bias vector. The apices $\pi \in \{f, i, o, c\}$ indicate the belonging to the equation f, i, c and o. σ_g is the hyperbolic tangents, while σ_h and σ_c are hard sigmoids.

$$
\begin{aligned}
f_t &= \sigma_g(W^f x_t + U^f h_{t-1} + b^f) \\
i_t &= \sigma_g(W^i x_t + U^i h_{t-1} + b^i) \\
o_t &= \sigma_g(W^o x_t + U^o h_{t-1} + b^o) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(W^c x_t + U^c h_{t-1} + b^c) \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}
\tag{7.2}
$$

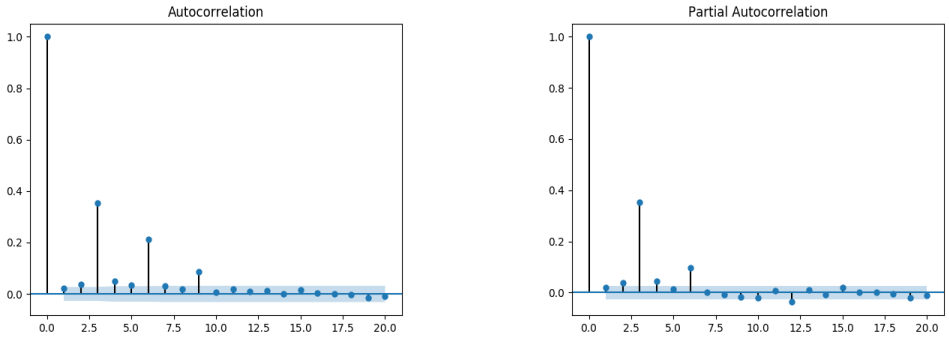
The GRU model is described in (7.3). Reusing the notation of the LSTM: $x_t \in \mathbb{R}^d$ is the input vector, $h_t \in \mathbb{R}^p$ is the output vector, $z_t \in \mathbb{R}^p$ is the update gate vector, $r_t \in \mathbb{R}^p$ is the reset gate vector. σ_g is the hyperbolic tangent function, and σ_h is the hard sigmoid function.

$$
\begin{aligned}
z_t &= \sigma_g(W^z x_t + U^z h_{t-1} + b^z) \\
r_t &= \sigma_g(W^r x_t + U^r h_{t-1} + b^r) \\
h_t &= (1 - z_t) \circ h_{t-1} + z_t \circ \sigma_h(W^h x_t + U^h(r_t \circ h_{t-1}) + b^h)
\end{aligned}
\tag{7.3}
$$

The output vector, h_t, in both the architectures LSTM and GRU, is connected to a densely connected layer, with linear activation and output of unitary size.

7.3 Results and discussion

We developed the ARMA model using the Box-Jerkins method (Box and Jenkins, 1976). Since this time series is verified as stationary, there is no need to adopt a differencing operator. We need to derive the order s of the autoregressive process and the order q of the moving average process. Fig 7.7 shows the autocorrelation function (Fig 7.7a, ACF) and the partial autocorrelation function (Fig 7.7b, PACF). The PACF plot gives insights about the order of the autoregressive side. According to this plot, the first significant spike is on the third lag, thus we set $s = 3$. To decide the order of the moving average side, we look at the ACF plot. In this plot, the first significant spike is at the third lag, thus we set $q = 3$.



(A) autocorrelation function (B) partial autocorrelation function

FIGURE 7.7: PACF and ACF.

The case study dataset is split in training and testing (80% - 20%). The training part is used to train the models, and the testing part is used to test the models with unknown data. The regression error is measured using the mean squared error (MSE).

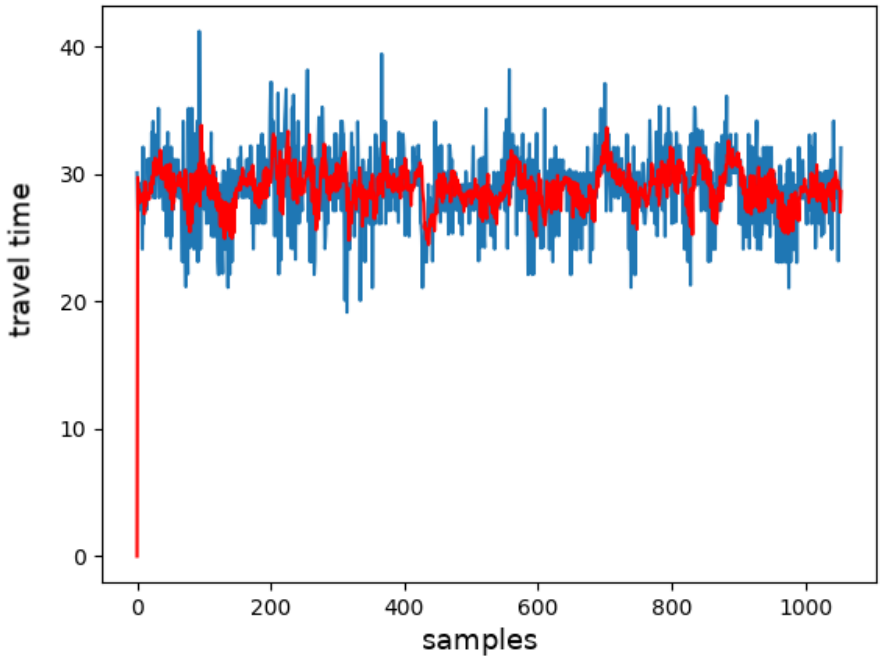
The ARMA(3, 3) model is fitted using the maximum likelihood method (Jones, 1980). The regression error in the test dataset has been $MSE = 8.969508$. Fig 7.8 summarizes the results for the ARMA(3, 3) model: Fig 7.8a plots the test dataset with the overlapped prediction, Fig 7.8b and Fig 7.8c plot, respectively, the PACF and ACF of the residuals.

The framework to exploit the RNN models has the following structure: 1) preprocess the dataset using a min-max scaler, with the output ranges between 0 and 1; 2) feed data to the RNN architecture (training-testing); 3) postprocess the results: inverse scaling to the original output range.

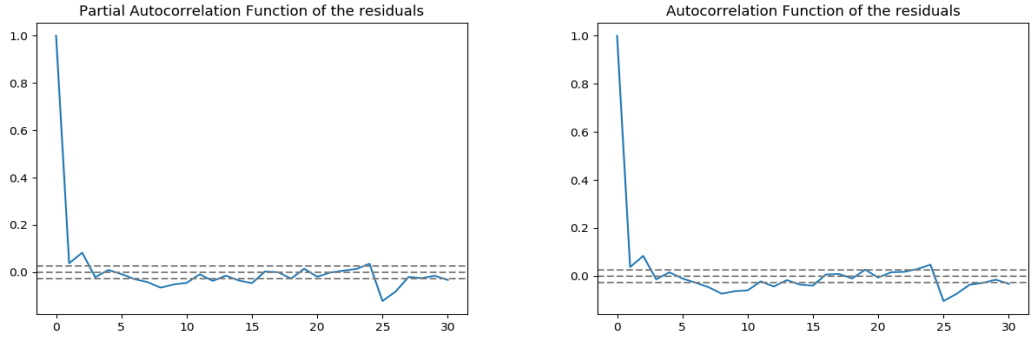
The LSTM and GRU models have been trained using the Adam optimiser (Kingma and Ba, 2014) with default parameterisation (learning rate = 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, learning rate decay=0.0). The loss function is the mean squared error.

We have explored the effects of different combinations of parameters without further hyper-tuning. The number of neurons has been set with the following values: 1, 2, 4, 8, 16, 32, 64, 128, 180, 260. The batch size: 8, 16, 64, 32. The training epochs: 30, 60, 120. The input dimension (as the number of the previous values to consider for the forecast): 1, 3, 5, 7, 8, 9, 11, 16, 32. Table 7.7 presents the results for the first ten best architectures.

There is a consistent gap between the ARMA model and the RNN architectures. RNN architectures perform better than the ARMA model, even without tuning them further or using more elaborate techniques.



(A) Plot of the test dataset with the overlapped prediction.



(B) PACF of the residuals from the test dataset. (C) ACF of the residuals from the test dataset.

FIGURE 7.8: ARMA(3,3)

Fig 7.9 shows the best performing RNN, a GRU architecture. In this figure we report the plot of the dataset overlapped with the prediction, distinguishing the training from the test in different colours (Fig 7.9a); the PACF (7.9b) and ACF (7.9c) of the residuals.

It is interesting to highlight that the best performing RNNs have an input size that is around the third spike of the ACF of the dataset (Fig: 7.7a). Moreover, all the input sizes greater than 11 achieve worse results, in a few combinations even worse than the ARMA model. The reason is that greater input sizes require greater effort in terms of the computational power and memory to process the dataset. This observation suggests that using the Box-Jerkins method to define a restricted exploration range of the input size may be beneficial to save computing power.

TABLE 7.7: comparison

Architecture	MSE test	batch size	epoch	input size	neurons
GRU	7,70128342164419	64	30	9	8
LSTM	7,70244964133797	8	30	8	8
GRU	7,70538247941626	8	30	7	128
LSTM	7,70686516569197	8	30	11	260
GRU	7,70909780976722	32	30	8	8
LSTM	7,71495684949493	8	30	11	32
LSTM	7,71557511850318	8	30	9	32
LSTM	7,71604104974153	8	30	9	16
LSTM	7,71630973211366	64	30	7	8
LSTM	7,71952616013033	8	30	9	260

We notice that the RNN architectures have something in common with the ARMA model. The output h_t of the models (7.3) and (7.2) can be seen as a non-linear coefficient θ_j of the ARMA (7.1). The input size is the amount of the autoregressive behaviour of the series, which can be seen by an ACF plot. The PACF and ACF of the residuals of the best RNN and the ARMA model are similar: the plots of the RNN are a smoothed version of the plots of the ARMA model.

Using more advanced techniques, such as using the early stopping to prevent overfitting, preprocessing with other methods, adding drop-out, regularizer layers, and hypertuning the parameters of the optimiser, the RNN architectures can achieve better results. In the next chapter we will show an example.

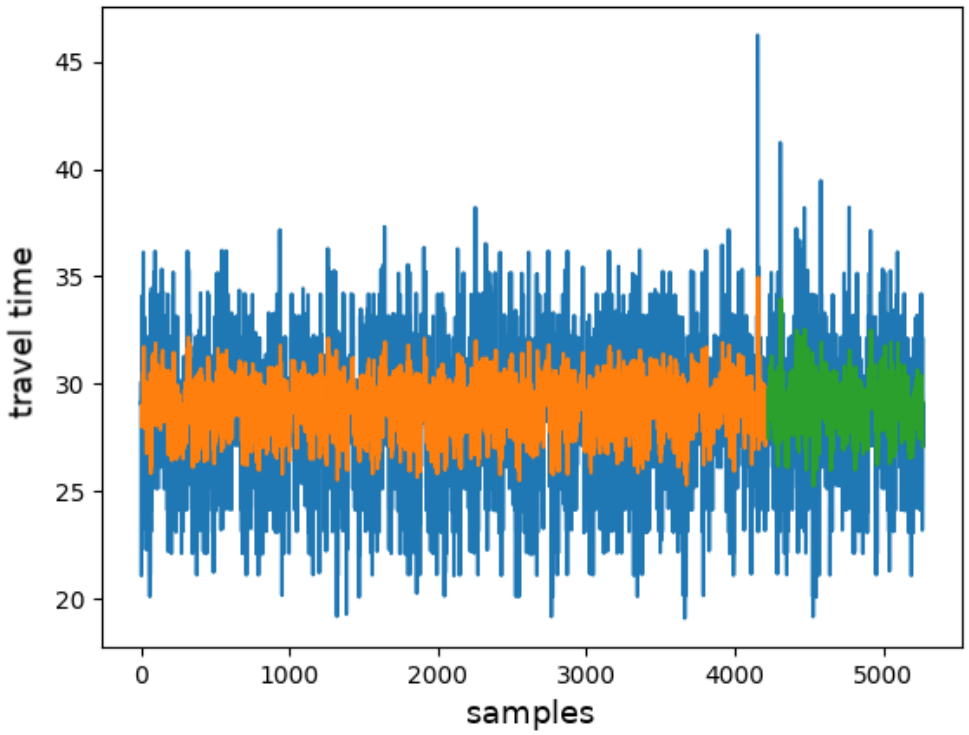
7.4 Summary

In this chapter, we have presented a novel method to estimate the amount of time that a train is going to spend in a berth.

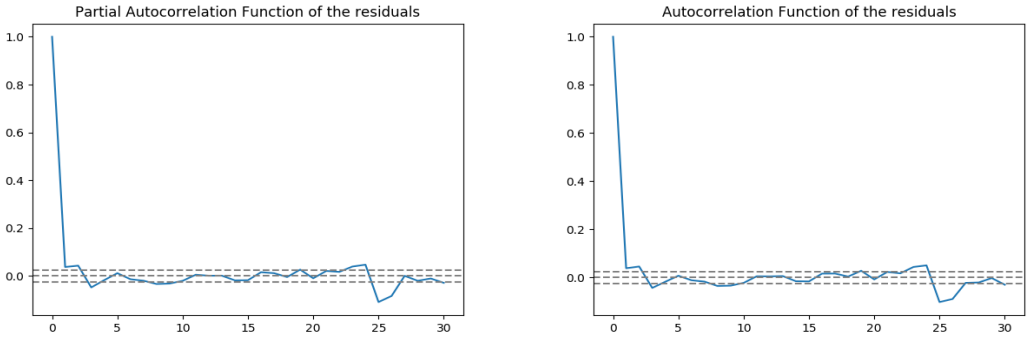
We have exploited a real-world case study, one berth that represents a busy junction in the Merseyside region. We have described the dataset in detail, and we have justified each of our choices with statistical analyses.

We compared the ARMA model with two of the Recurrent Neural Networks models, the Gated Recurrent Unit and the Long short-term memory. We tested both approaches on our case study. The results showed that RNN architectures outperform the ARMA model, even in their simpler configurations.

The comparison has been beneficial to analyse the strength and similarities of the two approaches. We have found that adapting the Box-Jerkins method to estimate the input



(A) Plot of the dataset with the overlapped prediction, divided in training and test.



(B) PACF of the residuals from the test dataset. (C) ACF of the residuals from the test dataset.

FIGURE 7.9: Best performing model, GRU, parameterised with: batch size=64, training epochs=30, input size = 9, number of neurons = 8.

size range of the RNN may be beneficial to better manage the computational resources. In detail, we have shown that the best results can be obtained by networks with input sizes that can cover the statistically significant spikes of the AutoCorrelation Function.

Our method is the building block to estimate expected delays for trains. The importance of using the TD's berth information is that TD delivers the highest accuracy possibly achievable, i.e. to the second, without changing the existing technology stack.

Chapter 8

Train journey running time forecasting using Train Describers and Gated Recurrent Units

In Chapter 7, we analysed the data of a single berth, and compared different forecasting models to predict the train's running time. In this chapter, we exploit these results. We present an adaptive method to forecast the running time of a train's journey using the Gated Recurrent Units (GRU). We tested the system on a real-world case study based in the Merseyside region.

We show a sequential training procedure that is a trade-off between accuracy and training time. Each round improves the model knowledge of the time series. The motivation for building such a procedure is that the berths have different distributions: for some berths, there may be a lack of samples or a very low variance distribution, while other berths may have richer variability.

We have compared two different forecasting systems. The first one utilises an input of the immediate previous n running times. The second one, instead, uses the previous n running times available 4 hours before the start of the journey. The experimental results show that the second method is not significantly less accurate than the first method. We also compared both methods with the standard train time prediction system currently used by the UK rail industry, called Darwin. The results showed that for most of the tested journeys, our proposed systems provide more accurate prediction than the industry system Darwin.

We have concluded that a balanced procedure that filters the communication of the new forecasting based on the exceeding of a threshold could be an effective trade-off between accuracy, delivery costs, and stability of the forecast.

This work is important because forecasting the train running time is a very important task to both controllers and end-users. Moreover, our method is the building block to estimate expected delays also. The importance of using the Train Describer's berth information is that it delivers the highest possible accuracy that can be achieved using the current UK rail infrastructure, i.e. to the second, without changing the existing Train Detection System technology stack.

The novelties are the following. Firstly, we propose a training procedure that effectively balances accuracy and training time. Secondly, we propose, analyse and compare two different forecasting concepts. The first one forecasts the running time, for each berth, based on the immediate previous n running times. The second forecasts the running time, for each berth, based on the previous n running times available 4 hours before the start of the journey. Thirdly, we evaluate the efficiency of the proposed methods against the standard train time prediction system currently used by the UK rail industry, called Darwin. We conclude that our proposed systems can provide more accurate prediction than the industry system Darwin, and that a balanced procedure that filters the communication of the new forecasting based on the exceeding of a threshold could be an effective trade-off between accuracy, delivery costs, and stability of the forecast.

The chapter is organised as follows. Section 8.1 introduces the case study utilised. Section 8.2 describes the methodology and Section 8.3 shows the experiments and results. The chapter will end with the conclusions.

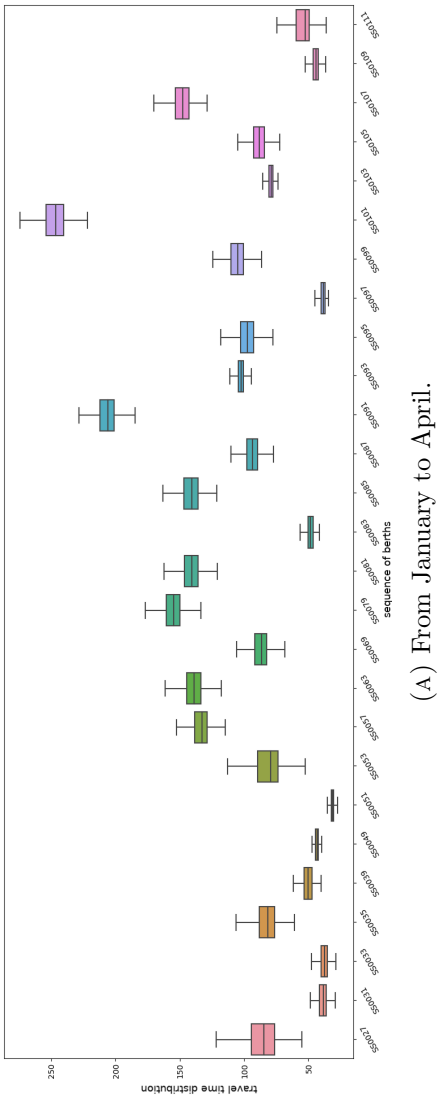
8.1 Case study

We have restricted the case study on the journeys of the Merseyrail's Northern Line, from Liverpool Central to Southport. The journeys start from the berth SS0027, located in Liverpool, to end with berth SS0111, located in the entrance of Southport station. We have chosen the Northern Line because it is a very busy part of the network. We have included every train that travels in this route, in the period from 2018-01-01 08:01:24 to 2018-04-30 22:56:59. The dataset contains 6,753 journeys, of which 5,287 in the training months (January, February, and March), and 1,466 in the testing month (April).

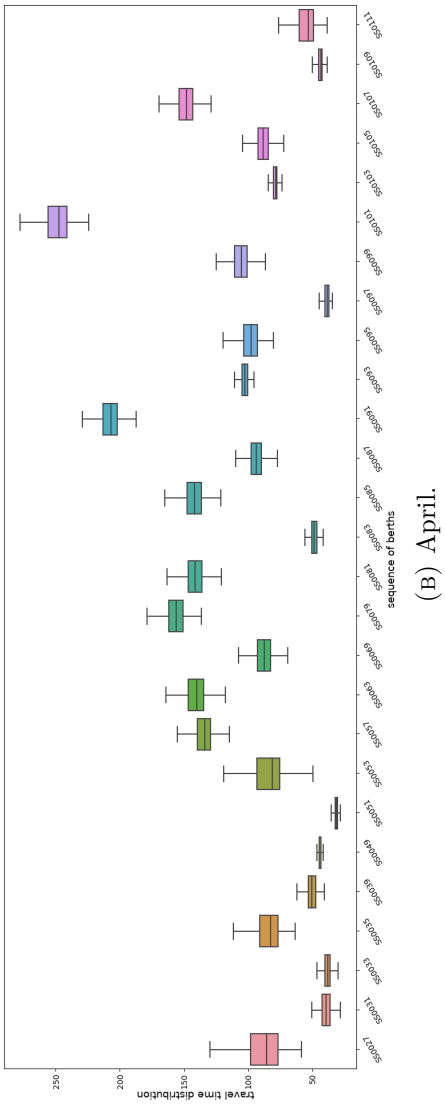
The time on each berth has been deducted by substraction between successive berth steps: i.e. using the values in Table 6.3. For example, the time extracted for the berth

SS0036 is 81.232 seconds by having the *timestamp* of the third row minus the *timestamp* of the second row.

Fig 8.1 shows in a box plot the distribution of time for each berth in the path. Fig 8.1a considers the full dataset from January to April, while Fig 8.1b includes only the month of April. There are very small differences between the two distributions, so the two periods can be considered as having the same distribution. The figure shows the journeys with train class 2, the most frequent.



(A) From January to April.



(B) April.

FIGURE 8.1: Distribution of times per berth: (8.1a) shows the full dataset, while (8.1b) only the month of April

Table 8.1 shows the complete path. For each berth we show its size and, when available, the corresponding station. The size of the berths has been measured by analysing the On Train Data Recorder (the train's black box) that covers the track, with an accuracy of around 40 metres.

The size of the berths becomes useful to compare the forecasting results of our proposed models with the official forecasting result provided by the current industry system Darwin. Darwin is the official system that feeds live and forecasting train running time information (at station level, i.e. from station to station) to most station screens, web applications, and mobile applications in the UK.

TABLE 8.1: Northern Line case study. Forecasting path from berth *SS 0027* to berth *SS 0111*.

Berth (area & code)	Berth Size (meters)	Station	Overall frequency	
SS 0027	167	Liverpool Central		
SS 0031	233	-		
SS 0033	400	-		
SS 0035	324	Moorfields		
SS 0039	527	-		
SS 0049	851	-		
SS 0051	644	-		
SS 0053	459	Sandhills		
SS 0057	1012	Bank Hall		
SS 0063	1217	Bootle Oriel Road		
SS 0069	577	Bootle New Strand		
SS 0079	1662	Seathforth & Litherland		
SS 0081	1664	Waterloo		
SS 0083	579	-	6753	1466
SS 0085	1589	Blundellsands & Crosby		
SS 0087	911	Hall Road		
SS 0091	3221	HighTown		
SS 0093	2394	-		
SS 0095	867	Formby		
SS 0097	547	-		
SS 0099	927	Freshfield		
SS 0101	4402	Ainsdale		
SS 0103	1589	-		
SS 0105	927	Hillside		
SS 0107	1505	Birkdale		
SS 0109	487	-		
SS 0111	734	-		

8.2 Methodology

Let us formalize the problem: in the following definitions we will use the term *berth* to refer to a berth path as discussed in Chapter 7. A journey is a finite sequence $a_{i\in\{0,\dots,m\}}$ of m berths, a connected path in the model discussed in Chapter 6. Let π_i be the running time, which is the time required for a train to travel across the berth a_i. Let $s_i \in \mathbb{N}$ be the length of berth a_i. Let us define b_i as the identifier of the station if berth a_i includes the station's platform, or an empty string otherwise.

The objective is to forecast the running time $\pi_i, \forall i \in \{0, \dots, m\}$. The running time with the sequence of the travelled berths leads to the forecast of the arrival time (8.1) in each berth-step (the transition point between berth a_i and berth a_{i+1}).

$$\tau_{i,i+1} = \sum_{0}^{i} \pi_i \tag{8.1}$$

Regarding the berth-step position σ it is defined as in (8.2), which represents the sum of the berth sizes.

$$\sigma_{i,i+1} = \sum_{0}^{i} s_i \tag{8.2}$$

The arrival time at the platform, ω_i, is defined as (8.3),

$$\omega_i = \tau_{i-1,i} + c_i \tag{8.3}$$

where c_i is a constant term that defines the time required for the train to get from the previous berth-step to the stopping point. It should be noted that, as to be discussed later in the experimental section, our proposed methodology forecasts train running time from berth to berth, and to the second, while Darwin, the current industry system, forecasts train running time from station to station, and to the minute. Despite this difference, it is still possible to compare the two methodologies by plotting their forecast results against the actual running time of a train, as recorded from the train's black box (OTDR). The proposed method should provide more detailed forecasts, since there are more berth steps than stations, and since it can provide results rounded to the second.

It should be noted that if berth a_i includes also station b_i within it, then the running time of the train on berth a_i will include also the dwell time, i.e the time taken for passengers to get on and off the train at that station.

The first berth, a_0, is the berth starting from the platform of the departure station, while the last berth, a_m, is the last berth just before the terminal station. This simplification does not affect the generality or accuracy of the method or the results.

In Chapter 7 we have shown that the Gated Recurrent Units (GRU)(Cho et al., 2014) and the Long Short-Term Memory (LSTM)(Hochreiter and Schmidhuber, 1997) architectures perform better than Autoregressive Moving Average models. In this chapter we use the GRU architecture. It achieved previously the best results and it is faster to be trained.

Let us formally define the GRU architecture. Let d be the number of input features, p be the number of hidden units, and t the time step. Let $\circ$ define the element-wise product (Hadamard product). The GRU model is described in (8.4). $x_t \in \mathbb{R}^d$ is the input vector, $h_t \in \mathbb{R}^p$ is the output vector, $z_t \in \mathbb{R}^p$ is the update gate vector, $r_t \in \mathbb{R}^p$ is the reset gate vector. $W \in \mathbb{R}^{p \times d}$, $U \in \mathbb{R}^{p \times p}$ are the weight matrixes, and $b \in \mathbb{R}^p$ is the bias vector. σ_g is the hyperbolic tangent function, and σ_h is the hard sigmoid function.

$$
\begin{aligned}
z_t &= \sigma_g(W_z x_t + U_z h_{t-1} + b_z) \\
r_t &= \sigma_g(W_r x_t + U_r h_{t-1} + b_r) \\
h_t &= (1 - z_t) \circ h_{t-1} + z_t \circ \sigma_h(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h)
\end{aligned}
\tag{8.4}
$$

The output vector, h_t, is connected to a densely connected layer, with linear activation and output, $\tilde{\pi}$, of unitary size.

For every berth i, we train a GRU model such that the mean squared error between the output of the GRU ($\tilde{\pi_i}$) and the running time (π_i) is minimised: $min\ MSE(\tilde{\pi_i}, \pi_i)$. The learning phase has been optimised using the Adam (Kingma and Ba, 2014) optimiser.

8.3 Experiments, results and discussion

As previously introduced, the dataset contains 6753 journeys (excluding any journey with unpredictable delays, e.g. cancellation, train broken down, etc.), covering the first four months of the year 2018. The first 5287 journeys have been exploited in the training

phase, while the last 1466 journeys have been utilized for the testing phase. The training dataset is further divided into two parts: 90% for training, and 10% for validation. The validation is required to measure how successful the training phase has been. The testing dataset measures how good the model is in dealing with unknown data.

For every berth we trained a model using the procedure in 8.3.1. Then we combined the results in two forecasting experiments, and we compared the results with the industry official forecast (the Darwin system) in 8.3.2. In the following subsections we are going to analyse these points separately.

8.3.1 Training one model

We are now considering the procedure for training one model, and the dataset is a time series of train running times for one berth.

The testing dataset has been scaled such that the features follow a normal distribution (in the scikit-learn library, a quantile transformer), using 100 quantiles. Then in the testing phase, the quantile transformer is applied locally to each input vector. That is because in a real-time forecast, the only available data are historical data and the distribution of the future data would be unknown. The distribution of the data in the Fig 7.1 shows the minimal variations between the testing and the training dataset. For each epoch of the training procedure, the dataset is shuffled, and the batches are extracted for the training.

In Chapter 7, we have shown that the time series autoregression has spikes up to the 9^{th} previous element, and the best results are those using input sizes between 9 and 11. We utilized an input size $d = 11$. The number of hidden units p is set to 180.

The first objective of this chapter is to build a training procedure that is a trade-off between accuracy and training time. The training procedure is sequential: it combines early stopping with varying batch sizes, number of training epochs, and learning rate.

These factors greatly influence the speed of the training procedure: larger batches can be computed efficiently with graphic cards. Moreover, a batch is a partition of the dataset, thus increasing the batch implies reducing the number of batches per epoch. We remind that in the training procedure, the gradients are computed for every sample of the batch, while the update of the matrix backpropagation is done by averaging the gradients of the batch. This implies that a larger batch is more likely to catch the average features, which are the more common ones among random samples of the dataset.

Reducing the batch size, reducing the learning rate, and increasing the number of epochs increase the accuracy, but the time required to complete the training increases significantly.

Our solution is to sequentially train the model following the training map in Table 8.2. For each training round we set a different combination of batch size, training epochs, learning rate, and early stopping configuration.

The early stopping monitors the improvements in the loss function, and keeps a counter: the number reported in the table is the maximum number of epochs allowed without improvements in the loss function. If the threshold is met then the procedure resets the matrices to the best achieved, and the training continues to the next round.

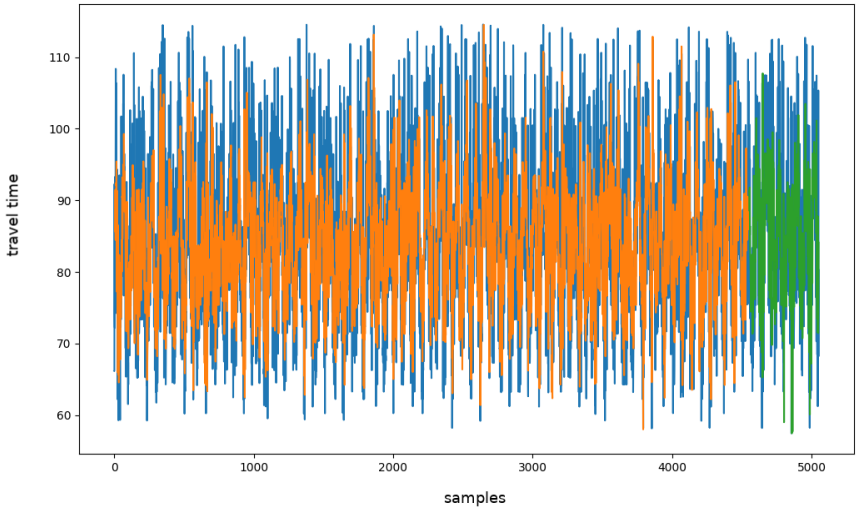
It should be noted that the stopping criteria for a training round are the number of epochs, and the threshold of the early stopping procedure. In fact, since the learning rate is reduced in each round, the number of epochs becomes an upper bound for the training length that is rarely met. Having a reduced learning rate implies that the variations in the loss are minimal, thus the early stopping becomes the actual stopping criteria. Reducing the batch size makes the epoch more sensitive to local variations. This can be compensated by the reduced learning rate, and the increased number of epochs available.

TABLE 8.2: Training map

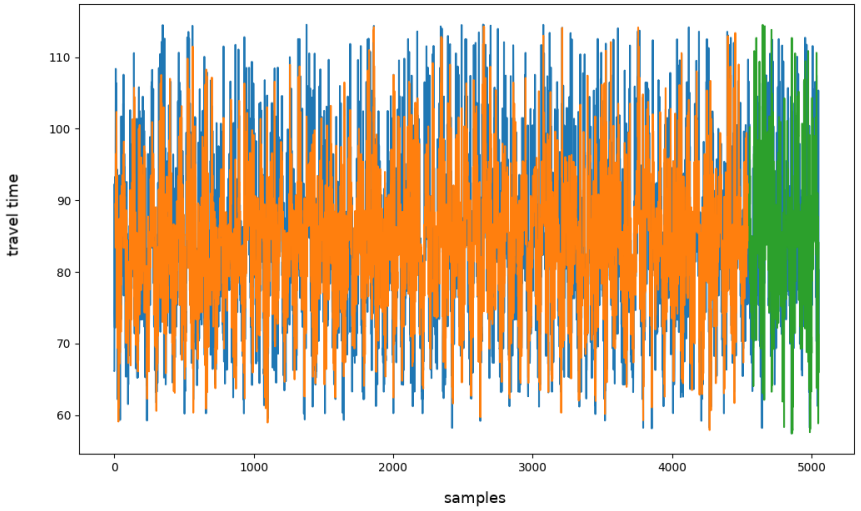
Round	Batch	Epochs	Learning rate	Early stopping
1	64	200	0.005	10
2	32	350	0.003	30
3	16	500	0.0009	50
4	8	800	0.0005	60
5	4	1200	0.0003	80
6	2	1500	0.0001	100

The combinations of these factors result in a training procedure that, in each round, provides a model that has learnt more details of the underlying time series. As the tracking progresses, the last rounds may no longer be necessary. This is because, due to the variability of the distributions of the time series, some berths may have very low variance which can be captured direcly with a larger batch size, while in smaller batches they may end up overfitting the regression. The early stopping criteria prevent the overfitting case and restore the weights of the best validation result. Fig 8.2 shows the progression of the training of the berth SS0027, round 1 and round 2, whilst Fig 8.3

shows round 3 and round 4. Round 5 and 6 failed to achieve a better result, and the procedure kept the matrices of round 4.



(A) Round 1

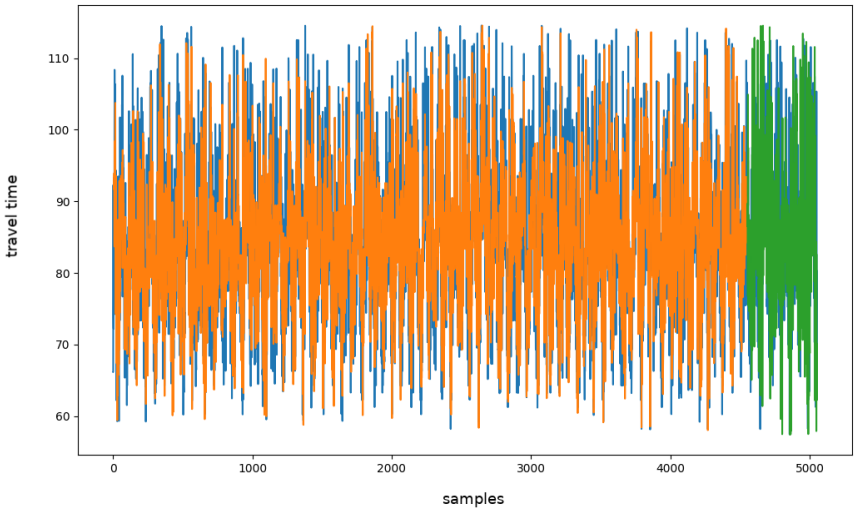


(B) Round 2

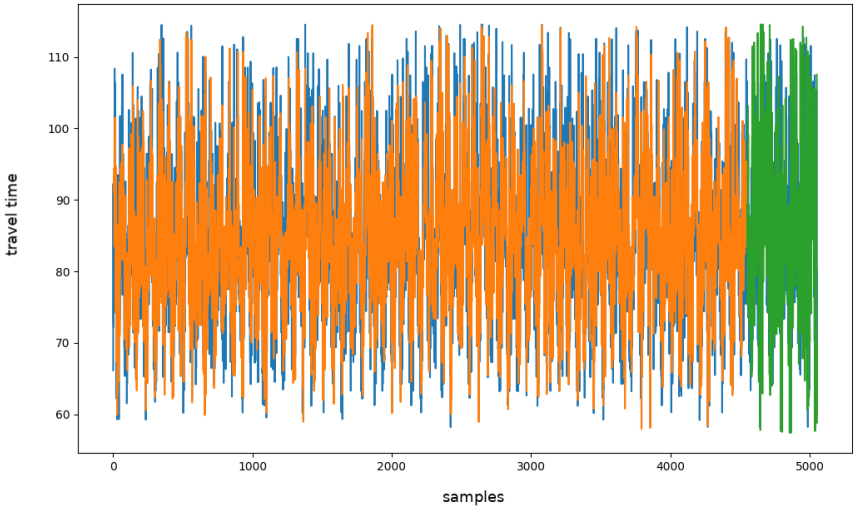
FIGURE 8.2: Training results for berth SS0027. The blue plot is the real time series;the orange plot is the fitted training dataset; and the green plot is the forecasted validation dataset. While progressing through the training rounds, the model learns more details of the underlying distribution.

8.3.2 Forecasting a journey and comparison with actual train running data and industry system (Darwin)

After we have trained all the models, the next objective is to combine the results to provide a journey forecast. We set up two different experiments using the previously trained



(A) Round 3

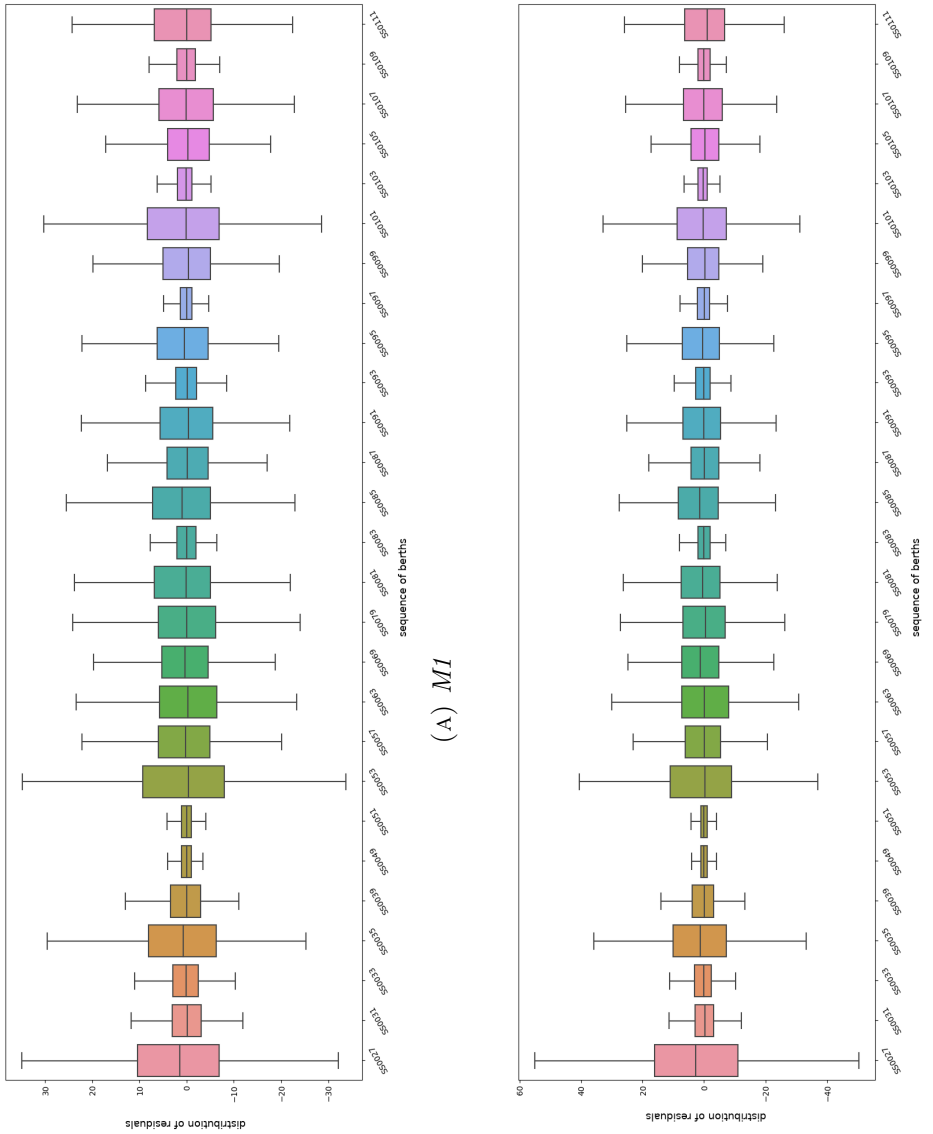


(B) Round 4

FIGURE 8.3: Training results for berth SS0027. The blue plot is the real time series;the orange plot is the fitted training dataset; and the green plot is the forecasted validation dataset. While progressing through the training rounds, the model learns more details of the underlying distribution.

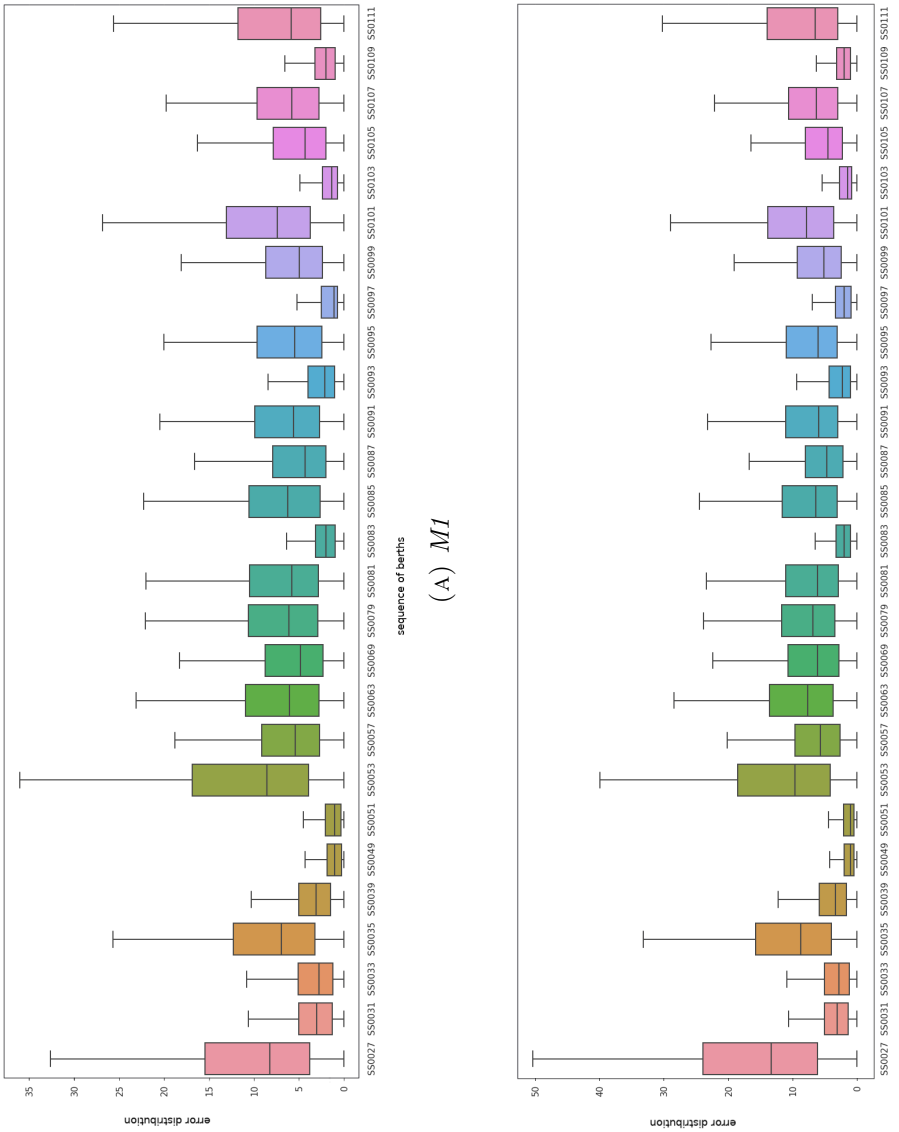
models, named *M1* and *M2*: *M1* is the forecast considering as input, independently for each berth, the immediate previous 11 berth times; *M2*, converserly, considers the previous 11 berth times that are available 4 hours before the start of the journey. This comparison is very useful to understand whether the freshness of inputs can influence the usefulness of the forecast, and if so, by how much.

The first objective of this section is to show how accurate is the forecast. Fig 8.4 shows the box plot of the residuals, the difference between $\tilde{\pi_i}$ and π_i. *M2*, as expected degrades in comparison to *M1*. The surprising fact is that the degradation is minimal. Fig 8.5 compares the errors (against actual train running time) of *M1* and *M2*. The error is defined as the absolute value of the residual (the difference between the forecasted time and the actual train running time). The degradation can be seen more clearly in Fig. 8.5: in the *M1* experiment (Fig 8.5a) the third quartile is over 10 seconds in 9 berths, whilst in the *M2* (Fig 8.5b) the third quartile is over 10 seconds in 13 berths. Note that Fig 8.4 and Fig 8.5 are built from the results of the testing dataset.



(A) *M1* (B) *M2*

FIGURE 8.4: Comparison between the residuals of *M1*, and *M2*.



(A) *M1*

(B) *M2*

FIGURE 8.5: Comparison between the error of *M1*, and *M2*.

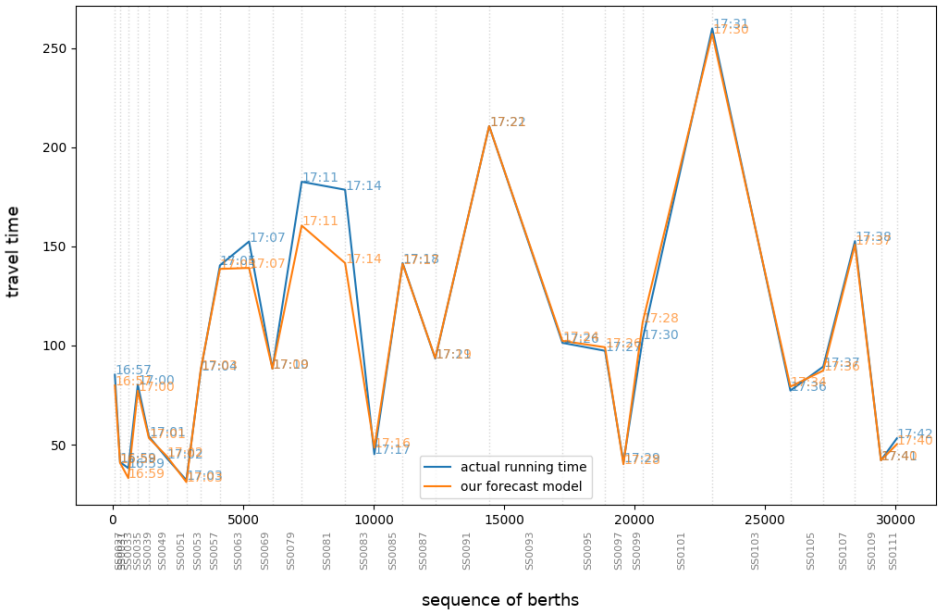
Table 8.3 compares the Root Mean Squared Error (RMSE) of the forecasting of M1 and M2 on all journeys with $running\ time \leq (average\ running\ time + 3 * standard\ deviation)$. This condition is to exclude journeys with exceptional delays that are not representative. In total 1137 journeys were selected for this comparison. As can be seen in the table, for most of the berths M2 perform less well than M1, but the differences are mostly insignificant. There are two berths, SS0109 and SS0051, where M2 performs better than M1. We believe this is due to a combination of factors: the time series of these berths have a very narrow variance and a lucky initialisation of the weights during the training of M2 may have lead to a better fitting.

	M1		M2	
	RMSE	std	RMSE	std
SS0027	14.581836	23.075769	20.668506	27.695899
SS0031	4.361917	5.113927	4.388976	5.210449
SS0033	5.224612	9.631311	5.951244	11.491995
SS0035	11.512732	17.055514	13.573504	18.054848
SS0039	4.548265	5.716148	4.966396	6.084083
SS0049	1.776982	3.007798	1.810412	3.035663
SS0051	3.676582	7.820575	3.586933	7.576574
SS0053	15.710668	20.770596	16.788535	22.555885
SS0057	8.353682	11.391245	8.620071	11.473983
SS0063	9.989990	16.018613	11.907013	15.939782
SS0069	8.086827	12.674306	9.267214	12.690157
SS0079	9.494750	14.591014	10.355410	16.077068
SS0081	8.954740	11.309755	9.389406	11.651676
SS0083	2.888621	3.789270	2.988745	3.976358
SS0085	9.598039	15.316348	10.317552	15.411000
SS0087	7.237614	12.604651	7.376964	12.547221
SS0091	9.060519	15.005347	9.859909	15.257585
SS0093	3.965536	5.577765	4.144572	5.756028
SS0095	9.107166	12.833648	9.998929	13.625749
SS0097	2.476155	4.120374	3.013623	4.576232
SS0099	8.648233	14.768626	9.042913	14.903215
SS0101	11.893781	17.971506	12.349709	18.003396
SS0103	2.258325	3.150573	2.411672	3.299971
SS0105	6.538481	8.012071	6.732601	8.284258
SS0107	9.270532	14.247824	9.679752	13.972443
SS0109	3.168467	6.478378	3.153379	5.931737
SS0111	13.647585	23.035171	14.502992	23.185706

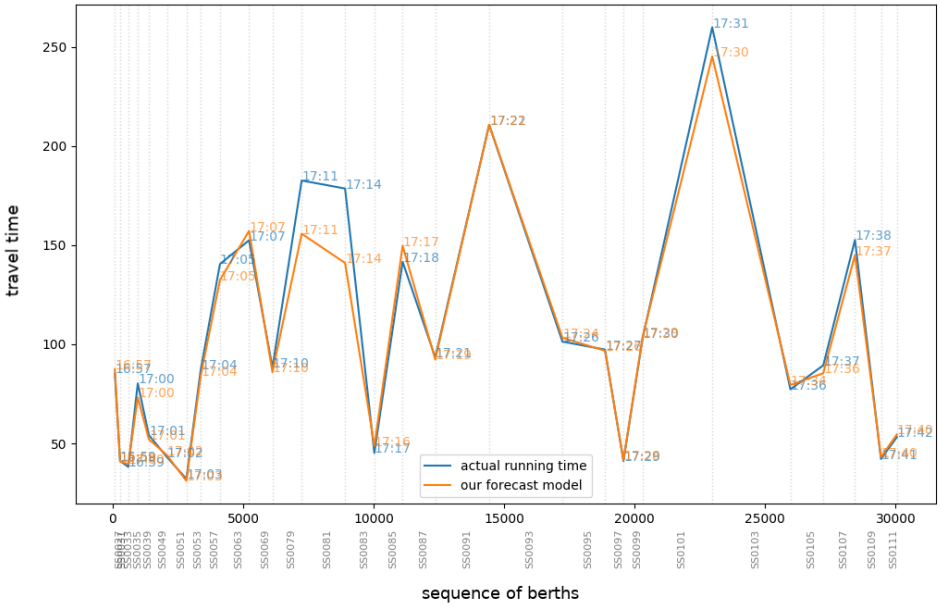
TABLE 8.3: Comparison of M1 and M2 in forecast accuracy against actual running time (RMSE). The number of tested journeys is 1137. The unit of measure is second.

Fig 8.6 shows a typical example of the comparison between the forecasted values by *M1* and *M2* against the actual train running times. The journey is the 362S48MV01 (a train with headcode 2S48 that ran on 1st April 2018). The times reported are rounded to

the minute, and are calculated using Eq.(8.1). The vertical axis is the time in seconds, while the horizontal axis is the position of berths along the journey (Eq.(8.2)). The example illustrates that firstly the two models can forecast train times effectively with small errors, and secondly although there is some degrades in *M2*, it is minimal.



(A) *M1*



(B) *M2*

FIGURE 8.6: Comparison between *M1*, and *M2*: the forecast for the journey 362S48MV01.

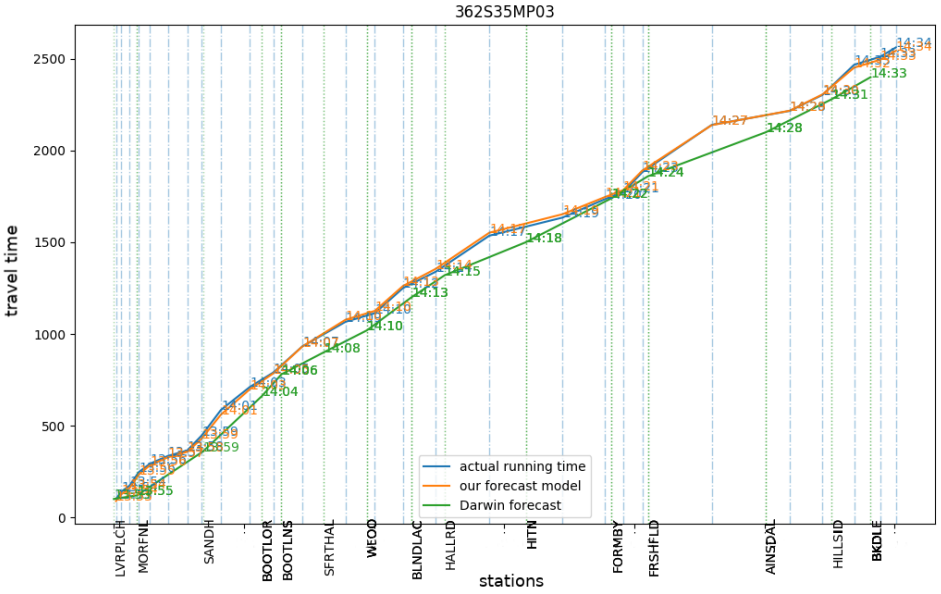
Figs 8.7-8.9 compare the forecasts of M1, M2, and the industry system Darwin against the actual train running time. The vertical axis shows the time progression of the

journey, in seconds, while the horizontal axis shows spatial progression, in metres. In the horizontal axis we use the blue dashed lines to show the positions of the berth steps, and the green dotted lines to show the positions of the stations, measured along the reference line of the rail track. We remind that the Darwin forecast is rounded to the minute, thus, the progression on the plots are all shown in minutes to allow comparisons. The baseline for the first estimation is the first berth of the journey.

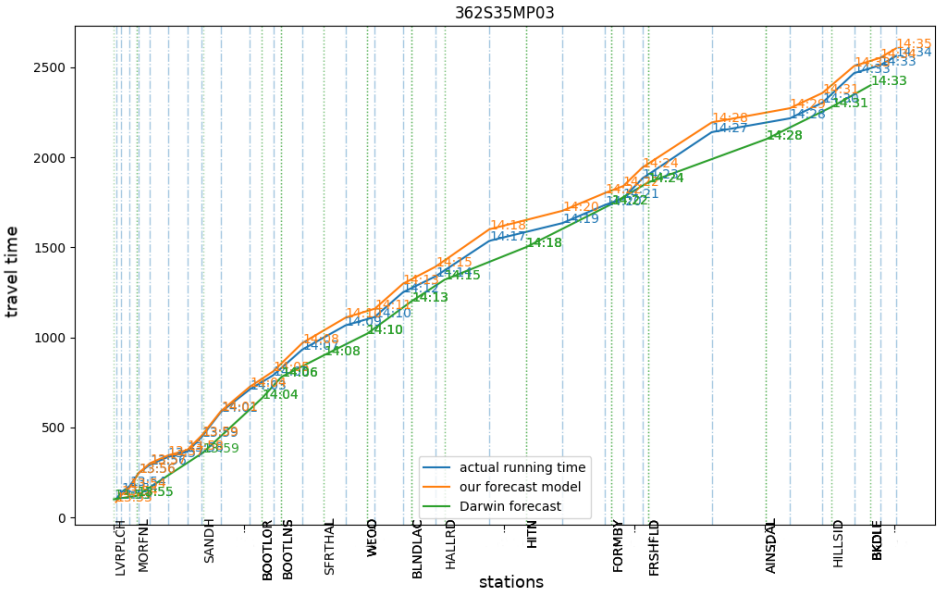
We reported here only three typical journeys, but the behaviours are the same for all 200 journeys that we have analysed. The first one shows *M1*, Fig 8.7a, forecasts the running time well and matches almost exactly the actual running time, while *M2*, Fig 8.7b, slightly overestimates the arrival time $\tau_{i,i+1}$. Both models forecast better than Darwin, which underestimates the running time. The second journey shows some interesting behaviours. Throughout the journey, *M1* (Fig 8.8a) starts well and maintains only a small gap to the actual running time. However, when the journey progresses toward the end, from the last four berth steps (corresponding to the last two stations), it starts to further underestimate and eventually ends up with an underestimation of 2 minutes at the end of the journey. *M2* (Fig 8.8b), on the other hand, is less accurate than *M1* for most of the journey (although the gap to actual time still remains small), but when the journey approaches its end (last four berth steps), *M2* becomes better than *M1* and ends up underestimating only 1 minute at the end of the journey. Darwin remains less accurate than both proposed models and ends up underestimating by 3 minutes at the end. In the third journey (Figs 8.9a, 8.9b), both *M1* and *M2* forecast well and match almost exactly the actual running time. Darwin again performs worse, but what is interesting is that the plot shows Darwin updates its forecasts multiple times during the journey, based on real time inputs. This reflects how Darwin works in reality.

In summary, the experiments show that both proposed models can work well in forecasting train running time. Both models work better than the current industry system, Darwin, in the tested data. The experiments also show that as expected using more recent inputs (*M1*) will provide more accuracy than using inputs from 4 hours before the start of the journey (*M2*), but the difference is shown to be minimal.

It is possible to make the proposed model to be even more accurate. This can be done by updating the model with real time arrival of trains for the incoming berths, each time the train crosses a new berth-step. This way, in (8.1), the forecasts of π_i will be substituted with the real values achieved. This substitution, obviously, reduces the overall error, because it reduces the sources of errors. Given a journey of n berths, the procedure requires $\frac{n\cdot(n+1)}{2}$ recomputations of forecasts (each time a new berth step is crossed). Moreover, given m endpoint devices (CIS), the number of transmissions will be $m \cdot \frac{n\cdot(n+1)}{2}$.



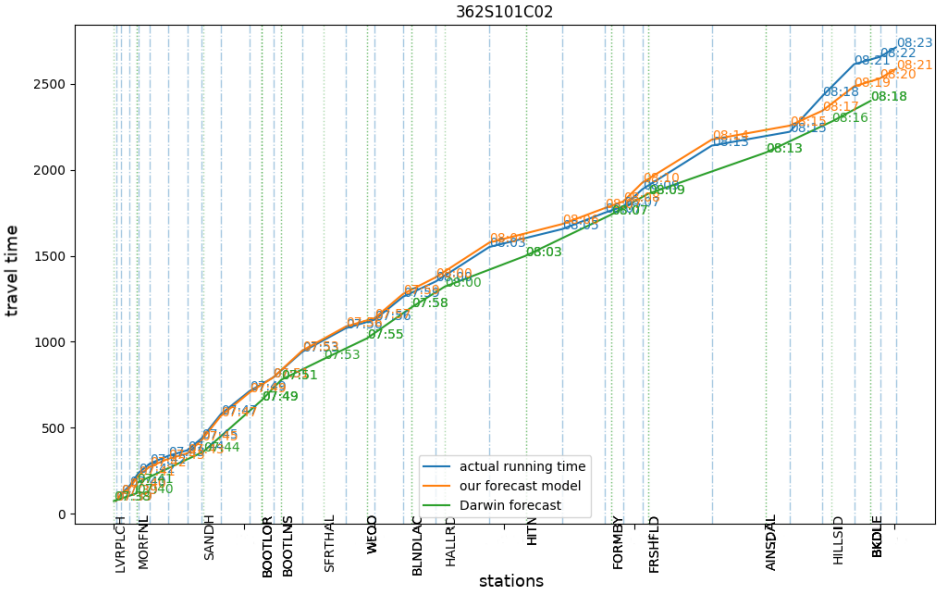
(A) *M1*, 362S35MP03



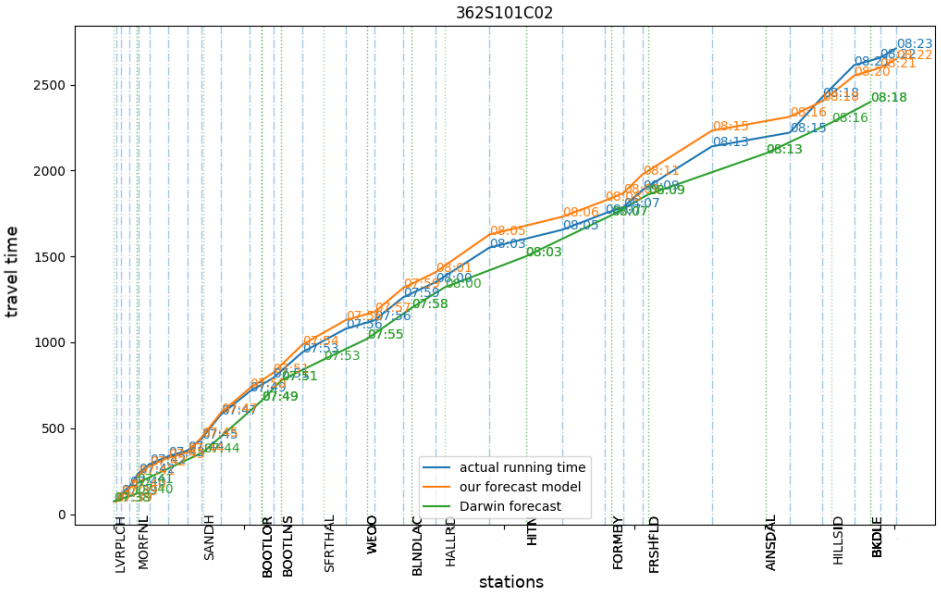
(B) *M2*, 362S35MP03

FIGURE 8.7: Comparison of M1, M2, and Darwin forecasting on real data of train running time. The ID of the journeys presented here is 362S35MP03.

This recomputation can be reduced by comparing the actual value $\tau_{i,i+1}$ with the forecasted value $\tau'_{i,i+1}$: if the distance between them is under a fixed threshold ϵ then is not necessary to recompute and propagate the new results. Conversely, if the distance is over the threshold, the recomputations are necessary to keep the forecast error contained, especially in very long journeys. This improvement, however, is out of the scope of this thesis.



(A) *M1*, 362S101C02

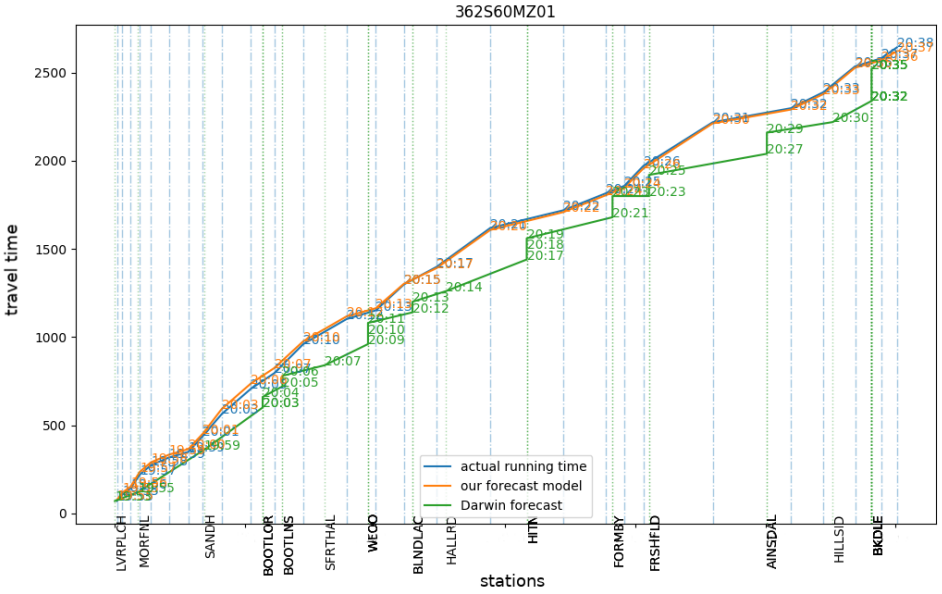


(B) *M2*, 362S101C02

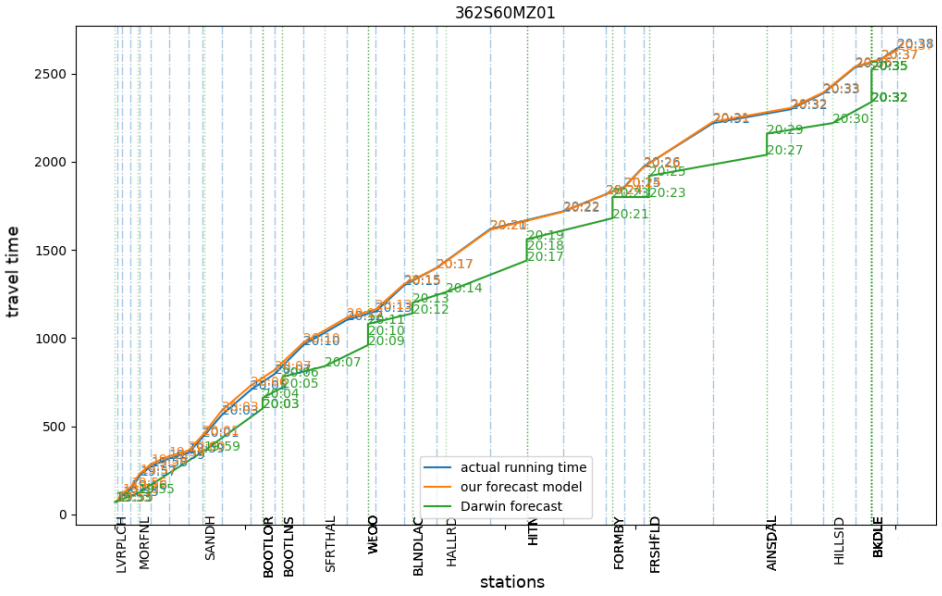
FIGURE 8.8: Comparison of M1, M2, and Darwin forecasting on real data of train running time. The ID of the journeys presented here is 362S101C02.

8.4 Summary

In this chapter we proposed a forecasting system based on the higher detail level available real-time data feeds, the Train Describer system. We have tested this system on real data from the Northern line on the Merseyrail network in the UK.



(A) *M1*, 362S60MZ01



(B) *M2*, 362S60MZ01

FIGURE 8.9: Comparison of M1, M2, and Darwin forecasting with on real data of train running time. The ID of the journeys presented here is 362S60MZ01.

We have firstly built a Gated Recurrent Unit model to forecast the train running time. We have also proposed a training procedure that is a trade-off between accuracy, and training time. The procedure is a sequential training that in each round improves the model knownledge of the time series. The motivation for building such a procedure is that the berths have different distributions: for some berths, there may be a lack of samples or a very low variance distribution, while other berths may have richer variability.

Secondly, we have compared two different forecasting concepts. The first one, *M1*, forecasts the running time, for each berth, based on the immidiate previous n running times. The second one, *M2*, forecasts the running time, for each berth, based on the previous n running times available 4 hours before the time of the start of the journey. Testing on real data, most concepts have shown to be effective and can forecast train time with low errors. The experimental results have also shown that the difference in terms of accuracy between *M1* and *M2* is minimal. We have also compared both methods with Darwin, the forecast system that is being used in most UK stations. The results show that both of our proposed methods are more accurate than Darwin on the tested real data.

We have concluded that a balanced procedure that filters the communication of the new forecasting based on the exceeding of a threshold could be an effective trade-off between accuracy, delivery costs, and stability of the forecast.

This work is important because forecasting the train running time is a very important task to both traffic controllers and passengers. Traffic controllers can exploit a precise forecast to better plan and control the railway traffic, during normal operating conditions and during distruption management. Passengers use the forecast every time that they look for train runtime, either from one of the screens at stations, or from a journey planner app or website.

Our method is the building block to estimate also expected delays. The importance of using the Train Describer's berth information is that it delivers the forecasts to the second, the highest accuracy possibly achievable with the current UK infrastructure.

Chapter 9

Final conclusions and future work

This thesis researches into four real problems coming from the supply chain management and railway industry. The suggested solutions exploit computational intelligence techniques and methodologies.

The first problem is the Multiple Heterogeneous Knapsack Problem, giving priority to smaller bins and considering the following constraints: non overlapping, boundaries and positioning (both constrained and free), rotations (around the vertical axis), orthogonal displacement, weight limit, static stability, weight distribution in a pyramidal region and load bearing considering items' arbitrary centre of mass. The contributions are:

- A mixed-integer linear model for the aforementioned problem
- A study of the trade-off of adding more constraints to make the problem more realistic and the complexity of finding a solution.
- New metrics that facilitate the comparison of datasets used in experiments.
- A constructive heuristic named Weight First Best Fit to handle large scale instances in a reasonable time.

The second problem is how to estimate if a particular packing solution is feasible in a constant O(1) computational time. Given that traditional feasibility checking for packing solutions is an NP-Hard problem, the aim is to significantly save time and computational effort. The contributions are:

- A novel two-stage strategy: the first stage exploits the master/slave Bender's decomposition to build a dataset of knapsack solutions; the second stage exploits the dataset to train a classifier for checking the feasibility of a packing solution.

- A new dataset of packing solutions and benchmark different classification algorithms.
- An analysis on the classification performances in the new dataset for the following algorithms: decision trees (DT), random forest (RF), support vector machine with radial basis function kernel (SVM-R), support vector machine with polynomial kernel (SVM-P), three different architectures of convolutional neural networks (CNN), feed forward fully connected neural networks (FFNN) with one, two and three hidden layers.
- We also suggest some research directions on how to improve the methodology in its various stages.

The third problem is to improve the train seat reservation system. The contributions are:

- A mixed-integer linear model for the Group Seat Reservation Knapsack Problem with Price on Seat, an extension of the Offline Group Seat Reservation Knapsack Problem. We introduce a profit evaluation dependent on not only the space occupied, but also on the individual profit brought by each reserved seat.
- A new GRASP based algorithm that solves the original problem and the newly proposed one.
- Problem instances that represent real world scenarios more realistically.

The fourth problem is to exploit the recently opened datasets to improve the United Kingdom railway system. The contributions are:

- An algorithm to automatically generate the *berths graph*, a graph that represents the feasible train routing strategies through the network of berths.
- Two different approaches to estimate the amount of time that a train is going to spend on a berth. The first approach uses an AutoRegressive Moving Average model, the second approach uses the Gated Recursive Unit and the Long short-term memory.
- An analysis that reveals that the best results have been obtained by networks with input sizes that were covering the statistically significant spikes of the AutoCorrelation Function.

- Two different forecasting systems based on Gated Recurrent Unit models and the *berths graph*. The first one utilises an input of the immediate previous n running times. The second one, instead, uses the previous n running times available 4 hours before the start of the journey.
- A sequential training procedure that is a trade-off between accuracy and training time. Each round improves the model knowledge of the time series. The motivations for building such a procedure is that the berths have different distributions. For some berths, there may be lack of samples or a very low variance distribution, while other berths may have a richer variability.
- A comparison of the methods with official forecast system currently used by the industry.
- A procedure that filters the communication of the new forecasting based on the exceeding of a threshold.

The work done in this thesis can be further improved as follows.

- The model in section 3.1.1 can be further generalised by introducing a priority term p_j. In this thesis the priority is given to smaller bins $p_j = \frac{1}{(L_j \cdot W_j \cdot H_j)}$, we could change the priority to bigger bins by setting $p_j = 1$. We published this generalization of the model and the heuristic in Deplano et al. (2019). However, a complete study on the general model could form part of a future study, since we focused on the priority of smaller bins.
- The problem in chapter 3 could be further studied by performing a sensitivity analysis. In particular it could be interesting to show the relationship between the problem instance complexity, the position of the items' centre of mass, their weight, and the shape of the bin.
- The heuristic Weight First Best Fit presented in section 3.2 does not take into account that the global centre of mass changes in space after one item is put in. The heuristic could further be improved by adopting a look-ahead strategy that takes into account the combination of the next n items. For example, using an $n = 2$ could solve the limitation shown in Figure 3.7.
- In section 4.2 we have shown that in the second stage we built an unbalanced dataset. If one requires a balanced dataset, the following approach can be used: 1) exploiting the works done in Atamtürk (2004, 2005); Atamtürk and Bhardwaj (2015) to derive, for each bin type in J, and for each couple of variables in I,

a valid cover, 2) then sample a fixed number of points under the cover inequality and augment the sampled set with unfeasible points, 3) eventually extend the sampling to random combinations of variables, e.g. uniformly sampling as in Xu et al. (2018). Some covers may be easier to find by exploiting the work on harmonic packing in Barnes (1982a,b), and applying a lifting procedure (Zemel, 1989; Kaparis and Letchford, 2010). The work on harmonic packing is useful also for reducing the number of classes of items and bins considered, since some of them may be multiples of others by Barnes (1982a,b).

- The sequential master/slave Bender's decomposition described in stage one of chapter 4, combined with a reinforcement learning, can be exploited to build a cover lifting procedure. The policies may be defined to lift a feasible/unfeasible packing until it becomes unfeasible/feasible, the state represents the candidate facet, while the reward/penalty drives the optimisation.

- The two stage strategy described in chapter 4 can be enhanced with a third stage: the exploitation of the classifiers into the optimisation process. A direct way could be, as mentioned in section 4.1, to bind the classifiers in the mathematical programming model, e.g. implementing the piecewise linearisation of the classifier in constraint (4.10). Another strategy that may be effective with evolutionary methods is to bind the classifiers in the evaluation procedure: it will become a Monte Carlo evaluation, where the more time the best solution survives, the more likely a certificate of feasibility is produced. The benefit in this case is to avoid certifications until the exploitation phase, while the exploration would benefit from checking done in constant time.

- The classifier of chapter 4 could be enhanced by considering the class of items by their packing efficiency. The transformation can be achieved by dividing the items' dimension by the bins' dimensions. Given m classes of bins and n classes of items, the classifier would be one with an input of $m \cdot n$ elements. This formulation would achieve a multiplication invariance: packing an item 4x4x4 on a bin 10x10x10, would be the same as packing an item 8x8x8 into a bin 20x20x20.

- The problem studied in chapter 5 could be improved by considering seat reservation over time. The model proposed in section 5.1 could be adapted by considering the problem as sequential packing, where in each stage the function is maximized considering the allocations done in the previous stages as constraints. A better strategy would be possible if we have access to the distribution of future reservations. The problem in this scenario could be approached considering sequential packing, where in each stage a certain number of reservations, and the distribution of the future reservations is available. The objective would be transformed in

maximising the expected value of the filling rate in each stage given the previous allocations as constraints.

- The algorithm proposed in section 5.2 tries to pack items in the order of the permutation and stops when it encounters the first infeasible item. The algorithm could be improved by changing the stopping criterion of the evaluation function to test all the remaining items until there is free space. This change would reduce the searching space and thus will improve the overall convergence speed.

- The forecasting model in chapter 8 forecasts the journey running time of a train. The model is not suitable to directly forecast the running time in a what-if analysis considering as example, skipping a station, a prolonged dwell-time, a temporary speed restriction, the effects of reactionary delays, etc. To enable this type of analysis the model has to be exploited into a simulation, where the scenarios can be parametrized more easily. As example, the running time in each berth implies a train speed, a speed restriction could be modelled by multiplying a rational speed factor. Skipping a station can be modelled in a similar way in which the penalty factor has to be defined per berth. The modelling of the reactionary delay is more complex, it implies the consideration of the trains' interaction in the complete network and with the timetable. The interactions are in the junctions and in station. A discrete event simulation could correctly model the problem.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Alvarez-Valdés, R., Parreño, F., and Tamarit, J. M. (2013). A grasp/path relinking algorithm for two-and three-dimensional multiple bin-size bin packing problems. *Computers & Operations Research*, 40(12):3081–3090.

Atamtürk, A. (2004). Sequence independent lifting for mixed-integer programming. *Operations Research*, 52(3):487–490.

Atamtürk, A. (2005). Cover and pack inequalities for (mixed) integer programming. *Annals of Operations Research*, 139(1):21–38.

Atamtürk, A. and Bhardwaj, A. (2015). Supermodular covering knapsack polytope. *Discrete Optimization*, 18:74–86.

Balfe, N. (2010). *Appropriate automation of rail signalling systems: a human factors study*. PhD thesis, University of Nottingham.

Barnes, F. W. (1982a). Algebraic theory of brick packing i. *Discrete Mathematics*, 42(1):7–26.

Barnes, F. W. (1982b). Algebraic theory of brick packing ii. *Discrete Mathematics*, 42(2-3):129–144.

Beasley, J. (1985). Algorithms for unconstrained two-dimensional guillotine cutting. *Journal of the Operational Research Society*, 36(4):297–306.

Becker, M. and Schreckenberg, M. (2018). Analytical method for the precise and fast prediction of railway running times and its applications. *IEEE Transactions on Intelligent Transportation Systems*, (99):1–10.

Bello, I., Pham, H., Le, Q. V., Norouzi, M., and Bengio, S. (2016). Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*.

Belov, G., Kartak, V., Rohling, H., and Scheithauer, G. (2013). Conservative scales in packing problems. *OR spectrum*, pages 1–38.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische mathematik*, 4(1):238–252.

Berger, A., Gebhardt, A., Müller-Hannemann, M., and Ostrowski, M. (2011). Stochastic delay prediction in large train networks. In *11th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

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

Box, G. E. and Jenkins, G. M. (1976). Time series analysis: Forecasting and control san francisco. *Calif: Holden-Day*.

Boyar, J. and Larsen, K. S. (1999). The seat reservation problem. *Algorithmica*, 25(4):403–417.

Cambazard, H. and Jussien, N. (2005). Integrating benders decomposition within constraint programming. *Principles and Practice of Constraint Programming-CP 2005*, pages 752–756.

Ceschia, S. and Schaerf, A. (2013). Local search for a multi-drop multi-container loading problem. *Journal of Heuristics*, pages 1–20.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Chollet, F. et al. (2015). Keras. https://keras.io.

Christensen, H. I., Khan, A., Pokutta, S., and Tetali, P. (2016). Multidimensional bin packing and other related problems: A survey.

Christensen, H. I., Khan, A., Pokutta, S., and Tetali, P. (2017). Approximation and online algorithms for multidimensional bin packing: A survey. *Computer Science Review*, 24:63–79.

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

Clausen, T., Hjorth, A. N., Nielsen, M., and Pisinger, D. (2010). The off-line group seat reservation problem. *European Journal of Operational Research*, 207(3):1244–1253.

Clautiaux, F., Alves, C., and de Carvalho, J. V. (2010). A survey of dual-feasible and superadditive functions. *Annals of Operations Research*, 179(1):317–342.

Cochrane, E. and Beasley, J. (2003). The co-adaptive neural network approach to the euclidean travelling salesman problem. *Neural Networks*, 16(10):1499 – 1525.

Coffman Jr, E. G., Csirik, J., Galambos, G., Martello, S., and Vigo, D. (2013). Bin packing approximation algorithms: survey and classification. In *Handbook of combinatorial optimization*, pages 455–531. Springer.

Coffman Jr, E. G., Csirik, J., Johnson, D. S., and Woeginger, G. J. (2004). An introduction to bin packing. *Bibliographie. Siehe www. inf. u-szeged. hu/˜ csirik*.

Corman, F. and Kecman, P. (2018). Stochastic prediction of train delays in real-time using bayesian networks. *Transportation Research Part C: Emerging Technologies*, 95:599 – 615.

Côté, J.-F., Dell'Amico, M., and Iori, M. (2014). Combinatorial benders' cuts for the strip packing problem. *Operations Research*, 62(3):643–661.

Crainic, T., Perboli, G., and Tadei, R. (2012a). A greedy adaptive search procedure for multi-dimensional multi-container packing problems.

Crainic, T. G., Perboli, G., and Tadei, R. (2009). Ts2pack: A two-level tabu search for the three-dimensional bin packing problem. *European Journal of Operational Research*, 195(3):744–760.

Crainic, T. G., Perboli, G., and Tadei, R. (2012b). Recent advances in multi-dimensional packing problems. In *New technologies-trends, innovations and research*. InTech.

Croes, G. A. (1958). A method for solving traveling-salesman problems. *Operations Research*, 6(6):791–812.

Daamen, W., Goverde, R. M. P., and Hansen, I. A. (2008). Non-discriminatory automatic registration of knock-on train delays. *Networks and Spatial Economics*, 9(1):47–61.

Dantzig, G. B. and Thapa, M. N. (2006a). *Linear programming 1: introduction*. Springer Science & Business Media.

Dantzig, G. B. and Thapa, M. N. (2006b). *Linear programming 2: theory and extensions*. Springer Science & Business Media.

Dantzig, G. B. and Wolfe, P. (1960). Decomposition principle for linear programs. *Operations Research*, 8(1):101–111.

de Queiroz, T. A. and Miyazawa, F. K. (2013). Two-dimensional strip packing problem with load balancing, load bearing and multi-drop constraints. *International Journal of Production Economics*, 145(2):511–530.

Deane, J. and Agarwal, A. (2013). Neural, genetic, and neurogenetic approaches for solving the 0-1 multidimensional knapsack problem. *International Journal of Management & Information Systems (Online)*, 17(1):43.

Delorme, M., Iori, M., and Martello, S. (2015). Combinatorial benders' decomposition for the orthogonal stock cutting problem. *Proceedings of the XLVII Simpósio Brasileiro de Pesquisa Operacional (SBPO)*.

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

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

Deplano, I., Lersteau, C., and Nguyen, T. T. (2019). A mixed-integer linear model for the multiple heterogeneous knapsack problem with realistic container loading constraints and bins' priority. *International Transactions in Operational Research*, n/a(n/a).

Dyckhoff, H. (1990). A typology of cutting and packing problems. *European Journal of Operational Research*, 44(2):145–159. Cutting and Packing.

Eremin, A. and Wallace, M. (2001). Hybrid benders decomposition algorithms in constraint logic programming. In *International Conference on Principles and Practice of Constraint Programming*, pages 1–15. Springer.

Fekete, S. P. and Schepers, J. (2004). A combinatorial characterization of higher-dimensional orthogonal packing. *Mathematics of Operations Research*, 29(2):353–368.

Fekete, S. P., Schepers, J., and Van der Veen, J. C. (2007). An exact algorithm for higher-dimensional orthogonal packing. *Operations Research*, 55(3):569–587.

Feo, T. A. and Resende, M. G. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations research letters*, 8(2):67–71.

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

Fraga-Lamas, P., Fernández-Caramés, T. M., and Castedo, L. (2017). Towards the internet of smart trains: A review on industrial iot-connected railways. *Sensors*, 17(6):1457.

Geoffrion, A. M. (1972). Generalized benders decomposition. *Journal of optimization theory and applications*, 10(4):237–260.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

Gonçalves, J. F. and Resende, M. G. (2011a). Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, 17(5):487–525.

Gonçalves, J. F. and Resende, M. G. (2013). A biased random key genetic algorithm for 2d and 3d bin packing problems. *International Journal of Production Economics*, 145(2):500–510.

Gonçalves, J. F. and Resende, M. G. C. (2011b). A parallel multi-population genetic algorithm for a constrained two-dimensional orthogonal packing problem. *Journal of Combinatorial Optimization*, 22(2):180–201.

Goverde, R., Daamen, W., and Hansen, I. (2007). Automatic identification of secondary delays based on train describer systems. In *Forms/Format'07 Conference*.

Goverde, R. and Hansen, I. (2000). Tnv-prepare: analysis of dutch railway operations based on train detection data. *Computers in Railways*, 7:779–788.

Goverde, R. M. and Hansen, I. A. (2001). Delay propagation and process management at railway stations. In *5th World Conference on Railway Research (WCRR 2001), Köln, November 25-29, 2001*. WCRR.

Goverde, R. M. and Meng, L. (2011). Advanced monitoring and management information of railway operations. *Journal of Rail Transport Planning & Management*, 1(2):69–79.

Goyal, S. (2018). Essays on the online multiple knapsack problem & the online reservation problem.

Gupta, I. K., Choubey, A., and Choubey, S. (2017). Clustered genetic algorithm to solve multidimensional knapsack problem. In *Computing, Communication and Networking Technologies (ICCCNT), 2017 8th International Conference on*, pages 1–6. IEEE.

Han, J., Jentzen, A., and Weinan, E. (2018). Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences of the United States of America*, 115 34:8505–8510.

Hansen, I. A., Goverde, R. M., and van der Meer, D. J. (2010). Online train delay recognition and running time prediction. In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pages 1783–1788. IEEE.

Hatano, L. (2004). Complexity versus choice: Uk rail fares. *Japan Railway and Transport Review*, 37:26–34.

Hifi, M., Kacem, I., Nègre, S., and Wu, L. (2010). A linear programming approach for the three-dimensional bin-packing problem. *Electronic Notes in Discrete Mathematics*, 36:993–1000.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hopfield, J. J. and Tank, D. W. (1985). "neural" computation of decisions in optimization problems. *Biol. Cybern.*, 52(3):141–152.

Hu, H., Duan, L., Zhang, X., Xu, Y., and Wei, J. (2018). A multi-task selected learning approach for solving new type 3d bin packing problem. *CoRR*, abs/1804.06896.

Hu, H., Zhang, X., Yan, X., Wang, L., and Xu, Y. (2017). Solving a new 3d bin packing problem with deep reinforcement learning method. *arXiv preprint arXiv:1708.05930*.

Jegadeshwari, S. and Jaisree, D. (2014). Heuristic algorithm for constrained 3d container loading problem: A genetic approach. *International Journal of Computing Algorithm*, 3(1):1016–1020.

Jin, R. (2017). Deep learning at alibaba. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 11–16. AAAI Press.

Jin, Z., Ito, T., and Ohno, K. (2003). The three-dimensional bin packing problem and its practical algorithm. *JSME International Journal Series C Mechanical Systems, Machine Elements and Manufacturing*, 46(1):60–66.

Jones, R. H. (1980). Maximum likelihood fitting of arma models to time series with missing observations. *Technometrics*, 22(3):389–395.

Junqueira, L., Morabito, R., and Yamashita, D. S. (2012). Three-dimensional container loading models with cargo stability and load bearing constraints. *Computers & Operations Research*, 39(1):74–85.

Kaparis, K. and Letchford, A. N. (2010). Cover inequalities. *Encyclopedia of Operations Research and Management Science*, 2:1074–1080.

Kecman, P. and Goverde, R. M. (2013a). Adaptive, data-driven, online prediction of train event times. In *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on*, pages 803–808. IEEE.

Kecman, P. and Goverde, R. M. (2013b). Process mining of train describer event data and automatic conflict identification. *Computers in Railways XIII: Computer System Design and Operation in the Railway and Other Transit Systems*, 127:227.

Kecman, P. and Goverde, R. M. (2015). Predictive modelling of running and dwell times in railway traffic. *Public Transport*, 7(3):295–319.

Kecman, P., Goverde, R. M. P., and Hansen, I. A. (2011). Train describer records as a source of information for infrastructure monitoring, performance analysis and traffic management. In *5th IET Conference on Railway Condition Monitoring and Non-Destructive Testing (RCM 2011)*, pages 1–5.

Kellerer, H., Pferschy, U., and Pisinger, D. (2004). Introduction to np-completeness of knapsack problems. In *Knapsack problems*, pages 483–493. Springer.

Kennedy, M. P. and Chua, L. O. (1988). Neural networks for nonlinear programming. *IEEE Transactions on Circuits and Systems*, 35(5):554–562.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, NIPS'12, pages 1097–1105, USA. Curran Associates Inc.

Kumar, M. and Yadav, N. (2011). Multilayer perceptrons and radial basis function neural network methods for the solution of differential equations: a survey. *Computers & Mathematics with Applications*, 62(10):3796–3811.

Lachhwani, K. (2019). Application of neural network models for mathematical programming problems: A state of art review. *Archives of Computational Methods in Engineering*.

Landschützer, C., Ehrentraut, F., and Jodin, D. (2015). Containers for the physical internet: requirements and engineering design related to fmcg logistics. *Logistics Research*, 8(1):1–22.

Laterre, A., Fu, Y., Jabri, M. K., Cohen, A., Kas, D., Hajjar, K., Dahl, T. S., Kerkeni, A., and Beguir, K. (2018). Ranked reward: Enabling self-play reinforcement learning for combinatorial optimization. *CoRR*, abs/1807.01672.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lessan, J., Fu, L., and Wen, C. (2018). A hybrid bayesian network model for predicting delays in train operations. *Computers & Industrial Engineering*.

Leung, J. Y., Tam, T. W., Wong, C. S., Young, G. H., and Chin, F. Y. (1990). Packing squares into a square. *Journal of Parallel and Distributed Computing*, 10(3):271–275.

Li, D., Daamen, W., and Goverde, R. M. (2016). Estimation of train dwell time at short stops based on track occupation event data: A study at a dutch railway station. *Journal of Advanced Transportation*, 50(5):877–896.

Li, D., Goverde, R. M., Daamen, W., and He, H. (2014). Train dwell time distributions at short stop stations. In *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, pages 2410–2415. IEEE.

Li, D., Yin, Y., and He, H. (2018). Testing the generality of a passenger disregarded train dwell time estimation model at short stops: Both comparison and theoretical approaches. *Journal of Advanced Transportation*, 2018.

Lin, Y.-H., Meller, R. D., Ellis, K. P., Thomas, L. M., and Lombardi, B. J. (2014). A decomposition-based approach for the selection of standardized modular containers. *International Journal of Production Research*, 52(15):4660–4672.

Looi, C.-K. (1992). Neural network methods in combinatorial optimization. *Computers & Operations Research*, 19(3):191 – 208.

Maas, A. L., Hannun, A. Y., and Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3.

Martello, S. and Toth, P. (1990). *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, Inc., New York, NY, USA.

Martin, L. J. (2016). Predictive reasoning and machine learning for the enhancement of reliability in railway systems. In *International Conference on Reliability, Safety, and Security of Railway Systems*, pages 178–188. Springer.

Masutti, T. A. and de Castro, L. N. (2009). A self-organizing neural network using ideas from the immune system to solve the traveling salesman problem. *Information Sciences*, 179(10):1454 – 1468. Including Special Issue on Artificial Imune Systems.

Milinković, S., Marković, M., Vesković, S., Ivić, M., and Pavlović, N. (2013). A fuzzy petri net model to estimate train delays. *Simulation Modelling Practice and Theory*, 33:144–157.

MoDULUSHCA (2012). The modulushca project.

Montreuil, B. (2011). Toward a physical internet: meeting the global logistics sustainability grand challenge. *Logistics Research*, 3(2-3):71–87.

Montreuil, B. (2012). Physical internet manifesto.

Nesterov, Y. (1983). A method for unconstrained convex minimization problem with the rate of convergence o (1/kˆ 2). In *Doklady AN USSR*, volume 269, pages 543–547.

Ohlsson, M., Peterson, C., and Söderberg, B. (1993). Neural networks for optimization problems with inequality constraints: The knapsack problem. *Neural Comput.*, 5(2):331–339.

Oneto, L., Fumeo, E., Clerico, G., Canepa, R., Papa, F., Dambra, C., Mazzino, N., and Anguita, D. (2018). Train delay prediction systems: a big data analytics perspective. *Big data research*, 11:54–64.

Open Rail Data Wiki, n.d. Open Rail Data Wiki, n.d. https://wiki.openraildata.com. Accessed: 2019-04-25.

Palmer, J. (2010). The need for train detection. In *Railway Signalling and Control Systems (RSCS 2010), IET Professional Development Course on*, pages 52–64. IET.

Paquay, C., Schyns, M., and Limbourg, S. (2016). A mixed integer programming formulation for the three-dimensional bin packing problem deriving from an air cargo application. *International Transactions in Operational Research*, 23(1-2):187–213.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Perboli, G., Crainic, T. G., and Tadei, R. (2011). An efficient metaheuristic for multi-dimensional multi-container packing. In *Automation Science and Engineering (CASE), 2011 IEEE Conference on*, pages 563–568. IEEE.

Pochet, Y. and Weismantel, R. (1998). The sequential knapsack polytope. *SIAM Journal on Optimization*, 8(1):248–264.

Pochet, Y. and Wolsey, L. A. (1995). Integer knapsack and flow covers with divisible coefficients: polyhedra, optimization and separation. *Discrete Applied Mathematics*, 59(1):57 – 74.

Pongnumkul, S., Pechprasarn, T., Kunaseth, N., and Chaipah, K. (2014). Improving arrival time prediction of thailand's passenger trains using historical travel times. In *2014 11th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, pages 307–312. IEEE.

Qin, S., Feng, J., Song, J., Wen, X., and Xu, C. (2018). A one-layer recurrent neural network for constrained complex-variable convex optimization. *IEEE transactions on neural networks and learning systems*, 29(3):534–544.

Rahmaniani, R., Crainic, T. G., Gendreau, M., and Rei, W. (2017). The benders decomposition algorithm: A literature review. *European Journal of Operational Research*, 259(3):801–817.

Rail, N. Network Rail real-time data feed. https://datafeeds.networkrail.co.uk/. Accessed: 2019-04-25.

Rail, N. Ppm series, periods 28 days.

Ramanujam, J. and Sadayappan, P. (1988). Optimization by neural networks. *IEEE 1988 International Conference on Neural Networks*, pages 325–332 vol.2.

Rashidy, R. A. H. E., Hughes, P., Figueres-Esteban, M., Harrison, C., and Van Gulijk, C. (2018). A big data modeling approach with graph databases for spad risk. *Safety science*, 110:75–79.

Resende, M. G. and Ribeiro, C. C. (2016). *Optimization by GRASP*. Springer.

Resende, M. G. and Ribeiro, C. C. (2019). Greedy randomized adaptive search procedures: Advances and extensions. In *Handbook of Metaheuristics*, pages 169–220. Springer.

Rudd, K. and Ferrari, S. (2015). A constrained integration (cint) approach to solving partial differential equations using artificial neural networks. *Neurocomputing*, 155:277–285.

Sallez, Y., Pan, S., Montreuil, B., Berger, T., and Ballot, E. (2016). On the activeness of intelligent physical internet containers. *Computers in Industry*, 81:96–104.

Shen, Y., Xu, J., Wu, X., and Ni, Y. (2019). Modelling travel time distribution and its influence over stochastic vehicle scheduling. *Transport*, 34(2):237–249.

Smith, K. A. (1999). Neural networks for combinatorial optimization: A review of more than a decade of research. *INFORMS Journal on Computing*, 11(1):15–34.

Smith-Miles, K. and Lopes, L. (2012). Measuring instance difficulty for combinatorial optimization problems. *Computers & Operations Research*, 39(5):875–889.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Sweeney, P. E. and Paternoster, E. R. (1992). Cutting and packing problems: a categorized, application-orientated research bibliography. *Journal of the Operational Research Society*, 43(7):691–706.

Szegedy, C., , , Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9.

Toossi, A., Barson, L., Hyland, B., Fung, W., and Best, N. (2017). *Infrastructure/Train Borne Measurements in Support of UK Railway System Performance-Gaining Insight Through Systematic Analysis and Modelling*, pages 223–244.

Travacca, B. and Moura, S. (2018). Dual hopfield method for large-scale mixed-integer programming [preprint, accepted to cdc 2018].

Trenn, S. (2008). Multilayer perceptrons: Approximation order and necessary number of hidden units. *IEEE transactions on neural networks*, 19(5):836–844.

Trivella, A. and Pisinger, D. (2016). The load-balanced multi-dimensional bin-packing problem. *Computers & Operations Research*, 74:152–164.

Tromp, J. (2004). Validation of a train simulation model with train detection data. *WIT Transactions on The Built Environment*, 74.

UK, R. D. G. (2016). http://data.atoc.org/. Technical report, Rail Delivery Group.

Van Gulijk, C., Hughes, P., Figueres-Esteban, M., Dacre, M., and Harrison, C. (2015). Big data risk analysis for rail safety? In *Proceedings of ESREL 2015*. CRC/Balkema.

Vinyals, O., Fortunato, M., and Jaitly, N. (2015). Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Wang, H. and Chen, Y. (2010). A hybrid genetic algorithm for 3d bin packing problems. In *Bio-Inspired Computing: Theories and Applications (BIC-TA), 2010 IEEE Fifth International Conference on*, pages 703–707. IEEE.

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

Wäscher, G., Haußner, H., and Schumann, H. (2007). An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183(3):1109 – 1130.

Wei, L., Oon, W.-C., Zhu, W., and Lim, A. (2013). A goal-driven approach to the 2d bin packing and variable-sized bin packing problems. *European Journal of Operational Research*, 224(1):110–121.

Wolpert, D. H. and Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82.

Xu, H., Koenig, S., and Kumar, T. S. (2018). Towards effective deep learning for constraint satisfaction problems. In *International Conference on Principles and Practice of Constraint Programming*, pages 588–597. Springer.

Yaghini, M., Khoshraftar, M. M., and Seyedabadi, M. (2013). Railway passenger train delay prediction via neural network model. *Journal of advanced transportation*, 47(3):355–368.

Yazdani, D., Omidvar, M. N., Deplano, I., Lersteau, C., Makki, A., Wang, J., and Nguyen, T. T. (2019). Real-time seat allocation for minimizing boarding/alighting time and improving quality of service and safety for passengers. *Transportation Research Part C: Emerging Technologies*, 103:158–173.

Zemel, E. (1989). Easily computable facets of the knapsack polytope. *Mathematics of Operations Research*, 14(4):760–764.

Zhang, Z., Zheng, L., Li, L., Deng, X., Xiao, L., and Huang, G. (2018). A new finite-time varying-parameter convergent-differential neural-network for solving nonlinear and nonconvex optimization problems. *Neurocomputing*, 319:74–83.

Zhao, X., Bennell, J. A., Bektaş, T., and Dowsland, K. (2016a). A comparative review of 3d container loading algorithms. *International Transactions in Operational Research*, 23(1-2):287–320.

Zhao, Y., Stow, J., and Harrison, C. (2016b). Improving the understanding of spad risks using red aspect approach data. In *Safety and Reliability*, volume 36, pages 199–212. Taylor & Francis.

Zhao, Y., Stow, J., and Harrison, C. (2018). A method for classifying red signal approaches using train operational data. *Safety science*, 110:67–74.

Zhu, W., Oon, W.-C., Lim, A., and Weng, Y. (2012). The six elements to block-building approaches for the single container loading problem. *Applied Intelligence*, 37(3):431–445.

Appendix A

Minimizing the wasted space giving priority to smaller bins is equivalent to maximizing the packing efficiency

In this appendix we show the proof of the following sentence: "Minimizing the wasted space giving priority to smaller bins is equivalent to maximizing the packing efficiency" (A.1)-(A.4).

$$\text{minimize} \quad \sum_{j \in J} \frac{(L_j \cdot W_j \cdot H_j - \sum_{i \in I}(l_i \cdot w_i \cdot h_i \cdot C_{i,j}))}{(L_j \cdot W_j \cdot H_j)} = \tag{A.1}$$

$$\sum_{j \in J} \frac{L_j \cdot W_j \cdot H_j}{L_j \cdot W_j \cdot H_j} - \sum_{j \in J} \frac{\sum_{i \in I}(l_i \cdot w_i \cdot h_i \cdot C_{i,j})}{L_j \cdot W_j \cdot H_j} = \tag{A.2}$$

$$n - \sum_{j \in J} \frac{\sum_{i \in I}(l_i \cdot w_i \cdot h_i \cdot C_{i,j})}{L_j \cdot W_j \cdot H_j} \Leftrightarrow \tag{A.3}$$

$$\text{maximize} \quad \sum_{j \in J} \frac{\sum_{i \in I}(l_i \cdot w_i \cdot h_i \cdot C_{i,j})}{L_j \cdot W_j \cdot H_j} \tag{A.4}$$

Where $\frac{l_i \cdot w_i \cdot h_i}{L_j \cdot W_j \cdot H_j}$ is the packing efficiency.

Appendix B

Slave Problem

Let $I = \{1, \cdots, n\}$ be a set that indexes n rectangular cuboids (items), the slave problem is to determine if exists a configuration such that these items can be packed in a specific rectangular cuboid (bin). For every item $i \in I$ and for the considered bin, let us define their dimensions as width w_i (W), height h_i (H) and length l_i (L). The problem is constrained to avoid any two items i, k to overlap; all the items must be allocated inside the bin; any item $i \in I$ is allowed to rotate orthogonally along the vertical axis.

It is preferable that the solution includes the items' absolute positioning using one of the bin's corners as the origin of the Cartesian system. We define x_i, y_i, z_i, the coordinates of the front-bottom-left item corner.

$\varphi_{o,i} \in \{0, 1\}$ is a binary variable that equals 1 if the orientation of item $i \in I$ follows orientation type $o \in O = \{1, \ldots, r\}$, where r is the number of the feasible rotations for an item (i.e 2). $\varphi_{o,i}$ will be 0 otherwise. The orientation types are defined in Table B.1.

TABLE B.1: Table of orientations

Orientation Type	Parallel to x-axis	Parallel to y-axis	Parallel to z-axis
1	Length	Width	Height
2	Width	Length	Height

Table B.2 summarises all the parameters, while Table B.3 all the variables. The problem is modeled as a constraint satisfation problem (B.1-B.14), where the result is feasible/unfeasible.

Table B.2: List of parameters

Name	Description
I	Set of items indexes
O	Set of available rotations
m	Cardinality of I
L	Length of the bin
W	Width of the bin
H	Height of the bin
l_i	Length of item i
w_i	Width of item i
h_i	Height of item i

Table B.3: List of variables

Name	Description
x_i, y_i, z_i	Position of the item i, x_i, y_i, z_i, are the respective coordinates of the front-bottom-left edge
$x_i^{'}, y_i^{'}, z_i^{'}$	Coordinate of the back-top-right edge of the item i.
$\varphi_{o,i}$	1 if the applied rotation for the item i is $o \in O$, otherwise 0
$x^p_{i,k}, y^p_{i,k}, z^p_{i,k}$	1 if item i is placed after item k along the coordinate x, y and z, which means respectively $x_k^{'} \leq x_i$, $y_k^{'} \leq y_i$, $z_k^{'} \leq z_i$, otherwise 0

subject to

$$x_i^{'} - x_i = l_i \cdot \varphi_{0,i} + w_i \cdot \varphi_{1,i} \qquad \forall i \in I \qquad \text{(B.1)}$$

$$y_i^{'} - y_i = l_i \cdot \varphi_{1,i} + w_i \cdot \varphi_{0,i} \qquad \forall i \in I \qquad \text{(B.2)}$$

$$z_i^{'} - z_i = h_i \qquad \forall i \in I \qquad \text{(B.3)}$$

$$x_i^{'} \leq W \qquad \forall i \in I \qquad \text{(B.4)}$$

$$y_i^{'} \leq L \qquad \forall i \in I \qquad \text{(B.5)}$$

$$z_i^{'} \leq H \qquad \forall i \in I \qquad \text{(B.6)}$$

$$x_{i,k}^p + y_{i,k}^p + z_{i,k}^p + x_{k,i}^p + y_{k,i}^p + z_{k,i}^p \geq 1 \qquad \forall i,k \in I | i \neq k \qquad \text{(B.7)}$$

$$x_k^{'} \leq x_i + (1 - x_{i,k}^p) \cdot W \quad x_i + 1 \leq x_k^{'} + x_{i,k}^p \cdot W \qquad \forall i,k \in I | i \neq k \qquad \text{(B.8)}$$

$$y_k^{'} \leq y_i + (1 - y_{i,k}^p) \cdot L \quad y_i + 1 \leq y_k^{'} + y_{i,k}^p \cdot L \qquad \forall i,k \in I | i \neq k \qquad \text{(B.9)}$$

$$z_k^{'} \leq z_i + (1 - z_{i,k}^p) \cdot H \qquad \forall i,k \in I | i \neq k \qquad \text{(B.10)}$$

$$z_{i,k}^p = 0, x_{i,k}^p = 0, y_{i,k}^p = 0 \qquad \forall i,k \in I | i = k \qquad \text{(B.11)}$$

$$x_i, y_i, z_i, x_i^{'}, y_i^{'}, z_i^{'} \in \mathbb{N}, \qquad \forall i \in I \qquad \text{(B.12)}$$

$$x_{i,k}^p, y_{i,k}^p, z_{i,k}^p \in \{0,1\}, \qquad \forall i,k \in I \qquad \text{(B.13)}$$

$$\varphi_{o,i} \in \{0,1\}, \qquad \forall i,k \in I, o \in O \qquad \text{(B.14)}$$

Constraints (B.1)-(B.3) define the borders of the items along the axes, taking into account the possible rotation. Constraints (B.4)-(B.6) ensure that the shape of each item is contained inside the borders of the bin. Constraint (B.7) states that two items do not overlap if the boundaries along at least one dimension do not overlap. The overlapping boundaries are characterized by the constraints (B.8)-(B.10).

For further reading there are many examples of three-dimensional orthogonal packing available in the literature, e.g. (Junqueira et al., 2012; Paquay et al., 2016; Hifi et al., 2010; Lin et al., 2014).

Appendix C

Relation of equivalence with rotation along the vertical axis

Let $I = \{1, \cdots, n\}$ be a set that indexes n rectangular cuboids (items). For every item $i \in I$, let us define their dimensions as width w_i, height h_i and length l_i. Two items $i, k \in I$ are equivalent for the relation of equivalence $\cong (i, k)$ defined in (C.1), if and only if there exists an orthogonal rotation r along the vertical axis that overlaps the dimensions of i with the dimensions of k.

$$\begin{aligned} i \cong k \Leftrightarrow &((l_i = l_k \wedge w_i = w_k) \vee \\ &(l_i = w_k \wedge w_i = l_k)) \wedge h_i = h_k \end{aligned} \tag{C.1}$$

Reflexivity, symmetry and transitiveness of (C.1) are trivial to prove. □

Appendix D

Application of the results of this thesis

The work presented in this thesis has been used in the following research projects:

Chapter 5 is one of the two results of the project "INTELLIGENT REAL-TIME SEAT ALLOCATION, T. T. Nguyen (PI), A. A. Makki, D. Yazdani, C. Lersteau, I. Deplano, J. J. Peiro Ramada, R. Wang, Y. Ancele, funded by Department for Transport via the T-TRIG programme. 2017". The second result has been published in Yazdani et al. (2019).

Chapter 6, Chapter 7 and Chapter 8 have been developed under the project "Anticipating and mitigating reactionary delays – a case study on the Northern line of Merseyrail, Dr. Trung Thanh Nguyen (PI), Mr. Igor Deplano and Dr. Qian Zhang. Funded by the Rail Safety and Standards Board, 2018-2019.". The results of Chapter 5 and Chapter 7 are the core components of a simulator for the estimation of reactionary delays, that is exploited to build a support decision system to mitigate the overall spreading effect of the reactionary delays.

The results of Chapter 6 and Chapter 8 are used in the project "COINS - Customer-Operational Information System for Railway Stations, Dr.Trung Thanh Nguyen (PI), Dr. Alison Hardy, Dr.Charly Lersteau, Mr. Igor Deplano, Dr. Qian Zhang, Mr. Yannis Ancele and Mr. Ahmed Makki. InnovateUK SBRI: First of A Kind, March – November 2019." to develop a backup system to estimate the train arrival time at station. The system is enabled when the real-time data feed of DARWIN is down.

The results of Chapter 6, Chapter 7 and Chapter 8 are used in the project "ANTI-SLIP: A study on using Network Rail's and train borne information to anticipate and mitigate the impact of slippery rail, Dr. Trung Thanh Nguyen (PI), Mr. Igor Deplano, Dr.

Charly Lersteau and Dr. Qian Zhang. Rail Safety and Standards Board, 2019-2020." to estimate performance-related impact of low-adhesion events.

Appendix E

Publications resulting from this thesis

In the course of completeing the work presented in this thesis, the contents of Chapter 3 have been published in a refereed journal:

Igor Deplano, Charly Lersteau, Trung Thanh Nguyen (2019), A mixed-integer linear model for the Multiple Heterogeneous Knapsack Problem with realistic container loading constraints and bins' priority. *International Transactions in Operational Research*. doi:10.1111/itor.12740

The contents of Chapter 5 have been published in a refereed journal:

Igor Deplano, Danial Yazdany, Trung Thanh Nguyen. The Offline Group Seat Reservation Knapsack Problem with Profit on Seats. *IEEE access*, vol. 7, pp. 152358-152367, 2019.